

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de Doctorat:

AUTOMÀTICA, ROBÒTICA I VISIÓ

Tesi doctoral

**Robust navigation for industrial service robots**

Jérémie Deray

Directors: Joan Solà Ortega  
Juan Andrade-Cetto

· July 14, 2020 ·



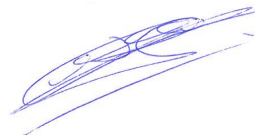
# Declaration of Authorship

I, Jérémie Deray, declare that this thesis titled, “Robust navigation for industrial service robots” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

---

Signed:



---

Date: July 14, 2020



# Acknowledgements

I would like to thank first and foremost my supervisors, Joan Solà Ortega, Juan Andrade-Cetto and Luca Marchionni, for their constant support and infinite patience throughout my doctoral studies. Their help has been invaluable and so is the knowledge they transmitted me.

Additionally I would like to express my gratitude and thank Francesco Ferro and through him everyone at PAL Robotics for taking me onboard for this incredible journey through robotics, making me part of their family, and helping shaping my understanding and competences as a roboticist. I can not forget that I entered the office as a Master student and left a prepared software engineer in robotics.

I also want to thank the members of the jury for accepting to examine the present thesis and thus dedicating time and efforts to the culmination of the work presented along these pages.

I also address a warm thanking to the many people, students, researchers, interns, engineers, friends, that I met over the past years. In particular my friend Bence Magyar, with whom I share insightful discussions and mutual understanding.

I have a tender thought for my family and thank them for their loving encouragements, despite the distance between us. I am grateful to my parents for letting me own my choices and pushing me toward their realization.

Finally, I thank my love, Elisabeth, for being by my side during the completion of this work, through the happy times and the more difficult ones, across countries and continents.



# Abstract

As one of the fundamental problems of robotics, the different challenges that constitute navigation have been studied for decades. Robust, reliable and safe navigation is a key factor for the enablement of higher level functionalities for robots that are going to evolve around humans on a daily basis. Throughout the present thesis, we tackle the problem of navigation for robotic industrial mobile-bases. We identify its components and analyze their respective challenges in order to address them. The research work presented here ultimately aims at improving the overall quality of the navigation stack of a commercially available industrial mobile-base.

To introduce and survey the overall problem we first break down the navigation framework into clearly identified smaller problems. We examine the problem of simultaneously mapping the environment and localizing the robot in it by exploring the state of the art. Doing so we recall and detail the mathematical grounding of the *Simultaneous Localization and Mapping* (SLAM) problem. We then review the problem of planning the trajectory of a mobile-base toward a desired goal in the generated environment representation. Finally we investigate and clarify the concepts and mathematical tools of the Lie theory, which we use extensively to provide rigorous mathematical foundation to our developments, focusing on the subset of the theory that is useful to state estimate in robotics.

As the first identified space for improvements, the problem of place recognition for closing loops in SLAM is addressed. Loop closure concerns the ability of a robot to recognize a previously visited location and infer geometrical information between its current and past locations. Using only a 2D laser range finder sensor, the task is challenging as the perception of the environment provided by the sensor is sparse and limited. We tackle the problem using a technique borrowed from the field of *Natural Language Processing* (NLP) which has been successfully applied to image-based place recognition, namely the Bag-of-Words. We further improve the method with two proposals inspired from NLP. Firstly the comparison of places is strengthened by taking into account the natural relative order of features in each individual sensor readings. Secondly, topological correspondences between places in a corpus of visited places are established in order to promote together instances that are ‘close’ to one another. We evaluate both our proposals separately and jointly on several data sets, with and without noise, and show an improvement over the state of the art.

We then tackle the problem of motion model calibration for odometry estimation. Given a mobile-base embedding an exteroceptive sensor able to observe ego-motion,

we propose a novel formulation for estimating the intrinsic parameters of an odometry motion model. Resorting to an adaptation of the pre-integration theory initially developed for the IMU motion sensor, we employ iterative nonlinear on-manifold optimization to estimate the wheel radii and wheel separation. The method is further extended to jointly estimate both the intrinsic parameters of the odometry model together with the extrinsic parameters of the embedded sensor. The method is validated in simulation and on a real robot and is shown to converge toward the true values of the parameters. It is then shown to accommodate to variation in model parameters quickly when the vehicle is subject to physical changes during operation.

Following the generation of a map in which the robot is localized, we address the problem of estimating trajectories for motion planning. We devise a new method for estimating a sequence of robot poses forming a smooth trajectory. Regardless of the Lie group considered, the trajectory is seen as a collection of states lying on a spline with non-vanishing  $n$ -th derivatives at every points. Formulated as a multi-objectives nonlinear optimization problem, it allows for the addition of cost functions such as velocity and acceleration limits, collision avoidance and more. The proposed method is evaluated for two different motion planning tasks, the planning of trajectories for a mobile-base evolving in the  $SE(2)$  manifold, and the planning of the motion of a multi-link robotic arm whose end-effector evolves in the  $SE(3)$  manifold. Furthermore, each task is evaluated in increasingly complex scenarios. In either cases, it is shown to perform comparably or better than the state of the art while producing more consistent results.

From our Lie theory study, we push further the idea of enablement introducing a new, ready to use, programming library called `manif`. The library is open source, publicly available and is developed following software programming good practices. It is designed so that it is easy to integrate and manipulate, and allows for flexible use while facilitating the possibility to extend it beyond the already implemented Lie groups. Furthermore the library is shown to be efficient compared to other existing solutions.

At last, we come to the conclusion of the doctoral study. We examine the research work and draw lines for future investigations. We also take a look over the past years and share a personal view and experience of *the PhD*.



# Resumen

La navegación autónoma es uno de los problemas fundamentales de la robótica, y sus diferentes desafíos se han estudiado durante décadas. El desarrollo de métodos de navegación robusta, confiable y segura es un factor clave para la creación de funcionalidades de nivel superior en robots diseñados para operar en entornos con humanos. A lo largo de la presente tesis, abordamos el problema de navegación para bases robóticas móviles industriales; identificamos los elementos de un sistema de navegación; y analizamos y tratamos sus desafíos. El trabajo de investigación presentado aquí tiene como último objetivo mejorar la calidad general del sistema completo de navegación de una base móvil industrial disponible comercialmente.

Para estudiar el problema de navegación, primero lo desglosamos en problemas menores claramente identificados. Examinamos el subproblema de mapeo del entorno y localización del robot simultáneamente (SLAM por sus siglas en inglés) y estudiamos el estado del arte del mismo. Al hacerlo, recordamos y detallamos la base matemática del problema de SLAM. Luego revisamos el subproblema de planificación de trayectorias hacia una meta deseada en la representación del entorno generada. Además, como una herramienta para las soluciones que se presentarán más adelante en el desarrollo de la tesis, investigamos y aclaramos el uso de teoría de Lie, centrándonos en el subconjunto de la teoría que es útil para la estimación de estados en robótica.

Como primer elemento identificado para mejoras, abordamos el problema de reconocimiento de lugares para cerrar lazos en SLAM. El cierre de lazos se refiere a la capacidad de un robot para reconocer una ubicación visitada previamente e inferir información geométrica entre la ubicación actual del robot y aquellas reconocidas. Usando solo un sensor láser 2D, la tarea es desafiante ya que la percepción del entorno que proporciona el sensor es escasa y limitada. Abordamos el problema utilizando 'bolsas de palabras', una técnica prestada del campo de procesamiento del lenguaje natural (NLP) que se ha aplicado con éxito anteriormente al reconocimiento de lugares basado en imágenes. Nuestro método incluye dos nuevas propuestas inspiradas también en NLP. Primero, la comparación entre lugares candidatos se fortalece teniendo en cuenta el orden relativo natural de las características en cada lectura individual del sensor; y segundo, se establece un corpus de lugares visitados para promover juntos instancias que están "cerca" la una de la otra desde un punto de vista topológico. Evaluamos nuestras propuestas por separado y conjuntamente en varios conjuntos de datos, con y sin ruido, demostrando mejora en la detección de cierres de lazo para sensores láser 2D, con respecto al estado del arte.

Luego abordamos el problema de la calibración del modelo de movimiento para la estimación de la odometría. Dado que nuestra base móvil incluye un sensor exteroceptivo capaz de observar el movimiento de la plataforma, proponemos una nueva formulación que permite estimar los parámetros intrínsecos del modelo cinemático de la plataforma durante el cómputo de la odometría del vehículo. Hemos recurrido a una adaptación de la teoría de preintegración inicialmente desarrollado para unidades inerciales de medida, y aplicado la técnica a nuestro modelo cinemático. El método nos permite, mediante optimización iterativa no lineal, la estimación del valor del radio de las ruedas de forma independiente y de la separación entre las mismas. El método se amplía posteriormente para identificar de forma simultánea, estos parámetros intrínsecos junto con los parámetros extrínsecos que ubican el sensor láser con respecto al sistema de referencia de la base móvil. El método se valida en simulación y en un entorno real y se muestra que converge hacia los verdaderos valores de los parámetros. El método permite la adaptación de los parámetros intrínsecos del modelo cinemático de la plataforma derivados de cambios físicos durante la operación, tales como el impacto que el cambio de carga sobre la plataforma tiene sobre el diámetro de las ruedas.

Como tercer subproblema de navegación, abordamos el reto de planificar trayectorias de movimiento de forma suave. Desarrollamos un método para planificar la trayectoria como una secuencia de configuraciones sobre una spline con  $n$ -ésimas derivadas en todos los puntos, independientemente del grupo de Lie considerado. Al ser formulado como un problema de optimización no lineal con múltiples objetivos, es posible agregar funciones de coste al problema de optimización que permitan añadir límites de velocidad o aceleración, evasión de colisiones, etc. El método propuesto es evaluado en dos tareas de planificación de movimiento diferentes, la planificación de trayectorias para una base móvil que evoluciona en la variedad  $SE(2)$ , y la planificación del movimiento de un brazo robótico cuyo efector final evoluciona en la variedad  $SE(3)$ . Además, cada tarea se evalúa en escenarios con complejidad de forma incremental, y se muestra un rendimiento comparable o mejor que el estado de la arte mientras produce resultados más consistentes.

Desde nuestro estudio de la teoría de Lie, desarrollamos una nueva biblioteca de programación llamada `manif`. La biblioteca es de código abierto, está disponible públicamente y se desarrolla siguiendo las buenas prácticas de programación de software. Está diseñado para que sea fácil de integrar y manipular, y permite flexibilidad de uso mientras se facilita la posibilidad de extenderla más allá de los grupos de Lie inicialmente implementados. Además, la biblioteca se muestra eficiente en comparación con otras soluciones existentes.

Por fin, llegamos a la conclusión del estudio de doctorado. Examinamos el trabajo de investigación y trazamos líneas para futuras investigaciones. También echamos un vistazo en los últimos años y compartimos una visión personal y experiencia del desarrollo de un doctorado industrial.

# Publications

The work presented along the chapters of this thesis has given rise to two journal publications, two conference publications, one technical report and one software publication. They are listed hereafter,

---

J. Deray, J. Solà, and J. Andrade-Cetto, “Word ordering and document adjacency for large loop closure detection in 2D laser maps,” *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1532–1539, 2017.

*Abstract* — We address in this paper the problem of loop closure detection for laser-based *Simultaneous Localization and Mapping* (SLAM) of very large areas. Consistent with the state of the art, the map is encoded as a graph of poses, and to cope with very large mapping capabilities, loop closures are asserted by comparing the features extracted from a query laser scan against a previously acquired corpus of scan features using a *Bag-of-Words* (BoW) scheme. Two contributions are here presented. First, to benefit from the graph topology, feature frequency scores in the BoW are computed not only for each individual scan but also from neighboring scans in the SLAM graph. This has the effect of enforcing neighbor relational information during document matching. Secondly, a weak geometric check that takes into account feature ordering and occlusions is introduced that substantially improves loop closure detection performance. The two contributions are evaluated both separately and jointly on four common SLAM datasets, and are shown to improve the state-of-the-art performance both in terms of precision and recall in most of the cases. Moreover, our current implementation is designed to work at nearly frame rate, allowing loop closure query resolution at nearly 22 Hz for the best case scenario and 2 Hz for the worst case scenario.

---

J. Deray, J. Solà, and J. Andrade-Cetto, “Joint on-manifold self-calibration of odometry model and sensor extrinsics using pre-integration,” *European Conference on Mobile Robots*, pp. 1–6, Prague, 2019.

*Abstract* — This paper describes a self-calibration procedure that jointly esti-

mates the extrinsic parameters of an exteroceptive sensor able to observe ego-motion, and the intrinsic parameters of an odometry motion model, consisting of wheel radii and wheel separation. We use iterative nonlinear on-manifold optimization with a graphical representation of the state, and resort to an adaptation of the pre-integration theory, initially developed for the IMU motion sensor, to be applied to the differential drive motion model. For this, we describe the construction of a pre-integrated factor for the differential drive motion model, which includes the motion increment, its covariance, and a first-order approximation of its dependence with the calibration parameters. As the calibration parameters change at each solver iteration, this allows *a posteriori* factor correction without the need of re-integrating the motion data.

We validate our proposal in simulations and on a real robot and show the convergence of the calibration towards the true values of the parameters. It is then tested online in simulation and is shown to accommodate to variations in the calibration parameters when the vehicle is subject to physical changes such as loading and unloading a freight.

---

J. Deray, B. Magyar, J. Solà, and J. Andrade-Cetto, “Timed-elastic smooth curve optimization for mobile-base motion planning” *IEEE International Conference on Intelligent Robots and Systems*, pp. 3143–3149, Macau, 2019.

*Abstract* — This paper proposes the use of piece-wise  $C^n$  smooth curve for mobile-base motion planning and control, coined *Timed-Elastic Smooth Curve* (TESC) planner. Based on a *Timed-Elastic Band*, the problem is defined so that the trajectory lie on a spline in SE(2) with non-vanishing  $n$ -th derivatives at every point. Formulated as a multi-objective non-linear optimization problem, it allows imposing soft constraints such as collision-avoidance, velocity, acceleration and jerk limits, and more. The planning process is realtime-capable allowing the robot to navigate in dynamic complex scenarios.

The proposed method is compared against the state-of-the-art in various scenarios. Results show that trajectories generated by the TESC planner have smaller average acceleration and are more efficient in terms of total curvature and pseudo-kinetic energy while being produced with more consistency than state-of-the-art planners do.

---

B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. Lane, “Timed-elastic bands for manipulation motion planning,” *IEEE Robotics and Automation Letters*, vol. 4, pp. 3513–3520, 2019.

*Abstract* — Motion planning is one of the main problems studied in the field of robotics. However, it is still challenging for the state-of-the-art methods to handle multiple conditions that allow better paths to be found. For example, considering joint limits, path smoothness and a mixture of Cartesian and joint-space constraints at the same time, pose a significant challenge for many of them. This work proposes

to use *Timed-Elastic Bands* for representing the manipulation motion planning problem, allowing to apply continuously optimized constraints to the problem during the search for a solution. Due to the nature of our method, it is highly extensible with new constraints or optimization objectives.

The proposed approach is compared against state-of-the-art methods in various manipulation scenarios. Results show that it is more consistent and less variant while performing in a comparable manner to the state-of-the-art. This behavior allows the proposed method to set a lower bound performance guarantee for other methods to build upon.

---

J. Solà, J. Deray, and D. Atchuthan, “A micro Lie theory for state estimation in robotics,” Tech. Rep. IRI-TR-18-01, Institut de Robòtica i Informàtica Industrial, Barcelona, 2018.

*Abstract* — A Lie group is an old mathematical abstract object dating back to the XIX century, when mathematician Sophus Lie laid the foundations of the theory of continuous transformation groups. Its influence has spread over diverse areas of science and technology many years later. In robotics, we are recently experiencing an important trend in its usage, at least in the fields of estimation, and particularly in motion estimation for navigation. Yet for a vast majority of roboticists, Lie groups are highly abstract constructions and therefore difficult to understand and to use.

In estimation for robotics it is often not necessary to exploit the full capacity of the theory, and therefore an effort of selection of materials is required. In this paper, we will walk through the most basic principles of the Lie theory, with the aim of conveying clear and useful ideas, and leave a significant corpus of the Lie theory behind. Even with this mutilation, the material included here has proven to be extremely useful in modern estimation algorithms for robotics, especially in the fields of SLAM, visual odometry, and the like.

Alongside this micro Lie theory, we provide a chapter with a few application examples, and a vast reference of formulas for the major Lie groups used in robotics, including most jacobian matrices and the way to easily manipulate them. We also present a new C++ template-only library implementing all the functionality described here.

---

J. Deray and J. Solà, “Manif: A micro Lie theory library for state estimation in robotics applications,” *The Journal of Open Source Software*, vol. 5, no. 46 pp. 1371, 2020.

*Abstract* — `manif` is a micro Lie theory library targeted at state estimation in robotics applications. Developed as a header-only library, with minimal dependency and a requirement on C++11 only, it is easy to integrate it to existing projects.



# Contents

<b>Acknowledgements</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>Resumen</b>	<b>v</b>
<b>Publications</b>	<b>vii</b>
<b>List of Figures</b>	<b>xvii</b>
<b>List of Tables</b>	<b>xix</b>
<b>List of Abbreviations</b>	<b>xxi</b>
<b>Notation</b>	<b>xxiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Objectives . . . . .	2
1.3 Resources . . . . .	3
1.4 Outline . . . . .	7
<b>2 Background</b>	<b>9</b>
2.1 The navigation problem . . . . .	11
2.1.1 SLAM . . . . .	11
2.1.2 Odometry . . . . .	16
2.1.3 Loop-closure . . . . .	18
2.1.4 Planning . . . . .	20
2.2 Lie Theory . . . . .	22
2.2.1 Lie groups . . . . .	23
2.2.2 The tangent spaces and the Lie algebra . . . . .	23
2.2.3 The exponential map . . . . .	26
2.2.4 The capitalized Exponential map . . . . .	26
2.2.5 The plus and minus operators . . . . .	27
2.2.6 Derivatives on Lie groups . . . . .	27

2.2.7	Uncertainty on Lie groups . . . . .	31
<b>3</b>	<b>Loop-Closure</b>	<b>33</b>
3.1	Related work . . . . .	34
3.2	Feature comparison for BoW-based visual place recognition . . . . .	34
3.2.1	Experiments . . . . .	34
3.2.2	Conclusion . . . . .	35
3.3	BoW-based loop-closure for 2D laser scans . . . . .	36
3.3.1	Feature sequence encoding as a Hidden Markov Model . . . . .	36
3.3.2	Pose-graph database augmentation . . . . .	39
3.3.3	Experiments . . . . .	40
3.3.4	Conclusion . . . . .	47
<b>4</b>	<b>Odometry Calibration</b>	<b>49</b>
4.1	Related work . . . . .	50
4.2	Abstraction on Lie groups of the pre-integration theory . . . . .	51
4.3	Pre-integration for the differential drive motion model . . . . .	52
4.3.1	Delta pre-integration . . . . .	52
4.3.2	Delta Jacobian pre-integration . . . . .	53
4.3.3	Delta covariance pre-integration . . . . .	53
4.3.4	Residual . . . . .	53
4.4	Joint calibration of differential drive intrinsic and sensor extrinsic parameters . . . . .	53
4.4.1	Jacobians and covariance propagation . . . . .	54
4.4.2	Residual . . . . .	54
4.5	Calibration . . . . .	55
4.5.1	Batch calibration process . . . . .	55
4.5.2	Online calibration process . . . . .	55
4.6	Experiments . . . . .	57
4.7	Conclusion . . . . .	58
<b>5</b>	<b>Motion Planning</b>	<b>61</b>
5.1	Related work . . . . .	62
5.2	Problem formulation . . . . .	62
5.2.1	Timed-Elastic Smooth Curve . . . . .	62
5.3	TESC for mobile-base motion planning . . . . .	66
5.3.1	Mobile-base specific constraints . . . . .	66
5.3.2	Experiments . . . . .	67
5.3.3	Conclusion . . . . .	70
5.4	TESC for manipulation motion planning . . . . .	73
5.4.1	Related work . . . . .	73
5.4.2	A different take on Timed-Elastic Band . . . . .	74
5.4.3	Arm specific constraints . . . . .	75
5.4.4	Experiments . . . . .	76
5.4.5	Conclusion . . . . .	79



<b>6</b>	<b>The manif library</b>	<b>81</b>
6.1	Related work . . . . .	82
6.2	Design . . . . .	83
6.3	Evaluation . . . . .	87
6.4	Conclusion . . . . .	89
<b>7</b>	<b>Conclusion</b>	<b>91</b>
7.1	Summary of contributions . . . . .	92
7.2	Future work . . . . .	93
7.3	Closing words . . . . .	94
<b>A</b>	<b>A brief history of ROS</b>	<b>97</b>
<b>B</b>	<b>Reminders</b>	<b>101</b>
B.1	Jacobians on vector spaces . . . . .	101
B.2	The chain rule . . . . .	102
<b>C</b>	<b>Lie theory take away</b>	<b>103</b>
<b>D</b>	<b>The manif library</b>	<b>107</b>
D.1	Implementation details . . . . .	107
D.2	Writing generic code . . . . .	109
D.3	On-Manifold optimization using Ceres and manif . . . . .	112



# List of Figures

1.1	The REEM robot surrounded at a fair. . . . .	2
1.2	PAL's robot family. . . . .	5
1.3	The TIAGo robot. . . . .	6
2.1	The SLAM structure. Hollow figures correspond to the truth whereas filled figures correspond to the estimates. . . . .	13
2.2	Iterative construction of a factor graph and its support sparse Hessian matrix. . . . .	15
2.3	A factor graph representation of Fig. 2.1. <i>Left</i> : Nodes representing known data have been replaced by factors (squares) which depends on the unknown variables (or states) (circles). <i>Right</i> : The same graph. Unknown states are labeled with a single index $i \in [0, 7]$ and factors are labeled with a single index $k \in [1, 10]$ . . . . .	16
2.4	Illustration of a loop closure. . . . .	19
2.5	The TEB is a sequence of robot poses forming a trajectory. Consecutive poses are tied to one another by a time interval. . . . .	21
2.6	Representation of the relation between the Lie group and the Lie algebra. The Lie algebra $\mathcal{T}_{\mathcal{M}}(\mathcal{E})$ (red plane) is the tangent space to the Lie group's manifold $\mathcal{M}$ (blue sphere) at the identity $\mathcal{E}$ . Through the exponential map, each straight path $\mathbf{v}t$ through the origin on the Lie algebra produces a path $\exp(\mathbf{v}t)$ around the manifold which runs along the respective geodesic. Conversely, each element of the group has an equivalent in the Lie algebra. This relation is so profound that (nearly) all operations in the group, which is curved and nonlinear, have an exact equivalent in the Lie algebra, which is a linear vector space. Note that the sphere in $\mathbb{R}^3$ is not a Lie group, it is however a convenient figural representation that can be drawn on paper. . . . .	24
2.7	Mappings between the manifold $\mathcal{M}$ and the representations of its tangent space at the origin $\mathcal{T}_{\mathcal{M}}(\mathcal{E})$ (Lie algebra $\mathfrak{m}$ and Cartesian $\mathbb{R}^m$ ). Maps hat $(\cdot)^\wedge$ and vee $(\cdot)^\vee$ are the linear invertible maps or <i>isomorphisms</i> (2.52–2.53), $\exp(\cdot)$ and $\log(\cdot)$ map the Lie algebra to/from the manifold, and $\text{Exp}(\cdot)$ and $\text{Log}(\cdot)$ are shortcuts to map directly the vector space $\mathbb{R}^m$ to/from $\mathcal{M}$ . . . . .	27

3.1	Comparison of <code>OpenCV</code> features for BoW-based place recognition - recall at 99+% precision. . . . .	36
3.2	Top: Clockwise ordered words of two scans and their matches. Middle: The resulting hidden Markov model. Each cell represents the product $\phi_{s_n-x s_n} \cdot \theta_{s_n o_m}$ e.g cells $Y-A$ & $A-A$ . Green squared cells represent the best path across the complete model. Bottom: The final sequence of states given the observations $O_n$ . . . . .	37
3.3	FLIRT features (violet spheres) extracted from a scan (blue dots and segments) superimposed on a map. . . . .	41
3.4	Variation of the average $F1$ score with respect to branching factor and tree level. . . . .	42
3.5	Recall versus precision for 20 candidates varying the inliers threshold.	43
3.6	Recall versus number of candidates at 99% precision. . . . .	43
3.7	Occupancy grid generated from the Intel-lab data set before and after the addition of virtual obstacles. . . . .	46
3.8	Occupancy grid of a large mall floor (approx. $2900m^2$ ). Red dots represent robot key frames and green edges loop-closures. During mapping, 415 loop closures were detected of which only 2 were false positives, easily eliminated using distance constraints. . . . .	47
4.1	The pre-integrated delta $\Delta_{ij} \in \mathcal{M}$ contains all motion increments from time $i$ up to time $j$ , so that $\mathbf{x}_j = \mathbf{x}_i \circ \Delta_{ij}$ . The current delta $\delta_k \in \mathcal{M}$ contains the motion from time $j$ to $k$ , computed from the last motion measurement at time $k$ . We have that $\Delta_{ik} = \Delta_{ij} \oplus \delta_k$ . . . . .	51
4.2	A differential drive robot moves from pose $\mathbf{x}_i$ to $\mathbf{x}_j$ . It mounts an exteroceptive sensor at pose $\mathbf{T}$ (red) with respect to the robot base (blue). It holds that $\Delta_{ij} \circ \mathbf{T} = \mathbf{T} \circ \Delta_{ij}^S$ . . . . .	54
4.3	Factor graph for the estimation problem. Two state blocks $\mathbf{c}$ and $\mathbf{T}$ are linked by a number of factors, each computing a residual of the type $\mathbf{r}_k = \mathbf{\Omega}_k^{\top/2}(\Delta_k(\mathbf{c}) \circ \mathbf{T} - \mathbf{T} \circ \Delta_k^S)$ (see Fig. 4.2). An absolute factor (grey) of the type $\mathbf{r}_0 = \mathbf{\Omega}_0^{\top/2}(\mathbf{c} - \mathbf{c}_0)$ keeps $\mathbf{c}$ close to its nominal values $\mathbf{c}_0$ . . . . .	55
4.4	Evolution and effects of the batch calibration. . . . .	58
4.5	Integrated odometry of the real robot before and after calibration - respectively red and blue. . . . .	59
4.6	Comparison of the evolution of vehicle kinematic parameters and the aggregated cost factor $F$ for a fixed window size and a dynamic one. . . . .	60
5.1	Decomposition of TIAGo's motion traversing through a scene with obstacles marked by dark regions on the floor . . . . .	63
5.2	The pose $\mathbf{x}_i$ is constrained towards the smooth curve defined by $\mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \boldsymbol{\tau}_{i-1}, \boldsymbol{\tau}_{i+1}$ and $s$ (tangents are illustrated by the arrows). . . . .	64
5.3	Ternary occupancy grid and Euclidean distance grid of a mapped office. Obstacles are shown in black on both figures. . . . .	67
5.4	Experiment setups. TEB-planner is depicted with violet arrow while TESC is with red arrows. The initial pose at the center of the grid is depicted with a larger black arrow. . . . .	68
5.5	Decomposition of maneuvers. . . . .	71
5.6	Experiment results of both planners in each of the three scenarios. . . . .	72

5.7	Experiment metrics for the evaluation of TEB2MP in the empty environment. . . . .	77
5.8	Experiment metrics for the evaluation of TEB2MP in the box environment. . . . .	78
5.9	Experiment metrics for the evaluation of TEB2MP in the industrial environment. . . . .	80
6.1	Comparison of the time required to evaluate the value and Jacobians of (6.6) for $N \in [1, 10]$ . . . . .	88
A.1	Re-inventing the Wheel, by Jorge Cham. . . . .	98
A.2	A collection of metrics showing the growth of the ROS community <sup>2</sup> . . . . .	99



# List of Tables

1.1	TIAGo-base technical specifications. . . . .	5
3.1	Selected combination of detectors and descriptors from <code>OpenCV</code> used in the experiments. . . . .	35
3.2	Public data sets used for the experiments [85]. . . . .	40
3.3	Aliases for the different methods compared. . . . .	40
3.4	First experiment : Algorithms statistics for each dataset at max F1 score. <i>Top-N</i> : Number of candidates - <i>Inl</i> : Inliers threshold - <i>TP</i> : True Positive - <i>FP</i> : False Positive - <i>F1</i> : F1 score (per mille ‰) . . .	44
3.5	Second experiment : Algorithms statistics for each indoor dataset at max F1 score. <i>Top-N</i> : Number of candidates - <i>Inl</i> : Inliers threshold - <i>TP</i> : True Positive - <i>FP</i> : False Positive - <i>F1</i> : F1 score (per mille ‰)	46
3.6	Average query time (seconds). . . . .	47
5.1	Metrics used in the experiments evaluation. . . . .	69
5.2	Metrics used in the experiments evaluation. . . . .	76
6.1	Lie theory C++ libraries comparison. ‘Any Scalar’ indicates a header-only library and the symbol ‘*’ means partial. . . . .	83
6.2	Time to evaluate the value and Jacobians of (6.8). . . . .	89
C.1	Elementary Jacobian blocks . . . . .	103
C.2	Typical Lie groups used in robotics, including the trivial $\mathbb{R}^n$ . . . . .	103
C.3	Operating on typical Lie groups used in robotics. . . . .	104
C.4	Adjoint and Jacobians of typical Lie groups used in robotics. . . . .	105





# List of Abbreviations

**CPU** *Central Processing Unit*

**API** *Application Programming Interface*

**NIP** *Nonlinear Programming*

**CRTP** *Curiously Recurring Template Pattern*

**ROS** *Robot Operating System*

**OSRF** *Open Source Robotics Foundation*

**SQP** *Sequential Quadratic Programming*

**AD** *Automatic Differentiation*

**IMU** *Inertial Measurement Unit*

**SLAM** *Simultaneous Localization and Mapping*

**V-SLAM** *Visual-Simultaneous Localization and Mapping*

**BoW** *Bag-of-Words*

**LRF** *Laser Range Finder*

**DoF** *Degree of Freedom*

**IMU** *Inertial Measurement Unit*

**WOLF** *Windowed Localization Frames*

**HMM** *Hidden Markov Model*

**RANSAC** *Random Sample Consensus*

**ICP** *Iterative Closest Point*

**PnP** *Perspective-n-Point*

**ICL** *Iterative Closest Line*

**EKF** *Extended Kalman Filter*  
**EIF** *Extended Information Filter*  
**TF** *Term Frequency*  
**IDF** *Inverse Document Frequency*  
**TF-IDF** *Term Frequency - Inverse Document Frequency*  
**NLP** *Natural Language Processing*  
**ODE** *Ordinary Differential Equation*  
**TESC** *Timed-Elastic Smooth Curve*  
**RRT** *Rapidly-exploring Random Tree*  
**RRTConnect** *Rapidly-exploring Random Tree Connect*  
**RRT\*** *Rapidly-exploring Random Tree\**  
**PF** *Potential Fields*  
**OG** *Occupancy Grid*  
**PRM** *Probabilistic Road Maps*  
**PRMS** *Probabilistic Road Maps\**  
**TEB** *Timed Elastic Band*  
**NURBS** *Non-Uniform Rational B-Splines*  
**SDF** *Signed Distance Field*  
**EDG** *Euclidean Distance Grid*  
**MPC** *Model Predictive Control*  
**TEB2MP** *Timed Elastic Bands for Manipulation Motion Planning*  
**FK** *Forward Kinematics*  
**IK** *Inverse Kinematics*  
**AABB** *Axis-Aligned Bounding Box*  
**STOMP** *Stochastic Optimization for Motion Planning*  
**TrajOpt** *Trajectory Optimization for Motion Planning*  
**GPMP2** *Gaussian Process Motion Planner 2*  
**CHOMP** *Covariant Hamiltonian Optimization for Motion Planning*  
**GP** *Gaussian Process*  
**MAP** *Maximum a Posteriori*  
**SLoM** *Sparse Least Squares on Manifold*

# Notation

Throughout the thesis the following notation is used,

- Lower case regular letters denote **scalars**,  $a = 42$ , or a function when accompanied by parenthesis,  $f(x) = x + 1$
- Lower case bold letters denote **column vectors**,  $\mathbf{v} = [1, 2, 3]^T$ .
- Capital bold letters denote **matrices**  $\mathbf{M} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ .
- Identity and zero matrices are respectively denoted  $\mathbf{I}$  and  $\mathbf{0}$ . Their dimensionality are explicated where necessary, *e.g.*  $\mathbf{I} \in \mathbb{R}_{3 \times 3}$  and  $\mathbf{0}_{3 \times 3}$  are both  $3 \times 3$  matrices.
- Given  $y = f(x)$  we note  $\mathbf{J}_x^y \triangleq \frac{\partial y}{\partial x}$  the Jacobian matrix of  $y$  with respect to  $x$ .

At a given time  $t$ ,

- $\mathbf{x}_t$  - A state vector representing the robot position and orientation - *e.g.*  $\mathbf{x}_t = [x, y, \theta] \in \mathbb{R}^3$ . It may include extra variables such as the robot velocity.  
The set  $X_T = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T\}$  is the history of robot poses for  $t \in [t_0, t_T]$ .
- $\mathbf{u}_t$  - A control signal executed at  $t - 1$  inducing the robot motion to  $\mathbf{x}_t$ .  
The set  $U_T = \{\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_T\}$  is the history of input commands.
- $\mathbf{l}_n$  - A state vector representing a landmark position and orientation.  
The set  $L = \{\mathbf{l}_0, \mathbf{l}_1, \dots, \mathbf{l}_N\}$  embeds all landmarks,  
while  $L_t = \{\mathbf{l}_{t,0}, \mathbf{l}_{t,1}, \dots, \mathbf{l}_{t,n}\}$  only includes landmarks observed at time  $t$ .
- $\mathbf{z}_{t,n}$  - A measurement of landmark  $\mathbf{l}_n$  at  $\mathbf{x}_t$ .  
The set  $Z_M = \{\mathbf{z}_{0,0}, \mathbf{z}_{1,1}, \dots, \mathbf{z}_{T,N}\}$  embeds all observations.



# Chapter 1

## Introduction

---

*Since a few years we see an acceleration in the development and spreading of mobile robots evolving alongside humans, both in public and private spaces. Automation has began decades ago in industries such as the automotive industry with in-situ robots (e.g. robotic arms). Despite a rapid growth thanks to the extremely structured environment of factories and warehouses, limiting the inherent uncertainty of the world to a set of well established rules, integration happened almost exclusively by modifying the environment (metal cage, painted railway etc.). Robots were strictly separated from workers in order to prevent any dramatic accident. Today's novelty lies in the mobility aspect of the robots. With the current advances in compliance, mapping, perception, together with the interest of the public, robots are slowly but surely getting out of their metal cage. This evolution obviously raises many different challenges both technical and societal. In this work we focus on one of the fundamental problems in robotics: Navigation.*

---

The thesis disclosed here is framed in the *Industrial PhD program* managed by the *Generalitat de Catalunya*. The work is developed in a collaborative framework between a University and a company. The partners are, the *Institut de Robòtica i Informàtica Industrial*, a joint University Research institute from *CSIC* and the *Universitat Politècnica de Catalunya* (UPC), and *PAL Robotics* on the company side. The goals of the research are aligned with the mid-term plans of the company.

## 1.1 Motivation

Although service robots began to appear in semi-structured spaces opened to public, such as shopping malls or museums, their presence is still unusual. Most of the times a robot appears in public, it is quickly surrounded by a crowd of people being curious, leading to a partial or complete occlusion of its sensors. In this condition the robot is nearly blind and naive navigation methods fail to localize it. Fig. 1.1 depicts such case, highlighting the difficulties of such situation.



Figure 1.1: The REEM robot surrounded at a fair.

Before being able to cope with the fairly unstructured and dynamic environment that homes are, robots are going to be massively deployed in semi-structured spaces (hospitals, retail stores, malls, museums *etc.*). Even omitting extreme and punctual examples such as the one depicted in Fig. 1.1, it is very challenging for a robot to evolve in human environments. These environments were designed for human crowds and thus do not incorporate any facilities dedicated to helping robots. Furthermore, the authorities in charge of these places are often reluctant to installing any markers or other sensors, especially if they are visible. A robotic system must therefore be self-contained. To be deployed, robots capacity to sense and adapt to dynamic scenes, together with their reliability and robustness, must be improved for both the people and the machine safety during long-term unsupervised deployment. To adapt to dynamic environments, robots must be able to recognize places and move around despite static and/or dynamic changes (change in the furniture arrangement, people passing by, seasonal change *etc.*). Moreover, since places as those aforementioned are commonly very large, robots must be able to create, organize and update fairly large representations of the environment.

## 1.2 Objectives

The main objective of the thesis is to investigate novel methods to improve and bolster the overall navigation system of industrial mobile-bases. These methods must ensure the robustness over time of the framework as they will be implemented on commercially available service robots which are deployed in real environments.

Whereas the research on navigation benefits from the large and ever growing variety of sensors available, industrial mobile-bases more often embeds a minimal set of well known and well proven industrial sensors. A common such set includes wheel encoders, to compute the robot odometry, and a 2D *Laser Range Finder* (LRF) to

observe the robot's surroundings. Regular cameras are also commonly found but usually employed for related, non-critical tasks. A consequence of the discrepancy between researchers exploring the benefits of new sensors and the cautious conservatism of industrial is that current solutions to planar LRF-based navigation do not benefit from all of the latest advances and concepts developed.

From the aforementioned motivations, the following main objectives for the doctoral study were identified:

- Improve the estimated odometry and calibration of mobile bases.
- Develop a place recognition module for LRF-based navigation. Such module should be usable for both re-localization and/or loop-closure.
- Enhance the motion planning capability of mobile-bases.

## 1.3 Resources

### The Robot Operating System

The *Robot Operating System* (ROS) [1] is a software ecosystem that aims at standardizing and facilitating the development of robotics applications. ROS essentially defines an *Application Programming Interface* (API) through which one may access a robot hardware, sensors and actuators, but also various kind of higher-level information such as detected objects, integrated odometry and much more. Over the course of the past decade, it has become the *de-facto* standard robotics framework, both in research and industry. The reader may find a brief history of the project in Appendix A.

#### ROS in a nutshell

Despite its name, the ROS is not an actual operating system, nor is it really a full framework. ROS is better described as an ecosystem composed of core components which form a *middleware* and a collection of higher level libraries that benefit from the middleware. A middleware can be thought of as a low-level framework built on top of an existing operating system. By defining an API, it organizes the communication between programs in a distributed system. The primary operating system supported by ROS is Ubuntu<sup>1</sup>. This architecture allows one to develop a robotics system as a collection of interconnected sub-programs, one for each sub task/functionality of the system, as opposed to a monolithic block. The connections between sub-programs are then defined and handled by the ROS middleware.

The ROS terminology is as follows, individual programs are called *nodes* which communicate with each other through,

- *Topics*. A publisher emits a stream of data on a topic that a subscriber(s) can retrieve. E.g. A camera driver (publisher) emits an image stream that an object recognition module (subscriber) will process.
- *Services*. A synchronous, per-request, client/server communication. E.g. The navigation stack (client) sends a request to the mobile-base controller (server) in order to lower the maximum speed authorized.

---

<sup>1</sup><https://ubuntu.com/>

- *Actions.* An asynchronous, per-call, client/server communication. The client can asynchronously monitor the state of the server, or cancel the request anytime. E.g. The grasping module (client) asks the manipulation planning (server) to find a trajectory to safely move the arm to a desired configuration.

Furthermore the ROS environment includes *parameters* which allows the user to set some values which are then retrievable by the nodes. Finally, the ecosystem comes with a load of tools, including system logging, orchestration, 3D simulation and visualization and much more. The overall ROS middleware is programming language agnostic, although it mainly supports C++ and Python.

Finally it is worth noting that ROS is totally Open Source and most of the core packages are released under a BSD license. This fact is most likely another key aspect of its wide adoption.

## PAL Robotics

Founded in 2004, PAL Robotics is a worldwide leading company in biped humanoid and service robots based in Barcelona. Aiming at enhancing people's quality of life, the company's team is composed of passionate engineers that create research platforms as well as service robots for tasks such as order fulfillment in warehouse or inventory making.

Since 2004 and the release of the first version of the REEM-A robot, PAL Robotics developed several robotic platforms including:

- REEM-A Officially released in 2005 it won the following year the walking challenge of the RoboCup.
- REEM-B The strongest robot of its time since it could carry a load about 20% of its own weight.
- REEM-H1 The first wheel-based mobile humanoid robot of the company.
- REEM The second wheel-based mobile humanoid robot and one of the current platforms.
- REEM-C A human-size biped humanoid robot.
- TIAGo-base A mobile base platform developed targeting both industry and research needs.
- TIAGo A mobile manipulator that adapts to research needs.
- StockBot A service robot that ease inventory making.
- Talos A human-size biped humanoid robot and one of the most leading-edge robotic platforms in the world.

The whole range of platforms is depicted on a timeline in Fig. 1.2. All of the recent robots are a 100% ROS-based.

## TIAGo

The TIAGo robot is the primary platform used throughout this thesis and is therefore briefly described here.

TIAGo, depicted in Fig. 1.3, is a modular mobile-manipulator developed by PAL Robotics since 2015. Its base, branded TIAGo-base, is a complete robot by itself.



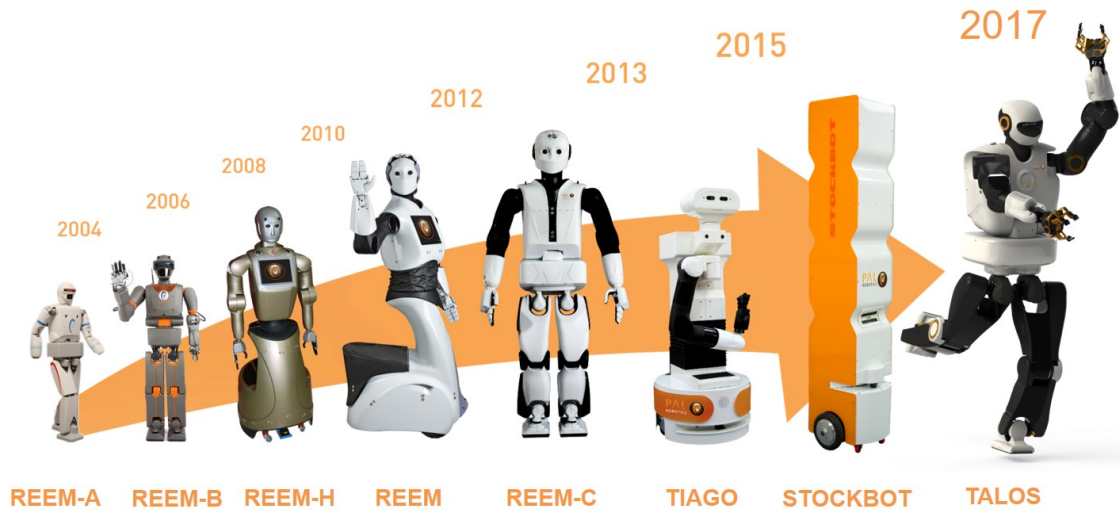


Figure 1.2: PAL's robot family.

Payload	100 kg
Max speed	1.5 m/s
Run time	10 h
Charging time	0 – 100 4.5 h
Footprint	540 mm
Weight	40 kg
Communication	WiFi, Bluetooth, I/O, CAN

Table 1.1: TIAGo-base technical specifications.

It is a two-wheel differential drive and embeds several sensors together with the onboard computer. Sensors include sonars on the rear side and a LRF on the front. TIAGo-base is primarily used for any kind of indoor delivery tasks. The base technical specifications are listed in Table 1.1. The TIAGo manipulator further expands the base capacity and list of sensors. It has a pan-tilt head which host a RGB-D camera, a laptop tray offering a wide variety of connectic, an elevating column for torso and a 7 *Degree of Freedom* (DoF) arm with exchangeable end-effectors. The robot is fully based on ROS and its simulation model is publicly available on internet<sup>2</sup>.

<sup>2</sup><http://wiki.ros.org/Robots/TIAGo>



Figure 1.3: The TIAGo robot.

## Institut de Robòtica i Infomàtica Industrial

IRI is a joint university research center focused on human-centered robotics research. Within its mobile robotics group, effort is placed in developing a full *Simultaneous Localization and Mapping* (SLAM) framework called *Windowed Localization Frames* (WOLF). This framework helps solving various optimization problems in robotics such as SLAM, visual odometry, sensor calibration, path planning and more. It includes a structure for having the data accessible and organized, plus some functionality for managing this data. Some parts of this thesis are contributions to the whole WOLF architecture.

## 1.4 Outline

While the reader may find an overview of the SLAM framework together with a general state of the art in Chapter 2, it has to be noted that a literature review specific to each contribution is presented at the beginning of each chapter. This allows for contextualizing the contribution with its related work. Furthermore, contributions – thus chapters – are presented in a chronological order following the course of the doctoral study.

The present manuscript is structured as follows:

**Chapter 1.** Introduces the topic of the thesis, its context, motivation and objectives. Finally it details the structure of the present document.

**Chapter 2.** Reviews the problem of navigation for mobile-bases. It characterizes the SLAM framework’s mathematical foundations and draws up a state of the art overview. The chapter also offer a study of the Lie theory for state estimation in robotics, a key tool used throughout the thesis.

**Chapter 3.** Details the *Bag-of-Words* (BoW) scheme applied to the task of place recognition for loop closure. A visual feature comparison is conducted to verify if the most commonly encountered feature in Visual BoW is also the best choice. In a second time, the BoW scheme is applied to the case of LRF sensors and improvements to the classical algorithm are proposed.

**Chapter 4.** Presents an algorithm for estimating the intrinsic parameters of a mobile-base motion model. It allows for both batch and online calibration. An extension to the algorithm is also presented so that it simultaneously estimates the motion model intrinsic parameters and the extrinsic parameters of a sensor able to observe ego motion.

**Chapter 5.** Proposes a novel algorithm for motion planning that estimates smooth curves on Lie groups. The algorithm is first employed for the motion planning of a mobile-base, then for the motion planning of a robotic arm.

**Chapter 6.** Introduces the `manif` library, a C++ library developed during the course of the thesis for using Lie theory in various estimation problem in robotics.

**Chapter 7.** Concludes the thesis by summarizing the contributions and discussing their consequences. Furthermore, the lines for future work are drawn. At last, the closing words offer a personal view of the doctoral study.



# Chapter 2

## Background

---

*For a mobile robot, Navigation encompasses several different problems. The robot has to be able to generate a representation of its environment, a map, to localize itself within it. Ultimately it has to move within the environment avoiding collisions. Mapping and localization are tackled at once, by a so called SLAM framework. Once a map is generated and the robot properly localized, it can move autonomously by planning a trajectory between its current pose and a targeted one. To deal with robot states, the mathematical tools employed require the use of a specific formalism, the Lie theory.*

---

Robotics navigation encompasses three main and complementary problematic:

- **Mapping**      the problem of representing the environment as the robot perceives it from its sensor readings.
- **Localization** the problem of localizing the robot within the aforementioned representation of the environment.
- **Planning**      the problem of finding a feasible trajectory between at least two configurations in a map.

These complementary problems are extensively active research areas and are fundamental in the sense that many high level tasks depend on them. How could a robot bring us a drink, clean a room or be a guide in a museum if it does not know its environment, what is its state in this environment or how to move in it?

Tremendous efforts and progresses characterized the past years of the research community, especially for its branch employing cameras as the main and often unique sensor. Nowadays, state-of-the-art algorithms are able to accurately localize a robot online and produce a rich representation of the environment. Despite these impressive results, the overall navigation framework is still very challenged by the possible combinations of:

- Robots
  - dynamics : mobile-base, biped, unmanned aerial vehicle, underwater vehicle ...
  - available sensors : rotary/linear encoders, RGB/D/event-cameras, sonars, 2D/3D LRF, *Inertial Measurement Unit* (IMU), radars ...
  - resources : one/many *Central Processing Unit* (CPU)-cores, memory, cloud, power, communication ...
- Environments
  - indoor : warehouse, museum, hospital, house ...
  - outdoor : city, countryside, forest, surface/underwater, space ...
  - weather : day, night, sunny, cloudy, foggy, rainy ...
- Task-driven specifications
  - precision of the localization and/or map
  - size of the map
  - execution/decision speed
  - robustness, reliability

Furthermore, with so many high level algorithms involved, one may quickly face the *curse of parameters*<sup>1</sup>.

---

<sup>1</sup>Analogous to the *curse of dimensionality*, it encompasses the problems of having highly parameterizable algorithms leading to a necessary fine tuning for a particular use-case, environment, scenario *etc.*

## 2.1 The navigation problem

Both problems of mapping and localization are tackled at once by a so called *Simultaneous Localization and Mapping* (SLAM) framework. SLAM is a family of algorithms which knows different approaches (*e.g.* filtering-based, optimization-based), which can be further broken down into specific pieces – *e.g.* the distinction between localization and re-localization – and as many different sub-problems as aforementioned scenarios. The SLAM problem has known several formulations and many algorithms were developed over the years to tackle it. References to some of these formulations are given in this section. However, only the optimization-based formulation, which has become the *de-facto* standard formulation, is further detailed. The reader can find exhaustive and historical reviews in [2–6].

From an engineering point of view, they are usually presented as a composition of two distinct parts, a *front-end* and a *back-end*. The front-end manages the sensors' raw data and extracts, interprets and organizes information in order to build a mathematical representation of the problem which can then be solved by the back-end.

Before further details, a different representation that better fits the reality of the implementation of SLAM algorithms is proposed. Indeed, the front-end is composed of two main sub-modules which operate at different pace. The first sub-module aims at tracking the robot current state, possibly using the concurrently built map representation, hence must be able to operate at sensor-frame, performing short-term data association. The second module on the other hand performs long-term data association, trying to recognize places visited by the robot in the past history of the environment exploration. This operation is usually time and computation expensive, therefore must not prevent the tracking from operating at sensor(s) rate. The module subdivision may be summarized as follows:

- **Core.** Builds the actual estimation problem and eventually solving it.
- **Odometry.** Tracks the sensor motion and selects information to be added to the overall problem.
- **Loop-closure/re-localization.** Detects loop closures and re-localizes the robot.

Most modern SLAM algorithms such as [7, 8] rely on the concurrency of these three modules.

The problem of planning is then treated independently, usually after the map covers almost fully the environment in which the robot is meant to evolve autonomously.

### 2.1.1 SLAM

The SLAM problem is best formulated in terms of probabilities with Gaussian random variables of the form,

$$\mathbf{x} \sim \mathcal{N}(\bar{\mathbf{x}}, \Sigma) \in \mathbb{R}^k, \quad (2.1)$$

where  $\bar{\mathbf{x}} \in \mathbb{R}^k$  is the mean - also known as expectation - of the distribution and the covariance matrix  $\Sigma_{\mathbf{x}}$  is a symmetric positive definite matrix. Equivalently one

can look at  $\mathbf{x}$  as being composed of a noise-free mean component and a ‘small’, zero-mean, noisy component,

$$\mathbf{x} = \bar{\mathbf{x}} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Sigma}), \quad (2.2)$$

Gaussian random variables are given by the *normal distribution* which probability density is,

$$p(\mathbf{x}|\bar{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}_{\mathbf{x}}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T \boldsymbol{\Sigma}_{\mathbf{x}}^{-1}(\mathbf{x} - \bar{\mathbf{x}})\right), \quad (2.3)$$

where  $|\boldsymbol{\Sigma}_{\mathbf{x}}|$  is the covariance matrix determinant. The expectation reads,

$$\bar{\mathbf{x}} = \mathbb{E}[\mathbf{x}] = [\mathbb{E}[\mathbf{x}_1], \mathbb{E}[\mathbf{x}_2], \dots, \mathbb{E}[\mathbf{x}_k]]^T \quad (2.4a)$$

$$= \int_{-\infty}^{\infty} \mathbf{x} \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}_{\mathbf{x}}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T \boldsymbol{\Sigma}_{\mathbf{x}}^{-1}(\mathbf{x} - \bar{\mathbf{x}})\right) d\mathbf{x}, \quad (2.4b)$$

and the covariance,

$$\boldsymbol{\Sigma} \triangleq \mathbb{E}[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T]. \quad (2.5a)$$

In SLAM, such a variable might be *i.e.* the robot state vector, which update is a Markov process depending only on the previous robot state and the input control command:

$$P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \Leftrightarrow \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{w}_t), \quad \mathbf{w}_k \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\mathbf{w}}). \quad (2.6)$$

where  $f(\cdot)$  is usually non-linear and models the robot kinematics and  $\mathbf{w}_k$  is a perturbation considered Gaussian with zero-mean and covariance  $\boldsymbol{\Sigma}_{\mathbf{w}}$ .

The observation model describes the probability of making an observation  $\mathbf{z}_t$  knowing the robot and landmarks poses:

$$P(\mathbf{z}_t|\mathbf{x}_t, L) \Leftrightarrow \mathbf{z}_t = h(\mathbf{x}_t, L) + \mathbf{v}_t, \quad \mathbf{w}_k \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\mathbf{v}}). \quad (2.7)$$

where  $h(\cdot)$  is usually non-linear and models the geometry of the observation and  $\mathbf{v}_k$  is an additive noise considered Gaussian with zero-mean and covariance  $\boldsymbol{\Sigma}_{\mathbf{v}}$ .

The complete probabilistic SLAM model, that is, at time  $t$ , the joint posterior density of the landmarks, the robot pose given the history of input controls commands and the observations is then:

$$P(X_T, L|Z_t, U_T, \mathbf{x}_0). \quad (2.8)$$

This formulation, exemplified in Fig. 2.1, is known as Full-SLAM as it estimates the whole history of the robot poses together with the landmarks poses. The adjective ‘full’ is employed here as opposed to early solutions that only maintain few landmarks and the current robot pose in the problem [9]. In the case one estimates only the robot poses history by marginalizing the landmarks, such formulation is known as Pose-SLAM [10–13].



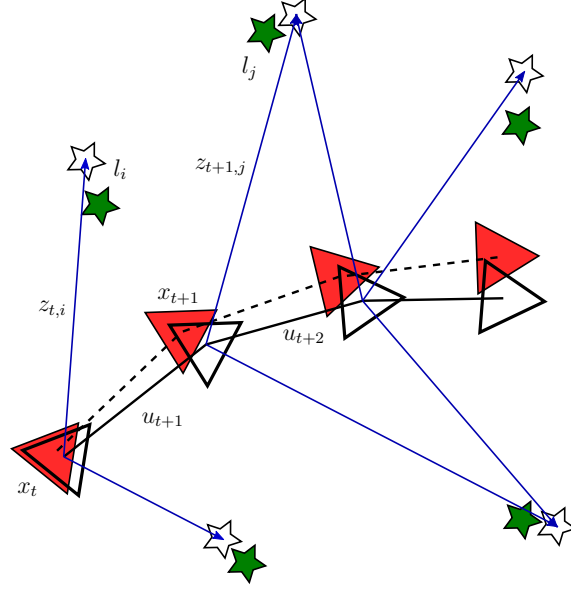


Figure 2.1: The SLAM structure. Hollow figures correspond to the truth whereas filled figures correspond to the estimates.

During what [6] refers to as “*the classical age*”, early solution to SLAM used filtering-based algorithms to solve the problem. Such formulation included the use of an *Extended Kalman Filter* (EKF) [14–16], particle filters [17, 18] or later on the *Extended Information Filter* (EIF) [10, 13].

### Filtering-based SLAM

The EKF prediction of the mean state directly results from (2.6):

$$\hat{\mathbf{x}}_{t|t-1} = f(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{u}_t, \mathbf{v}_t) . \quad (2.9)$$

while the uncertainty is propagated as:

$$\Sigma_{xx,t|t-1} \leftarrow \mathbf{J}_x^{f(\cdot)} \Sigma_{xx,t-1|t-1} \mathbf{J}_x^{f(\cdot)T} + \mathbf{J}_v^{f(\cdot)} \Sigma_w \mathbf{J}_v^{f(\cdot)T} . \quad (2.10)$$

where  $\mathbf{J}_x^{f(\cdot)}$  and  $\mathbf{J}_v^{f(\cdot)}$  are respectively the Jacobians of  $f(\cdot)$  with respect to state  $\mathbf{x}$  and with respect to  $\mathbf{v}$ .

The EKF update step is then:

$$\begin{bmatrix} \hat{\mathbf{x}}_{t|t} \\ \hat{\mathbf{L}}_t \end{bmatrix} \leftarrow \begin{bmatrix} \hat{\mathbf{x}}_{t|t-1} \\ g(\hat{\mathbf{x}}_{t-1}, \mathbf{z}_t) \end{bmatrix} + \mathbf{K}_t [\mathbf{z}_t - h(\hat{\mathbf{x}}_{t|t-1}, \hat{\mathbf{L}}_{t-1})] \quad (2.11)$$

$$\Sigma_{s,t|t} \leftarrow \Sigma_{s,t|t-1} - \mathbf{K}_t \mathbf{S}_t^{-1} \mathbf{K}_t^T . \quad (2.12)$$

with,

$$\mathbf{K}_t = \Sigma_{t|t-1} \mathbf{J}_x^{h(\cdot)T} \mathbf{S}_t^{-1} \quad (2.13)$$

$$\mathbf{S}_t = \mathbf{J}_x^{h(\cdot)} \Sigma_{t|t-1} \mathbf{J}_x^{h(\cdot)T} + \Sigma_v . \quad (2.14)$$

where  $\mathbf{J}_x^{h(\cdot)}$  is the Jacobian of  $h(\cdot)$  with respect to the state  $\mathbf{x}_t$ .

Whereas EKF-based SLAM solutions are widely popular, a key issue of the method is its scalability. In (2.12),  $\mathbf{S}_t$  dimension grows with the number of observation variables hence its inversion becomes more costly and the multiplication has a quadratic cost.

In order to cope with this issue, EIF methods were proposed [10, 13]. Rather than using the covariance matrix, EIF methods use the sparsity of the covariance inverse, namely the *Information Matrix* noted  $\Lambda$ , to design efficient algorithms. These methods are related to the EKF methods by:

$$\Lambda = \Sigma^{-1}, \quad \boldsymbol{\eta} = \Lambda \hat{\mathbf{x}}. \quad (2.15)$$

where  $\boldsymbol{\eta}$  is the information vector.

Despite these improvements, another type of formulation has become the *de-facto* standard SLAM formulation nowadays. Relying on optimization, it is known as *Graph-based SLAM*.

### Graph-based SLAM

The SLAM problem may also be solved through non-linear sparse optimization. It is usually pictured graphically; the robot poses and landmarks locations are nodes in a graph, tied together by edges representing their relative relationships - a motion (2.6) or an observation (2.7). Such representation is depicted in Fig. 2.3.

**Factor Graph** The SLAM-graph is only constituted of two types of nodes: states connected to a small subset of other states through constraints. Such bipartite-graph representation is called a *factor graph*.

Without further derivation and recalling that the noises in (2.6) and (2.7) are Gaussian, the SLAM problem (2.8) can be written in a quadratic form,

$$\begin{aligned} \log(\mathbb{P}(X_T, L|Z_t, U_T)) &= \underbrace{\sum_t [\mathbf{x}_t - f(\mathbf{x}_{t-1}, \mathbf{u}_t)]^T \boldsymbol{\Omega}_f [\mathbf{x}_t - f(\mathbf{x}_{t-1}, \mathbf{u}_t)]}_{\text{motions}} + \\ &\quad \underbrace{\sum_t [\mathbf{z}_t - h(\mathbf{x}_t, L_t)]^T \boldsymbol{\Omega}_h [\mathbf{z}_t - h(\mathbf{x}_t, L_t)]}_{\text{observations}} + \text{const}. \end{aligned} \quad (2.16)$$

From the probabilities in (2.6–2.7) the following factors  $\Phi$  are derived:

$$\Phi_t = \mathbb{P}(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t) \propto \exp([\mathbf{x}_t - f(\mathbf{x}_{t-1}, \mathbf{u}_t)]^T \boldsymbol{\Omega}_w [\mathbf{x}_t - f(\mathbf{x}_{t-1}, \mathbf{u}_t)])^{-\frac{1}{2}} \quad (2.17)$$

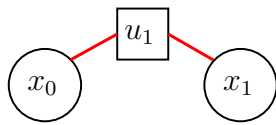
$$\Phi_n = \mathbb{P}(\mathbf{z}_t | \mathbf{x}_t, L_t) \propto \exp([\mathbf{z}_t - h(\mathbf{x}_t, L_t)]^T \boldsymbol{\Omega}_v [\mathbf{z}_t - h(\mathbf{x}_t, L_t)])^{-\frac{1}{2}}. \quad (2.18)$$

where  $\boldsymbol{\Omega}_w = \Sigma_w^{-1}$  and  $\boldsymbol{\Omega}_v = \Sigma_v^{-1}$  are the information matrices of the observed data. (2.17–2.18) lead to a unique form of the error formulation:

$$e_k(\mathbf{x}_{t-1}, \mathbf{x}_t) = f(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t \quad (2.19)$$

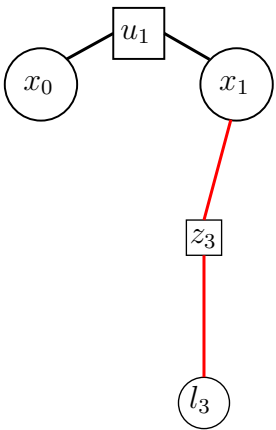
$$e_k(\mathbf{x}_t, \mathbf{l}_n) = h(\mathbf{x}_t, \mathbf{l}_n) - \mathbf{z}_n \quad (2.20)$$

$$\Phi_k = \exp(e_k^T \boldsymbol{\Omega}_k e_k)^{-\frac{1}{2}}. \quad (2.21)$$



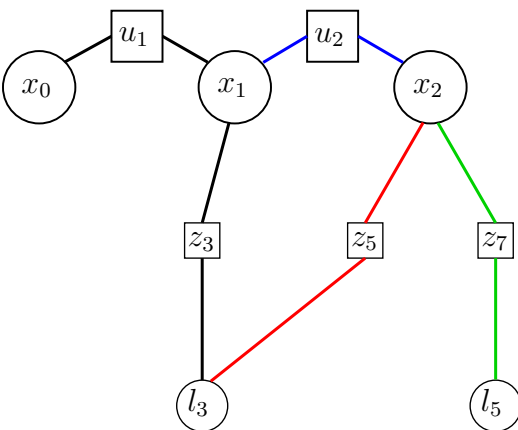
	$x_0$	$x_1$	$x_2$	$l_3$	$l_5$
$x_0$					
$x_1$					
$x_2$					
$l_3$					
$l_5$					

(a) Robot moves from  $\mathbf{x}_0$  to  $\mathbf{x}_1$  under the control command  $\mathbf{u}_1$ .



	$x_0$	$x_1$	$x_2$	$l_3$	$l_5$
$x_0$					
$x_1$					
$x_2$					
$l_3$					
$l_5$					

(b) At time  $t_{+1}$  the robot senses landmark  $l_3$



	$x_0$	$x_1$	$x_2$	$l_3$	$l_5$
$x_0$					
$x_1$					
$x_2$					
$l_3$					
$l_5$					

(c) Robot moves from  $\mathbf{x}_1$  to  $\mathbf{x}_2$  under the control command  $\mathbf{u}_2$ . At  $t_{+2}$  the robot senses landmark  $l_3$  again and  $l_5$ .

Figure 2.2: Iterative construction of a factor graph and its support sparse Hessian matrix.

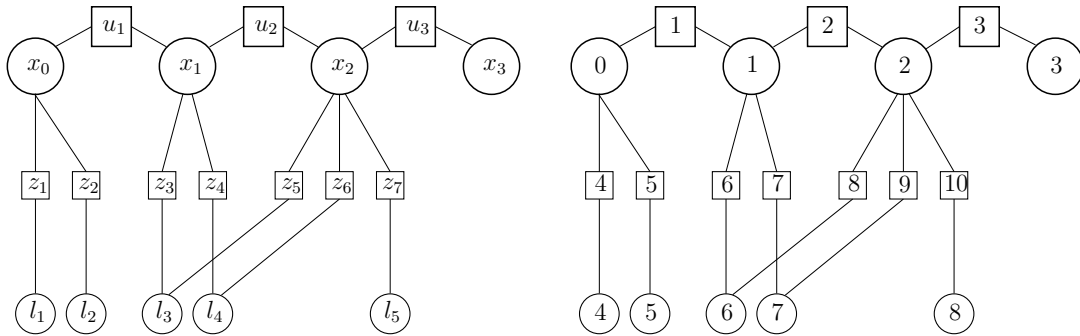


Figure 2.3: A factor graph representation of Fig. 2.1. *Left*: Nodes representing known data have been replaced by factors (squares) which depends on the unknown variables (or states) (circles). *Right*: The same graph. Unknown states are labeled with a single index  $i \in [0, 7]$  and factors are labeled with a single index  $k \in [1, 10]$ .

Finally, the SLAM problem is reduced to solving the equation:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{k=1}^K e_k(\mathbf{x}_i, \mathbf{x}_j)^T \boldsymbol{\Omega}_k e_k(\mathbf{x}_i, \mathbf{x}_j) . \quad (2.22)$$

where the summed terms are of the form of the Mahalanobis distance.

**Solving the SLAM problem** As mentioned in Section 2.1.1, the graph-SLAM problem can be solved by means of iterative non-linear optimization. Rather than detailing the optimization algorithms, which is beyond the scope of this thesis, here we provide some references shall the reader require further insights. Such optimization frameworks usually implement a Gauss-Newton or a Levenberg-Marquardt least squares minimization, using either QR [19] or Cholesky [20, 21] decomposition of the sparse Jacobian matrices of the problem. Important speed-ups have been obtained with incremental operation of these algorithms, where newly acquired information is added directly on the already-factored problem [22, 23].

## 2.1.2 Odometry

A key issue in robotics navigation is the ability of the robot to estimate its current pose in an unknown environment, the so called self-localization. The odometry module, generically expressed by (2.6), aims at providing an estimate of the robot state at a given time with respect to the previous one. It may estimate the robot displacement from an input control command and/or from sensors readings.

Alongside the motion integration, one can also integrate the associated uncertainty. It is done by linearizing the motion model ( $f(\cdot)$  in (2.6)) and integrating a Gaussian estimate of the state  $\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \boldsymbol{\Sigma})$  as follows:

$$\hat{\mathbf{x}} \leftarrow f(\hat{\mathbf{x}}, \mathbf{u}, 0) \quad (2.23)$$

$$\boldsymbol{\Sigma}_{\mathbf{x}} \leftarrow \mathbf{J}_{\mathbf{x}}^{f(\cdot)} \boldsymbol{\Sigma}_{\mathbf{x}} \mathbf{J}_{\mathbf{x}}^{f(\cdot)T} + \mathbf{J}_{\mathbf{v}}^{f(\cdot)} \boldsymbol{\Sigma}_{\mathbf{v}} \mathbf{J}_{\mathbf{v}}^{f(\cdot)T} . \quad (2.24)$$

Where  $\mathbf{J}_{\mathbf{x}}^{f(\cdot)}$  and  $\mathbf{J}_{\mathbf{v}}^{f(\cdot)}$  are, respectively, the Jacobians of  $f(\cdot)$  with respect to  $\mathbf{x}$  and the perturbation  $\mathbf{v}$ .  $\boldsymbol{\Sigma}_{\mathbf{x}}$  is the covariance matrix associated to  $\mathbf{x}$ .

The main platform used during this thesis is a TIAGo robot equipped with a differential drive base and a 2D LRF. Hence, the differential drive odometry model is presented here, together with model agnostic means to compute odometry from a LRF.

### Differential Drive Odometry

The differential drive model for a ground robot consists of two actuated wheels on a single axle, one on each side of its base. The robot's frame is defined at the center of the axle, with the X-axis looking forward, *i.e.*, perpendicular to the wheels' axle. The model is parameterized by the wheels radii  $(r_l, r_r)$  and the wheel separation  $d$ . To each of these values is associated a calibration parameter,  $\mathbf{c} = (c_l, c_r, c_d)$ , obtaining the calibrated parameters  $(c_l r_l, c_r r_r, c_d d)$ .

The wheels motion are measured by wheel encoders reporting noisy incremental wheel angles  $\mathbf{u} = \delta\boldsymbol{\psi} = (\delta\psi_l, \delta\psi_r)$  every time step  $\delta t$ .

Assuming constant wheel velocities between times  $t_j$  and  $t_k$ , the motion of the vehicle can be described by a small arc of length  $\delta l_k$ , angle  $\delta\theta_k$ , and radius  $\delta l_k / \delta\theta_k$ ,

$$\begin{aligned}\delta l_k &= \frac{1}{2}(c_r r_r \delta\psi_{r,k} + c_l r_l \delta\psi_{l,k}) \\ \delta\theta_k &= \frac{1}{c_d d}(c_r r_r \delta\psi_{r,k} - c_l r_l \delta\psi_{l,k}) .\end{aligned}\tag{2.25}$$

This arc can be expressed in the tangent or velocity space of SE(2), *i.e.*, the Lie algebra  $\mathfrak{se}(2)$ , with

$$f(\mathbf{u}_k, \mathbf{c}) = [\delta l_k, \delta s_k, \delta\theta_k]^\top \in \mathfrak{se}(2) ,\tag{2.26}$$

where  $\delta s_k$  is a zero-mean perturbation corresponding to lateral wheel slippage. The associated Jacobians read,

$$\mathbf{J}_{\delta\boldsymbol{\psi}_k}^{f(\cdot)} = \begin{bmatrix} \frac{c_l r_l}{2} & \frac{c_r r_r}{2} \\ 0 & 0 \\ -\frac{c_l r_l}{c_d d} & \frac{c_r r_r}{c_d d} \end{bmatrix}\tag{2.27a}$$

$$\mathbf{J}_{\mathbf{c}}^{f(\cdot)} = \begin{bmatrix} \frac{\delta\psi_{l,k} r_l}{2} & \frac{\delta\psi_{r,k} r_r}{2} & 0 \\ 0 & 0 & 0 \\ -\frac{\delta\psi_{l,k} r_l}{c_d d} & \frac{\delta\psi_{r,k} r_r}{c_d d} & -\frac{\delta\theta_k}{c_d} \end{bmatrix} .\tag{2.27b}$$

The uncertainty covariance  $\boldsymbol{\Sigma}_f$  used for uncertainty integration in (2.24) is obtained from the uncertainty of the wheel angle measurements,

$$\boldsymbol{\Sigma}_f = \mathbf{J}_{f(\cdot)}^{\mathbf{u}} \boldsymbol{\Sigma}_{\boldsymbol{\psi}} \mathbf{J}_{f(\cdot)}^{\mathbf{u}T} \in \mathbb{R}^{2 \times 2} .\tag{2.28}$$

with  $\boldsymbol{\Sigma}_{\boldsymbol{\psi}}$  the wheel measurement covariance:

$$\mathbf{Q}_{\boldsymbol{\psi}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\psi_l}^2 + \alpha^2 & 0 \\ 0 & \boldsymbol{\Sigma}_{\psi_r}^2 + \alpha^2 \end{bmatrix} \in \mathbb{R}^{2 \times 2}\tag{2.29}$$

$$\boldsymbol{\Sigma}_{\psi_l}^2 = k_l |\delta\psi_l| \quad \boldsymbol{\Sigma}_{\psi_r}^2 = k_r |\delta\psi_r| \quad \alpha = \frac{1}{2}(\mu_l + \mu_r) .$$

where  $k_r$  and  $k_l$  are wheels intrinsic parameters,  $\alpha$  acts as an offset equal to half the wheels encoders resolution  $\mu_l$  and  $\mu_r$  [24].

## LRF-based Odometry

The odometry estimation can also be addressed by tracking a sensor pose directly from its readings. Laser-scan based odometry is the process of estimating the robot trajectory from consecutive LRF readings. Given two consecutive readings, the odometry algorithm is two-fold; first a data association (scan-to-scan(s), scan-to-map) is established, then the relative transform that best aligns the associated data is estimated. Doing so over time allows one to compute pair-wise relative transforms which, once integrated, provide an estimate of the robot trajectory.

In the literature this problem has been addressed in many ways, such as an adaptation of the *Iterative Closest Point* (ICP) algorithm [25] to the problem of motion estimation [26]. Other methods constitute direct variations of the original ICP formulation; [27] uses a custom metric in place of the Euclidean distance to lessen the effect of sensor rotation compared to translation on the distance of associated readings. Other variants include *Iterative Closest Line* (ICL) proposed in [28] whose error function relies on a point-to-line distance rather than point-to-point as in the classical ICP. Other LRF-based odometry estimation algorithms include feature-based data association [29] which extracts features from the readings. Doing so, they discard the need for an iterative process inherent of ICP-based methods. Rather than considering the range-readings in the Cartesian plane, [30, 31] perform the data association in polar coordinates. A more recent work aligns consecutive scans by means of correlation [32]. The scans are projected on 2D occupancy grids, which are then correlated. Recently, an Optical-Flow-based formulation has been successfully proposed for 2D scan-to-scan alignment [33], later improved by aligning one scan to many [34].

### 2.1.3 Loop-closure

Loop closure detection is an essential module of any SLAM system. It is the process of (re)identifying a place from a generated corpus of places visited in the past. Unlike the odometry module mentioned in Section 2.1.2 which estimates short term robot poses, the loop closure module estimates the robot pose with regards to the global map therefore comparing the current pose against the entire history of poses.

Closing such loops reduces the uncertainty in the estimated map, accumulated during open loop mapping. This is exemplified in Fig. 2.4. The robot trajectory estimated (highlighted with dashes) drifts over time. When revisiting a place, the module identifies it as being part of its corpus and estimates the relative transform from the current pose to the identified one (the red segment in Fig. 2.4). Doing so it creates and adds a constraint to the problem allowing to correct the accumulated drift.

This is closely related to the place recognition problem up to the difference that in the case of place recognition, one only seeks for a topological match hence do not requires to compute the relative transform between the two matched places.

Loop closure detection has been tackled with geometric methods [35], correlation methods [36], and with appearance-based methods. Appearance can be considered either globally [37–40] or as a set of local distinctive features [12, 41, 42] possibly extracted from different sensor modalities [43]. After the initial work of the computer vision community on the use of the *Bag-of-Words* (BoW) method for object recognition [44–46], the SLAM community found in BoW an efficient manner to

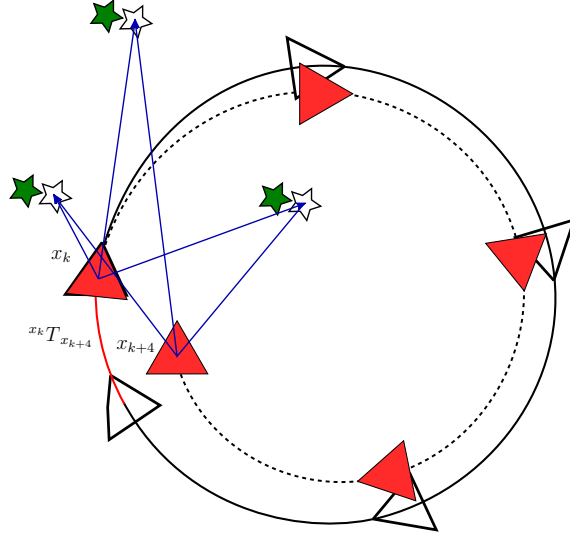


Figure 2.4: Illustration of a loop closure.

query large corpus of places visited by a robot while mapping [47, 48], hence its amenity for the solution of the loop closure problem. Recent state-of-the-art *Visual-Simultaneous Localization and Mapping* (V-SLAM) algorithms have relied on BoW for their loop closure and re-localization modules. ORB-SLAM [8] for instance uses DBoW2 [49], whereas LSD-SLAM [7] relies on FAB-MAP [50]. It is worth noting that after ORB-SLAM and LSD-SLAM, most modern V-SLAM algorithms rely on the BoW scheme for their loop-closure module. The reader can find a recent and extensive survey of visual place recognition in [51].

### Bag-of-Words

In the BoW framework, the objective is to find a snapshot of a sensor readings in a corpus of snapshots. To that end, it includes two distinct elements. First, a *vocabulary*,  $W = \{w_1, \dots, w_k\}$ , composed of cluster centers or *words*,  $w_k$ , representing the feature's descriptor space. The vocabulary of words is built offline from a data set unrelated to the later use by employing a hierarchical k-means [46, 52]. The second element consists of a *database* composed of *documents*,  $D = \{d_1, \dots, d_N\}$ , where each document  $d$  is the BoW associated to a sensor readings snapshot at a known pose of the robot in the current map. That is, the set of local features in the vocabulary detected in a given sensor readings and their local coordinates.

The database keeps a record of each word occurrence in every document by means of two frequency scores. The *Term Frequency* (TF) relates to how frequent a given word is within a document, while the *Inverse Document Frequency* (IDF) refers to how frequent is a given word in the whole database. Given a word  $w_i$  in document  $d_j$ , these frequencies are computed as follows:

$$tf_{ij} = \frac{n_{ij}}{\sum_i n_{ij}}, \quad (2.30)$$

$$idf_i = \log \left( \frac{|D|}{\sum_j |n_{ij} > 0|} \right), \quad (2.31)$$

where  $n_{ij}$  is the occurrence of the word  $i$  in document  $j$ ,  $|D|$  is the size of the database and  $|n_{ij} > 0|$  evaluates to 1 if  $w_i$  occurs in  $d_j$  and 0 otherwise. The weighting of every word  $w_i$  in each document  $d_j$  is given by its *Term Frequency - Inverse Document Frequency* (TF-IDF) score,

$$x_{ij} = tf_{ij} \cdot idf_i. \quad (2.32)$$

A document  $d_j$  is characterized by its *signature*, a vector containing its TF-IDF weights,  $sig_j = [x_{j1}, x_{j2}, \dots, x_{jk}]^T$ . The comparison of two documents  $d_j$  and  $d_k$  is performed by computing the cosine similarity of their signatures:

$$sim_{jk} = \frac{sig_j^T sig_k}{\|sig_j\| \|sig_k\|}. \quad (2.33)$$

Given a new sensor reading (a query), its feature descriptors are extracted, quantized into words, and its signature compared to those of every document in the database; the  $N$  most similar documents are returned by the BoW scheme.

Finally, a geometrical consistency check is performed to assert which, if any, of the returned documents is a good match to the query sensor reading. In the case of visual place recognition, the consistency check can be for instance the estimation of the *Essential matrix* [53], a *trifocal tensor* [53] or a *Perspective-n-Point* (PnP) projection [54].

## 2.1.4 Planning

Motion planning is one of the most important and most studied aspects for any robotic system as it enables interactions with the real world. Operating in a real environment requires safe and efficient methods that find plans for the robot to reach a desired configuration. While there exist many approaches solving point-to-point type planning, often at the cost of additional requirements, they generally lack in speed, complexity or guarantees.

Initially designed for obstacle avoidance, the method of *Potential Fields* (PF) proved to be very valuable for trajectory planning [55]. Two artificial potential fields are superimposed, one repulses the robot from obstacles while the second attracts it toward the desired goal. While being simple and efficient, their main drawbacks are the presence of local-minima in the field in which the robot gets stuck. The method typically fails to find a path between close obstacles.

*Rapidly-exploring Random Tree* (RRT)-based methods were first presented in [56, 57]. They originally aimed at solving general nonholonomic and kinodynamic planning problems which earlier randomized approaches struggled with - *e.g. Probabilistic Road Maps* (PRM) [58]. By subsequently growing a tree randomly from the initial configuration, RRT methods offer a good exploration of the search space with relative ease. Later improvements include *Rapidly-exploring Random Tree Connect* (RRTConnect) [57] which extends previous work by constructing trees from both ends - initial and final configurations - until they connect to each other. While these methods are able to generate trajectories in complex, high-dimensional spaces, they are by nature non-guided therefore non-optimal, and have to be instrumented to provide time guarantees. The quality of the generated trajectories is often poor in that they are not optimal and often redundant and jerky. Tackling the non-optimality defect of sampling-based methods, asymptotically optimal planners were



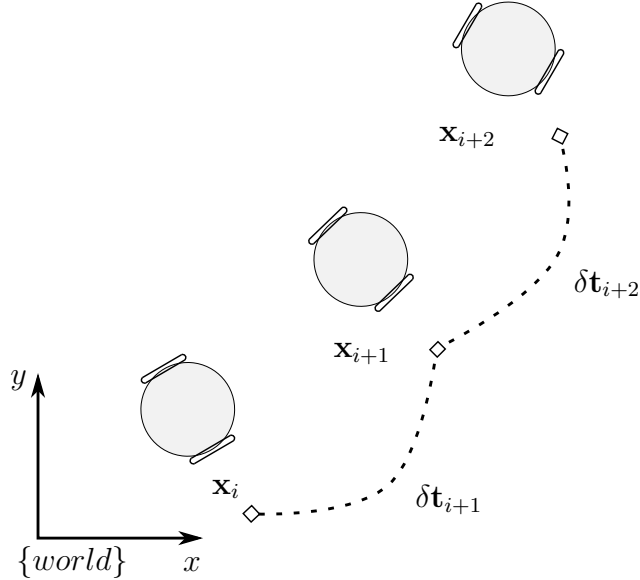


Figure 2.5: The TEB is a sequence of robot poses forming a trajectory. Consecutive poses are tied to one another by a time interval.

proposed in [59]. These planners, *Rapidly-exploring Random Tree\** (RRT\*) and *Probabilistic Road Maps\** (PRMS), were shown to tend to an optimal solution given a cost metric. Further improvements of the RRT\* algorithms include kinematics consideration for simple models such as mobile-bases [60].

### Timed-Elastic Band

Originated in [61], the *Timed Elastic Band* (TEB) is an increasingly popular method for estimating trajectories [62, 63]. The band refers to a sequence of  $n+1$  robot poses  $\mathbf{x}_i \in \text{SE}(m)$  linking together an initial and a final configuration:

$$\mathbf{Q} = \{\mathbf{x}_i\}_{i=0\dots n} . \quad (2.34)$$

With  $\mathbf{x}_0$  fixed at the origin. Consecutive poses are tied to one another by  $n$  time intervals  $\delta t$ ,

$$\Delta \mathbf{T} = \{\delta t_i\}_{i=1\dots n} , \quad (2.35)$$

with  $\delta t_i$  denoting the time interval required for the robot to move from a pose  $\mathbf{x}_{i-1}$  to  $\mathbf{x}_i$  along the trajectory. The TEB is illustrated in Fig. 2.5.

Defining the pairs,

$$\mathbf{P} = \{(\mathbf{x}_i, \delta t_i)\}_{i=1\dots n} , \quad (2.36)$$

the TEB is formulated as a multi-objective optimization problem:

$$\mathbf{P}^* = \arg \min_{\mathbf{P}} \sum_{k,i} w_k c_{ki}(\mathbf{Q}_i, \Delta \mathbf{T}_i) \quad (2.37)$$

which can be solved by means of a least-squares non-linear solver. The terms  $w_k$  are weight factors used to balance the different cost contributions  $c_{ki}(\mathbf{Q}_i, \Delta \mathbf{T}_i)$ . These

costs depend on  $\mathbf{Q}_i$  and  $\Delta\mathbf{T}_i$ , which are subsets of respectively  $\mathbf{Q}$  and  $\Delta\mathbf{T}$  in the neighborhood of  $\delta t_i$ . The cost functions are built simply with  $c_k = \mathbf{e}_k^\top \mathbf{e}_k$ , with  $\mathbf{e}_k$  an error measure. The index  $k$  indicates the nature of each objective error during the interval  $\delta t_i$ .

## 2.2 Lie Theory

In SLAM, as for many other robotics problems, the robot state includes a rotational part to represent the robot orientation in space. This requires particular care as all of the mathematical tools to solve the SLAM problem rely on linear algebra (see Section 2.1.1). But rotations are not part of a *vector space* while maintaining some of its properties; the set of rotations belong to a *non-commutative* group. For this reason, over the last years, the robotics community has put a great deal of effort to formulate estimation problems properly. This is motivated by an increasing demand for precision, consistency and stability of solutions. Indeed, a proper modeling of the states and measurements, the functions relating them, and their uncertainties, is crucial to achieve high quality mapping. This has led to problem formulations involving what has been known as *manifolds*, which in this context are no less than the smooth topologic surfaces of the Lie groups where the state representations evolve.

Sophus Lie (1842-1899) was a Norwegian mathematician. He largely created the theory of continuous symmetry and applied it to the study of geometry and differential equations.

While the Lie theory literature is well-rounded – shall only one be cited, be it [64] –, a first introduction to it is often tedious. Indeed the vast majority of related documents has been written by mathematicians and physicists for audiences of the same crowd; rigorous descriptions and demonstrations are in order, treating at large many aspects that are not relevant at the moment for the particular tasks encountered in state estimation. However, over the years, many attempts to ease its understanding were made - as their titles sometime suggest -, *Very Basic Lie Theory* [65], *Basic Lie Theory* [66], *Naive Lie Theory* [67], *Lie Groups for 2D and 3D Transformations* [68], *State Estimation for Robotics* [69].

Despite these efforts, some aspects may still appear somewhat confusing to the reader of recent robotics publications as different fields employ a similar vocabulary for different considerations.

This section does not pretend at offering yet another, better, introduction to Lie Theory but rather to explicit the few concepts necessary throughout the remainder of the thesis. Following the description of necessary general concepts, the details for each of the most common group encountered in robotics are given in a ‘take-away’ form in Appendix C.

The interested reader is encouraged to read *A micro Lie theory for state estimation in robotics* [70] for a brief, clearly explained, (yet another) attempts at introducing the Lie Theory for the roboticist.

### 2.2.1 Lie groups

A Lie group  $\mathcal{G}$  encompasses a subset of elements of an *algebraic group* - defined by maps - that lie on the surface of a *smooth manifold* - also known as *differentiable*. Lie groups thus possess properties of an *algebraic group* but also some from *differential geometry*.

The manifold is a geometrical object, a topological smooth (hyper)-surface, with no edges or spikes, embedded in a higher dimensional space. Its smoothness implies the existence of a unique tangent space at each point that resembles a linear space, allowing one to do calculus. The reader should be able to visualize the manifold concept represented by a blue sphere in Fig. 2.6.

In a Lie group, the manifold *looks* the same at every point (similarly to the surface of a sphere), and therefore all tangent spaces at any point are alike. However there exists a special tangent space at the special element ‘Identity’, which is called the Lie algebra of the Lie group (red plane in Fig. 2.6). Lie groups join the local properties of smooth manifolds, allowing us to do calculus, with the global properties of groups, enabling nonlinear composition of distant objects.

The Lie groups are defined by the following axioms,

$$\text{Closure under } \circ \quad : \quad \chi \circ \gamma \in \mathcal{G} \quad (2.38)$$

$$\text{Identity } \mathcal{E} \quad : \quad \mathcal{E} \circ \chi = \chi \circ \mathcal{E} = \chi \quad (2.39)$$

$$\text{Inverse } \chi^{-1} \quad : \quad \chi^{-1} \circ \chi = \chi \circ \chi^{-1} = \mathcal{E} \quad (2.40)$$

$$\text{Associativity} \quad : \quad (\chi \circ \gamma) \circ \zeta = \chi \circ (\gamma \circ \zeta) , \quad (2.41)$$

for  $\chi, \gamma, \zeta$  and  $\mathcal{E} \in \mathcal{G}$ . They read,

- The composition of elements of the manifold is continuous, the resulting element remains on the manifold.
- There exists a special element ‘Identity’ so that the composition of a manifold element with the ‘Identity’ is the object itself.
- Inversion of a manifold element is continuous, the resulting element remains on the manifold. The composition of an element with its inverse results in the ‘Identity’.
- The order in which the composition operator is applied does not matter.

Hereafter, for simplicity and as it has been common in robotics works, Lie groups are referred to as just ‘manifolds’, denoted  $\mathcal{M}$ .

### 2.2.2 The tangent spaces and the Lie algebra

As aforementioned, the tangent space at the identity, which is noted  $\mathcal{T}_{\mathcal{M}}(\mathcal{E})$ , is called the Lie algebra of  $\mathcal{M}$  and is noted  $\mathfrak{m}$ ,

$$\mathfrak{m} \triangleq \mathcal{T}_{\mathcal{M}}(\mathcal{E}) . \quad (2.42)$$

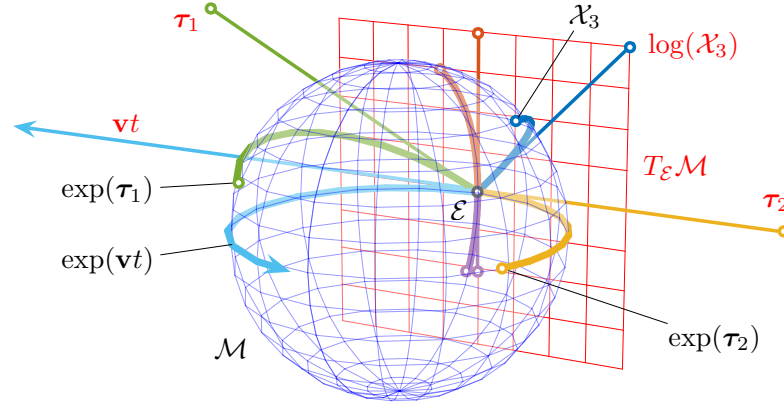


Figure 2.6: Representation of the relation between the Lie group and the Lie algebra. The Lie algebra  $\mathcal{T}_{\mathcal{M}}(\mathcal{E})$  (red plane) is the tangent space to the Lie group’s manifold  $\mathcal{M}$  (blue sphere) at the identity  $\mathcal{E}$ . Through the exponential map, each straight path  $\mathbf{v}t$  through the origin on the Lie algebra produces a path  $\exp(\mathbf{v}t)$  around the manifold which runs along the respective geodesic. Conversely, each element of the group has an equivalent in the Lie algebra. This relation is so profound that (nearly) all operations in the group, which is curved and nonlinear, have an exact equivalent in the Lie algebra, which is a linear vector space. Note that the sphere in  $\mathbb{R}^3$  is not a Lie group, it is however a convenient figural representation that can be drawn on paper.

It is a *vector space* equipped with a binary operation  $[\cdot, \cdot] : \mathfrak{m} \times \mathfrak{m} \rightarrow \mathfrak{m}$  called the *Lie bracket*, such that,

$$\text{Bilinearity} \quad : \quad [a\boldsymbol{\tau} + b\boldsymbol{\rho}, \boldsymbol{\sigma}] = a[\boldsymbol{\tau}, \boldsymbol{\sigma}] + b[\boldsymbol{\rho}, \boldsymbol{\sigma}] \quad (2.43a)$$

$$[\boldsymbol{\sigma}, a\boldsymbol{\tau} + b\boldsymbol{\rho}] = a[\boldsymbol{\sigma}, \boldsymbol{\tau}] + b[\boldsymbol{\sigma}, \boldsymbol{\rho}]$$

$$\text{Alternativity} \quad : \quad [\boldsymbol{\tau}, \boldsymbol{\tau}] = 0 \quad (2.43b)$$

$$\text{Jacobi identity} \quad : \quad [\boldsymbol{\tau}, [\boldsymbol{\rho}, \boldsymbol{\sigma}]] + [\boldsymbol{\sigma}, [\boldsymbol{\tau}, \boldsymbol{\rho}]] + [\boldsymbol{\rho}, [\boldsymbol{\sigma}, \boldsymbol{\tau}]] = 0 \quad (2.43c)$$

$$\text{Anticommutativity} \quad : \quad [\boldsymbol{\tau}, \boldsymbol{\rho}] = -[\boldsymbol{\rho}, \boldsymbol{\tau}] , \quad (2.43d)$$

for  $a, b \in \mathbb{R}$  and  $\boldsymbol{\tau}, \boldsymbol{\rho}, \boldsymbol{\sigma} \in \mathfrak{m}$ .

Its structure can be found computing the derivative with respect to time of the group axiom (2.40). Given  $\boldsymbol{\chi}(t)$  a point moving on Lie group’s manifold  $\mathcal{M}$ , its velocity  $\dot{\boldsymbol{\chi}} = \partial\boldsymbol{\chi}/\partial t$  belongs to the space tangent to  $\mathcal{M}$  at  $\boldsymbol{\chi}$ ,  $\mathcal{T}_{\mathcal{M}}(\boldsymbol{\chi})$ . Considering (2.40) for multiplicative groups,  $\boldsymbol{\chi}(t) \times \boldsymbol{\chi}(t)^{-1} = \mathcal{E}$ , the time derivation reads,

$$\dot{\boldsymbol{\chi}}(t) \times \boldsymbol{\chi}(t)^{-1} + \boldsymbol{\chi}(t) \times \dot{\boldsymbol{\chi}}(t)^{-1} = 0 \quad (2.44)$$

$$\dot{\boldsymbol{\chi}}(t) \times \boldsymbol{\chi}(t)^{-1} = -\boldsymbol{\chi}(t) \times \dot{\boldsymbol{\chi}}(t)^{-1} . \quad (2.45)$$

While (2.45) does not provide any insights on an eventual abstract Lie group structure, let us consider for a moment  $\boldsymbol{\chi} \in \text{SO}(m)$  the Special Orthogonal group of dimension  $m$  (or rotation group, since for  $m = 2$  and  $m = 3$  elements are the usual rotation). Then (2.45) can be further developed,

$$\dot{\boldsymbol{\chi}}(t)\boldsymbol{\chi}(t)^{-1} = -\boldsymbol{\chi}(t)\dot{\boldsymbol{\chi}}(t)^{-1} \quad (2.46)$$

$$\dot{\boldsymbol{\chi}}(t)\boldsymbol{\chi}(t)^T = -\boldsymbol{\chi}(t)\dot{\boldsymbol{\chi}}(t)^T \quad (2.47)$$

$$= -(\dot{\boldsymbol{\chi}}(t)\boldsymbol{\chi}(t)^T)^T , \quad (2.48)$$

where  $\mathbf{R}^{-1} = \mathbf{R}^T$  and  $(\mathbf{R}_a \mathbf{R}_b)^T = \mathbf{R}_b^T \mathbf{R}_a^T$  are properties of rotation matrices used in this development. The resulting form  $\mathbf{A} = -\mathbf{A}^T$  clearly shows that the matrix  $\mathbf{A}$  is skew-symmetric. Therefore, the vector space  $\mathfrak{so}(3)$  takes the shape of skew-symmetric matrices. While this results seems a little counter-intuitive for a vector-space, later on is demonstrated that elements of the Lie algebra can be expressed as linear combination of some base elements.

Let us define

$$\dot{\chi}(t)\chi(t)^{-1} = \boldsymbol{\tau}(t) \leftrightarrow \dot{\chi}(t) = \boldsymbol{\tau}(t)\chi(t) . \quad (2.49)$$

Considering the identity vicinity, that is  $\chi(t_0) = \mathcal{E}$ , one now has  $\dot{\chi}(t) = \boldsymbol{\tau}(t)$ . Computing the Taylor series of  $\chi(t)$  for  $t = t_0$  results in the first order approximation,

$$\chi(t_0 + dt) \approx \chi(t_0) + \dot{\chi}(t_0)dt \quad (2.50)$$

$$\approx \mathcal{E} + \boldsymbol{\tau}(t_0)dt , \quad (2.51)$$

meaning that infinitesimals around the identity are, locally, only dependent on the tangent space.

The Lie algebra *is* a vector space and as such its elements can be *identified* with vectors in  $\mathbb{R}^m$  whose dimension  $m$  is the number of degrees of freedom of  $\mathcal{M}$ .

### The ‘vee’ and ‘hat’ operators

From now on, elements of the algebra  $\boldsymbol{\tau}^\wedge \in \mathfrak{m}$  are differentiated from elements of the real vector space  $\boldsymbol{\tau} \in \mathbb{R}^m$  with a ‘hat’ decorator. This is to emphasize that  $\boldsymbol{\tau}$  and  $\boldsymbol{\tau}^\wedge$  are the same element, expressed in two different isomorphic spaces.

As shown in (2.48), elements of the Lie algebra have non-trivial structures (see also Table C.2), for this reason it is often preferred to express elements of  $\mathfrak{m}$  in the isomorphic space  $\mathbb{R}^m$ . This is done by using a linear map – or isomorphism – commonly called ‘vee’,

$$\boldsymbol{\tau}^\wedge \in \mathfrak{m} \rightarrow \boldsymbol{\tau} \in \mathbb{R}^m \quad ; \quad \boldsymbol{\tau}^\wedge \rightarrow (\boldsymbol{\tau}^\wedge)^\vee = \boldsymbol{\tau} = \sum_{i=1}^n \tau_i \mathbf{e}_i , \quad (2.52)$$

where  $\mathbf{e}_i$  are the vectors of the base of  $\mathbb{R}^m$ .

The inverse mapping, commonly called *hat*, reads,

$$\boldsymbol{\tau} \in \mathbb{R}^m \rightarrow \boldsymbol{\tau}^\wedge \in \mathfrak{m} \quad ; \quad \boldsymbol{\tau} \rightarrow \boldsymbol{\tau}^\wedge = \sum_{i=1}^n \tau_i E_i , \quad (2.53)$$

where  $E_i$  are the base elements of the vector space  $\mathfrak{m}$  - also known as generators - obtained by differentiating  $\chi$  around the origin in the  $i$ -th direction. Note that  $\mathbf{e}_i^\wedge = E_i$ .

This linear relation comes handy as vectors in  $\mathbb{R}^m$  are easier to manipulate, allowing the use of the more familiar linear algebra using matrix operators. For the tasks at hand, elements in  $\mathbb{R}^m$  are preferred over those in  $\mathfrak{m}$ , to the point that most of the operators and objects defined subsequently are in  $\mathbb{R}^m$ . Thus  $\mathfrak{m} \cong \mathbb{R}^m$  and  $\boldsymbol{\tau}^\wedge \cong \boldsymbol{\tau}$ .

To the vector space  $\mathfrak{m}$  is associated an inner product  $\langle \boldsymbol{\tau}, \boldsymbol{\tau} \rangle$  so that,

$$\langle \boldsymbol{\tau}, \boldsymbol{\tau} \rangle = \boldsymbol{\tau}^T \cdot \mathbf{W} \cdot \boldsymbol{\tau} , \quad (2.54)$$

with  $\mathbf{W}$ , the matrix of the inner product  $\langle \cdot, \cdot \rangle$  relative to the space basis  $E$ , defined as,

$$\mathbf{W} = \begin{bmatrix} \langle E_1, E_1 \rangle & \langle E_1, E_2 \rangle & \dots & \langle E_1, E_i \rangle \\ \langle E_2, E_1 \rangle & \langle E_2, E_2 \rangle & \dots & \langle E_2, E_i \rangle \\ \dots & \dots & \ddots & \vdots \\ \langle E_i, E_1 \rangle & \langle E_i, E_2 \rangle & \dots & \langle E_i, E_i \rangle \end{bmatrix}, \quad (2.55)$$

with  $\langle \mathbf{A}, \mathbf{B} \rangle := \text{trace}(\mathbf{A}\mathbf{B}^T)$ .

### 2.2.3 The exponential map

The exponential map  $\exp(\cdot)$  allows one to exactly transfer elements of the algebra to the group. Intuitively, the exponential map wraps the tangent element from the tangent space onto the manifold following a geodesic (similarly to wrapping a string around a ball). The inverse operation is the logarithm map  $\log(\cdot)$  (similarly to unwrapping the string while holding it straight).

$$\exp : \quad \boldsymbol{\tau}^\wedge \in \mathfrak{m} \rightarrow \boldsymbol{\chi} \in \mathcal{M} \quad ; \quad \boldsymbol{\chi} = \exp(\boldsymbol{\tau}^\wedge) \quad (2.56)$$

$$\log : \quad \boldsymbol{\chi} \in \mathcal{M} \rightarrow \boldsymbol{\tau}^\wedge \in \mathfrak{m} \quad ; \quad \boldsymbol{\tau}^\wedge = \log(\boldsymbol{\chi}) . \quad (2.57)$$

From the above discussion one knows,  $\dot{\boldsymbol{\chi}}(t) = \boldsymbol{\tau}^\wedge \boldsymbol{\chi}(t)$  and  $\boldsymbol{\chi}(t_0) = \mathcal{E}$ . This is a linear *Ordinary Differential Equation* (ODE) with initial state at  $\mathcal{E}$  whose solution is given by  $\boldsymbol{\chi}(t) = \exp(\boldsymbol{\tau}^\wedge t)$ . For groups commonly encountered in robotics, closed form of the exponential are obtained by unrolling the absolutely convergent Taylor series,

$$\exp(\boldsymbol{\tau}^\wedge) = \mathcal{E} + \boldsymbol{\tau}^\wedge + \frac{1}{2!} \boldsymbol{\tau}^{\wedge 2} + \frac{1}{3!} \boldsymbol{\tau}^{\wedge 3} + \dots = \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\tau}^\wedge)^n, \quad (2.58)$$

and benefiting from the algebraic properties of the powers of  $\boldsymbol{\tau}^\wedge$ .

Conversely, the logarithm reads,

$$\log(\boldsymbol{\chi}) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} (\boldsymbol{\chi} - \mathcal{E})^n . \quad (2.59)$$

### 2.2.4 The capitalized Exponential map

The capitalized Exp and Log maps are handy shortcuts to map elements  $\boldsymbol{\tau} \in \mathbb{R}^m$  directly to  $\boldsymbol{\chi} \in \mathcal{M}$ . They respectively combine the ‘hat’ together with the ‘exp’ operations and the ‘log’ together with ‘vee’,

$$\text{Exp} : \quad \boldsymbol{\tau} \in \mathbb{R}^m \rightarrow \boldsymbol{\chi} \in \mathcal{M} \quad ; \quad \boldsymbol{\chi} = \text{Exp}(\boldsymbol{\tau}) = \exp(\boldsymbol{\tau}^\wedge) \quad (2.60)$$

$$\text{Log} : \quad \boldsymbol{\chi} \in \mathcal{M} \rightarrow \boldsymbol{\tau} \in \mathbb{R}^m \quad ; \quad \boldsymbol{\tau} = \text{Log}(\boldsymbol{\chi}) = \log(\boldsymbol{\chi})^\vee . \quad (2.61)$$

Fig. 2.7 recalls how a Lie group, its Lie algebra and the real vector space are related to each other with the maps to convert from one to another.

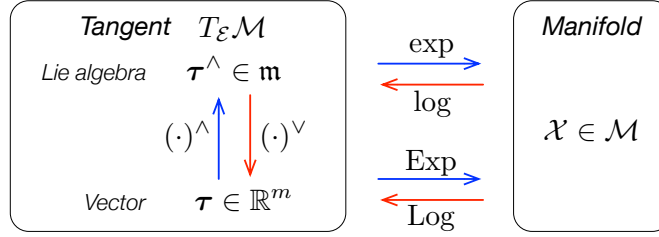


Figure 2.7: Mappings between the manifold  $\mathcal{M}$  and the representations of its tangent space at the origin  $\mathcal{T}_{\mathcal{M}}(\mathcal{E})$  (Lie algebra  $\mathfrak{m}$  and Cartesian  $\mathbb{R}^m$ ). Maps hat  $(\cdot)^\wedge$  and vee  $(\cdot)^\vee$  are the linear invertible maps or *isomorphisms* (2.52–2.53),  $\exp(\cdot)$  and  $\log(\cdot)$  map the Lie algebra to/from the manifold, and  $\text{Exp}(\cdot)$  and  $\text{Log}(\cdot)$  are shortcuts to map directly the vector space  $\mathbb{R}^m$  to/from  $\mathcal{M}$ .

### 2.2.5 The plus and minus operators

Plus and minus operators allow us to introduce increments of Lie group's elements which take place in the tangent space. Denoted  $\oplus$  and  $\ominus$ , they combine an  $\text{Exp}/\text{Log}$  operation with a composition. Because of the non-commutativity of the composition, they are defined in *right-* and *left-* version.

The right operators are,

$$\text{right-}\oplus \quad : \quad \gamma = \chi \oplus {}^x\tau \triangleq \chi \circ \text{Exp}({}^x\tau) \in \mathcal{M} \quad (2.62)$$

$$\text{right-}\ominus \quad : \quad {}^x\tau = \gamma \ominus \chi \triangleq \text{Log}(\chi^{-1} \circ \gamma) \in \mathcal{T}_{\mathcal{M}}(\chi), \quad (2.63)$$

where  ${}^x\tau$  belongs to the tangent space at  $\chi$ . It is said to be expressed in the *local* frame at  $\chi$ , following the convention of frame transformation, the reference frame is noted with a left superscript. Those operators are referred here as *right-*version since  ${}^x\tau$  appears on the right hand-side of the  $\oplus$  operator and  $\ominus$  is its counterpart.

The left operators are,

$$\text{left-}\oplus \quad : \quad \gamma = {}^\mathcal{E}\tau \oplus \chi \triangleq \text{Exp}({}^\mathcal{E}\tau) \circ \chi \in \mathcal{M} \quad (2.64)$$

$$\text{left-}\ominus \quad : \quad {}^\mathcal{E}\tau = \gamma \ominus \chi \triangleq \text{Log}(\gamma \circ \chi^{-1}) \in \mathcal{T}_{\mathcal{M}}(\mathcal{E}), \quad (2.65)$$

where  ${}^\mathcal{E}\tau$  is expressed in the *global* frame, it now appears on the left of the  $\oplus$  operator hence the *left* adjective of those operators.

Notice that while *right-* and *left-*  $\oplus$  are distinguished by the operands order, the notation in (2.63) and (2.65) is ambiguous. In general, in the context of SLAM, *local* perturbations are considered and thus the *right-*version of the operators is used as it will be the case hereafter unless otherwise stated. Similarly the frame superscript for  $\tau$  is dropped as it is assumed to be expressed in a *local* frame unless otherwise specified.

### 2.2.6 Derivatives on Lie groups

Inspired by the standard derivation (see Appendix B.1), one can now use the  $\oplus$  and  $\ominus$  operators to define Jacobians of functions  $f(\cdot) : \mathcal{M} \rightarrow \mathcal{N}$  acting on manifolds.

### Right Jacobians on Lie groups

Using the *right*-operators  $\oplus$  and  $\ominus$ , the derivative reads,

$$\frac{\partial f(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}} \triangleq \lim_{\boldsymbol{\tau} \rightarrow 0} \frac{f(\boldsymbol{\chi} \oplus \boldsymbol{\tau}) \ominus f(\boldsymbol{\chi})}{\boldsymbol{\tau}} \in \mathbb{R}^{n \times m}, \quad (2.66a)$$

which is equivalent to

$$= \lim_{\boldsymbol{\tau} \rightarrow 0} \frac{\text{Log}(f(\boldsymbol{\chi})^{-1} \circ f(\boldsymbol{\chi} \circ \text{Exp}(\boldsymbol{\tau})))}{\boldsymbol{\tau}} \in \mathbb{R}^{n \times m} \quad (2.66b)$$

and

$$= \lim_{\boldsymbol{\tau} \rightarrow 0} \frac{\log(f(\boldsymbol{\chi})^{-1} \circ f(\boldsymbol{\chi} \circ \exp(\boldsymbol{\tau}^\wedge)))^\vee}{\boldsymbol{\tau}} \in \mathbb{R}^{n \times m}. \quad (2.66c)$$

This Jacobian is called the *right Jacobian of  $f(\cdot)$*  – notice that the perturbation  $\boldsymbol{\tau}$  lie on the right side of the  $\oplus$  operator. When comparing the notations of (2.66a) and (2.66c), one easily see the benefit of the succinct form that offer the  $\oplus$  and  $\ominus$  operators. Not only is the notation in (2.66a) more compact than in (2.66c), but most importantly it looks more familiar and intuitive; it is the derivative of  $f(\boldsymbol{\chi})$  with respect to  $\boldsymbol{\chi}$  whose infinitesimals are expressed in the tangent space. Variations in  $\boldsymbol{\chi}$  and  $f(\boldsymbol{\chi})$  are expressed as vectors in the *local* tangent spaces, *i.e.* tangent respectively at  $\boldsymbol{\chi}$  and  $f(\boldsymbol{\chi})$ . This derivative is then a proper Jacobian matrix  $\mathbb{R}^{n \times m}$  mapping the local tangent spaces  $\mathcal{T}_{\mathcal{M}}(\boldsymbol{\chi}) \rightarrow \mathcal{T}_{\mathcal{N}}(f(\boldsymbol{\chi}))$ .

For small values of  $\boldsymbol{\tau}$ , the following approximation holds,

$$f(\boldsymbol{\chi} \oplus \boldsymbol{\tau}) \rightarrow f(\boldsymbol{\chi}) \oplus \frac{\partial f(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}} \boldsymbol{\tau} \in \mathcal{N}. \quad (2.67)$$

### Left Jacobians on Lie groups

Similarly to Section 2.2.6, derivatives can also be defined from the *left* -operators leading to,

$$\frac{\partial f(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}} \triangleq \lim_{\boldsymbol{\tau} \rightarrow 0} \frac{f(\boldsymbol{\tau} \oplus \boldsymbol{\chi}) \ominus f(\boldsymbol{\chi})}{\boldsymbol{\tau}} \in \mathbb{R}^{n \times m}, \quad (2.68a)$$

which is equivalent to

$$= \lim_{\boldsymbol{\tau} \rightarrow 0} \frac{\text{Log}(f(\text{Exp}(\boldsymbol{\tau}) \circ \boldsymbol{\chi}) \circ f(\boldsymbol{\chi})^{-1})}{\boldsymbol{\tau}} \in \mathbb{R}^{n \times m}, \quad (2.68b)$$

or fully developed,

$$= \lim_{\boldsymbol{\tau} \rightarrow 0} \frac{\log(f(\exp(\boldsymbol{\tau}^\wedge) \circ \boldsymbol{\chi}) \circ f(\boldsymbol{\chi})^{-1})^\vee}{\boldsymbol{\tau}} \in \mathbb{R}^{n \times m}. \quad (2.68c)$$

It is called the *left Jacobian of  $f(\cdot)$*  – notice that the perturbation  $\boldsymbol{\tau}$  lie on the left side of the  $\oplus$  operator. The perturbation here lies in the *global* frame, thus the left Jacobian is a  $n \times m$  matrix mapping the *global* tangents spaces.

For small values of  $\boldsymbol{\tau}$ , the following approximation holds,

$$f(\boldsymbol{\tau} \oplus \boldsymbol{\chi}) \rightarrow \frac{\partial f(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}} \boldsymbol{\tau} \oplus f(\boldsymbol{\chi}) \in \mathcal{N}. \quad (2.69)$$



### The Adjoint

A relation between local and the global tangent elements can be identified from (2.62) and (2.64),

$${}^{\mathcal{E}}\boldsymbol{\tau} \oplus \boldsymbol{\chi} = \boldsymbol{\chi} \oplus {}^{\mathcal{X}}\boldsymbol{\tau} , \quad (2.70a)$$

which develops to,

$$\text{Exp}({}^{\mathcal{E}}\boldsymbol{\tau})\boldsymbol{\chi} = \boldsymbol{\chi} \text{Exp}({}^{\mathcal{X}}\boldsymbol{\tau}) \quad (2.70b)$$

$$\exp({}^{\mathcal{E}}\boldsymbol{\tau}^\wedge) = \boldsymbol{\chi} \exp({}^{\mathcal{X}}\boldsymbol{\tau}^\wedge)\boldsymbol{\chi}^{-1} \quad (2.70c)$$

$$= \exp(\boldsymbol{\chi} {}^{\mathcal{X}}\boldsymbol{\tau}^\wedge \boldsymbol{\chi}^{-1}) \quad (2.70d)$$

$${}^{\mathcal{E}}\boldsymbol{\tau}^\wedge = \boldsymbol{\chi} {}^{\mathcal{X}}\boldsymbol{\tau}^\wedge \boldsymbol{\chi}^{-1} , \quad (2.70e)$$

where

$$\boldsymbol{\chi} \exp({}^{\mathcal{X}}\boldsymbol{\tau}^\wedge)\boldsymbol{\chi}^{-1} = \exp(\boldsymbol{\chi} {}^{\mathcal{X}}\boldsymbol{\tau}^\wedge \boldsymbol{\chi}^{-1}) , \quad (2.71)$$

is a property of the exponential map. This leads to defining the adjoint matrix

$$\mathbf{Ad}_\boldsymbol{\chi} \boldsymbol{\tau} = (\boldsymbol{\chi} {}^{\mathcal{X}}\boldsymbol{\tau}^\wedge \boldsymbol{\chi}^{-1})^\vee , \quad (2.72)$$

that linearly maps the tangent vectors  ${}^{\mathcal{E}}\boldsymbol{\tau}$  and  ${}^{\mathcal{X}}\boldsymbol{\tau}$ , allowing for a seamless transformation from a global to a local frame,

$${}^{\mathcal{E}}\boldsymbol{\tau} = \mathbf{Ad}_\boldsymbol{\chi} {}^{\mathcal{X}}\boldsymbol{\tau} . \quad (2.73)$$

The adjoint is effectively the Jacobian of the transformation of tangent vectors through Lie group's elements.

One can therefore show that the left and right Jacobians are related by the adjoint of  $\mathcal{M}$  and  $\mathcal{N}$ ,

$$\frac{{}^{\mathcal{E}}\partial f(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}} \mathbf{Ad}_\boldsymbol{\chi} = \mathbf{Ad}_{f(\boldsymbol{\chi})} \frac{{}^{\mathcal{X}}\partial f(\boldsymbol{\chi})}{\partial \boldsymbol{\chi}} . \quad (2.74)$$

Additional properties of the adjoint matrix are,

$$\boldsymbol{\chi} \oplus \boldsymbol{\tau} = (\mathbf{Ad}_\boldsymbol{\chi} \boldsymbol{\tau}) \oplus \boldsymbol{\chi} \quad (2.75)$$

$$\mathbf{Ad}_{\boldsymbol{\chi}^{-1}} = \mathbf{Ad}_\boldsymbol{\chi}^{-1} \quad (2.76)$$

$$\mathbf{Ad}_{\boldsymbol{\chi}\boldsymbol{\gamma}} = \mathbf{Ad}_\boldsymbol{\chi} \mathbf{Ad}_{\boldsymbol{\gamma}} . \quad (2.77)$$

### Elementary Jacobians

For all typical manifolds  $\mathcal{M}$  used in robotics, one can determine the closed form for the Jacobians of *inversion*, *composition*, *exponentiation* and *action*. Once these derivative ‘blocks’ are found, all other Jacobians can be inferred from them following the chain rule.

All Jacobians developed here are right-Jacobians, *i.e.*, defined by (2.66a).

**Inverse** The Jacobian of the inverse reads,

$$\mathbf{J}_\chi^{\chi^{-1}} \triangleq \frac{\partial \chi^{-1}}{\partial \chi} \in \mathbb{R}^{m \times m}. \quad (2.78)$$

It can be determined from the adjoint using (2.71) and (2.72),

$$\mathbf{J}_\chi^{\chi^{-1}} = \lim_{\tau \rightarrow 0} \frac{\text{Log}((\chi^{-1})^{-1}(\chi \text{Exp}(\tau))^{-1})}{\tau} \quad (2.79a)$$

$$= \lim_{\tau \rightarrow 0} \frac{\text{Log}(\chi \text{Exp}(-\tau)\chi^{-1})}{\tau} \quad (2.79b)$$

$$= \lim_{\tau \rightarrow 0} \frac{(\chi(-\tau)^\wedge \chi^{-1})^\vee}{\tau} \quad (2.79c)$$

$$= -\mathbf{Ad}_\chi. \quad (2.79d)$$

**Composition** The Jacobians of the composition function read,

$$\mathbf{J}_\chi^{\chi \circ \gamma} \triangleq \frac{\partial \chi \circ \gamma}{\partial \chi} \in \mathbb{R}^{m \times m} \quad (2.80)$$

$$\mathbf{J}_\gamma^{\chi \circ \gamma} \triangleq \frac{\partial \chi \circ \gamma}{\partial \gamma} \in \mathbb{R}^{m \times m}, \quad (2.81)$$

and are determined using (2.72) and (2.76),

$$\mathbf{J}_\chi^{\chi \circ \gamma} = \mathbf{Ad}_{\gamma^{-1}} \quad (2.82)$$

$$\mathbf{J}_\gamma^{\chi \circ \gamma} = \mathbf{I}. \quad (2.83)$$

**Exponentiation** The Jacobian of the ‘Exp’ function is defined as the right Jacobian of  $\mathcal{M}$ ,

$$\mathbf{J}_r(\tau) \triangleq \frac{\tau \partial \text{Exp}(\tau)}{\partial \tau} \in \mathbb{R}^{m \times m}. \quad (2.84)$$

The right Jacobian maps variations of  $\tau$  into variations in the local tangent space at  $\text{Exp}(\tau)$ . For small  $\delta\tau$ , the following approximations hold,

$$\text{Exp}(\tau + \delta\tau) = \text{Exp}(\tau) \text{Exp}(\mathbf{J}_r(\tau)\delta\tau) \quad (2.85)$$

$$\text{Exp}(\tau) \text{Exp}(\delta\tau) = \text{Exp}(\tau + \mathbf{J}_r^{-1}(\tau)\delta\tau) \quad (2.86)$$

$$\text{Log}(\text{Exp}(\tau) \text{Exp}(\delta\tau)) = \tau + \mathbf{J}_r^{-1}(\tau)\delta\tau. \quad (2.87)$$

Complementarily, the left Jacobian  $\mathbf{J}_l$  maps variations of  $\tau$  into variations in the global tangent space,

$$\mathbf{J}_l(\tau) \triangleq \frac{\varepsilon \partial \text{Exp}(\tau)}{\partial \tau} \in \mathbb{R}^{m \times m}, \quad (2.88)$$

leading to the following approximations,

$$\text{Exp}(\tau + \delta\tau) = \text{Exp}(\mathbf{J}_l(\tau)\delta\tau) \text{Exp}(\tau) \quad (2.89)$$

$$\text{Exp}(\delta\tau) \text{Exp}(\tau) = \text{Exp}(\tau + \mathbf{J}_l^{-1}(\tau)\delta\tau) \quad (2.90)$$

$$\text{Log}(\text{Exp}(\delta\tau) \text{Exp}(\tau)) = \tau + \mathbf{J}_l^{-1}(\tau)\delta\tau. \quad (2.91)$$

From (2.85) and (2.89), one can relate the left- and right- Jacobians with the Adjoint,

$$\mathbf{Ad}_{\text{Exp}(\boldsymbol{\tau})} = \mathbf{J}_l(\boldsymbol{\tau})\mathbf{J}_r^{-1}(\boldsymbol{\tau}) . \quad (2.92)$$

Also, the chain rule allows us to develop

$$\mathbf{J}_r(-\boldsymbol{\tau}) \triangleq \mathbf{J}_{-\boldsymbol{\tau}}^{\text{Exp}(-\boldsymbol{\tau})} = \mathbf{J}_{\boldsymbol{\tau}}^{\text{Exp}(-\boldsymbol{\tau})}\mathbf{J}_{-\boldsymbol{\tau}}^{\boldsymbol{\tau}} \quad (2.93a)$$

$$= \mathbf{J}_{\boldsymbol{\tau}}^{\text{Exp}(\boldsymbol{\tau})^{-1}}(-\mathbf{I}) \quad (2.93b)$$

$$= -\mathbf{J}_{\text{Exp}(\boldsymbol{\tau})}^{\text{Exp}(\boldsymbol{\tau})^{-1}}\mathbf{J}_{\boldsymbol{\tau}}^{\text{Exp}(\boldsymbol{\tau})} \quad (2.93c)$$

$$= \mathbf{Ad}_{\text{Exp}(\boldsymbol{\tau})}\mathbf{J}_r(\boldsymbol{\tau}) \quad (2.93d)$$

$$= \mathbf{J}_l(\boldsymbol{\tau}) . \quad (2.93e)$$

Closed forms of  $\mathbf{J}_r$ ,  $\mathbf{J}_r^{-1}$ ,  $\mathbf{J}_l$  and  $\mathbf{J}_l^{-1}$  exist for the typical manifolds used in robotics (see *e.g.* Table C.4).

**Action** Given  $\boldsymbol{\chi} \in \mathcal{M}$  and  $v \in \mathcal{V}$ , the Jacobians read,

$$\mathbf{J}_{\boldsymbol{\chi}}^{\boldsymbol{\chi} \cdot v} \triangleq \frac{\boldsymbol{\chi} \partial \boldsymbol{\chi} \cdot v}{\partial \boldsymbol{\chi}} \quad (2.94)$$

$$\mathbf{J}_v^{\boldsymbol{\chi} \cdot v} \triangleq \frac{v \partial \boldsymbol{\chi} \cdot v}{\partial v} . \quad (2.95)$$

Action is thus the effect of transforming an object  $v$  by means of a manifold element  $\boldsymbol{\chi}$ . For instance, rotating a vector  $v \in \mathbb{R}^2$  with an element  $\boldsymbol{\chi} \in \text{SO}(2)$ . Since the group actions depend on the set  $\mathcal{V}$ , the Jacobian expressions above cannot be generalized.

### Inferred Jacobians

**Log map** For  $\boldsymbol{\tau} = \text{Log}(\boldsymbol{\chi})$ ,

$$\mathbf{J}_{\boldsymbol{\chi}}^{\text{Log}(\boldsymbol{\chi})} = \mathbf{J}_r^{-1}(\boldsymbol{\tau}) . \quad (2.96)$$

**Plus and minus** The Jacobians read,

$$\mathbf{J}_{\boldsymbol{\chi}}^{\boldsymbol{\chi} \oplus \boldsymbol{\tau}} = \mathbf{J}_{\boldsymbol{\chi}}^{\boldsymbol{\chi} \circ \text{Exp}(\boldsymbol{\tau})} = (\mathbf{Ad}_{\text{Exp}(\boldsymbol{\tau})})^{-1} \quad (2.97)$$

$$\mathbf{J}_{\boldsymbol{\tau}}^{\boldsymbol{\chi} \oplus \boldsymbol{\tau}} = \mathbf{J}_{\text{Exp}(\boldsymbol{\tau})}^{\boldsymbol{\chi} \circ \text{Exp}(\boldsymbol{\tau})} \mathbf{J}_{\boldsymbol{\tau}}^{\text{Exp}(\boldsymbol{\tau})} = \mathbf{J}_r(\boldsymbol{\tau}) , \quad (2.98)$$

and given  $\boldsymbol{\zeta} = \boldsymbol{\chi}^{-1} \circ \boldsymbol{\gamma}$  and  $\boldsymbol{\tau} = \boldsymbol{\gamma} \ominus \boldsymbol{\chi} = \text{Log}(\boldsymbol{\zeta})$ ,

$$\mathbf{J}_{\boldsymbol{\chi}}^{\boldsymbol{\gamma} \ominus \boldsymbol{\chi}} = \mathbf{J}_{\boldsymbol{\zeta}}^{\text{Log}(\boldsymbol{\zeta})} \mathbf{J}_{\boldsymbol{\chi}^{-1}}^{\boldsymbol{\zeta}} \mathbf{J}_{\boldsymbol{\chi}}^{\boldsymbol{\chi}^{-1}} = -\mathbf{J}_l^{-1}(\boldsymbol{\tau}) \quad (2.99)$$

$$\mathbf{J}_{\boldsymbol{\tau}}^{\boldsymbol{\gamma} \ominus \boldsymbol{\chi}} = \mathbf{J}_{\boldsymbol{\zeta}}^{\text{Log}(\boldsymbol{\zeta})} \mathbf{J}_{\boldsymbol{\gamma}}^{\boldsymbol{\zeta}} = \mathbf{J}_r^{-1}(\boldsymbol{\tau}) . \quad (2.100)$$

### 2.2.7 Uncertainty on Lie groups

As seen in Section 2.1 the SLAM problem is best formulated in terms of probabilities. Especially, the robot state  $\boldsymbol{\chi} \in \mathcal{M}$  takes the form of a Gaussian random variable for which (2.2) no longer holds for Lie groups that are not closed under the  $+$  operation.

One therefore defines local perturbations  $\boldsymbol{\tau}$  in the tangent vector space at  $\bar{\boldsymbol{\chi}}$  such that,

$$\boldsymbol{\chi} = \bar{\boldsymbol{\chi}} \oplus \boldsymbol{\tau}, \quad \boldsymbol{\tau} = \boldsymbol{\chi} \ominus \bar{\boldsymbol{\chi}} \in \mathcal{T}_{\mathcal{M}}(\bar{\boldsymbol{\chi}}). \quad (2.101)$$

From there, because the perturbation  $\boldsymbol{\tau} \in \mathcal{T}_{\mathcal{M}}$  lies in a vector space, one can leverage all the usual tools from probabilities.

Covariance matrices can be properly defined on the tangent space at  $\bar{\boldsymbol{\chi}}$  through the standard expectation operator  $\mathbb{E}[\cdot]$  (2.5a),

$$\boldsymbol{\Sigma}_{\boldsymbol{\chi}} \triangleq \mathbb{E}[\boldsymbol{\tau}\boldsymbol{\tau}^T] = \mathbb{E}[(\boldsymbol{\chi} \ominus \bar{\boldsymbol{\chi}})(\boldsymbol{\chi} \ominus \bar{\boldsymbol{\chi}})] \in \mathbb{R}^{n \times n}, \quad (2.102)$$

allowing the definition of Gaussian random variables on manifolds,  $\boldsymbol{\chi} \sim \mathcal{N}(\bar{\boldsymbol{\chi}}, \boldsymbol{\Sigma}_{\boldsymbol{\chi}})$ . Notice that although we write  $\boldsymbol{\Sigma}_{\boldsymbol{\chi}}$ , the covariance is rather that of the tangent perturbation  $\boldsymbol{\tau}$ .

Perturbations can also be expressed in the global reference frame using the left-operators (2.64) and (2.65) yielding global covariance specification. Similarly to the tangent vectors in (2.73), covariances can be transformed from a local to a global reference frame according to,

$${}^{\mathcal{E}}\boldsymbol{\Sigma}_{\boldsymbol{\chi}} = \mathbf{Ad}_{\boldsymbol{\chi}} \boldsymbol{\Sigma}_{\boldsymbol{\chi}} \mathbf{Ad}_{\boldsymbol{\chi}}^T. \quad (2.103)$$

Covariance propagation through a function  $f(\cdot) : \mathcal{M} \rightarrow \mathcal{N}$  ;  $\boldsymbol{\chi} \rightarrow \boldsymbol{\gamma} = f(\boldsymbol{\chi})$  only requires the linearization (2.67) with the Jacobian matrix (2.66a) to yield the familiar formula,

$$\boldsymbol{\Sigma}_{\boldsymbol{\gamma}} \approx \mathbf{J}_{\boldsymbol{\chi}}^{\boldsymbol{\gamma}} \boldsymbol{\Sigma}_{\boldsymbol{\chi}} \mathbf{J}_{\boldsymbol{\chi}}^{\boldsymbol{\gamma}T} \in \mathbb{R}^{m \times m}. \quad (2.104)$$

# Chapter 3

## Loop-Closure

---

*The loop closure module is probably the strictest module in terms of correctness in a SLAM framework. Erroneous loop-closure may have a catastrophic effect on the problem estimation and are possibly difficult to identify and filter-out. The loop closure detection must then be strengthened targeting 99+% accuracy while maintaining a satisfactory recall so that the system retains the ability to close as many loops as possible.*

---

## 3.1 Related work

Unlike appearance-based place recognition using vision, there is little published work using solely 2D LRF. It is possibly due to the fact that reliable features for 2D laser scans were developed later than their image-based counterparts.

Similarly to images, scans may be described globally by global descriptors. The Geometrical Landmark Relations (GLARE) [71] encodes the geometrical relations of FLIRT corners in a histogram of relative distance over relative orientation. Extending GLARE, the Geometrical Surface Relations [72] descriptor considers every reading of the 2D laser scan rather than extracted corners. The Geometric Relation Distribution (GRD) feature [73] encodes geometric pairwise relations between landmark points into a continuous probability density function that serves as a signature to compared scans.

There also exists local features for 2D scans which allow for the use of the BoW scheme (see Section 2.1.3) for place the problem of place recognition. The local feature Fast Laser Interest Point Transform (FLIRT) [74] is robust to scale and orientation changes and as been employed to tackle the problem of place recognition [75]. More recently Kallasi *et al.* proposed the FALKO binary feature [76] designed to be efficient to compute and manipulate.

## 3.2 Feature comparison for BoW-based visual place recognition

While most modern V-SLAM algorithms rely on the BoW scheme for their loop-closure module, many of them use the recent ORB-based BoW implementation of [8] as an off-the-shelf component. This is great from a software re-usability point of view; however one may question the particular choice of the ORB features for this task, especially if the SLAM front-end does not use this particular feature.

Following the work of [49] this section presents an extended comparison of visual local features for the task of BoW-based place recognition. The comparison helps to better understand how the choice of a given feature type influences the recognition performance, together with the execution time.

### 3.2.1 Experiments

The experiments are conducted on the KITTI data set [77] using 15 meaningful combinations of the point-based local feature algorithms available in the OpenCV library [78]. The combinations of detector/descriptor evaluated are listed in Table 3.1. For each combination the experiment is as follows,

- The BoW vocabulary is trained for the descriptor type on a large data set of random images extracted from internet.
- The KITTI sequences are pre-processed in order to select a subset of images that are going to compose the BoW databases. Images are selected so that from two consecutive images the Essential matrix can be estimated in a *Random Sample Consensus* (RANSAC) scheme with at least  $N$  inlier features.

Detectors	ORB	SIFT	SURF	KAZE	AKAZE
Descriptors	ORB	SIFT	SURF	KAZE	AKAZE
Detectors	BRISK	MSER	FAST	AGAST	AGAST
Descriptors	BRISK	SIFT	DAISY	BRIEF	DAISY
Detectors	MSER	FAST	AGAST	FAST	AGAST
Descriptors	SURF	LATCH	BRISK	BRISK	LATCH

Table 3.1: Selected combination of detectors and descriptors from `OpenCV` used in the experiments.

- Each sequence is then processed by the BoW framework. The geometrical check is performed by estimating the Essential matrix in a RANSAC scheme again.

The vehicle poses ground-truth corresponding to the images are provided by the data set. The BoW recognition is considered a true positive if the recognized image position is within a 12 meters radius and 30 degree rotation from the query image position. Otherwise, it is considered a false positive. The performance is evaluated in terms of ‘precision’ and ‘recall’, two metrics used in the fields of pattern recognition and classification which capture an understanding of *relevance*. Precision – also known as positive predictive value – is the fraction of relevant instances among all retrieved instances, while recall – also known as sensitivity – is the fraction of the total amount of relevant instances that were actually retrieved. Precision and recall are computed respectively as,

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \quad (3.1)$$

$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \quad (3.2)$$

All user-defined parameters of the RANSAC scheme were set so that the precision scores are above 99% as one would want in a SLAM loop-closure context.

## Results

Results are presented in Fig. 3.1. They show that despite its common usage, the ORB feature may not be the best suited for this task. Indeed 9 other combinations exhibit a higher recall than ORB at 99+% precision. However most of them are also much slower than ORB to compute, by an order of magnitude, and thus are not suitable for the task. However, the combination of the AGAST detector and the BRIEF descriptor increases the recall by  $\sim 10\%$  at 99+% precision compared to ORB for a very similar cost in terms of execution time.

### 3.2.2 Conclusion

The results tend to suggest that despite its popularity, the ORB feature is not the best suited feature for the task of place recognition. With very little change one may gain  $\sim 10\%$  of recall for the same execution time.

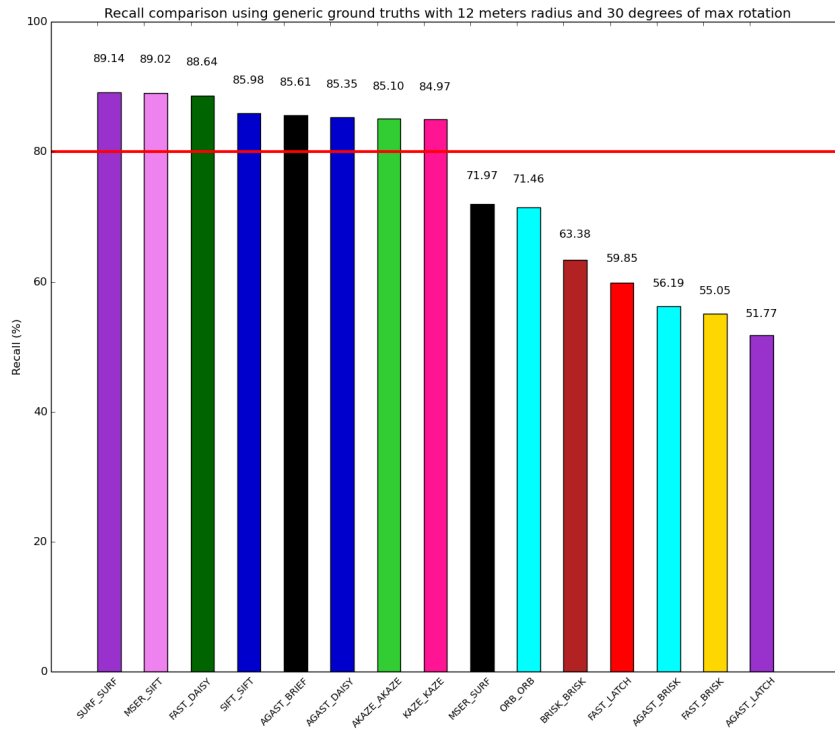


Figure 3.1: Comparison of OpenCV features for BoW-based place recognition - recall at 99+% precision.

### 3.3 BoW-based loop-closure for 2D laser scans

In contrast to other sensing modalities, 2D LRF data has a natural (counter-) clock-wise ordering of its readings which can be easily exploited to reinforce the computation of scan similarity. Some empirical observations concerning the local features ordering are drawn, and used in an algorithm that computes the best feature correspondence assignment between two scans.

The observations are the following; given local features extracted from a 2D scan (quantized into words) they are ordered clock-wise in a sequence. The ordering remains the same for a given scene observed from slightly different viewpoints. As the change in viewpoint increases, features shift their location in the sequence or possibly reorder, they may disappear and new features may appear.

#### 3.3.1 Feature sequence encoding as a Hidden Markov Model

The data association between two scans is done directly on words after quantization of the features. Therefore, a given descriptor quantized into a particular word  $w$ , can only match features also quantized as  $w$  and in no case could match another word in the vocabulary. This is exemplified in Fig. 3.2 (top). The problem of scan alignment is then analogous to finding the path that maximizes the sequence of feature matches in a *Hidden Markov Model* (HMM). Consider the query laser scan  $l_i$  and its extracted words  $w_{1i}, \dots, w_{Ni}$  as the set of states  $S_N$  in the model. Consider also the candidate scan match  $l_j$  with its words  $w_{1j}, \dots, w_{Mj}$  as a set of observations



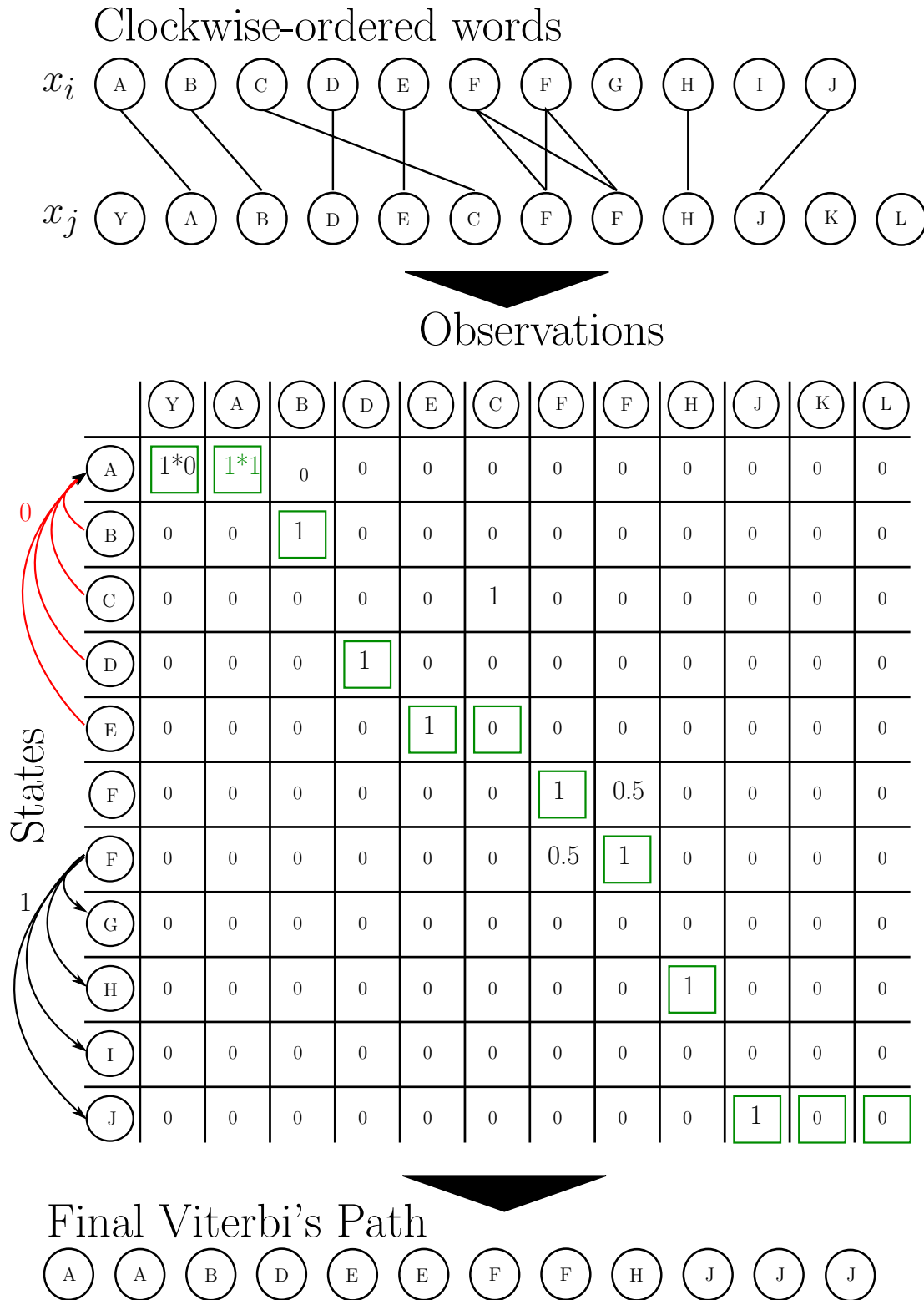


Figure 3.2: Top: Clockwise ordered words of two scans and their matches. Middle: The resulting hidden Markov model. Each cell represents the product  $\phi_{s_{n-x}|s_n} \cdot \theta_{s_n|o_m}$  e.g cells Y-A & A-A. Green squared cells represent the best path across the complete model. Bottom: The final sequence of states given the observations  $O_n$

$O_M$ . One can define a HMM such that:

- There are equal initial probabilities  $\delta_{s_n} = \frac{1}{N}$ .
- The transition from one state to another solely goes forward with respect to the clockwise ordering of the states. Self transitions have a lower probability to enforce the importance of alignments  $\phi_{s_n|s_n} = \frac{0.5}{F}$ ,  $\phi_{s_n|s_{n+x}} = \frac{1+\frac{0.5}{F-1}}{F}$ , and  $\phi_{s_n|s_{n-x}} = 0$ , where  $F$  is the number of states following the currently evaluated state in the ordered sequence.
- The output probability is defined such that a word mismatch has null probability whereas a word match has equal probability. Hence the emission probabilities are  $\theta_{s_n|o_m} = \frac{1}{C}$  for a match, and  $\theta_{s_n|o_m} = 0$  for a mismatch, where  $C$  is the number of matches of the currently evaluated word.

Fig. 3.2 (middle) gives an unnormalized representation of the HMM produced by the matching of words in scans  $l_i$  and  $l_j$ . Black downward pointing arrows indicate feasible transitions, and red upward pointing arrows indicate non-feasible transitions. Each cell is then filled by the product  $\phi_{s_{n-x}|s_n} \cdot \theta_{s_n|o_m}$ , where  $s_n$  is the currently evaluated state,  $s_{n-x}$  is the previous most likely state and  $\theta_{s_n|o_m}$  the output probability. Columns are filled recursively based on the previous iteration.

Unlike [79], the HMM is built based on the inner ordering of two independent sets of features extracted from raw sensor readings, whereas [79] builds a HMM based on the inner ordering of two independent sequences of key-frames, hence not impacting the frame-to-frame similarity measure. Once the HMM is built, the goal is then to find a sequence of states that maximizes the probability of a path across it.

### The Viterbi algorithm

In order to find the most probable path at a reasonable cost in terms of computation, one can use of the Viterbi algorithm [80]. This dynamic programming algorithm searches recursively for the most likely sequence of states given a sequence of events, by computing for each observation the partial probability with respect to the previous state that optimally induced the current state. Such sequence is called the Viterbi path. It is commonly used in *Natural Language Processing* (NLP) and decoding [81, 82].

Crossing edges as shown in Fig. 3.2 (top) highlight multi-matches. These might occur either because different features are quantized to the same word (*e.g.* words  $C$  and  $F$ ), or because the feature belongs to a moving object. The work in [75] does not discard such mismatches while constructing the offset histogram, and thus they can not be taken into account to compute a consistent relative transform. Thanks to the constraint of forward state transition, the Viterbi algorithm naturally discards such crossing edges. Note that in Fig. 3.2 (top), crossing edges for the sequence  $C$ - $D$ - $E$  can be resolved in at least two different ways, either by removing the match  $C$  and keeping  $D$  and  $E$ , or removing the latter keeping only  $C$ . The Viterbi algorithm maximizes the sequence of states in the Viterbi path hence would likely favor, in this case, keeping matches  $D$  and  $E$  and discarding  $C$ .

## Scoring

Once the Viterbi path is obtained, the candidate is scored based on three criteria:

- The number of correct matches that have not been discarded by the Viterbi algorithm.
- The number of sequences of consecutive words that have a correct match.
- The distribution of matches in the laser scans. The wider the better.

The second point is similar to the concept of phrases in [75], where a phrase represents a sequence of consecutive words, analogous to a  $n$ -grams model.

Considering such sequences and weighting them according to their length, one can add an extra layer of constraints to the geometric check. These criteria evaluate respectively to:  $score_{jk} = \frac{|M|}{|C|}$ , where  $|M|$  is the number of correct matches and  $|C|$  the number of features in the candidate scan;  $weight_{jk} = \frac{|CM|}{|C|}$ , where  $|CM|$  is the number of sequences of consecutive correct matches, *e.g.*, sequences  $A-B$  &  $D-E$  in Fig. 3.2 (top); and  $ratio_{jk} = \frac{Id_r - Id_l}{|C|}$ , where  $Id_r$  and  $Id_l$  are the indices of the rightmost- and leftmost- correct matches in the Viterbi path, respectively. These three criteria are aggregated into a final geometric score,

$$g_{jk} = \frac{score_{jk} + weight_{jk}}{2} \cdot ratio_{jk} . \quad (3.3)$$

While querying the BoW database, both the TF-based similarity and the geometric score in (3.3) are computed for each document in the database. They are aggregated into a single similarity term,

$$sg_{jk} = sim_{jk} \cdot g_{jk} . \quad (3.4)$$

This aggregated similarity term is then used to rank BoW candidates instead of the TF-IDF-based similarity.

### 3.3.2 Pose-graph database augmentation

In this section is proposed a topological augmentation of the BoW database. By database *augmentation* one refers to the fact of benefiting from common features in adjacent poses (documents) in the pose-graph of the map for the computation of the TF-IDF weights. Since the pose-graph is computed by a graph-based SLAM front end, finding adjacencies in the database involves no computation overhead.

Database augmentation taking the form of a similarity graph has been proposed in [83] and [84] for the task of image recognition. Graph edges are created by matching image features and asserting an affine transform between images. Direct edges represent document adjacencies; documents connected to an adjacent document then represents *2-adjacencies*, and so on. The set  $E_j$  of adjacencies of document  $d_j$  is used to emphasize the TF weight of the document,

$$m_{ij} = n_{ij} + \sum_{k \in E_j} n_{ik} \quad (3.5)$$

$$atf_{ij} = \frac{m_{ij}}{\sum_i m_{ij}} . \quad (3.6)$$

Dataset	# Scans	Length (m)	In/Out
FR-079	1464	390.8	Indoor
FR-CLINIC	6917	1437.6	Outdoor
INTEL-LAB	2672	360.7	Indoor
MIT-CSAIL-3rd-FLOOR	1051	382.9	Indoor

Table 3.2: Public data sets used for the experiments [85].

Alias	BoW	“weak” check	adjacent TF
tfidf	x	-	-
tfidfgraph	x	-	x
viterbi	x	x	-
vitgraph	x	x	x

Table 3.3: Aliases for the different methods compared.

This new score is coined *Adjacency-TF* (ATF). Adjacency-TF can be used as a direct drop-off replacement for (2.30) in (2.32), so that the TF-IDF weight becomes

$$x_{ij} = atf_{ij} \cdot idf_{ij} . \quad (3.7)$$

In the case of object recognition, the database augmentation is based on object appearance similarity. However in the case of place recognition within a SLAM framework the topological distribution of the places matters more. Since an edge in a pose-graph SLAM is computed from sensor readings and represents a spatial constraint, it embeds both the appearance-based similarity required by the BoW scheme (consecutive nodes share some common features) and the topological information that is emphasized by the database augmentation.

Whereas objects recognition usually considers a pre-trained database for which an offline database augmentation can be computed [83, 84], in the case of place recognition within a SLAM framework the database together with its augmentation are constructed online. Using the SLAM pose-graph built online by the framework allows for a database augmentation at no extra cost.

Turcot *et al.* [84] identify useful features (features belonging to a transformation inlier set) from the document adjacencies and discards the others in order to decrease the database size together with its noise. Since the database here is built online, all features are kept as they can become useful later on during mapping.

### 3.3.3 Experiments

Experiments were carried out over four standard 2D laser data sets (three indoors and one outdoor). Table 3.2 lists the data sets together with their details. First, the contributions are evaluated both separately and jointly against our own implementation of the classical TF-IDF-based BoW and against the publicly available *Geometrical FLIRT Phrase algorithm* (Gflip) [75] using the experiment proposed by [75]. Second, the robustness of the method with respect to changes in the environment is evaluated using synthetic data. Hereafter the aliases given in Table 3.3 are used for simplicity.

Each data set is pre-processed by a SLAM algorithm [86] in order to provide a baseline pose estimation against which to assess if a detected loop-closure is correct

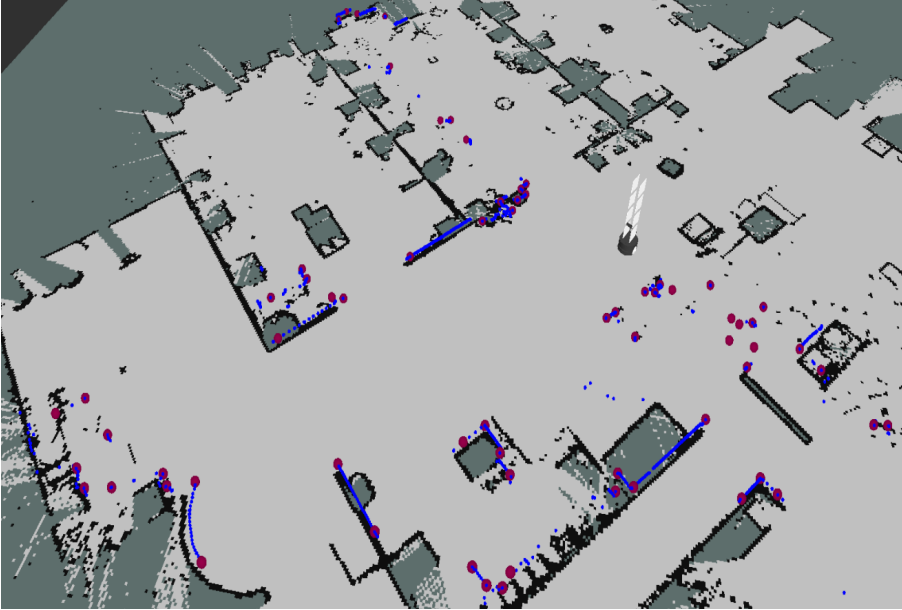


Figure 3.3: FLIRT features (violet spheres) extracted from a scan (blue dots and segments) superimposed on a map.

or not. Algorithms performance are compared with the following procedure for each data set used.

First, the data set is used to train the BoW database. For every scan, its FLIRT [74] features are extracted (see Fig. 3.3), then quantized to obtain their associated BoW and signature. During query, a scan used to query the database is first removed from it in order to avoid the obvious one-to-one matching. A consistency check is performed on the top  $N$  candidates returned by the query process. The candidate whose inlier set has the the smallest residual error, given that it meets an inliers threshold, is considered a loop closure.

To appraise the correctness of a recognition, the estimated rigid transform is compared to that of the aforementioned baseline algorithm from the pre-processed data set. It is considered correct if the difference between the estimated pose and that of the baseline lies within 0.5 m and 10 degrees.

The vocabulary is trained from 20.000 scans randomly sampled from a randomly selected subset of data sets among the vast database of the company PAL Robotics. These data sets were recorded in the form of Rosbags at different time and places, the great majority being recorded in indoor real case scenario. The training data sets are thus different than those used for evaluation. On average, 17.5 features are extracted per scan.

The hierarchical K-means tree architecture is chosen empirically using the aforementioned experiment. A total of 49 different trees are trained, varying the branching factor from 2 to 7 and the depth factor from 2 to 7 as well. Evaluating the  $F_1$  score (3.8) averaged over all experiments lead to the selection of the average optimal architecture of  $k = 4$  and  $d = 6$  (Fig. 3.4). As the number of tree leaves is rather small, they are parsed linearly, unlike [46], in order to reduce the quantization error.

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \quad (3.8)$$

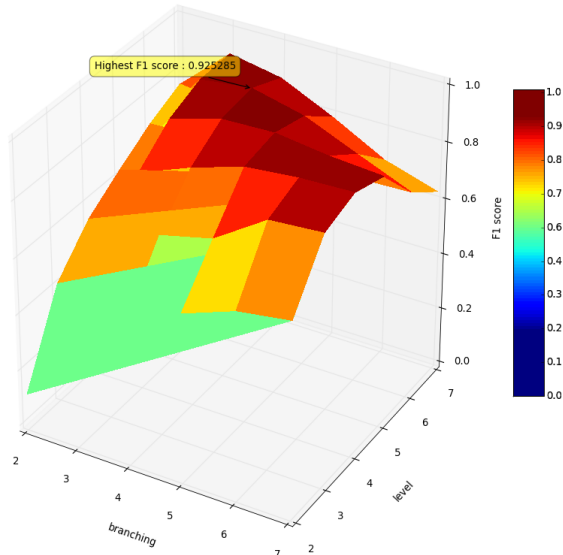


Figure 3.4: Variation of the average  $F1$  score with respect to branching factor and tree level.

Concerning the BoW database augmentation, only direct document adjacencies are considered as documents with 2-adjacencies do not show any improvements whereas 3-adjacencies decreased the quality of results. This is expected as 3-adjacencies end up linking observation in the map which seldom share common features.

### Implementation Details

The two proposals and *gflip* are implemented in C++. All experiments are conducted on a system having an Intel i7-870, 2.93 GHz CPU, 8 GiB of RAM and running on Ubuntu 16.04.

### Recognition performance

Precision over recall performance results are shown in Fig. 3.5 for each algorithm. The top candidate threshold is set as  $N = 20$  while the inlier threshold is varied. As can be seen in plots Fig. 3.5 (a-c), accounting for the indoor data sets, both the *viterbi* and *vitgraph* versions of the proposed method outperform *tfidf* and *gflip* both in terms of recall and precision. While *tfidfgraph* only slightly surpasses the performance of *tfidf* in two data sets, it outperforms *gflip* for the MIT-CSAIL-3rd-FLOOR data set.

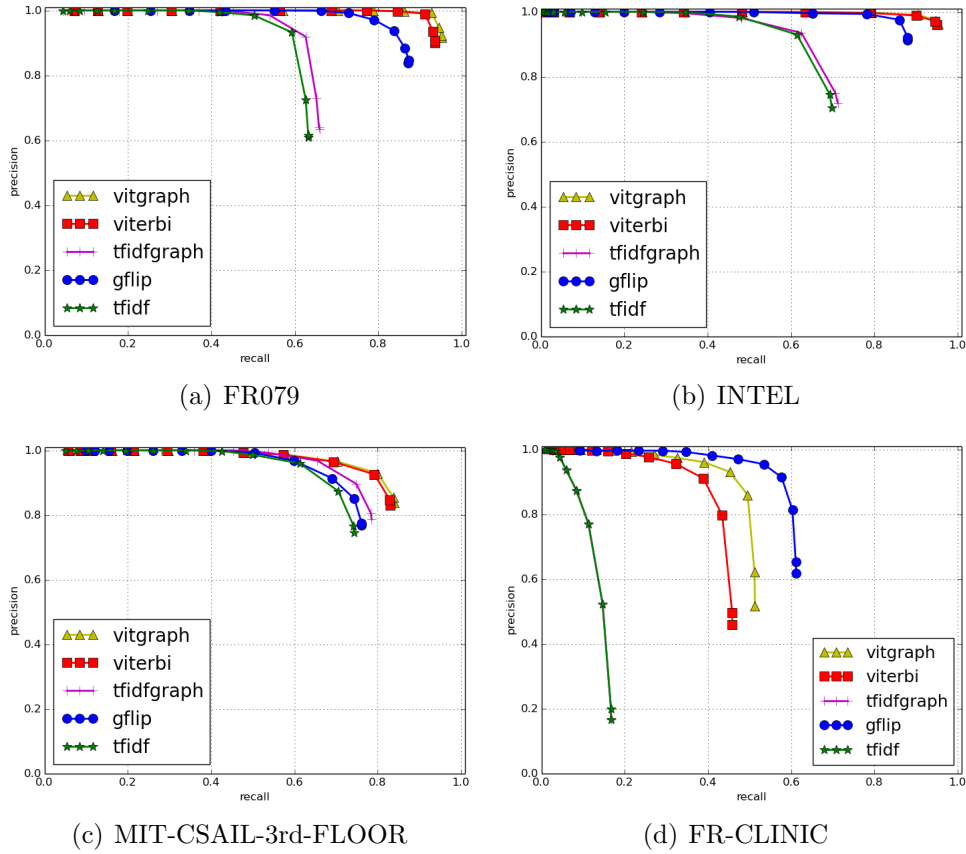


Figure 3.5: Recall versus precision for 20 candidates varying the inliers threshold.

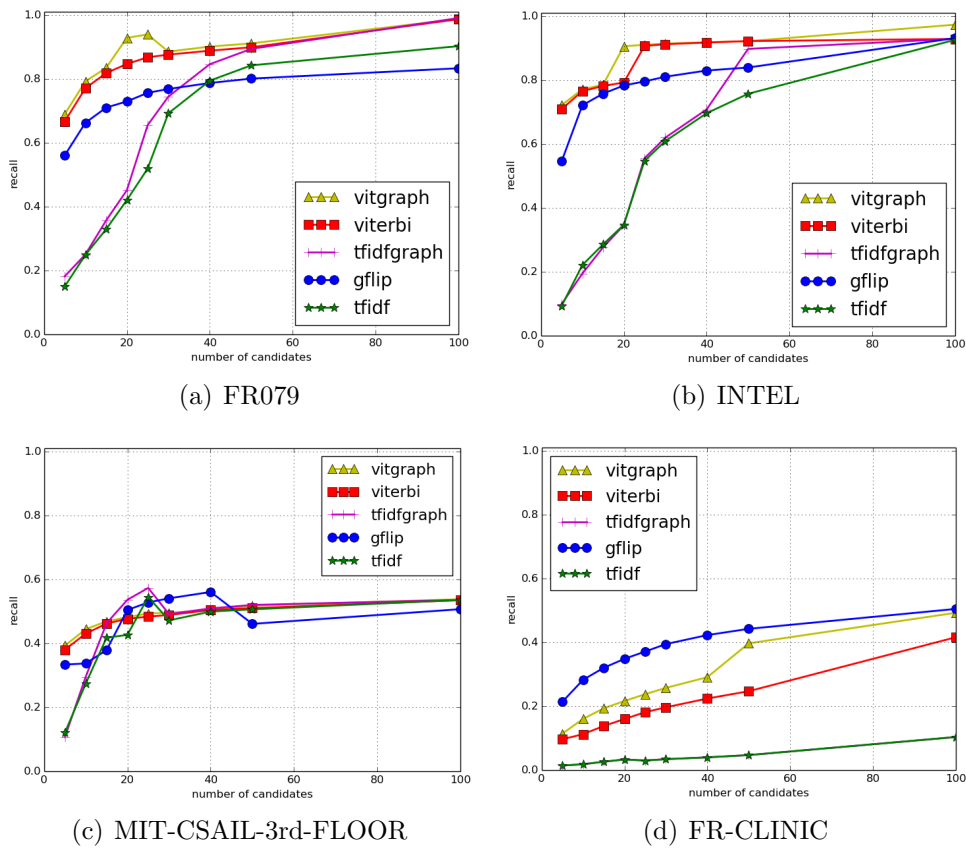


Figure 3.6: Recall versus number of candidates at 99% precision.

Dataset	GFLIP			TF-IDF			TF-IDF+GRAPH			VITERBI			VITGRAPH								
	Top-N	Inl	TP	FP	F1	Inl	TP	FP	F1	Inl	TP	FP	F1	Inl	TP	FP	F1				
FR079	5	9	911	53	772	7	386	178	393	7	324	36	368	7	1008	29	827	7	1033	26	<b>840</b>
	10	9	1070	73	843	7	562	36	562	7	565	70	555	7	1153	30	892	7	1185	25	<b>908</b>
	20	9	1211	159	884	7	831	60	725	7	877	77	745	7	1278	13	949	7	1300	10	<b>959</b>
	50	9	1276	66	932	7	1290	34	947	7	1342	22	971	7	1346	20	973	7	1357	18	<b>978</b>
	100	7	1360	78	958	5	1363	16	981	5	1414	28	<b>994</b>	7	1382	10	990	7	1383	11	990
INTEL	5	5	1947	72	832	3	897	628	428	3	791	816	370	3	2346	145	910	3	2365	126	<b>918</b>
	10	5	2136	73	877	5	1149	124	584	5	1061	139	549	3	2464	95	944	3	2473	88	<b>947</b>
	20	5	2292	58	914	5	1637	124	740	5	1666	116	749	3	2518	75	958	3	2525	69	<b>960</b>
	50	5	2437	28	950	3	2505	103	950	3	2523	86	957	3	2574	36	976	3	2575	36	<b>976</b>
	100	3	2512	54	961	3	2579	39	977	3	2588	29	980	3	2587	31	980	3	2591	26	<b>981</b>
MIT- CSAIL- 3rd-FLOOR	5	5	555	124	646	5	249	92	361	5	233	95	341	5	644	75	733	5	666	73	<b>749</b>
	10	5	679	139	731	5	457	116	567	5	478	126	582	5	747	74	803	5	771	70	<b>820</b>
	20	5	773	136	794	5	733	106	781	5	778	90	816	5	823	66	854	5	831	63	<b>860</b>
	50	5	858	128	847	5	859	73	872	5	882	62	890	5	865	72	876	3	917	112	<b>886</b>
	100	5	905	91	889	5	896	64	896	5	898	61	899	5	894	65	895	5	898	59	<b>900</b>
FRCLINIC	5	7	2555	261	<b>525</b>	5	361	365	095	5	361	365	095	5	1517	394	344	5	1796	365	396
	10	7	3326	322	<b>630</b>	5	629	625	154	5	629	625	154	5	2206	593	454	5	2558	487	514
	20	7	3987	376	<b>707</b>	5	1015	922	229	5	1015	922	229	5	3002	765	562	5	3424	559	629
	50	7	4838	426	<b>795</b>	5	1774	1260	357	5	1774	1260	357	5	4236	761	712	5	4674	495	774
	100	7	5345	423	843	5	2596	1402	476	5	2596	1402	476	5	5145	688	808	5	5535	405	<b>861</b>

Table 3.4: First experiment : Algorithms statistics for each dataset at max F1 score. *Top-N*: Number of candidates - *Inl*: Inliers threshold - *TP*: True Positive - *FP*: False Positive - *F1*: F1 score (per mille  $\%$ )



Table 3.4 reports statistics for each algorithm at different Top  $N$  values. Each row shows the top performance of the algorithm in terms of its  $F1$  score for a fixed Top  $N$  while varying the inlier threshold. The best  $F1$  score of each row is highlighted in boldface. The results show that the proposed Viterbi-based assertion allows for a drastic improvement of the recall and precision over the classical BoW’s performance and outperforms *gflip* for the three indoor data sets. We attribute this to the fact that the top  $N$  returned by the BoW query incorporate “weak” geometrical constraints that are fully leveraged within the purely geometrical consistency check. Adding the BoW database augmentation further improves these results, especially when the number of query candidates is small (lowest values of Top  $N$ ). However its effect is mitigated when used alone as it decreases the retrieval performance compared to *tfidf* for the lower Top  $N$  values while increasing it for the higher Top  $N$  values. Finally, Table 3.4 shows that *gflip* requires a higher number of inlier threshold ( $Inl$ ) to reach its optimal performance, by a number of 2 in average compared to *tfidf*. The proposed algorithm reaches its optimal performance for the same  $Inl$  value as *tfidf* or less.

Fig. 3.6 shows the recall over Top  $N$  for each algorithm with precision over 99%. Fig. 3.6 (a-c) show that both *viterbi* and *vitgraph* outperform *tfidf* and *gflip* recall for the indoor data sets. Moreover, *tfidfgraph* also clearly outperforms *gflip* for the higher Top  $N$  cases.

### Synthetic obstacles experiment setup

The second experiment aims at evaluating the robustness of the proposed *vitgraph* to substantial changes in the environment using synthetic data. For each of the three indoor datasets, an occupancy grid of 0.05m resolution is generated (see *e.g.* Fig. 3.7(a)). First, a set of synthetic scans is generated by the mean of ray-tracing using the occupancy grid. The set is used to train the BoW databases. Then, virtual obstacles are added (painted) to the occupancy-grid at random position (but on the robot trajectory) (Fig. 3.7(b)). Obstacles are of three different shapes - circles, rectangles and legs (two small circles side-by-side) and different size, from 0.05 to 1m for the first two shapes. Then, again, a new set of scans is generated from the new occupancy grid and used for querying the BoW database.

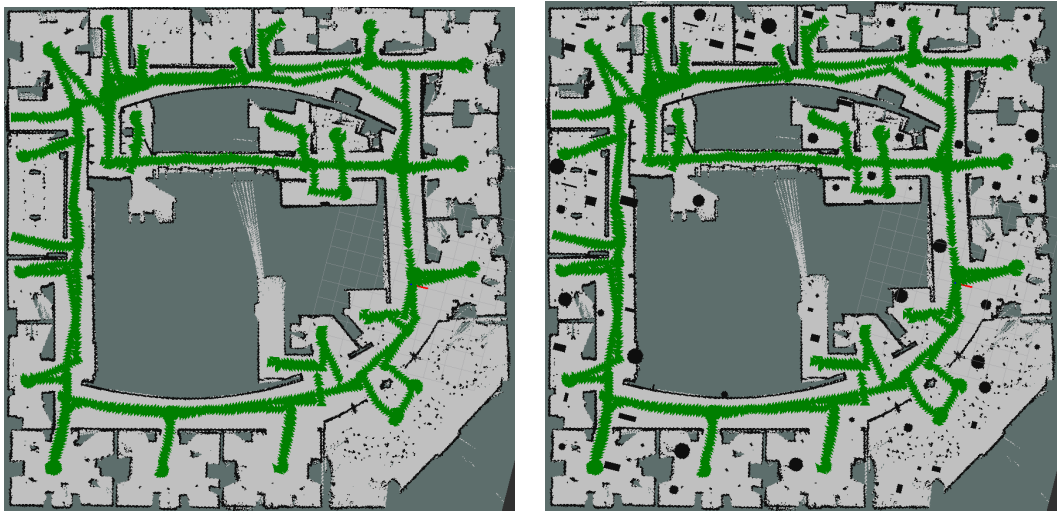
**Results** Table 3.5 reports statistics for both *gflip* and *vitgraph* for a selected subset of Top  $N$ . The results show an expected decrease of the recognition performance compared to the previous experiment, but remain strong enough to perform loop closures. The magnitude of the performance decrease is similar for both *vitgraph* and *gflip*. Since the proposed algorithm performs better than [75] in the previous experiment, it still does in this second experiment. The results show that *vitgraph* is robust to changes in the environment.

### Execution performance

Table 3.6 shows the average execution time per query for both *gflip* and *vitgraph* at 99% precision varying Top  $N$  considering both the smallest and the largest data sets. It also reports the average query time of our *tfidf* implementation as a comparison point for the overhead induced by the weak geometrical check. Both Fig. 3.8 and

		GFLIP				VITGRAPH			
Dataset	Top-N	Inl	TP	FP	F1	Inl	TP	FP	F1
FR079	20	9	716	412	570	7	861	369	<b>659</b>
	50	9	893	466	651	7	906	412	<b>670</b>
INTEL	20	5	1700	482	702	3	2022	503	<b>780</b>
	50	5	1935	562	750	3	2062	510	<b>788</b>
MIT-CSAIL-3rd-FLOOR	20	5	505	308	555	5	579	166	<b>662</b>
	50	5	615	338	628	5	640	168	<b>690</b>

Table 3.5: Second experiment : Algorithms statistics for each indoor dataset at max F1 score. *Top-N*: Number of candidates - *Inl*: Inliers threshold - *TP*: True Positive - *FP*: False Positive - *F1*: F1 score (per mille ‰)



(a) Intel-lab occupancy grid

(b) Intel-lab occupancy grid with painted obstacles

Figure 3.7: Occupancy grid generated from the Intel-lab data set before and after the addition of virtual obstacles.

the accompanying video<sup>1</sup> show results of the application of the method during the mapping of a large mall floor.

These results highlight that the computation overhead of the Viterbi path is linear in the number of scans. The complexity of the Viterbi algorithm is about  $O(Q \cdot C^2)$  where  $Q$  is the number of features of the query scan and  $C$  the number of features of the candidate scan. Hence for every query the total computation overhead is  $O(D \cdot Q \cdot C^2)$  with  $D$  the number of documents in the database. In our experiments, *vitgraph* runs at nearly 2 Hz for the largest data set and up to 22 Hz for the smallest one. This suggests that a carefully designed implementation could run at frame rate, around 10 Hz, as commonly encountered with LRF. However, in practice the target rate should rather be that of the solver (one should allow the solver to finish before issuing a new loop closure event) – around 1 Hz. In such context, the overall impact of improving the precision/recall performance overcomes the penalty in execution time when compared to [75].

<sup>1</sup><http://goo.gl/DcCj8q>



Figure 3.8: Occupancy grid of a large mall floor (approx.  $2900m^2$ ). Red dots represent robot key frames and green edges loop-closures. During mapping, 415 loop closures were detected of which only 2 were false positives, easily eliminated using distance constraints.

### 3.3.4 Conclusion

In this section were presented two contributions to the BoW-based place recognition using 2D LRF only. First, a ‘weak’ geometrical check that emphasizes BoW candidates which share a reliable static sequence of features with a given query scan. Sequences are detected and matched through the use of the Viterbi algorithm, bringing further together place recognition and natural language processing.

The second contribution is a topological augmentation of the BoW database which permits topological neighbors to empower each other in the BoW candidates ranking list. By using the graph provided by the SLAM algorithm such augmentation comes at no extra computation cost.

The addition of both contributions to the classical TF-IDF scheme outperforms

Dataset	# Candidates	GFLIP	TF-IDF	VITGRAPH
MIT-CSAIL-3rd-FLOOR	5	0.0038	0.0188	0.0276
	50	0.0125	0.0264	0.0367
	100	0.0233	0.0347	0.0453
FR-CLINIC	5	0.0337	0.1220	0.3369
	50	0.0571	0.1323	0.3545
	100	0.0848	0.1548	0.3686

Table 3.6: Average query time (seconds).

the state of the art loop closure detection methods both in terms of recall and precision for three indoor data sets, while drastically improving the classical TF-IDF results for all data sets.

Interestingly, both *viterbi* and *gflip* seem to reach a plateau for the *FR079* and *INTEL* data sets for the higher Top  $N$  cases. However, *vitgraph*'s recall keeps increasing. This unveils the limitation of geometrical properties such as the one inferred in [75] or in this work and suggests that a closer attention to word-to-word matching should be considered. It also highlights the capability of neighbor documents to empower each other in the candidate ranking list of the BoW scheme, and thus supports the idea of using a topological augmentation of the BoW database as proposed here.

Despite the improvement in recognition of the proposed method over *tfidf*, *gflip* appears to perform better for the outdoor data set *FRCLINIC*. We conjecture this as being due to the fact that the vocabulary tree used is trained from scans captured in indoor environments in the vast majority. Effectively biasing its performance towards features encountered indoor.

# Chapter 4

## Odometry Calibration

---

*For mobile robots, odometry is a key element in the computation of localization estimates. By integrating and composing the wheels displacements measured during a time step to a known pose at the previous step, one can have a rough estimate of the robot trajectory. As odometry algorithms solely integrate and compose small displacements, they have the main drawback of accumulating error unboundedly. Moreover, the integration requires the a-priori knowledge of robot's specific kinematic model parameters. Such parameters are estimated from often tedious and complex calibration procedures performed by experts. A small error in calibration will lead to a large inaccuracy in the pose estimation over time, preventing other tasks to be performed successfully (e.g. localization, mapping).*

---

## 4.1 Related work

Most popular odometry calibration procedures rely on the execution by the mobile base of a predetermined trajectory, taking measures of the pose error along the path either manually or with an external sensor. Borenstein and Feng [87] proposed the so called UMBmark test, a calibration procedure requiring the robot to drive along a square path both clockwise and counter-clockwise. Ideally, by the end of the procedure the robot has come back to its initial pose. Measuring and minimizing the pose error between the initial and final pose allows to calibrate the kinematic parameters. Kelly [88] generalized this procedure by replacing the squared shape trajectory by a list of repeatedly visited way-points along a trajectory of any shape.

Martinelli *et al.* [89] proposed to augment the state of a Kalman Filter used for localization with the kinematic parameters. It however requires an *a-priori* known map. Later, the use of an EKF allowed Martinelli *et al.* [90] to simultaneously estimate the systematic and non systematic odometry errors of a mobile robot.

Censi *et al.* [91] proposed the simultaneous calibration of the differential drive kinematics together with the relative 2D pose of a sensor that estimates the robot ego-motion. They formulate the calibration problem in terms of a maximum likelihood and solve it in closed form. Their method does not require any predefined path nor an external sensor but is suited only for parameters that do not vary over time.

In a graph-based SLAM context, Kümmerle *et al.* [92] append the unknown parameters to key-frames states. Doing so allows to consider the kinematic parameters as varying in time (dynamic) if, *e.g.*, the robot is loaded with a cargo heavy enough to alter the previous estimate.

Recently, Cicco *et al.* [93] proposed an unsupervised calibration procedure. By exploring and recording the effects of elementary motions on the uncertainty of the parameters estimate, their method chooses autonomously at every time the best next motion for the robot to perform to further reduce the uncertainty.

The calibration presented here can be seen as a middle ground between those of Censi *et al.* [91] and Kümmerle *et al.* [92], in that it allows for the calibration of dynamic kinematic parameters and static sensor extrinsics without a complete SLAM framework around it. The proposed method does not require a predefined trajectory, an *a-priori* known map, nor any external sensor/landmarks to perform the calibration.

To approach the calibration of the motion model, we get inspired by the IMU pre-integration theory initiated by [94], later improved in [95], then [96]. We apply it to differential drive motion estimation, and use the mechanisms initially conceived for IMU bias estimation to achieve the self-calibration of the motion model parameters. In this regard, we improve over the formulation in [95] in the sense that we provide recursive integration formulae, and an integration pipeline divided in small steps. This results in equivalent but simpler formulae, especially for the Jacobians through the use of the chain rule. We make systematic use of Lie theory as exposed in Section 2.2, making this work a true on-manifold estimation approach.

## 4.2 Abstraction on Lie groups of the pre-integration theory

In a typical mobile robot, motion measurements are acquired at high rate, typically at 100 – 1000 Hz. A measurement  $\mathbf{u}_k$  corresponding to a single time increment  $\delta t$  at time  $t_k$  produces a local state increment  $\delta_k \in \mathcal{M}$ , named the ‘current delta’ through a kinematic motion model characterized by a set of parameters  $\mathbf{c}$ ,

$$\delta_k = \delta(\mathbf{u}_k, \mathbf{c}) \in \mathcal{M} . \quad (4.1)$$

By now, let  $\mathcal{M}$  be an abstract manifold where the robot state evolves. In fact, measurements typically come in the form of velocities or local increments and can be easily expressed as vectors  $\mathbf{b}_k$  in the Lie algebra of  $\mathcal{M}$ , so that  $\delta_k = \text{Exp}(\mathbf{b}_k(\mathbf{u}_k, \mathbf{c}))$ . Between two distant time instants  $t_i$  and  $t_k$ , several  $\delta$  can be integrated into a single ‘delta’ or increment  $\Delta_{ik} \in \mathcal{M}$  expressing the robot state at time  $t_k$  relative to the robot state at time  $t_i$  (Fig. 4.1). This integral can be computed recursively through  $\Delta_{ik} = \Delta_{ij} \oplus \delta_k$ , where  $\oplus$  indicates the composition law of the Lie group  $\mathcal{M}$  and  $\Delta_{ii}$  is its identity.

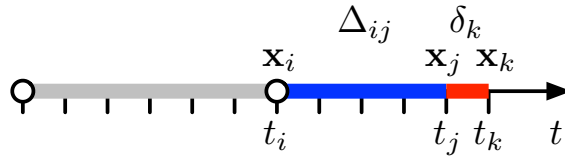


Figure 4.1: The pre-integrated delta  $\Delta_{ij} \in \mathcal{M}$  contains all motion increments from time  $i$  up to time  $j$ , so that  $\mathbf{x}_j = \mathbf{x}_i \circ \Delta_{ij}$ . The current delta  $\delta_k \in \mathcal{M}$  contains the motion from time  $j$  to  $k$ , computed from the last motion measurement at time  $k$ . We have that  $\Delta_{ik} = \Delta_{ij} \oplus \delta_k$ .

The pre-integration theory developed for the IMU sensor [94, 95] deals with the problem of producing motion factors for a factor graph from the aggregation of hundreds of motion measurements. At the time of evaluating the residuals of such factors, one realizes that the integrated IMU delta  $\Delta_{ik}$  has two undesired dependencies with the state [94]. On one side, it depends on the initial state  $\mathbf{x}_i$ ; on the other side, it depends on the sensor biases  $\mathbf{c}$ . This can be visualized as

$$\Delta_{ik}(U_{ik}, \mathbf{x}_i, \mathbf{c}) = \Delta_{ij}(U_{ij}, \mathbf{x}_i, \mathbf{c}) \oplus \delta_k(\mathbf{u}_k, \mathbf{c}) , \quad (4.2)$$

where  $U_{ik} = \{\mathbf{u}_i, \dots, \mathbf{u}_k\}$  is the set of all measurements in the interval. Since the estimates of  $\mathbf{x}_i$  and  $\mathbf{c}$  change during the optimization,  $\Delta_{ik}$  would need to be re-integrated at each solver iteration for the residual to be evaluated. This is addressed in two ways. First, a change of reference frame [96] allows us to write a delta that is independent of the initial states. Second, the effect of the change in the calibration  $\mathbf{c}$  is linearized around a value  $\bar{\mathbf{c}}$  so that the pre-integrated delta can be corrected *a-posteriori*. The result is a pre-integrated delta that only depends on the measured data and  $\bar{\mathbf{c}}$ ,

$$\bar{\Delta}_{ik}(U_{ik}, \bar{\mathbf{c}}) = \bar{\Delta}_{ij}(U_{ij}, \bar{\mathbf{c}}) \oplus \delta_k(\mathbf{u}_k, \bar{\mathbf{c}}) , \quad (4.3)$$

together with an expression for linearly correcting it when the estimations of bias  $\mathbf{c}$  deviate from the values  $\bar{\mathbf{c}}$  used during pre-integration,

$$\Delta_{ik}(U_{ik}, \mathbf{c}) \approx \bar{\Delta}_{ik}(U_{ik}, \bar{\mathbf{c}}) \oplus \mathbf{J}_{\mathbf{c}}^{\Delta_{ik}} \cdot (\mathbf{c} - \bar{\mathbf{c}}), \quad (4.4)$$

This pre-integration avoids the need of re-integrating all motion data at each iteration of the optimizer, as can be seen in the expression of the factor's expectation error,

$$\mathbf{e} = (\bar{\Delta}_{ik}(U_{ik}) \oplus \mathbf{J}_{\mathbf{c}}^{\Delta_{ik}}(\mathbf{c} - \bar{\mathbf{c}})) \ominus (\mathbf{x}_i^{-1} \circ \mathbf{x}_j), \quad (4.5)$$

which clearly separates the states  $\{\mathbf{x}_i, \mathbf{x}_j, \mathbf{c}\}$  from the measurements  $U_{ik}$ . The Jacobian matrix  $\mathbf{J}_{\mathbf{c}}^{\Delta_{ik}}$  required for this update is pre-integrated alongside  $\bar{\Delta}_{ik}$  during the motion phase. The same is true for the covariance matrix  $\mathbf{Q}_{\Delta_{ik}}$  required to compute the residual  $\mathbf{r} = \mathbf{Q}_{\Delta_{ik}}^{-\top/2} \mathbf{e}$ .

In the following the measurements  $U_{ij}$  and  $\mathbf{u}_k$  are dropped from the notation for simplicity.

### 4.3 Pre-integration for the differential drive motion model

Hereafter, the motion model considered is the the differential drive depicted in Section 2.1.2.

#### 4.3.1 Delta pre-integration

Contrary to the IMU case, the ‘deltas’ of pose in  $\text{SE}(n)$  are naturally independent of the initial pose  $\mathbf{x}_i$ . Thus we only need to address the dependency with the calibration parameters  $\mathbf{c}$ , which is done by setting  $\mathbf{b}_k = \mathbf{b}_k(\mathbf{u}_k, \bar{\mathbf{c}})$ .

The ‘current delta’  $\delta_k = (\delta x_k, \delta y_k, \delta \theta_k) \triangleq (\delta \mathbf{p}_k, \delta \theta_k) \in \text{SE}(2)$  is computed from the arc (2.26) using the exponential map  $\delta_k = \text{Exp}(\mathbf{b}_k)$  (2.60) with  $\delta s_k = 0$ ,

$$\begin{aligned} \delta \mathbf{p}_k &= \begin{bmatrix} \frac{\delta l_k}{\delta \theta_k} \sin(\delta \theta_k) \\ \frac{\delta l_k}{\delta \theta_k} (1 - \cos(\delta \theta_k)) \end{bmatrix} \approx \begin{bmatrix} \delta l_k \cos(\frac{1}{2} \delta \theta_k) \\ \delta l_k \sin(\frac{1}{2} \delta \theta_k) \end{bmatrix} \\ \delta \theta_k &= \delta \theta_k, \end{aligned} \quad (4.6)$$

where the right-hand side expressions account for suitable approximations when  $\delta \theta_k \rightarrow 0$ . The Jacobian of the current delta  $\delta_k$  is computed from the Jacobian of  $\text{Exp}(\mathbf{b})$ , which for  $\mathbf{b} = (u, v, \theta)^\top \in \mathfrak{se}(2)$  is given in Appendix C.

The pre-integrated delta  $\bar{\Delta}_{ij} = (\bar{\Delta} \mathbf{p}_{ij}, \bar{\Delta} \theta_{ij}) \in \text{SE}(2)$  is the discrete integration of several current deltas  $\delta_k$ . The operator  $\oplus$  in (4.3) is given by the composition law of  $\text{SE}(2)$ ,

$$\begin{aligned} \bar{\Delta} \mathbf{p}_{ik} &= \bar{\Delta} \mathbf{p}_{ij} + \bar{\Delta} \mathbf{R}_{ij} \delta \mathbf{p}_k \\ \bar{\Delta} \theta_{ik} &= \bar{\Delta} \theta_{ij} + \delta \theta_k, \end{aligned} \quad (4.7)$$

where  $\bar{\Delta} \mathbf{R}_{ij} = \mathbf{R}(\bar{\Delta} \theta_{ij}) \triangleq \begin{bmatrix} \cos \bar{\Delta} \theta_{ij} & -\sin \bar{\Delta} \theta_{ij} \\ \sin \bar{\Delta} \theta_{ij} & \cos \bar{\Delta} \theta_{ij} \end{bmatrix}$ . Integrated angles are systematically brought back to  $(-\pi, \pi]$ .



### 4.3.2 Delta Jacobian pre-integration

All Jacobian and covariance blocks hereafter are computed in the Lie-theoretic form (2.66a) They are of dimension  $3 \times 3$  unless otherwise stated.

The Jacobian  $\mathbf{J}_{\mathbf{c}}^{\Delta ik}$  in (4.4) is computed recursively starting at  $\mathbf{J}_{\mathbf{c}}^{\Delta ii} = \mathbf{0}$  and using the chain rule,

$$\mathbf{J}_{\mathbf{c}}^{\Delta ik} = \mathbf{J}_{\Delta ij}^{\Delta ik} \mathbf{J}_{\mathbf{c}}^{\Delta ij} + \mathbf{J}_{\delta_k}^{\Delta ik} \mathbf{J}_{\mathbf{b}_k}^{\delta_k} \mathbf{J}_{\mathbf{c}}^{\mathbf{b}_k}, \quad (4.8)$$

where  $\mathbf{J}_{\Delta ij}^{\Delta ik}$ ,  $\mathbf{J}_{\delta_k}^{\Delta ik}$ ,  $\mathbf{J}_{\mathbf{b}_k}^{\delta_k}$  and  $\mathbf{J}_{\mathbf{c}}^{\mathbf{b}_k}$  are respectively the Jacobian blocks of (4.7), (4.6) and (2.25–2.26). The Jacobians  $\mathbf{J}_{\Delta ij}^{\Delta ik}$ ,  $\mathbf{J}_{\delta_k}^{\Delta ik}$  of (4.7) are given by those of the composition law of SE(2) in Table C.4.

### 4.3.3 Delta covariance pre-integration

Let  $\mathbf{Q}_{\psi}$ ,  $\mathbf{Q}_{\delta}$  and  $\mathbf{Q}_{\Delta}$  be the covariances of respectively the measurement noise given in (2.29), the current delta and the pre-integrated delta. The covariance of the current delta  $\delta_k$  reads,

$$\mathbf{Q}_{\delta} = \mathbf{J}_{\mathbf{b}_k}^{\delta_k} (\mathbf{J}_{\delta\psi}^{\mathbf{b}_k} \mathbf{Q}_{\psi} \mathbf{J}_{\delta\psi}^{\mathbf{b}_k \top} + \mathbf{J}_s^{\mathbf{b}_k} \sigma_s^2 \mathbf{J}_s^{\mathbf{b}_k \top}) \mathbf{J}_{\mathbf{b}_k}^{\delta_k \top}, \quad (4.9)$$

with  $\mathbf{J}_{\delta\psi}^{\mathbf{b}_k}$ , and  $\mathbf{J}_s^{\mathbf{b}_k} = [0, 1, 0]^{\top} \in \mathbb{R}^3$ , the Jacobians of (2.26), and  $\sigma_s^2$  the perturbation variance of the wheel slippage  $\delta s_k$ . The motion covariance starts at  $\mathbf{Q}_{\Delta ii} = \mathbf{0}$  and is also pre-integrated recursively,

$$\mathbf{Q}_{\Delta ik} = \mathbf{J}_{\Delta ij}^{\Delta ik} \mathbf{Q}_{\Delta ij} \mathbf{J}_{\Delta ij}^{\Delta ik \top} + \mathbf{J}_{\delta_k}^{\Delta ik} \mathbf{Q}_{\delta} \mathbf{J}_{\delta_k}^{\Delta ik \top}. \quad (4.10)$$

### 4.3.4 Residual

The residual of the differential drive factors reads,

$$\mathbf{r}(\mathbf{c}) = \mathbf{\Omega}^{\top/2} (\Delta(\mathbf{c}) - \widehat{\Delta}) \in \mathbb{R}^3, \quad (4.11)$$

where  $\mathbf{\Omega} = \mathbf{Q}_{\Delta}^{-1}$  is the information matrix of the pre-integrated motion,  $\Delta(\mathbf{c})$  comes from (4.4); and  $\widehat{\Delta} = \mathbf{x}_k \ominus \mathbf{x}_i$  is an independent estimate of the platform motion typically obtained from another embedded sensor (laser scan registration *e.g.* [33], visual odometry, *etc.*) an external sensor (Vicon [97], *etc.*), or from a graph with nodes  $\mathbf{x}_i, \mathbf{x}_k$ .

## 4.4 Joint calibration of differential drive intrinsic and sensor extrinsic parameters

We assumed in Section 4.3 that both  $\widehat{\Delta}$  and  $\Delta$  are expressed in the differential drive's reference frame. From now on, we consider  $\widehat{\Delta}^S$  to be expressed in another sensor's reference frame,  $S$ , relative to the robot frame by the extrinsics  $\mathbf{T} \triangleq (x, y, \theta) \in \text{SE}(2)$ .

The aim is now to calibrate jointly the differential drive model  $\mathbf{c}$  together with the sensor extrinsics  $\mathbf{T}$ . From Fig. 4.2 we clearly see that  $\mathbf{T} \circ \Delta^S = \Delta \circ \mathbf{T}$ , so the

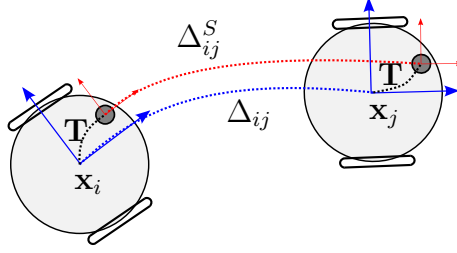


Figure 4.2: A differential drive robot moves from pose  $\mathbf{x}_i$  to  $\mathbf{x}_j$ . It mounts an exteroceptive sensor at pose  $\mathbf{T}$  (red) with respect to the robot base (blue). It holds that  $\Delta_{ij} \circ \mathbf{T} = \mathbf{T} \circ \Delta_{ij}^S$ .

error of each individual motion delta following this closed kinematic chain can be defined as,

$$\mathbf{e}(\mathbf{c}, \mathbf{T}) = (\mathbf{T} \circ \widehat{\Delta}^S) - (\Delta(\mathbf{c}) \circ \mathbf{T}) . \quad (4.12)$$

#### 4.4.1 Jacobians and covariance propagation

The Jacobians of each error  $\mathbf{e}_k$  with respect to the unknown  $\mathbf{c}$  and  $\mathbf{T}$  can be computed by steps, using the Jacobian blocks in Appendix C and the chain rule. Define with Fig. 4.2,

$$\mathbf{e} = \mathbf{U} - \mathbf{L} , \quad \mathbf{U} \triangleq \mathbf{T} \circ \widehat{\Delta}^S , \quad \mathbf{L} \triangleq \Delta \circ \mathbf{T} . \quad (4.13)$$

where Jacobians  $\mathbf{J}_T^U, \mathbf{J}_{\widehat{\Delta}^S}^U, \mathbf{J}_\Delta^L, \mathbf{J}_T^L$  are given by those of the composition law of SE(2) in Table C.4 and  $\mathbf{J}_U^e = \mathbf{I} , \mathbf{J}_L^e = -\mathbf{I}$ .

Then apply the chain rule to find all the Jacobians of  $\mathbf{e}$ ,

$$\mathbf{J}_\Delta^e = \mathbf{J}_L^e \mathbf{J}_\Delta^L = -\mathbf{J}_\Delta^L \quad (4.14a)$$

$$\mathbf{J}_{\widehat{\Delta}^S}^e = \mathbf{J}_U^e \mathbf{J}_{\widehat{\Delta}^S}^U = \mathbf{J}_{\widehat{\Delta}^S}^U \quad (4.14b)$$

$$\mathbf{J}_T^e = \mathbf{J}_L^e \mathbf{J}_T^L + \mathbf{J}_U^e \mathbf{J}_T^U = \mathbf{J}_T^U - \mathbf{J}_T^L \quad (4.14c)$$

$$\mathbf{J}_c^e = \mathbf{J}_L^e \mathbf{J}_\Delta^L \mathbf{J}_c^\Delta = -\mathbf{J}_\Delta^L \mathbf{J}_c^\Delta . \quad (4.14d)$$

Then propagate both  $\mathbf{Q}_{\widehat{\Delta}^S}$  and  $\mathbf{Q}_\Delta$  to the space of  $\mathbf{e}$ ,

$$\mathbf{Q}_e = \mathbf{J}_{\widehat{\Delta}^S}^e \mathbf{Q}_{\widehat{\Delta}^S} \mathbf{J}_{\widehat{\Delta}^S}^{e \top} + \mathbf{J}_\Delta^e \mathbf{Q}_\Delta \mathbf{J}_\Delta^{e \top} \quad (4.15)$$

#### 4.4.2 Residual

The residual is similar to (4.11) with  $\mathbf{\Omega} = \mathbf{Q}_e^{-1}$ ,

$$\mathbf{r} = \mathbf{\Omega}^{\top/2} \mathbf{e} \in \mathbb{R}^3 \quad (4.16a)$$

$$\mathbf{J}_c^r = \mathbf{\Omega}^{\top/2} \mathbf{J}_c^e , \quad \mathbf{J}_T^r = \mathbf{\Omega}^{\top/2} \mathbf{J}_T^e . \quad (4.16b)$$

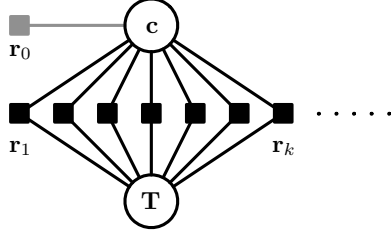


Figure 4.3: Factor graph for the estimation problem. Two state blocks  $\mathbf{c}$  and  $\mathbf{T}$  are linked by a number of factors, each computing a residual of the type  $\mathbf{r}_k = \boldsymbol{\Omega}_k^{\top/2}(\Delta_k(\mathbf{c}) \circ \mathbf{T} - \mathbf{T} \circ \Delta_k^S)$  (see Fig. 4.2). An absolute factor (grey) of the type  $\mathbf{r}_0 = \boldsymbol{\Omega}_0^{\top/2}(\mathbf{c} - \mathbf{c}_0)$  keeps  $\mathbf{c}$  close to its nominal values  $\mathbf{c}_0$ .

## 4.5 Calibration

### 4.5.1 Batch calibration process

The calibration problem can be modeled as a simple factor graph composed of only two nodes (Fig. 4.3), one holding the differential drive kinematic parameters  $\mathbf{c}$ , the second holding the sensor extrinsic parameters  $\mathbf{T}$ . The nodes are constrained one another by  $K$  factors, each containing the pre-integration of a (large) number of motion measurements, whose residuals are evaluated with (4.16a). Additionally, the  $\mathbf{c}$  node is constrained by an absolute factor attracting the calibration parameters towards their nominal values  $\mathbf{c}_0$ ,

$$\mathbf{r}_0(\mathbf{c}) = \boldsymbol{\Omega}_0^{\top/2}(\mathbf{c} - \mathbf{c}_0) \quad (4.17)$$

where  $\boldsymbol{\Omega}_0 = \text{diag}(\sigma_l^{-2}, \sigma_r^{-2}, \sigma_d^{-2})$  is chosen sufficiently small not to constrain the optimizer from reaching an adequate solution. In our experiments we chose  $\sigma_l = \sigma_r = \sigma_d = 0.01$ .

Collecting all factors, the estimation problem can be written as,

$$[\mathbf{c}^*, \mathbf{T}^*] = \arg \min_{\mathbf{c}, \mathbf{T}} \sum_{k=0}^K \mathbf{r}_k(\mathbf{c}, \mathbf{T})^\top \mathbf{r}_k(\mathbf{c}, \mathbf{T}) . \quad (4.18)$$

A simple solution can be implemented via Gauss-Newton optimization, by iterating until convergence,

$$\mathbf{J}_k = [\mathbf{J}_c^{\mathbf{r}_k} \quad \mathbf{J}_T^{\mathbf{r}_k}], \quad \mathbf{b} = \sum_k \mathbf{J}_k^\top \mathbf{r}_k, \quad \mathbf{H} = \sum_k \mathbf{J}_k^\top \mathbf{J}_k \quad (4.19)$$

$$\Delta \mathbf{x} = -\mathbf{H}^+ \mathbf{b} \quad (4.20)$$

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta \mathbf{x} \quad (4.21)$$

where  $\mathbf{x} = (\mathbf{c}, \mathbf{T})$ , and  $\mathbf{H}^+$  is the pseudo-inverse of  $\mathbf{H}$ .

### 4.5.2 Online calibration process

Online incremental self-calibration can be easily achieved by repetitively solving the problem as new factors are incorporated. In order to speed-up operation, the

```

Input:  $\mathbf{c}_0, \mathbf{T}_0, \{\Psi_k\}, \{\Delta_k^S\}$ 
 $\mathbf{c} = \mathbf{c}_0, \mathbf{T} = \mathbf{T}_0$ 
while new sensor reading  $\{\Delta_k^S, \mathbf{Q}_k^S\}$  do
   $\bar{\mathbf{c}} = \mathbf{c}$ 
  Pre-integrate diff. drive motion
   $\{\bar{\Delta}, \mathbf{Q}, \mathbf{J}_c^\Delta\}_k = \text{integrate}(\bar{\mathbf{c}}, \Psi_k)$  (2.25–4.10)
  Pack all info for factor  $k$ 
   $\{\bar{\Delta}, \mathbf{Q}, \mathbf{J}_c^\Delta, \bar{\mathbf{c}}, \Delta^S, \mathbf{Q}^S\}_k \rightarrow \Phi_k$ 
   $\mathbf{x} = (\mathbf{c}, \mathbf{T})$ 
  while not end condition do
     $\mathbf{b} = 0, \mathbf{H} = 0$ 
    for  $i \in W$  do
      Unpack info for factor  $i$ 
       $\{\bar{\Delta}, \mathbf{Q}, \mathbf{J}_c^\Delta, \bar{\mathbf{c}}, \Delta^S, \mathbf{Q}^S\} \leftarrow \Phi_i$ 
       $\Delta(\mathbf{c}) = \text{correct}(\bar{\Delta}, \mathbf{J}_c^\Delta, \mathbf{c}, \bar{\mathbf{c}})$  (4.4)
       $\{\mathbf{r}, \mathbf{J}_c^r, \mathbf{J}_T^r\}_i = \mathbf{r}_i(\Delta(\mathbf{c}), \mathbf{Q}, \mathbf{T}, \Delta^S, \mathbf{Q}^S)$ 
       $\mathbf{b} \leftarrow \mathbf{b} + \mathbf{J}_i^r \mathbf{r}_i, \mathbf{H} \leftarrow \mathbf{H} + \mathbf{J}_i^r \mathbf{J}_i$  (4.19)
    end
    Update  $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}^+ \mathbf{b}$  (4.20–4.21)
     $(\mathbf{c}, \mathbf{T}) \leftarrow \mathbf{x}$ 
  end
end
Output:  $\mathbf{c}, \mathbf{T}$ 

```

**Algorithm 1:** Incremental joint self-calibration

solver is set to escape after a small number of iterations. As factors get old, they gradually accumulate more optimization iterations. Very old factors are removed from the graph, thus creating a fixed window  $W$  of factors being evaluated. The overall algorithm is depicted in Alg. 1.

One key advantage of the incremental, windowed algorithm is that it allows to deal with dynamic variations of the estimated parameters. Indeed, in case of changes in the parameters to estimate, the different factors cannot reach a good consensus on the states, and the overall cost increases,

$$F(t) = \sum_{i \in W(t)} \mathbf{r}_i^\top \mathbf{r}_i . \quad (4.22)$$

By monitoring this cost, we are able to detect these changes, and act on the length of the window  $W$  appropriately. A trivial strategy is to reset the window after a significant cost increase. Favored strategies simply reduce the window length dynamically. Reducing  $W$  has a double effect: on one hand, the factors related to the old values are rapidly removed from the problem, thus canceling their adverse effect; on the other hand, shorter windows allow for faster convergence, at the cost of reduced accuracy. Increasing the window length gradually after observing a recovery in the cost allows to dynamically control the trade-off between convergence speed and accuracy.

## 4.6 Experiments

The batch calibration method is evaluated both on simulated and real data. For the simulation experiment, the ROS simulator Gazebo is used with the publicly available simulation of the TIAGo robot. The robot is equipped with a differentially driven base equipped with a SICK LMS561 LRF sensor and encoders on each wheel. The real experiment is also conducted on a TIAGo-base robot. Finally, the online calibration proposal is evaluated in simulation.

**Batch calibration experiment** For this experiment, the simulated robot is manually driven along a non-specific path, sufficiently diverse in motions to cover its kinematic space. The motion  $\widehat{\Delta}^S$  is tracked using a LRF by means of a laser scan matcher algorithm [33]. The precise kinematic parameters are known beforehand, thus the calibration is initialized far from the nominal values to test the viability of the method. From the robot description, the parameters are,

$$\begin{aligned} r_l = r_r &= 0.0985 \text{ m}, \quad d = 0.4044 \text{ m} \\ T_x &= 0.202 \text{ m}, \quad T_y = 0 \text{ m}, \quad T_\theta = 0 \text{ rad} . \end{aligned}$$

They are thus initialized to nominal values  $\mathbf{c} = [0.1, 0.1, 0.4]$  and  $\mathbf{T} = [0.22, 0.1, -0.1]$ .

The results are shown in Fig. 4.4 where the plots show the time evolution of the calibrated parameters. The method is capable of very accurately recovering the vehicle kinematic parameters

$$r_l^* = 0.0985 \text{ m}, \quad r_r^* = 0.0986 \text{ m}, \quad d^* = 0.4064 \text{ m} ,$$

as well as the LRF extrinsics, which converge to

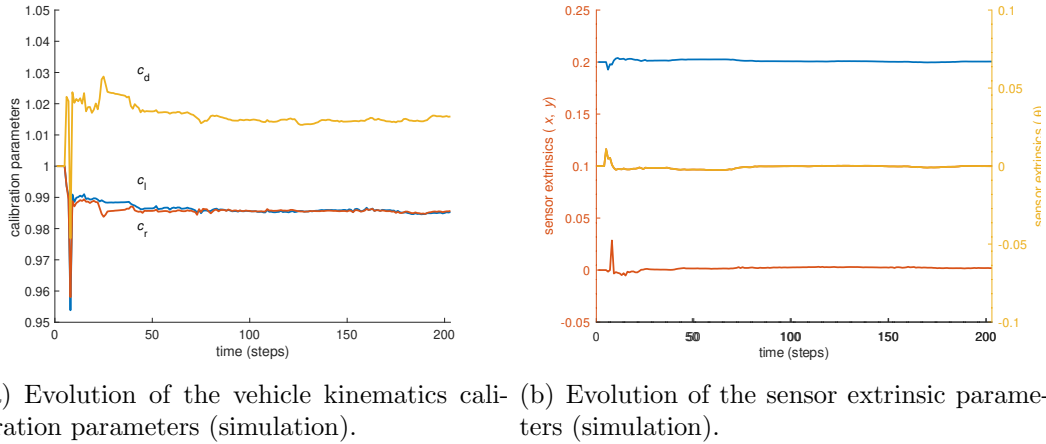
$$T_x^* = 0.2005 \text{ m}, \quad T_y^* = 0.0019 \text{ m}, \quad T_\theta^* = 0 \text{ rad} .$$

A similar setup is used to calibrate the real robot. The nominal values are initialized accordingly to the robot description to  $[0.0985 \text{ m}, 0.0985 \text{ m}, 0.4044 \text{ m}]$  for the differential drive model and  $[0.202 \text{ m}, 0 \text{ m}, 0 \text{ rad}]$  for the LRF extrinsic. The robot is manually driven along a path that covers its kinematic space. The motion  $\widehat{\Delta}^S$  is again tracked using [33].

The results of the calibration are,

$$\begin{aligned} r_l^* &= 0.0986 \text{ m}, & r_r^* &= 0.0978 \text{ m}, & d^* &= 0.4084 \text{ m} , \\ T_x^* &= 0.1975 \text{ m}, & T_y^* &= 0.0024 \text{ m}, & T_\theta^* &= -0.0108 \text{ rad} . \end{aligned}$$

Fig. 4.5 shows the integrated odometry of the robot along a trajectory in an office-like environment before and after calibration. The true initial and final pose of the robot are set to be nearly the same (see Fig. 4.5 middle-top). While the trajectory integrated before calibration clearly drifts, with sections of it going through walls and obstacles and its final pose far from the initial one; the trajectory integrated after calibration is much better, with a final pose very close to the initial one. Note how the improvements of the kinematic parameters provides a much better estimate. The results shown are entirely relying on odometry integration and does not include any loop closure to impose the final pose.



(a) Evolution of the vehicle kinematics calibration parameters (simulation). (b) Evolution of the sensor extrinsic parameters (simulation).

Figure 4.4: Evolution and effects of the batch calibration.

**Online calibration experiment** For this experiment, the simulated robot moves along a trajectory. During its course, the wheel radii slightly decrease to simulate the effect of loading and unloading a freight heavy enough to squeeze the rubber tires. Moreover, the freight is not perfectly centered on the robot, leading to an asymmetrical change of the wheel radii. This change in the differential drive model compels the online adaptation of the calibration parameters  $\mathbf{c}$  for the reported odometry to remain correct. The simulation runs over 500 time steps; the freight loading and unloading takes place at iterations 200 and 300, respectively. During this interval, the left and right wheel radii are decreased by 1% and 0.6%, respectively, so that

$$r_l = 0.0975 \text{ m}, \quad r_r = 0.0979 \text{ m}, \quad d = 0.4044 \text{ m} .$$

The results are reported in Fig. 4.6 and show an adaptation of the calibration as the freight is loaded and unloaded. The online calibration scheme described in Section 4.5.2 is applied both with a fixed size window of 50 factors (Fig. 4.6(a)) and a dynamically sized window (Fig. 4.6(b)) triggered by an increase in the cost (4.22). Apart from highlighting the benefit of a quicker transition of the dynamically sized window method, this comparison allows to better visualize the evolution of  $\mathbf{c}$  and how it effectively changes toward the true value as older factors escape the window.

## 4.7 Conclusion

This chapter detailed a method to jointly self-calibrate the extrinsic parameters of an exteroceptive sensor able to observe ego-motion, and the intrinsic parameters of a differential drive kinematic motion model. An incremental online variant of the method allows to self-calibrate the motion model while it is subject to physical change. The method benefit from a proposed abstraction of the IMU pre-integration theory allows for the application of pre-integration to a simpler case, 2D odometry, obtaining easily self-calibration. The proposal is evaluated in simulation and shown that it converges toward the true values of the parameters. It is then demonstrated to greatly improves the estimated odometry on a real robot. Moreover the online variant is shown to be able to quickly estimate changes of the motion model parameters.

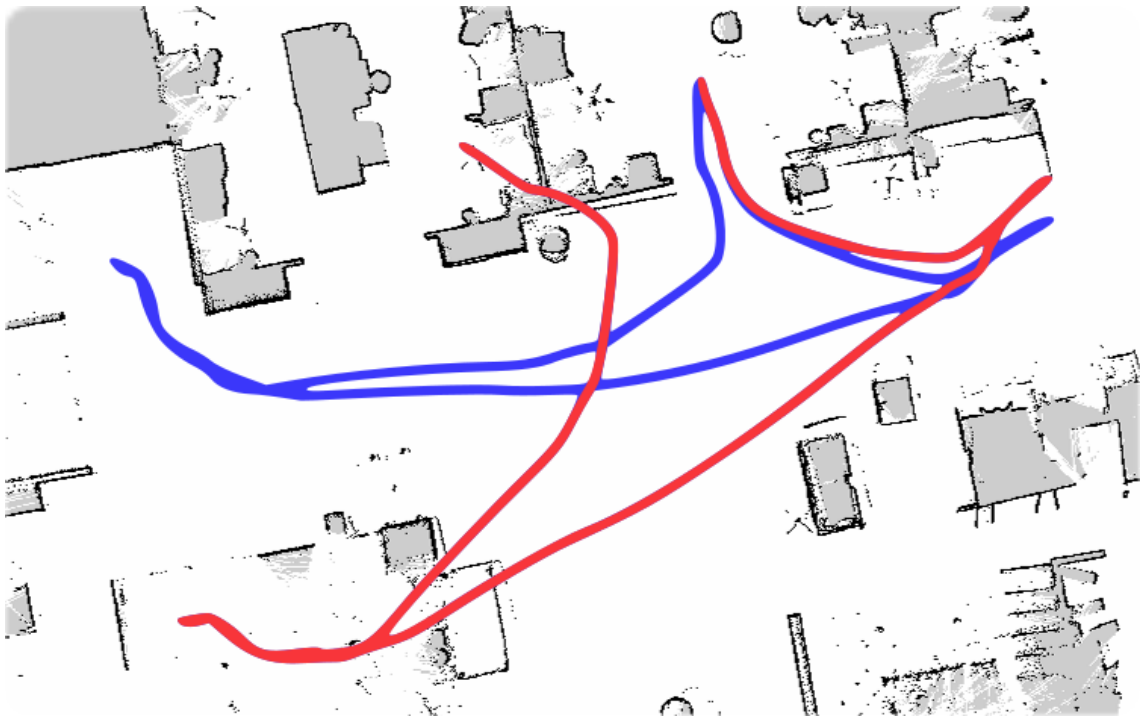
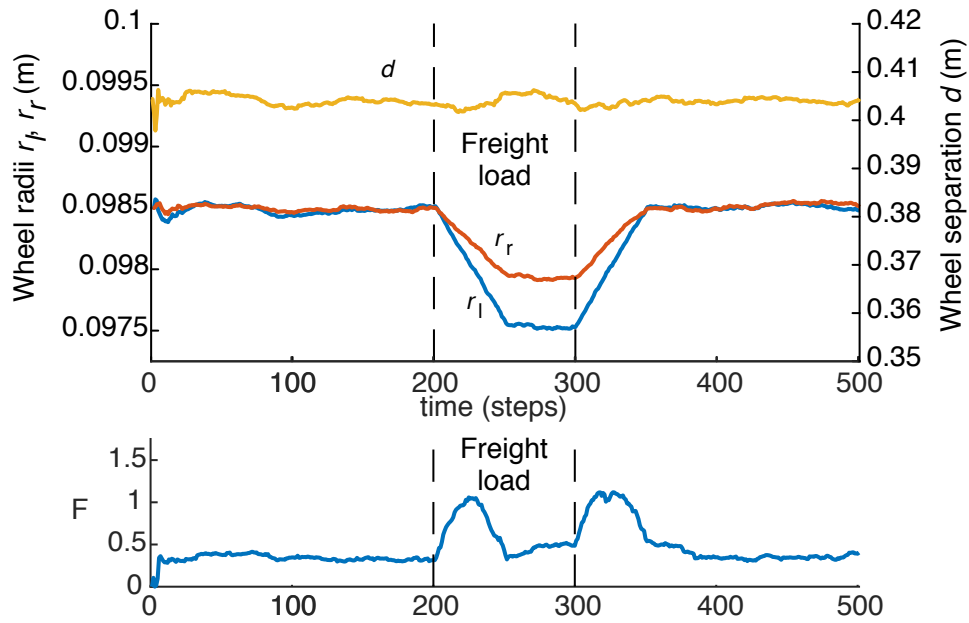
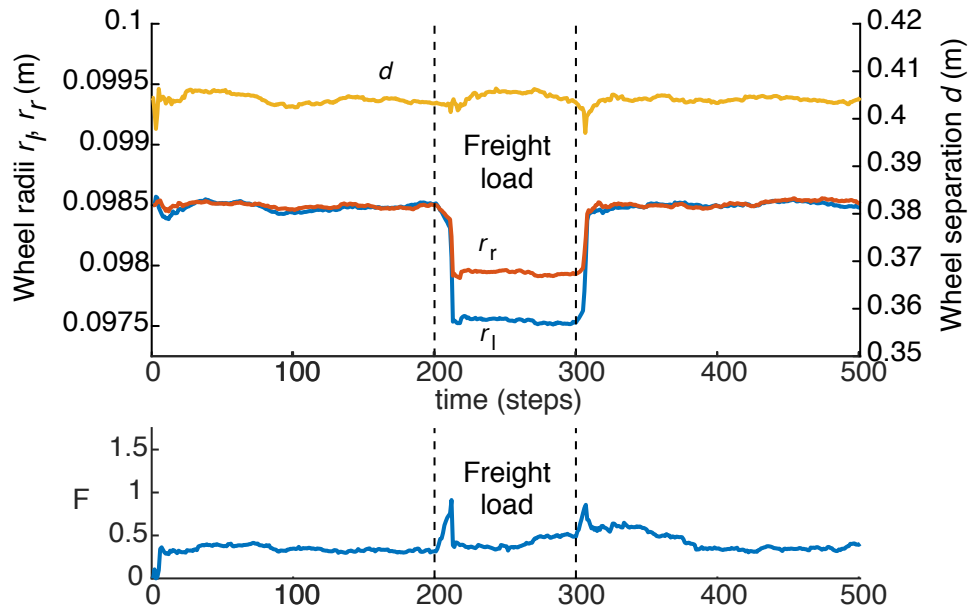


Figure 4.5: Integrated odometry of the real robot before and after calibration - respectively red and blue.



(a) Window size = 50.



(b) Dynamic window size.

Figure 4.6: Comparison of the evolution of vehicle kinematic parameters and the aggregated cost factor  $F$  for a fixed window size and a dynamic one.



# Chapter 5

## Motion Planning

---

*The capacity of planning a safe trajectory is of fundamental importance for autonomous robots as it enables high-level applications in service robotics or autonomous transportation. Not only should a robot be able to move toward a desired configuration, but it shall do it optimally, and most importantly safely. Such planning must happen online in order to react to the dynamic aspects of the environment.*

---

## 5.1 Related work

Differential geometry on manifolds for trajectory optimization has often been used in robotics, especially on Lie groups, with *e.g.* a smooth rigid-body motion on  $SE(3)$  [98] presented as early as twenty years ago. Most methods rely on estimating piecewise-smooth curves, based on piecewise polynomial functions [99–101] B-splines [102, 103], or *Non-Uniform Rational B-Splines* (NURBS) [104, 105]. Other methods model the trajectory using Bezier curves geometrically constructed on manifolds [106]. More recently, Popiel and Noakes [107] presented an extension of the method to construct splines. For either of the later two works, the derivatives do not exhibit the same properties as the Euclidean case such as continuous smoothness at the splines knots.

Splines, specifically with non-vanishing  $n$ -th derivatives are of primary interest as a trajectory backbone as they allow to define constraints such as velocity, acceleration but also jerk limits, which is desirable *e.g.* for autonomous people transportation vehicles in order to improve comfort and prevent motion sickness [108].

The *Timed Elastic Band* (TEB) planner [62, 63] rapidly became one of the most popular local planners for mobile-bases in the ROS community. Based on a TEB formulation (see Section 2.1.4), the TEB-planner allows to efficiently and rapidly estimate a discretized trajectory in the plan. It incorporates constraints such as trajectory feasibility from a robot’s kinematics point of view, together with obstacle avoidance.

## 5.2 Problem formulation

A sparse trajectory planning scheme whose discretization lies on piece-wise  $C^n$  curve on a Lie manifold is presented. The trajectory is considered as a sequence of points lying on a spline on a Lie manifold and exhibit the property of ensuring non-vanishing  $n$ -th derivatives at any point.

Coined *Timed-Elastic Smooth Curve* (TESC), the proposed method builds upon the TEB formulation and differs from the traditional polynomial or spline-based trajectory estimation in that it does not aim at estimating coefficients [102] nor control points which encompass and define the curve [106, 107] but rather a discrete collection of points which themselves lie on the piece-wise curve on a manifold and that forms the trajectory. TESC allows for a seamless calculus of a pose at any time along the TEB.

Formulated as a multi-objective nonlinear optimization problem, the core of the proposed method allows imposing soft constraints such as velocity, acceleration and jerk limits, and more. Additional constraints may easily be added to the core problem, such as obstacle avoidance in a 2D grid for a mobile-base defined in  $SE(2)$ .

### 5.2.1 Timed-Elastic Smooth Curve

#### $C^n$ smooth curve

Considering the TEB as a collection of discrete points, one aims at enforcing consecutive points to lie on a smooth curve in any Lie group, and the different pieces

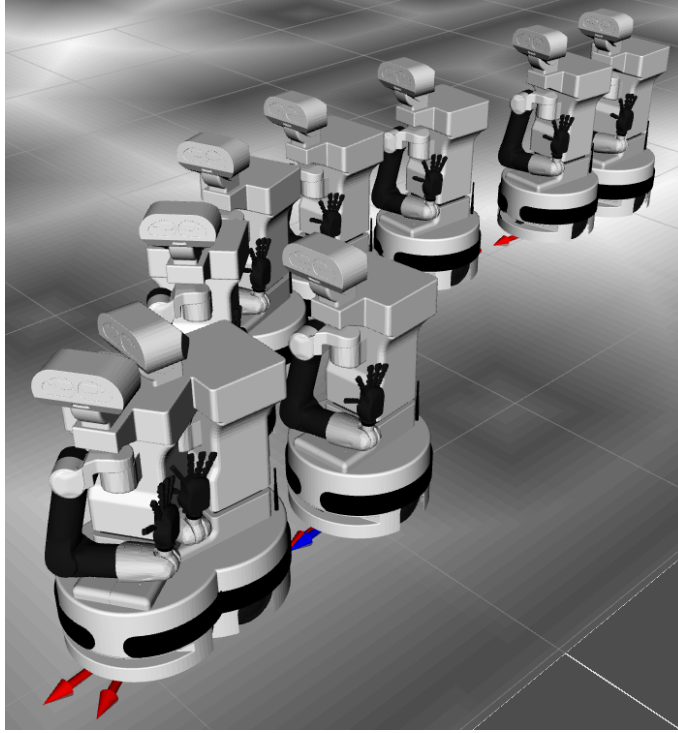


Figure 5.1: Decomposition of TIAGO's motion traversing through a scene with obstacles marked by dark regions on the floor

to form a continuous smooth spline. Jakubiak *et al.* [109] propose a geometric two-step algorithm to generate smooth splines on Riemannian manifolds, in particular Lie groups. Given two configurations  $\mathbf{x}_i$  and  $\mathbf{x}_{i+1}$ , the algorithm allows for interpolating a configuration  $\mathbf{x}_i$  so that it lies on a smooth curve connecting  $\mathbf{x}_i$  and  $\mathbf{x}_{i+1}$ .

The smoothness constraint proposed here results from a revisited formulation of the interpolation algorithm of [109]. Given three consecutive poses  $\mathbf{x}_{i-1}$ ,  $\mathbf{x}_i$  and  $\mathbf{x}_{i+1}$ , and their associated time intervals  $\delta t_i$  and  $\delta t_{i+1}$ , the constraint is as follows. First compute the tangent vectors  $\boldsymbol{\tau}_{i-1}$  and  $\boldsymbol{\tau}_{i+1}$ , which are approximated with backward differences, and the interpolation factor  $s$ ,

$$\boldsymbol{\tau}_i = \mathbf{x}_i \ominus \mathbf{x}_{i-1} \quad (5.1)$$

$$s = \delta t_i / (\delta t_i + \delta t_{i+1}) \in [0, 1] . \quad (5.2)$$

Then compute the interpolated point  $\hat{\mathbf{x}}_i$  that belong to the smooth curve with

$$\mathbf{l}(s) = \mathbf{x}_{i-1} \oplus (s \cdot \boldsymbol{\tau}_{i-1}) \quad (5.3)$$

$$\mathbf{r}(s) = \mathbf{x}_{i+1} \oplus ((s-1) \cdot \boldsymbol{\tau}_{i+1}) \quad (5.4)$$

$$\boldsymbol{\beta}(s) = \mathbf{r}(s) \ominus \mathbf{l}(s) \quad (5.5)$$

$$\hat{\mathbf{x}}_i = \mathbf{l}(s) \oplus (\phi(s) \cdot \boldsymbol{\beta}(s)) . \quad (5.6)$$

The resulting error is then:

$$\mathbf{e}_s = \hat{\mathbf{x}}_i \ominus \mathbf{x}_i , \quad (5.7)$$

where  $\oplus$  and  $\ominus$  operators are given in Section 2.2.5. While the definition of the real valued smoothing function  $\phi(t)$  in (5.6) is given hereafter, the reader can refer to

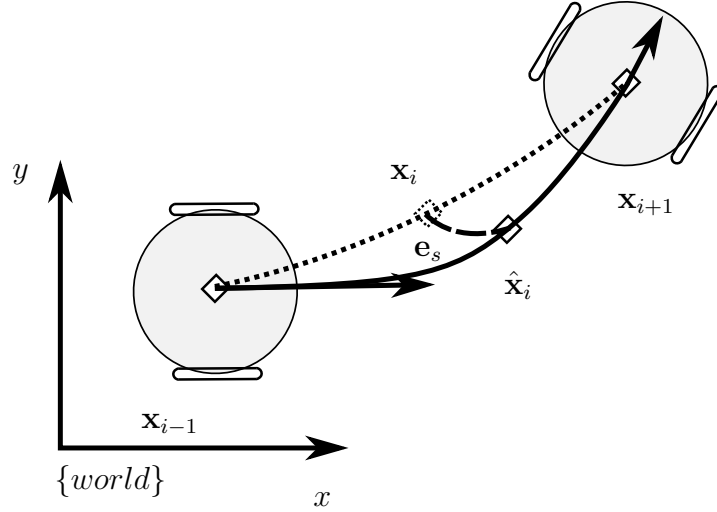


Figure 5.2: The pose  $\mathbf{x}_i$  is constrained towards the smooth curve defined by  $\mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \boldsymbol{\tau}_{i-1}, \boldsymbol{\tau}_{i+1}$  and  $s$  (tangents are illustrated by the arrows).

[109] for more details of its critical guarantees:

$$\phi(s) = \gamma \sum_{j=0}^m \frac{a_{m+1+j}}{m+1+j} s^{m+1+j}, \quad (5.8)$$

with

$$a_{m+1+j} = (-1)^j \binom{m}{j} s^{m+j} \quad (5.9)$$

$$\gamma^{-1} = \sum_{j=0}^m \frac{a_{m+1+j}}{m+1+j}, \quad (5.10)$$

where  $m$  is the smoothness degree ( $C^m$ ).

Fig. 5.2 illustrates (5.3–5.7) for the case of a mobile-base and how the intermediate (middle) pose  $\mathbf{x}_i$  is attracted towards the smooth curve defined by  $\mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \boldsymbol{\tau}_{i-1}, \boldsymbol{\tau}_{i+1}$  and  $s$ .

The trajectory smoothness at segments junctions (knots) is implicitly ensured by design as the final tangent of the  $i$ -th segment is to be equal to the initial tangent of the  $i+1$ -th (consecutive) segment. Therefore, TESC does not require to explicitly impose equality constraints on knots, unlike other spline-based frameworks such as [110].

### Curve's derivatives boundaries constraints

The formulation described in Section 5.2.1 ensures non-vanishing  $n$ -th derivatives at every point, allowing to enforce upper and lower boundaries (*i.e.* inequality constraints) on the curve's derivatives. The derivatives subject to inequality constraints are:

- $\mathbf{v}_k$  the average velocity over a  $\delta t$
- $\mathbf{a}_k$  the average acceleration over a  $\delta t$

- $\mathbf{j}_k$  the average jerk over a  $\delta t$ .

They are approximated using backward finite differencing through a sliding window over the TEB of respectively, the past two, three and four states,

$$\mathbf{v}_i = \frac{\mathbf{x}_i \ominus \mathbf{x}_{i-1}}{\delta t_i} \quad (5.11)$$

$$\mathbf{a}_i = 2 \cdot \frac{\mathbf{v}_i - \mathbf{v}_{i-1}}{\delta t_i + \delta t_{i-1}} \quad (5.12)$$

$$\mathbf{j}_i = 6 \cdot \frac{\mathbf{a}_i - \mathbf{a}_{i-1}}{\delta t_i + \delta t_{i-1} + \delta t_{i-2}} . \quad (5.13)$$

Similarly to [63], the TESC problem is optimized here using a non-linear least-squares solver. Inequality constraints are therefore approximated by two-sided quadratic penalties so that the error functions of an inequality constraint are,

$$e_\nu = \begin{cases} -\nu + \nu^L, & \text{if } \nu < \nu^L \\ +\nu - \nu^U, & \text{if } \nu > \nu^U \\ 0, & \text{otherwise,} \end{cases} \quad (5.14)$$

with  $\nu$  a constrained value,  $\nu^L$  and  $\nu^U$  respectively  $\nu$ 's lower and upper bounds, and  $<$ ,  $>$  are element-wise comparisons. From (5.11–5.14) results the following error functions:

$$\mathbf{e}_v \text{ subject to } \mathbf{v}^L < \mathbf{v}_i < \mathbf{v}^U \quad (5.15)$$

$$\mathbf{e}_a \text{ subject to } \mathbf{a}^L < \mathbf{a}_i < \mathbf{a}^U \quad (5.16)$$

$$\mathbf{e}_j \text{ subject to } \mathbf{j}^L < \mathbf{j}_i < \mathbf{j}^U . \quad (5.17)$$

### Minimizing time and trajectory length

Not only shall a trajectory be feasible but also should it be as short as possible, both in terms of execution time and traveled distance. While a double objective of distance and time, *i.e.*,  $c = w_l \langle \boldsymbol{\tau}, \boldsymbol{\tau} \rangle + w_t \delta t^2$  seems natural, experiments shown that having them enforce each other gives better results in terms of smoothness and stability of the solutions in the optimization framework used here. The joint length-time error is defined as,

$$e_l = \langle \boldsymbol{\tau}_i \delta t, \boldsymbol{\tau}_i \delta t \rangle = \langle \boldsymbol{\tau}_i, \boldsymbol{\tau}_i \rangle \delta t^2 , \quad (5.18)$$

where  $\langle \boldsymbol{\tau}_i, \boldsymbol{\tau}_i \rangle$  is the arc-length of the  $i$ -th segment squared. While this cost function has no particular physical meaning, it is preferable for a numerical optimization process. Having two different cost functions for time and length, each having their own weight, leads to optimizing  $2(n - 1)$  unique cost functions, each one having a fairly large residual with respect to the other cost functions. This is so since, *e.g.*, time will not reduce as much as the distance to the final target. Instead, by merging those two cost function into (5.18), there are only  $n - 1$  unique cost functions to minimize, a single weight to tune and a residual of a similar order as the others.

### Distance to the target configuration

While the previous cost functions constrain the overall shape of the trajectory, this cost function pulls the TEB's tip,  $\mathbf{x}_n$ , toward the desired configuration – or *goal* –,  $\mathbf{x}_g$ . It is defined as follows:

$$\mathbf{e}_g = \mathbf{x}_g \ominus \mathbf{x}_n . \quad (5.19)$$

## 5.3 TESC for mobile-base motion planning

Building upon the work of Rösman *et al.* [62, 63] the TESC formulation is extended to the case of mobile-base motion planning. The Lie group abstraction is thus dissipated here by considering  $\chi \in \text{SE}(2)$ . All of the cost functions defined in Section 5.2 remains valid while new ones, specific to the task at hand, are added.

### 5.3.1 Mobile-base specific constraints

#### Non-holonomic constraints

Since the trajectory optimized is that of a mobile-base, one has to take the kinematic constraint into account so that the trajectory is physically feasible. In the case of an omnidirectional base (*e.g.* using mecanum wheels), no further constraints are required. Considering a differential or bicycle-like base, the first of such kinematic constraints aims at enforcing non-holonomy since such base cannot move sideways. It is imposed as,

$$e_h \text{ subject to } \mathbf{v}_{yi} = 0 , \quad (5.20)$$

with  $\mathbf{v}_{yi}$  the y-component of the velocity vector computed from (5.11). Given the optimization framework employed, equality constraints are obtained by setting both the lower and the upper bounds to the same value.

#### Minimum turning radius constraints

If the mobile-base considered is a bicycle-like model (*e.g.* car-like), one has to ensure a minimum turning radius. The equivalent condition is implemented on the inverse radius, since unlike  $R$ ,  $1/R$  crosses zero continuously as the robot transitions from a left turn to a right turn,

$$1/R = \mathbf{v}_{\omega i} / \mathbf{v}_{xi} \quad (5.21)$$

$$e_r \text{ subject to } -1/R_{min} < 1/R < 1/R_{min} , \quad (5.22)$$

with  $\mathbf{v}_{xi}$  and  $\mathbf{v}_{\omega i}$  respectively the x- and the angular-components of the velocity vector (5.11). For  $\mathbf{v}_{xi} \rightarrow 0$ ,  $\mathbf{v}_{\omega i}$  is constrained to zero in a way akin to (5.20) to avoid turning in place.

#### Obstacle avoidance

When dealing with mobile-base navigation, the robot's surrounding environment is often represented by a 2D *Occupancy Grid* (OG) (Fig. 5.3(a)). In its simplest ternary form, the OG has three distinct values denoting whether a cell is free (no obstacle), occupied (obstacle) or unknown. In order to avoid obstacles, the TESC

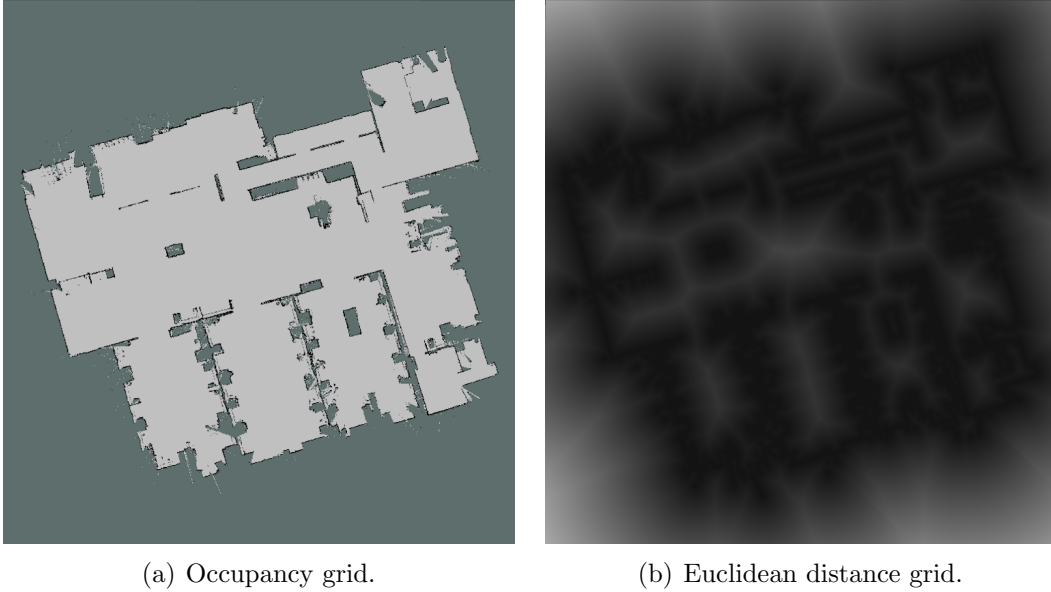


Figure 5.3: Ternary occupancy grid and Euclidean distance grid of a mapped office. Obstacles are shown in black on both figures.

planner represents the environment by means of an *Euclidean Distance Grid* (EDG) (Fig. 5.3(b)) which stores in each cell the distance to the closest obstacle in the grid. Cells representing an obstacle have a zero value. Such representation allows to efficiently evaluate whether a configuration is in collision with an obstacle or not. Given a 2D-OG, an EDG is computed using the distance transform algorithm described in [111]. This algorithm is of first choice as it is fast, efficient and computes an exact Euclidean distance. Given a distance grid and two consecutive poses  $\mathbf{x}_i$  and  $\mathbf{x}_j$  along the TEB, the obstacle avoidance constraint is computed as follows. First,  $k$  poses are interpolated between  $\mathbf{x}_i$  and  $\mathbf{x}_{i+1}$  as per [109], with  $k$  chosen in adequacy with the grid resolution. Then the EDG cells' distance corresponding to each of the  $k + 2$  poses are evaluated. The resulting error functions is

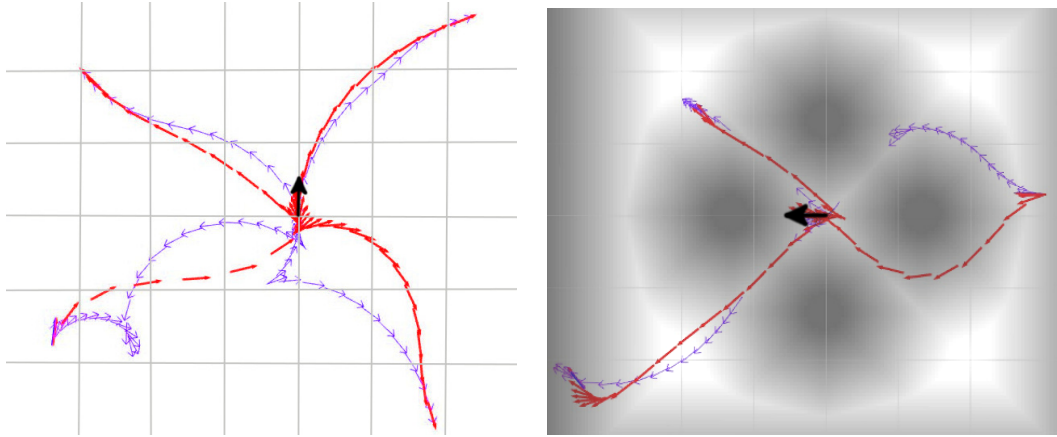
$$e_o = \begin{cases} r - d, & \text{if } d \leq r \\ 0, & \text{otherwise,} \end{cases} \quad (5.23)$$

with  $d$  the smallest distance evaluated over the  $k + 2$  poses and  $r$  the radius of the circle encompassing the robot footprint. Notice that complex footprint shapes can be considered and is left here to further work. Evaluating intermediate interpolated configurations allows to assert that a segment as a whole is obstacle free avoiding the common problem in discrete trajectory planning of having consecutive poses lying on each side of an obstacle.

### 5.3.2 Experiments

This section describes the experimental setup along with results from simulations in three different scenarios using the TIAGo simulation. The proposed TESC approach is compared against the state-of-the-art TEB planner [63].

Specifically, each planner is queried a 1000 times for each scenario and results



(a) Obstacle-free scenario, collage of four different goals. (b) Four obstacles scenario, collage of three different goals.

Figure 5.4: Experiment setups. TEB-planner is depicted with violet arrow while TESC is with red arrows. The initial pose at the center of the grid is depicted with a larger black arrow.

were compiled from these trials. To achieve a fair comparison TESC and TEB-planner are initialized with the same velocity and acceleration limits.

The evaluation covers eight metrics. First is the success rate, which indicates whether the planner has found a collision-free trajectory or not. The optimization time measures how much time the planner took to find a trajectory. As the robot operates in the real environment it is important to be able to find plans in a timely manner, especially for reactive control applications. The trajectory arc-length and the trajectory time show how much the robot has to move to reach the goal and the time it takes to do so. The average velocity and acceleration metrics encompass both their linear and angular components. Finally the energy is an approximation of the kinetic-energy, while the trajectory curvature highlights the smoothness of the trajectory's curve. Smoother trajectories require less acceleration/deceleration, therefore putting less stress on the mechanical parts of the robot. Moreover they allow people to feel safer in the robot's vicinity as it exhibit a more predictable behavior. The aforementioned height metrics are detailed in Table 5.1. Note that the curvature metric is approximated as the sum of acceleration's norm in the global reference frame.

The weight factors  $w_k$  for the TESC planner are empirically determined so that the costs  $w_k c_{ki}(\mathbf{Q}_i, \Delta \mathbf{T}_i)$  are all of the same order of magnitude. Similarly to the observations made in [63], experiments showed that the weights associated to both the kinematics (Section 5.3.1) and the goal (Section 5.2.1) must be an order of magnitude higher than other weights. The weight factors used for the TEB-planner are those presented as optimal in the original paper.

### Implementation Details

The TESC planner is implemented in C++ using the least-squares solver **Ceres** [112] as it is flexible and offers automatic-differentiation. It also relies on the **manif** library Chapter 6, a Lie-theory library for state-estimation. All the scenarios are simulated using ROS and TESC has been integrated within the ROS navigation stack so that



Table 5.1: Metrics used in the experiments evaluation.

Metric	Description
Success rate in %	$100 \cdot \frac{\text{success}}{\text{success} + \text{failure}}$
Planning time in s	—
Trajectory arc length	$\sum \ \mathbf{x}_i \ominus \mathbf{x}_{i-1}\ $
Trajectory time (s)	$\sum \delta t_i$
Average velocity	$\text{mean}(\ \mathbf{v}_i\ )$
Average acceleration	$\text{mean}(\ \mathbf{a}_i\ )$
Energy	$\sum \frac{\ \log(\mathbf{x}_i^{-1} \cdot \mathbf{x}_{i-1})\ }{2 \cdot \delta t_i^2}$
Trajectory curvature	$\sum \ 2 \cdot \frac{\log(\mathbf{x}_i^{-1} \cdot \mathbf{x}_{i-1}) - \log(\mathbf{x}_{i-1}^{-1} \cdot \mathbf{x}_{i-2})}{\delta t_i + \delta t_{i-1} + \delta t_{i-2}}\ $

its outputs are directly applicable to real robots using `ros_control` [113]. The source code of the planner is publicly available <sup>1</sup>.

All experiments are conducted on a system having an Intel i7-4702HQ, 2.2 GHz CPU, 8 GiB of RAM and running on Ubuntu 16.04.

### Obstacle-free Environment

In the first scenario, the robot is located at the center of an  $8 \times 8$  m grid with no obstacles and has to plan a trajectory toward a randomly generated pose. It is illustrated in Fig. 5.4(a) for four different goals. Statistics of the several metrics were summarized in Fig. 5.6, columns associated with this experiment are marked with the **OF** suffix. As Figs. 5.6(a)–5.6(b) show, the optimization time of both planners in this environment is well within realtime-capable requirements. Both planners performed very similar trajectory arc length, with a short advantage for TESC (Fig. 5.6(c)). Trajectory time as shown in Fig. 5.6(d) are also very similar. For the average velocity metric shown in Fig. 5.6(e), TESC has slightly higher values than TEB-planner but has much lower acceleration average (Fig. 5.6(f)). Finally, TESC outperforms TEB-planner for both the total energy cost and the trajectory curvature shown respectively in Fig. 5.6(g) and Fig. 5.6(h). These first results highlight that the smooth properties of the TESC approach improve the quality of the generated trajectories.

### Synthetic Obstacles Environment

The second environment is a  $8 \times 8$  m grid with four circular obstacles surrounding the robot. Goals are randomly generated and their feasibility is verified. Fig. 5.4(b) depicts this scenario for three different runs. Statistics of the several metrics were summarized in Fig. 5.6, columns associated with this experiment are marked with **O** suffix.

With the increase in challenge, the difficulty is reflected by both a decrease of success rates, as well as an increase of execution time. As Fig. 5.6(a) depicts, TEB-planner and TESC only succeed planning in 83% and 61% of the time respectively. TESC optimization time increases largely while TEB-planner’s remains fairly stable

<sup>1</sup>[https://github.com/artivis/tesc\\_planner](https://github.com/artivis/tesc_planner)

as visible in Fig. 5.6(b). Both planners perform in a similar manner in terms of trajectory time. Observations for the trajectory arc-length and time are similar to those in Section 5.3.2. With a lower average velocity (Fig. 5.6(e)) and a much smaller acceleration (Fig. 5.6(f)) TESC produces trajectories not only smoother but less prone to cause motion sickness and wear out vehicle hardware while consuming less energy (Fig. 5.6(g)). Note how even the deviation depicted in both Figs. 5.6(g)–5.6(h) is much smaller for TESC than TEB-planner, signifying consistent better results.

### Office Environment

The third experiment considers a more realistic scenario relying on the OG generated from a simulated small office. The environment is of size  $10, 2 \times 14, 85$  m, constituted of two distinct rooms connected by an open door, both filled with furniture such as shelves and tables. Once again goals are generated randomly while their feasibility is verified. Statistics are presented in Fig. 5.6, columns associated with this experiment are marked with **SO** suffix.

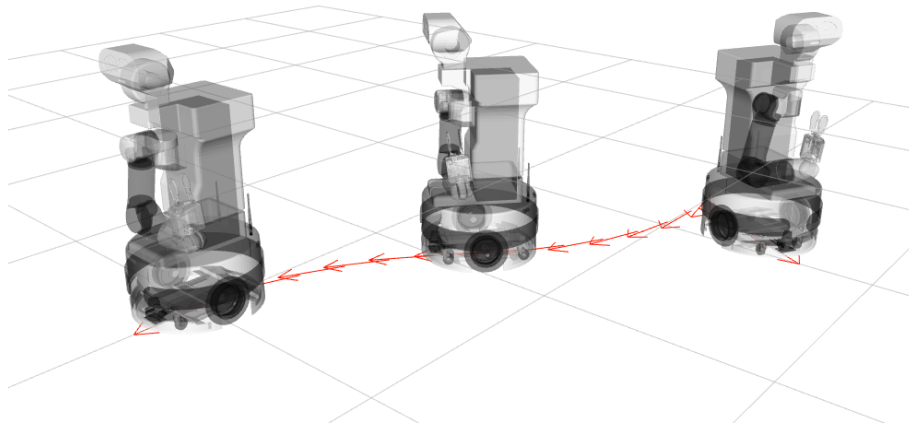
Unlike previous experiments, the success rate of both planners is much closer to one another with 77% and 71% respectively for TEB-planner and TESC. However the optimization time of TESC slightly increases again while TEB-planner’s remains fairly stable. Trajectory length and time for both planners show the same trend as for previous experiments. In the same manner TESC shows once again a smaller average velocity (Fig. 5.6(e)) and a much smaller acceleration (Fig. 5.6(f)) than TEB-planner, but also smaller and more consistent energy (Fig. 5.6(g)) and curvature (Fig. 5.6(h)).

### 5.3.3 Conclusion

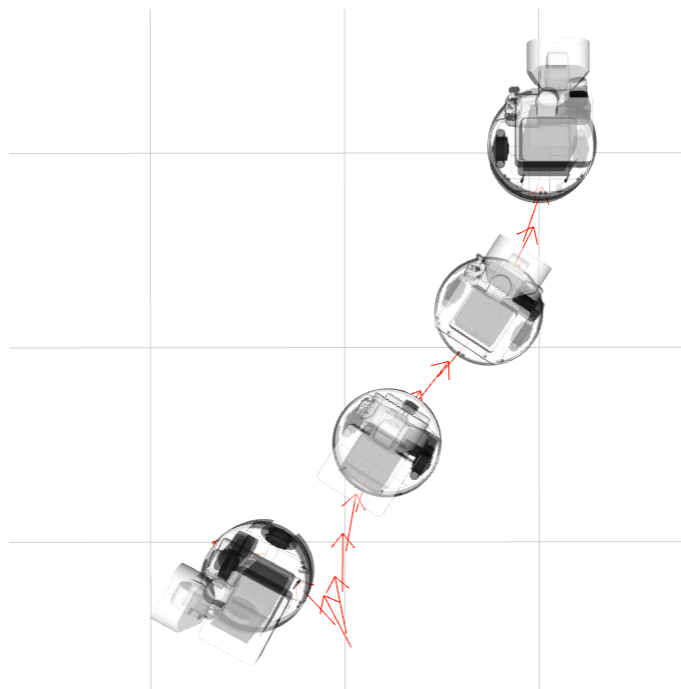
This Section presented a novel formulation, *Timed-Elastic Smooth Curve* (TESC), for  $C^n$  smooth trajectory optimization. The generated trajectory’s curve has non-vanishing  $n$ -th derivatives, allowing to constrain velocity, acceleration, jerk, *etc.*, with ease. Moreover, the continuity of the curve at its knots is ensured by design, which relieves from the addition of extra cost functions to do so. While relying on a discrete set of points, its formulation allows for interpolating points that do belong to the trajectory curve. This property allows *e.g.* to ensure that the whole trajectory is collision-free. The proposed TESC is benchmarked in a series of mobile base motion planning scenarios and is shown to prevail or matches the performance of the TEB-planner in most presented metrics. TESC has also proven to be more consistent in the quality of the generated trajectories.

However challenging, these experimental scenarios put both planners to the test. TESC could not uniformly prevail in all metrics, likely due to the environment representation in use. The *Euclidean Distance Grid* (EDG) discretization and unsigned-ness are both leading to discrete and possibly non-existing gradients, thus to optimization issues.

Experiments have shown that TESC is planning trajectories that are smoother and more energy-efficient than TEB-planner. TESC’s trajectories are of the same length but with a smaller average velocity and acceleration at the cost of increasing the optimization time.



(a) Side view.



(b) Top view.

Figure 5.5: Decomposition of maneuvers.

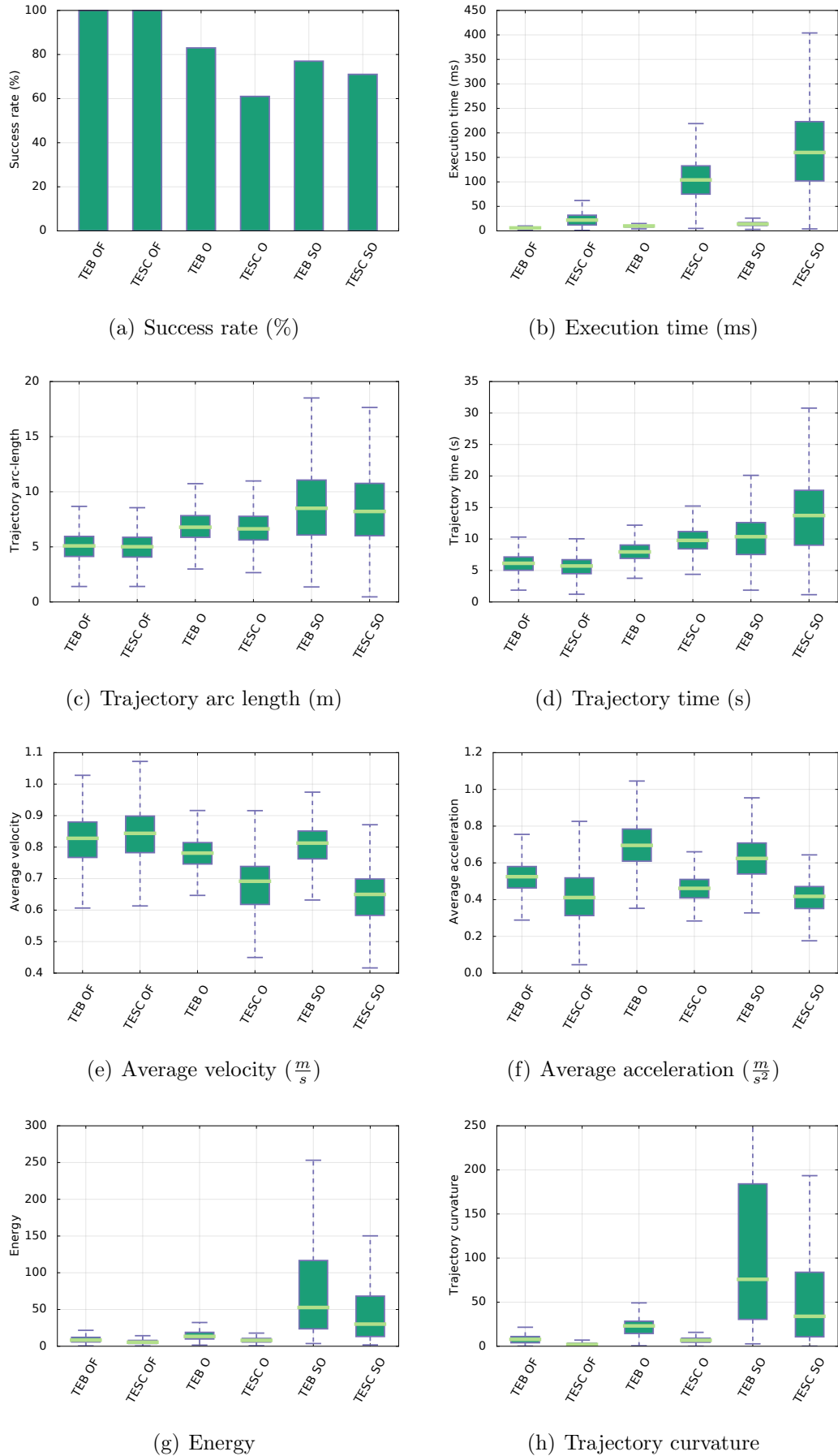


Figure 5.6: Experiment results of both planners in each of the three scenarios.

## 5.4 TESC for manipulation motion planning

Following the simpler case of a 2D mobile-base motion planning in Section 5.3, this section highlights the versatility of the concepts supporting the TESC planner and describes its extension to the case of motion planning for a robotic arm, thus in  $SE(3)$ . The task of motion planning for manipulation is more challenging in that the kinematics involved are more complex. Rather than controlling two wheels, the planner has to handle the many joints of an arm forming a kinematic chain. The extension of the TESC planner described hereafter is coined *Timed Elastic Bands for Manipulation Motion Planning* (TEB2MP).

### 5.4.1 Related work

Even though sampling-based methods are able to find (near-)optimal paths (see Section 2.1.4), the solutions of such method can potentially violate constraints imposed by the robot hardware. Indeed, the solution may require motions that are not doable from a kinematics point of view. Recent extension to RRT\* includes kinematics considerations for simple models such as a mobile-base [60]. However, to the best of the author knowledge, no sampling-method is able to handle the complicated kinematics chains of a robotic arm. In order to incorporate such constraint, one has to rely on optimization techniques.

A prominent line of work for this type of motion planners is *Covariant Hamiltonian Optimization for Motion Planning* (CHOMP) [114] and its variants [115, 116]. These methods optimize the cost function using covariant gradient descent. Later introduced, *Stochastic Optimization for Motion Planning* (STOMP) [117] performs optimization on non-differentiable constraints by drawing samples stochastically from a set of noisy trajectories. The main limitations of these methods is that they can potentially be very computationally expensive as they require a finely discretised trajectory to check if it is collision free. They may also fail to converge on even moderately difficult problems. Finally, the performance of STOMP is heavily dependent on the tuning of parameters used for noise generation for a given problem. The parameters fine tuning producing good results on a problem will, most of the time, performs badly on a different problem.

To avoid the potential computational complexity of CHOMP and STOMP the *Trajectory Optimization for Motion Planning* (TrajOpt) method was introduced in [118]. It formulates the optimization problem as a *Sequential Quadratic Programming* (SQP) problem with continuous-time collision checking. The reduced computational cost arise from exploiting the sparsity of the problem, where the trajectory considered is a coarse discretization of the actual continuous trajectory. In addition, the SQP formulation allows for imposing hard constraints such that the produced plan is guaranteed to respect them. TrajOpt was also proven versatile in a series of different tasks presented in [119].

The work of [120] proposes a joint-space trajectory representation using Reproducing Kernel Hilbert Spaces. Using such representations is strongly motivated as it turns the reasoning about smoothness, described by acceleration, jerk, snap, *etc.*, trivial. However, similarly to STOMP, the large number of parameters and the strong dependence of the generated results to their fine tuning make this solution very difficult to apply across different problems or even to replicate the results from

[120]. Nevertheless the method is shown to perform better than CHOMP.

More recently, [121] introduced a method for motion planning using *Gaussian Process* (GP). The trajectory is represented as a continuous valued function that maps time to robot states and its optimization is performed using probabilistic inference. A GP is used to provide a prior function that encourages smoothness while a likelihood function encourages collision free trajectories. The posterior distribution of the GP prior together with the likelihood function are used to calculate the *Maximum a Posteriori* (MAP) estimate of the trajectory. Coined *Gaussian Process Motion Planner 2* (GPMP2) the method provides comparable success rates with the state-of-the-art while requiring much less computational efforts. It however has several limitations. Joint limits are not explicitly taken into account as they are only clamped to their maximum values in the optimization initialization. An estimated trajectory may therefore violates the kinematics constraints. Similarly the trajectory smoothness is only encouraged using a prior having no acceleration. Finally, GPMP2 does not minimize the trajectory length in either Euclidean or joint space so that any trajectory, may it be very convoluted, can be a solution.

### 5.4.2 A different take on Timed-Elastic Band

Unlike the TEB framework described in Section 2.1.4, the states of the TEB employed here do not lie on a manifold but are rather represented in the *joint-space*. A joint-space trajectory is defined as,

$$\mathbf{Q} = \{\phi_i\}_{i=1\dots n} , \quad (5.24)$$

where  $\phi$  is a full joint state of the kinematic chain,

$$\phi = \{\vartheta_j\}_{j=1\dots J} . \quad (5.25)$$

Each of its entries  $\vartheta$  represents the state of an individual joint. Depending on the type of joint, a field of  $\phi$  corresponds to either,

- an angle (in radians) for revolute joints
- a distance (in meters) for linear joints

In this work, only these two simple joint types are considered.

The mapping from joint-space to Cartesian space is achieved by means of *Forward Kinematics* (FK) for any joint. The inverse mapping is called *Inverse Kinematics* (IK). FK and IK problems are beyond the scope of this thesis and are not further detailed. They are considered tools to map back and forth and are extensively used here – *e.g.* FK mapping is applied prior to the computation of any constraint described in Section 5.2.1. As an example, the distance to target configuration cost function (5.19) would look like,

$$\mathbf{e}_g = \text{FK}(\phi_g) \ominus \text{FK}(\phi_n) . \quad (5.26)$$

This allows the arm motion planning pipeline to operate with constraints defined in either of two complimentary spaces, the task-space and the joint-space.

### 5.4.3 Arm specific constraints

#### Joint-space limits

Similarly to Section 5.2.1, boundaries are imposed upon the velocity and acceleration of each joint in the joint-space. Furthermore, limits are also imposed on their position to reflect the real limit of the hardware, *e.g.* minimum and maximum translation of a linear joint.

The joint position limits are inspired by [122](83). It is adjusted so that the error cost is  $\in [0, 1]$  where 0 denotes that the joint is within the limits and 1 when outside,

$$\mathbf{e}_{l,j} = \begin{cases} 0, & \vartheta_j \in [\vartheta_j^L, \vartheta_j^U] \\ \exp\left(\frac{(\vartheta_j - \vartheta_j^L) * (\vartheta_j^U - \vartheta_j)}{(\vartheta_j^U - \vartheta_j^L)^2}\right) & \text{otherwise,} \end{cases} \quad (5.27)$$

where  $\vartheta_j^L$  and  $\vartheta_j^U$  are respectively the lower and upper joint limits of  $\vartheta_j$ .

Velocity and acceleration constraints are similar to those of Section 5.2.1,

$$\dot{\phi}_i = \frac{\theta_i - \theta_{i-1}}{\delta t_i} \quad (5.28)$$

$$\ddot{\phi}_i = 2 \cdot \frac{\dot{\phi}_i - \dot{\phi}_{i-1}}{\delta t_i + \delta t_{i-1}}. \quad (5.29)$$

Using (5.14), their respective errors are then,

$$\mathbf{e}_{\dot{\phi}} \text{ subject to } \dot{\phi}^L < \dot{\phi}_i < \dot{\phi}^U \quad (5.30)$$

$$\mathbf{e}_{\ddot{\phi}} \text{ subject to } \ddot{\phi}^L < \ddot{\phi}_i < \ddot{\phi}^U. \quad (5.31)$$

#### Collision avoidance

For a robotic arm, the collision avoidance not only must prevent collisions with the environment but also with its own body, also known as self-collisions. Furthermore, whereas collisions were only checked for a 2D space in Section 5.3.1, the space considered here is now 3D, increasing both the complexity and most importantly the computational effort. To mitigate this effort, the robot is modeled by an *Axis-Aligned Bounding Box* (AABB) representation allowing to efficiently detect collisions using the **Flexible Collision Library** [123]. This method efficiently approximates complex shapes using bounding box segments in otherwise expensive computation.

The resulting error function is,

$$\mathbf{e}_o = V(\phi) + \begin{cases} 0, & \Gamma(\phi) < d^L \\ -\Gamma(\phi) + d^L & \text{otherwise,} \end{cases} \quad (5.32)$$

where  $V(\phi)$  is the overlapping volume of the robot at state  $\phi$ ,  $\Gamma(\phi)$  is the distance to the closest collision point for all joints  $\vartheta_j \in \phi$  and  $d^L$  is the minimum distance to keep from obstacles.

Table 5.2: Metrics used in the experiments evaluation.

Metric	Description
Planning time in s	$\tau$
Success rate in %	$s_r$
Smoothness in $rad/s^2$	$\ddot{\phi}$
Distance from joint limits	$\frac{\sum_{i=1}^n \sum_{j=1}^m \frac{\vartheta_{i,j} - \vartheta_{i,j}^{\min}}{\vartheta_{i,j}^{\max} - \vartheta_{i,j}^{\min}}}{n \cdot m}$

#### 5.4.4 Experiments

Experiments were conducted in simulation using the TIAGo mobile manipulator robot in three different scenarios. For each scenario, the TEB2MP planner is compared against state-of-the-art planners, STOMP, TrajOpt and RRTConnect which serves as a baseline for sampling-based methods.

Given the probabilistic nature of some planners, multiple trials are performed for each scenario. Specifically, each planner is executed a 100 times for each scenario and statistical results were compiled.

STOMP, TrajOpt and TEB2MP planners are each initialized by linearly interpolating intermediate joint states between the initial and final configuration. TEB2MP is also evaluated with a cubic polynomial interpolation initialization in order to highlight the effect of different initialization on the proposed method.

Additionally, the trajectory size of the planners STOMP, TrajOpt and TEB2MP are set to be equal to 20 for fair comparison.

The evaluation covers four metrics. First is the success rate, which indicates whether the planner has found a collision-free trajectory or not. Second, the optimization time, measures how much time the planner took to find a trajectory. The third metric is the smoothness of the generated trajectory in  $rad/s^2$ . Smoother trajectories require less acceleration or deceleration, therefore putting less stress on the mechanical parts of the robot. It must be noted that for the smoothness metric, the lower the score the better. Finally, the fourth metric is the distance from the joint limits for each joint. Being further away from the joint limits is beneficial as plans that are close to the joint limits may be harder to execute due to hardware lock-in or precision errors. In addition, operating close to the joint limits may have a negative effect in the manipulability [122] and the ability to find new plans. The aforementioned four metrics are detailed in Table 5.2.

#### Implementation details

The TEB2MP planner has been implemented in C++ using G2o [124] for solving the nonlinear least-square problem and the `manif` library Chapter 6 for the Lie theory aspect. Experiments are carried in simulation using ROS and the `MoveIt!` framework [125]. The output of each planner can be used to control directly the real robot using `ros_control` [113]. All experiments are conducted on a system having an Intel i7-4710MQ, 2.5 GHz CPU, 8 GiB of RAM and running on Ubuntu 16.04.



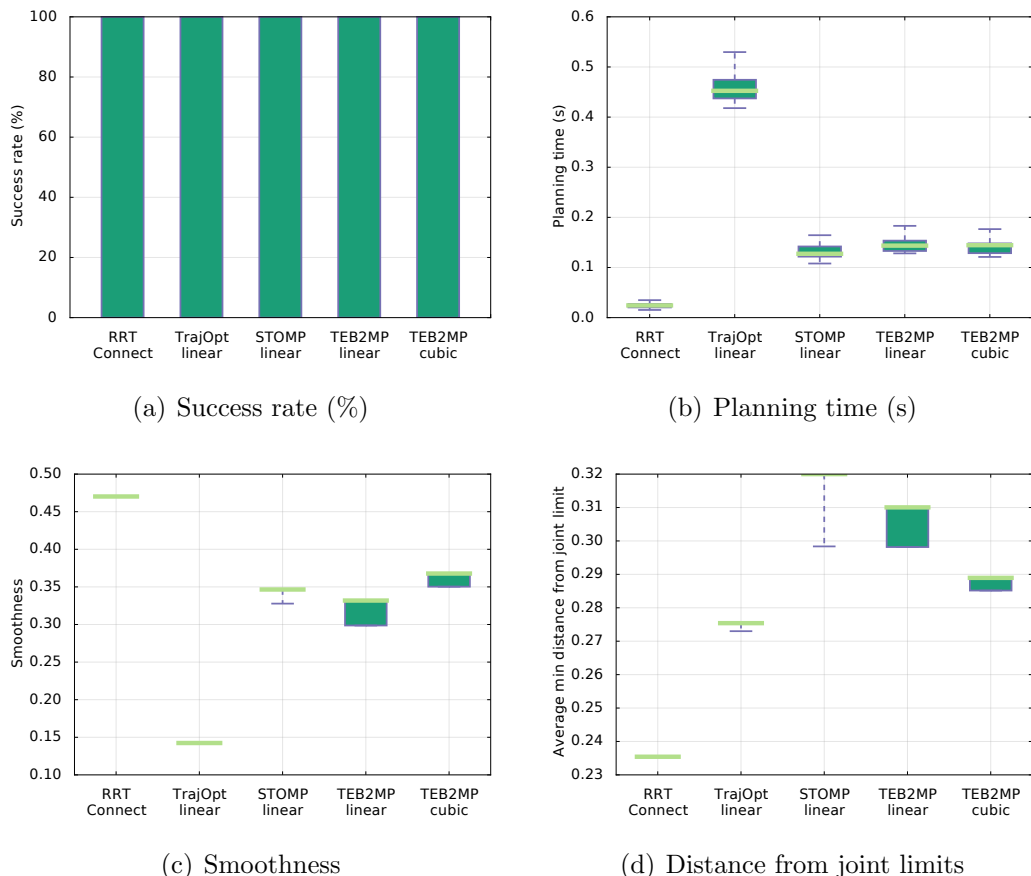


Figure 5.7: Experiment metrics for the evaluation of TEB2MP in the empty environment.

### Obstacle-free environment

For the first scenario, the robot is placed in an empty environment (*e.g.* no obstacles) where it has to move its end-effector from a predetermined initial pose to a predetermined final one. This setting is essentially the same as Fig. 5.9(a) without the green box.

Statistical results of the several metrics for the various tested planners are summarized in Fig. 5.7. One can see from Fig. 5.7(a) that each planner succeeds to find a trajectory each and every time, an expected results as the environment is empty and the goal feasible. However their respective execution time to do so varies largely (Fig. 5.7(b)). RRTConnect is by far faster than all the other planners. On average STOMP and TEB2MP are 6 times slower. TrajOpt is by far the slower planner, being in average 2.5 times slower than TEB2MP. For the smoothness metric, TrajOpt outperforms every other planners. STOMP and TEB2MP perform equally good and better than RRTConnect. At last, STOMP performs best on the joint distance metric, followed by the two versions of TEB2MP. TrajOpt ranks fourth, just behind cubic TEB2MP, and RRTConnect performs the worst of all.

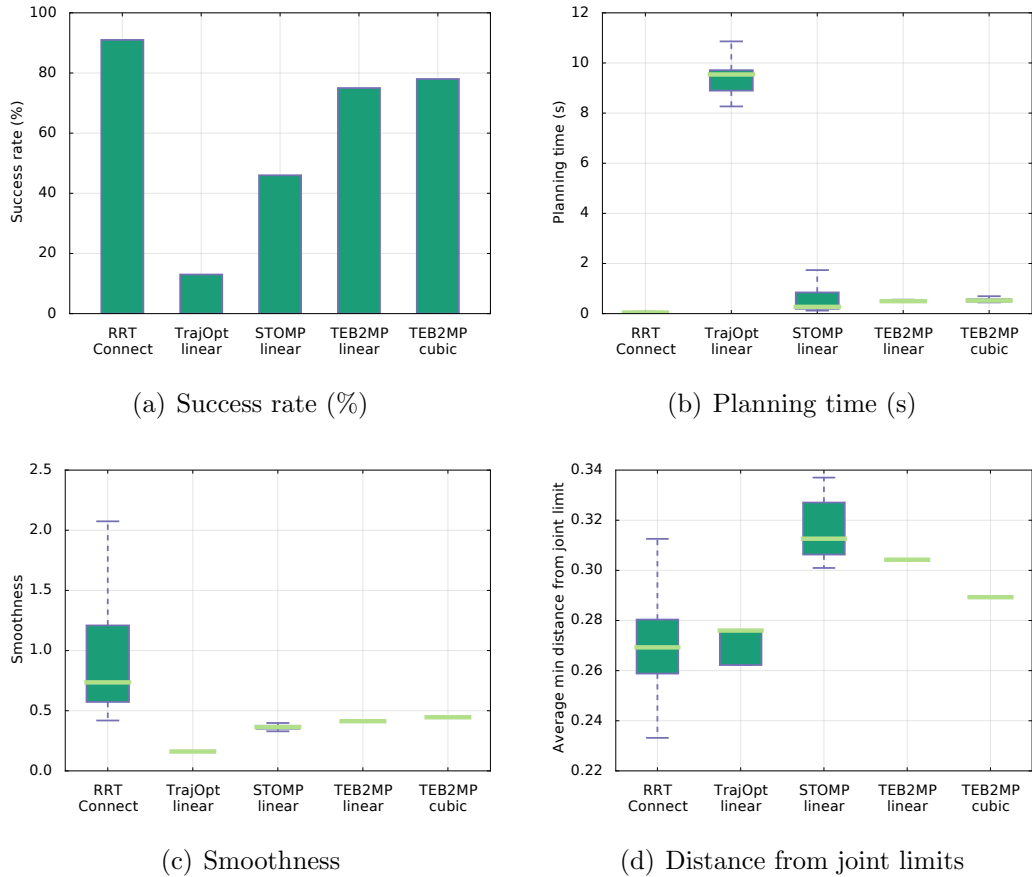


Figure 5.8: Experiment metrics for the evaluation of TEB2MP in the box environment.

### Box environment

The box scenario is similar to the empty one (Section 5.4.4) in that the robot has to move its end-effector from a predetermined initial pose to a desired final one. The main difference is that an obstacle, a large box, is placed along the trajectory. The setting is pictured in Fig. 5.9(a).

Statistical results for this experiment are summarized in Fig. 5.8. Repeating previous success, RRTConnect performs fastest but also best with 90% of success rate. TEB2MP arrives second with close to 80% of success, followed by STOMP with approximately 45% and far last is TrajOpt which laboriously passes the 10% bar. In terms of execution speed, STOMP is slightly faster than TEB2MP but the variance of this result is fairly large whereas TEB2MP performs much more consistently. TrajOpt ranks last with almost 10 seconds of planning time, while all other planners are sub-second in average. On the other hand, TrajOpt performs best on the smoothness metric, while STOMP and TEB2MP performs similarly well and RRTConnect performs worst with a fairly large variance. Finally, STOMP achieves the best results in the joint limit distance metric, closely followed by TEB2MP but - again - with more consistency. TrajOpt and RRTConnect come lasts with similar results in average.

## Desk environment

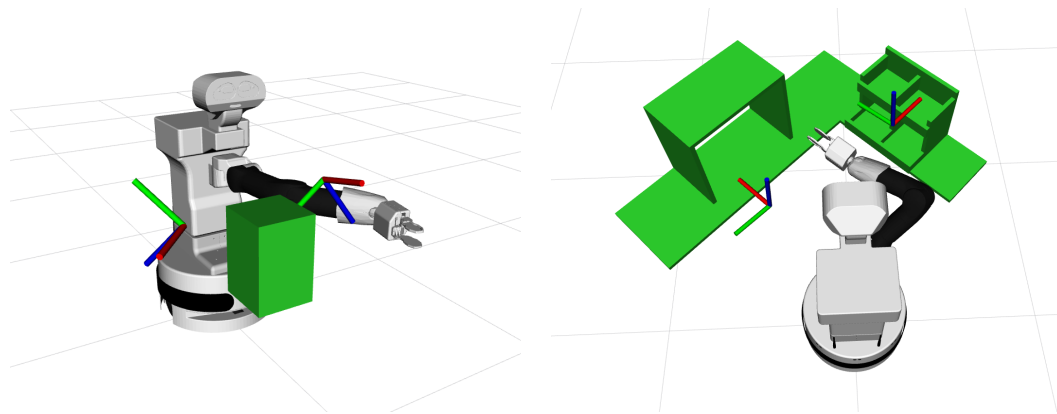
The desk scenario places the robot in an industrial-like setting. Facing a work station, the robot has to move its end-effector from the top of a shelf to a storage on the other side of the desk. Depicted in Fig. 5.9(b), the scenario was first introduced in [121]. Its use here allows for a comparison discussion with the method presented in [121].

Statistical results for this experiment are summarized in Fig. 5.9. RRTConnect successfully generated trajectories for all runs, achieving 100% success rate while TrajOpt performed remarkably well. TEB2MP and STOMP following closely behind. Similarly to the previous experiments, RRTConnect is the fastest method closely followed by STOMP and TEB2MP while TrajOpt lies far behind. TrajOpt performs best on the smoothness criterion, followed by STOMP and TEB2MP. However STOMP exhibits a very large variance on the quality of the results while TEB2MP is much more consistent. Similarly to the previous experiment, it is RRTConnect which comes last for this metric. Finally, in this scenario all planners exhibited a fairly good, similar performance on joint limit distance. A possible explanation to this is that a more complex environment naturally constraints the manipulator's feasible range of motion, resulting in larger distances from limits. It is worth noting that TEB2MP consistently provided a good performance throughout this experiment as well.

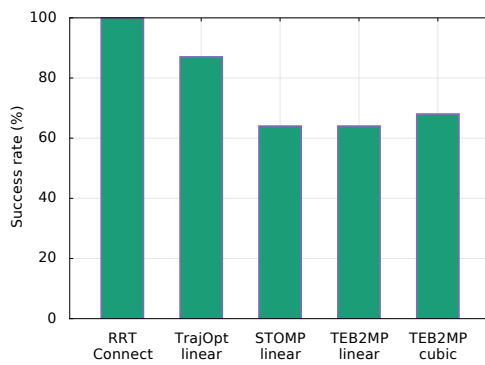
Since the environment used in this experiment was first presented by Dong *et al.* in [121] which experiments were conducted using the PR2 robot, a robot fairly similar to the TIAGo robot, a comparison with GPMP2 can be attempted. In the same environment, with the PR2 robot, GPMP2 is reported with an average planning time of 20ms. A caveat of these experiments is that the reported planning time did not include the time to compute GPMP2's *Signed Distance Field* (SDF). For a fair comparison, the pre-processing step was measured and added to the initially reported values, so that a GPMP2 plan takes approximately 800ms. That is in the same computational complexity range as TEB2MP. In terms of planning success rate, [121] reported a single number whereas two different scenes were used; therefore, a direct comparison cannot be made. Nevertheless, even in the case of GPMP2 having a consistently higher success rate, the optimization process does not consider the smoothness of the trajectory or distance from joint limits.

### 5.4.5 Conclusion

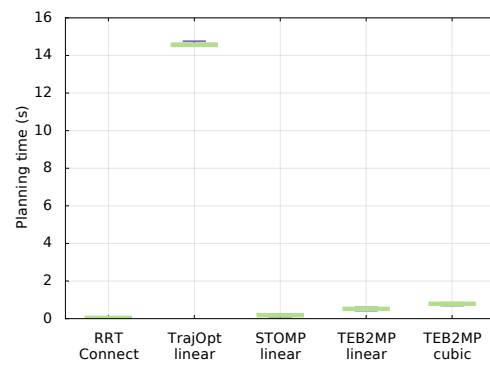
We presented an extension to the TESC planner addressing the problem of manipulation motion planning. Coined *Timed-Elastic Band for Manipulation Motion Planning* (TEB2MP) planner, it allows for defining cost function both in the Cartesian and joint spaces. The proposed approach is compared against state-of-the-art methods in three different scenarios of increasing difficulty. The results show that in the worst case it performs comparably with the state-of-the-art while providing more consistence performances.



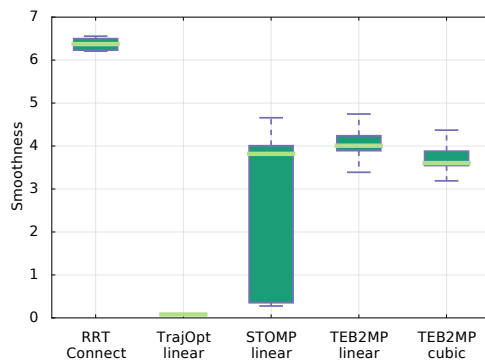
(a) Box scenario: Starting from the right hand side of the box, the robot has to reach the goal on the left side of the box. (b) Industrial manufacturing scenario: Starting from the top right shelf, the robot has to reach the goal on the left of the bench.



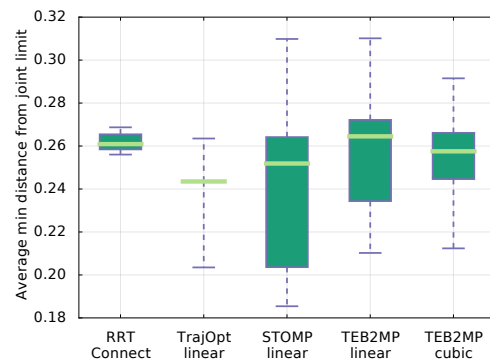
(c) Success rate (%)



(d) Planning time (s)



(e) Smoothness



(f) Distance from joint limits

Figure 5.9: Experiment metrics for the evaluation of TEB2MP in the industrial environment.

# Chapter 6

## The manif library

---

*manif* is a Lie theory library targeted at state estimation in robotics applications. Developed as a header-only library, with minimal dependency and a requirement on C++11 only, it is easy to integrate it to existing projects.

---

The `manif` library [126] is a C++ library that naturally results from the study conducted in Section 2.2, intreching the Lie theory in computer code. The library is distributed under a permissive open-source MIT license and is publicly available<sup>1</sup>.

## 6.1 Related work

With Lie theory becoming increasingly popular in robotics, it is not surprising to find many implementations in different programming languages. Most of the large *Nonlinear Programming* (NIP) framework that serves as a backbone to SLAM algorithms implementation offers a subset of the Lie theory relevant to SLAM and robotics in general. One may cite `Ceres` [112], `GTSAM` [127] or `G2o` [124] as examples. However, these implementations are deeply intertwined with the rest of the library making it very tedious to isolate only the Lie theory-related part with the intent to use it in another project. Similarly, some robotics projects may also provide their own implementation such as `mrpt` [128] or `kindr` [129]. While we will focus especially on C++ libraries as it is most often the chosen language for computationally intense work, one may however cite a few other projects and libraries available in diverse programming languages:

- `manopt` [130], a Matlab toolbox for optimization on manifolds.
- `MTKM` [131], a Matlab toolbox for manifold-based graph optimization for multi-sensor calibration and SLAM.
- `SymPy` [132] which offers a Python module for Lie Algebra symbolic computations.
- `Pymanopt` [133], a feature-rich Python toolbox for optimization on manifolds using *Automatic Differentiation* (AD).
- `censilib` [134] a stand-alone Python library handling various differentiable manifolds.

There exists a few standalone C++ libraries offering the subset of Lie theory relevant to robotics which are overviewed hereafter.

The `MTK` library [135] is a framework for on-manifold operations. Besides the traditional groups, it allows for the composition of arbitrary compound manifolds. First introduced in [136], which present the theory of on-manifold optimization, the library also provides an implementation of a *Sparse Least Squares on Manifold* (SLoM) algorithm. The library does not provide analytical Jacobians but instead rely on numerical AD.

Ethan Eade's `liegroups` library [137] - which we will refer as `eadelib` to avoid ambiguity - similarly to `manif`, results from a research and clarification work [68]. Written in C++, it does not offer the full set of on-manifold operations nor does it provide any Jacobians.

The `Sophus` [138] library is a C++11 implementation of Lie groups based on `Eigen3`. It allows for most common on-manifold operations but only offers analytical

---

<sup>1</sup><https://github.com/artivis/manif>

Table 6.1: Lie theory C++ libraries comparison. ‘Any Scalar’ indicates a header-only library and the symbol ‘\*’ means partial.

Library	Manifold operations	Manifold base class	Jacobians	Any scalar	Maps	C++11 and higher
Eigen				✓	✓	✓
MTK	✓					✓
eadelib	*			✓		✓
Sophus	✓		*	✓	✓	✓
wave_geometry	✓		✓	✓	✓	
manif	✓	✓	✓	✓	✓	✓

Jacobian matrices for some of them. Moreover, the Jacobian matrices are expressed with respect to the representation vector that underlies the implementation.

The `wave_geometry` [139, 140] library implements manifolds geometry with fast AD and coordinate frame semantics checking. Based on `Eigen3` and `Boost` [141], it requires a C++17-compatible compiler. As of the time of writing, `wave_geometry` only implements the groups  $SO(3)$  and  $SE(3)$ .

## 6.2 Design

The `manif` library is a Lie theory library targeted at state estimation in robotics applications. The library is designed with ease of integration and use as a primary concern. This reflects in the following features,

- A single external dependency on `Eigen3` [142].
- Header-only, no library object to link to.
- Templated underlying scalar type so that one can use its own.
- C++11, since not everyone gets to enjoy the latest C++ features, especially in industry.

The `manif` library has been developed to make easily accessible the most common operations on Lie groups in state estimation. The library is mathematically grounded in [70] whose formalism (see Section 2.2) molds the library API.

Following high software development standards, `manif` is thoroughly tested and supports several platforms and compilers: gcc and clang on Linux and OS X as well as msvc on Windows.

`manif` differs from all libraries mentioned in Section 6.1 in that all its classes inherit from a common templated base class which enforces a common minimal API. It also differs from each library in ways that are summarized in Table 6.1.

### Implementation

The `manif` library design is similar to that of `Eigen3`, in the sense that all classes defined have in common that they inherit from either of two templated base classes using static polymorphism through the *Curiously Recurring Template Pattern* (CRTP)

```

template <typename Derived>
struct CRTPBase
{
    // Declare the common API
    void interface() {
        derived().implementation(); }

    // Return a reference to the derived object
    Derived& derived() {
        return *static_cast<Derived*>(this); }
};

class Derived : public CRTPBase<Derived>
{
    // Define the function body
    void implementation() {
        std::cout << "Hello world."; }
};

```

Listing 6.1: CRTP example.

scheme [143](Section 16.3). This allows for the possibility to write generic code without paying the price of pointer indirection as they are resolved at compile-time. Thanks to this static polymorphism, the library remains open to extensions to Lie groups beyond the currently implemented  $SO(2)$ ,  $SE(2)$ ,  $SO(3)$  and  $SE(3)$ . The very same scheme is also used to implement their related tangent objects. An example of such CRTP class is given in Listing 6.1.

In `manif`, the inheritance hierarchy is threefold. First, two CRTP base classes, ‘LieGroupBase’ for Lie groups and ‘TangentBase’ for their related tangent objects. Both declare the API common to their respective sub-classes while also defining default implementation for some of the functions that can be constructed from chaining basic operations. These base classes do not have any members (they do not own any memory blocks) but instead operate on members owned by their derived classes.

Second, for each derived group and tangent, there exists an intermediate base class which inherits from ‘LieGroupBase’ and ‘TangentBase’ respectively. These intermediate classes define all group’s function together with some extra functions which are specific to a given group. Similarly to the CRTP base classes, these intermediate classes do not have any member.

Finally the third and last layer is the object which most *users* are expected to manipulate.

The inheritance scheme of `manif` is shown in pseudo-code in Appendix D.1 for the case of  $SE(2)$ .

### Integration to other projects

`manif` is a header-only library and as such is very easily integrated to other projects. Given a project managed with a ‘CMakeLists.txt’ file, it only takes 4 lines to start



```

project(foo)

# Find the Eigen library as manif depends on it
find_package(Eigen3 REQUIRED)
target_include_directories(${PROJECT_NAME}
  SYSTEM PUBLIC ${EIGEN3_INCLUDE_DIRS})

# Find the manif library
find_package(manif REQUIRED)

add_executable(${PROJECT_NAME} src/foo.cpp)
# Add manif include directories to the target
target_include_directories(${PROJECT_NAME}
  SYSTEM PUBLIC ${manif_INCLUDE_DIRS})

```

Listing 6.2: Integrating manif to a project.

```

template <typename Derived>
void print(const LieGroupBase<Derived>& g)
{
  std::cout << "Group degrees of freedom : "
    << g::DoF
    << "\n"
    << "Underlying representation vector size : "
    << g::RepSize
    << "\n"
    << "Current values : " << g << "\n";
}

```

Listing 6.3: A generic ‘print’ function

using the library as shown in Listing 6.2.

### Writing generic code

Beside the embodiment of the mathematics from Section 2.2 in computer code, one of the key features of `manif` is the possibility to write group-abstract generic code. This is possible thanks to the CRTP inheritance scheme described in Section 6.2.

Such generic function may be purely functional, *e.g.* allowing to introspect some properties of an abstract group object as shown in Listing 6.3. It also allows to write group abstract wrappers to interface with other libraries as shown in Appendix D.3. But more interestingly it also allows to define some helper functions to do computations on group abstract objects. An example is shown in Listing 6.4 that implements the following equation,

$$n = \|\chi \ominus \gamma\|^2 \quad (6.1)$$

A larger and more interesting example of writing group-abstract code is given in Appendix D.2.

```

template <typename DerivedA, typename DerivedB>
void ominusSquaredNorm(const LieGroupBase<DerivedA>& rhs,
                      const LieGroupBase<DerivedB>& lhs)
{
    return (rhs-lhs).coeffs().squaredNorm();
}

```

Listing 6.4: A generic multi-template function

```

class QuadraticCostFunction
: public ceres::SizedCostFunction<1, 1>
{
public:
    virtual bool Evaluate(double const* const* parameters,
                        double* residuals,
                        double** jacobians) const {
        const double x = parameters[0][0];
        residuals[0] = 10 - x;
        // Compute the Jacobian if asked for.
        if (jacobians != NULL && jacobians[0] != NULL) {
            jacobians[0][0] = -1;
        }
        return true;
    }
};

```

Listing 6.5: Ceres ‘CostFunction’

### On the use with Ceres

The **Ceres** library is very popular in SLAM research for its relative ease of use, and for offering AD. However, using **Ceres** for on-manifold optimization requires some special care.

For a given error function  $f(\cdot)$ , one is interested in linearizing it,

$$f(\boldsymbol{\chi} \oplus \boldsymbol{\tau}) = \mathbf{e} + \mathbf{J}_{\boldsymbol{\tau}}^e \boldsymbol{\tau} . \quad (6.2)$$

While the **manif** library computes Jacobians as per (2.66a), many non-linear solvers expect them to be expressed as per (B.3), that is with respect to the underlying representation vector of the group element (*e.g.* with respect to the quaternion parameters vector for  $\text{SO}(3)$ ). **Ceres** is one of them.

In the **Ceres** framework, the computation of  $\mathbf{J}_{\boldsymbol{\tau}}^e$  is decoupled in two folds as explained hereafter. The following terminology should sound familiar to **Ceres** users.

On one side, a ‘CostFunction’<sup>2</sup> is a class representing and implementing an error function as detailed in Listing 6.5. It produces the Jacobian

$$\mathbf{J}_{\boldsymbol{\chi} \oplus \boldsymbol{\tau}}^e . \quad (6.3)$$

<sup>2</sup>[http://ceres-solver.org/npls\\_modeling.html#costfunction](http://ceres-solver.org/npls_modeling.html#costfunction)

```

Eigen::Quaterniond state;

ceres::Problem::Options problem_options;
ceres::Problem problem(problem_options);

// Add the state to Ceres problem
problem->AddParameterBlock(state.data(), 4);

// Associate a LocalParameterization to the state vector
problem->SetParameterization(
    state.data(),
    new EigenQuaternionParameterization()
);

```

Listing 6.6: Ceres ‘EigenQuaternionParameterization’

On the other side, to retrieve the Jacobian  $\mathbf{J}_\tau^e$ , one must associate a ‘LocalParameterization’<sup>3</sup> to a state object as shown in Listing 6.6. The ‘LocalParameterization’ class (and derived) performs the state update step of the optimization –  $\chi \oplus \tau$ . While the function operates for any  $\chi$  and  $\tau$ , its Jacobian is always evaluated for  $\tau = \mathbf{0}$ , thus actually providing the Jacobian

$$\mathbf{J}_\tau^{\chi \oplus \tau} = \frac{\partial \chi \oplus \tau}{\partial \tau} = \lim_{\tau \rightarrow \mathbf{0}} \frac{\chi \oplus \tau - \chi}{\tau}. \quad (6.4)$$

Once both the ‘CostFunction’ and ‘LocalParameterization’’s Jacobians are evaluated, Ceres internally computes (6.2) from the chain rule (see Section 2.2.6),

$$\mathbf{J}_\tau^e = \mathbf{J}_{\chi \oplus \tau}^e \mathbf{J}_\tau^{\chi \oplus \tau}. \quad (6.5)$$

The intermediate Jacobians (6.3–6.4) that Ceres requires are not provided by `manif` as it would compute directly the final Jacobian (6.2).

However, one still wants to use `manif` in a Ceres-based project. For this reason, `manif` is compliant with Ceres AD scheme and the ‘ceres::Jet’<sup>4</sup> type. The AD scheme will take care to compute (6.3–6.4) and the composition (6.5) is performed automatically afterward. An example of writing an on-manifold problem with Ceres using `manif` is shown in Appendix D.3.

## 6.3 Evaluation

The computation performance of the `manif` library is evaluated on the benchmark proposed in [139]. The benchmark is initially designed to compare automatic differentiation libraries efficiency for any expression. While the `manif` library does not provide such feature, one can argue that it does provide semi-automatic differentiation since it implements the Jacobians for each operation of a complex expression which may then be composed following the chain rule. Computing the Jacobians of a complex expression then relates more to bookkeeping than partial-derivatives.

<sup>3</sup>[http://ceres-solver.org/npls\\_modeling.html#localparameterization](http://ceres-solver.org/npls_modeling.html#localparameterization)

<sup>4</sup>[http://ceres-solver.org/automatic\\_derivatives.html#dual-numbers-jets](http://ceres-solver.org/automatic_derivatives.html#dual-numbers-jets)

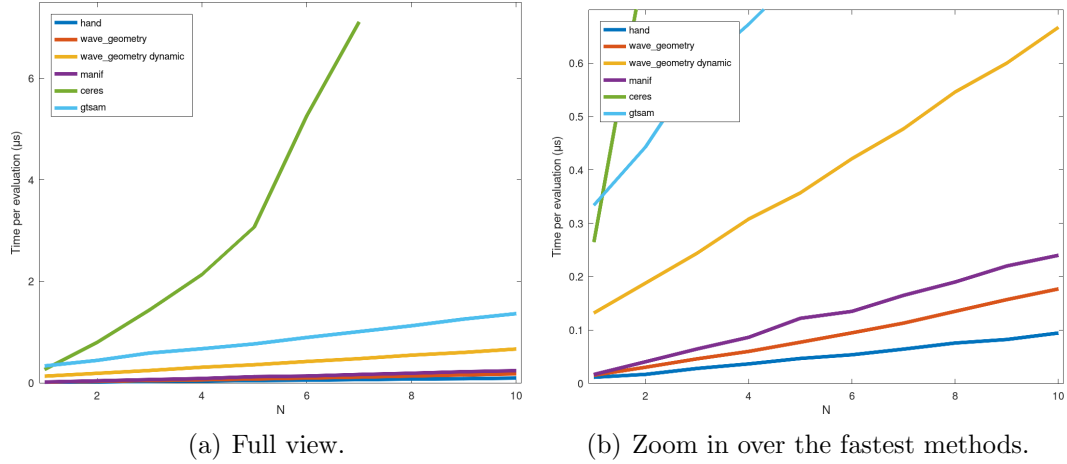


Figure 6.1: Comparison of the time required to evaluate the value and Jacobians of (6.6) for  $N \in [1, 10]$ .

Experiments are conducted on a system having an Intel i7-8850H, 2.6 GHz CPU, 16 GiB of RAM and running Ubuntu 18.04.

### Rotation chain

The rotation chain test transforms a vector  $\mathbf{v}_1 \in \mathcal{R}^3$  by a chain of rotation  $\boldsymbol{\chi} \in \text{SO}(3)$  of increasing length  $N$ ,

$$\mathbf{v}_2 = \left( \prod_{i=1}^N \boldsymbol{\chi}_i \right) \mathbf{v}_1, \quad (6.6)$$

so that for  $N = 3$ ,

$$\mathbf{v}_2 = \boldsymbol{\chi}_1 \boldsymbol{\chi}_2 \boldsymbol{\chi}_3 \mathbf{v}_1. \quad (6.7)$$

The transformed vector value is evaluated together with the  $N + 1$  Jacobians.

The results are given in Fig. 6.1. Fig. 6.1(a) shows that **manif** outperforms **Ceres**, **GTSAM** and **wave\_geometry**-dynamic for any length of the rotations chain. However, a closer look (Fig. 6.1(b)) shows that it is slightly slower than **wave\_geometry** while both are very close to the performance of a hand-written Jacobian computation.

### Pre-integrated IMU factor

The pre-integrated IMU factor test represents a scenario closer to a real use-case. The factor error cost is a simplified version of the one presented in [95](45).

Let  $\bar{\boldsymbol{\chi}}_{ij} \in \text{SO}(3)$  be a pre-integrated measurement of rotation between times  $i$  and  $j$ . Are associated to  $i$  and  $j$  the estimated orientation  $\boldsymbol{\chi}_i$  and  $\boldsymbol{\chi}_j$  in  $\text{SO}(3)$ . Let  $\boldsymbol{\tau}$  be an unknown small change of the IMU bias. The error of the updated pre-integrated factor reads,

$$e_{ij} = (\bar{\boldsymbol{\chi}}_{ij} \oplus \boldsymbol{\tau}) \ominus \boldsymbol{\chi}_i^{-1} \boldsymbol{\chi}_j, \quad (6.8)$$

for which each of the four Jacobians are evaluated. Table 6.2 reports the average computation time of the factor for both the **wave\_geometry** and the **manif** libraries. The results show that while both libraries performs really well, with execution times close to this of a hand-written Jacobian computation, **manif** performs

Algorithm	hand-coded	wave_geometry			manif
		forward	typed forward	reverse	
mean ( $\mu s$ )	229.058	531.052	471.596	271.962	367.879
std ( $\mu s$ )	5.273	17.539	9.150	9.150	9.870

Table 6.2: Time to evaluate the value and Jacobians of (6.8).

better than `wave_geometry` in its ‘forward’ and ‘typed-forward’ versions. However, `wave_geometry` ‘reverse’ version performs better than `manif`.

## 6.4 Conclusion

We presented a new C++11 library called `manif`. It implements the Lie theory for use in state estimation in robotic applications. The library is simple to integrate and use in larger projects. The general concepts supporting its design are overviewed and further detailed in Appendix D.

`manif` execution performance are evaluated and compared against similar libraries. The results show that it is comparable to the state of the art while offering a more complete API and more Lie groups to be used.



# Chapter 7

## Conclusion

---

*In closing, the work presented throughout the thesis is recalled and the main contributions are summarized. The lines for future research study are drawn.*

---

This thesis investigated the problem of robust navigation for industrial service robots. Besides being an active field that receives an increasing interest, the research topic is motivated by the need of the company PAL Robotics to see their robots evolve more robustly in real dynamic environments. We therefore considered industrial mobile-bases as the main research platform. Such robot embeds a somewhat minimal set of sensors to reliably perceive the robot ego-motion and its surrounding. That is, wheel encoders and a LRF. We addressed the problem of navigation by breaking it down to smaller sub-problems. We identified some of these sub-problems as points of improvements and detailed them. We then proposed methods that address each problem, detailed them at length and showed their benefits, effectively improving the navigation system.

Although we could not cover the entirety of the wide navigation field, our work did cover some of the key issues and resulted in substantial improvements over the state of the art.

## 7.1 Summary of contributions

The main contributions of this thesis address three problems of navigation,

- **Loop-closure** – In Chapter 3 we presented two contributions to BoW-based place recognition for loop closure using only 2D LRF (Section 3.3). The first contribution aims at benefiting from the natural relative ordering of features extracted from sensor readings. Pushing further the parallel of the BoW principles with NLP, we propose using the Viterbi algorithm and a weighting scheme to favor longer sequences of consecutive words when comparing two laser scans. Our second contribution is a database augmentation (Section 3.3.2) that empowers topological neighbors so that places that are ‘close’ to each other and share common features are better ranked in the BoW query phase. Our proposals substantially enhance the performances of the classical BoW method and performs better than the state of the art while being also more robust to changes in the environment.
- **Odometry calibration** – In Chapter 4 we proposed a method able to calibrate the odometry model of a mobile-base together with the extrinsic parameters of a sensor that observes ego-motion. The estimation can be performed online in order to compensate for the variations of the model due to physical change as they appear. The proposed method relies on an abstraction of the IMU pre-integration theory allowing it to be applied to any sensor.
- **Motion planning** – In Chapter 5 we detailed the *Timed-Elastic Smooth Curve* formulation for motion planning which generates smoothsplines on Lie groups. The genericity of the proposed method was demonstrated by employing it in two different planning tasks; motion planning for a mobile-base and motion planning for a multi-link robotic arm. Our proposal was shown to perform favorably against the state of the art for both applications while offering more consistent results.

Furthermore, we contributed in Chapter 6 a C++ programming library for using Lie theory in state estimation for robotic applications. At the time of writing,



the library has over 200 ‘stars’ on its Github repository<sup>1</sup> and receives external contributions demonstrating an interest of the community for this type of grounded mathematical software tools.

## 7.2 Future work

We highlight in this section possible lines of future research that follows the aforementioned contributions.

### Loop closure

The central idea behind the use of the Viterbi algorithm (Section 3.3.1) – emphasizing candidates by asserting co-occurrent sequences of ordered words – is very generic and could be applied – in a different modality – to other sensors. Features also exhibit a relative 2D ordering in the image plane or in 3D in point-cloud. The main challenge however is that the increase of dimension (a 2D LRF reading is 1D) also increases the complexity of characterizing and matching such ordering. Our second contribution, the database augmentation, can however be calqued without modification to any sensor modality as it takes fully place in the BoW database independently of the sensor in use.

Future work would be to investigate the use of soft clustering for feature quantization. Unlike the K-Means algorithm, soft clustering methods allow for assigning several labels to a single instance. The probability of belonging to a given cluster center can be directly injected in the output probability of the HMM. Doing so we expect to see the quantization error further reduced. Moreover, the reliability of word matches can be weighted by inferring a score from word labeling probabilities. For instance, two features, labeled  $C - D - E$  for one and  $E - F - G$  for the second, could be matched together.

### Odometry calibration

Future work would naturally aim at applying the pre-integration abstraction to different motion models and other self-calibration problems, including in 3D. Furthermore, the pre-integration scheme could be integrated in a larger SLAM framework.

### Motion planning

An improvement that could be brought to the mobile-base planner is to use *Signed Euclidean Distance Grid* (S-EDG) to represent the robot environment. Unlike the EDG used in this work, a S-EDG also encodes the distance to the closest unoccupied cell so that cells corresponding to obstacles are filled with negative values rather than zeros. This difference allows for computing a discrete gradient throughout the grid which would repulse the robot out of a collision state during the optimization process and greatly improve over the current solution.

In a similar manner, the arm planner could benefit from improving the environment representation. Its performance in highly cluttered environments could

---

<sup>1</sup><https://github.com/artivis/manif>

be improved by dynamically increasing the resolution of the planner around often colliding points or using locally updated collision environments.

A wider perspective for future research would be to merge together the two applications presented. By doing so, one would seek at fully benefiting from a mobile manipulator robot such as TIAGo. Integrating the mobile-base planning together with the arm planning, the robot could anticipate motions much like humans do, *e.g.* the robot could raise its arm toward an object while still being moving toward the table on which the object sits.

Generally, the TESC formulation would benefit from using an optimization initialization point that is closer to the solution. To do so, rather than initializing at identity or rely on interpolation, one could start the optimization process from the trajectory provided by a sampling-based algorithm such as an *Rapidly-exploring Random Tree* (RRT)-based algorithm.

A second point for improvement is the use of a SQP solver as proposed in the recent work of [144]. The first benefit of using such solver is that constraints (*e.g.* min/-max limits on velocity/acceleration *etc.*) would not be approximated anymore, thus, the optimization result can be certified before its execution on the robot. Another side benefit is that the TESC formulation would be closer to this of a pure *Model Predictive Control* (MPC) method.

Finally, since TESC already has a notion of time in the planning procedure, one could extend the planner capabilities taking into account dynamic information, *e.g.* moving obstacles. This way a fully dynamic environment can be supported.

## The manif library

Envisioning further developments for the library is fairly straightforward. First by extending the available Lie groups implemented, with a couple examples coming to mind,

- The trivial  $\mathbb{R}^n$  group, which is already under development at the time of writing.
- The Sim(2) and Sim(3) groups which are 2D and 3D similarity transforms, respectively. The latter is especially interesting for Vision-based works as it embeds the rotation and translation in 3D but also scaling, one of the variables in multi-view geometry.

Long-term development includes creating composite manifolds (see [70] Section IV.) from any number of Lie groups already implemented. As of the time of writing, the Jacobians matrices implemented in `manif` are right Jacobians (see Section 2.2.6). While it also provides means to transform a right Jacobian into its left counter-part, we envision the possibility for the user to easily choose which one (right- or left-Jacobian) to compute and retrieve.

## 7.3 Closing words

To write the period that concludes a work of several years is not an easy task. It forces me into a retrospection, reevaluate again the studies I conducted, the contributions I made, the decisions I took.

It is the occasion for me to revisit memories, dive into these constitutive moments of a PhD student life. The frustration facing the embodiment of an idea simply not working. The increasing sleep deprivation as a deadline comes closer. The anxiety upon reception of a mail containing a decision concerning a submission. The proud when such decision is positive. The incredible trips around the world. And repeat the cycle. There is something of a rite of passage to it.

I should recall here that the PhD took place as a collaboration between a university and a company. It allowed me to have a view of both sides and experience both ways of working. I must recall that the entirety of the doctoral study took place in Barcelona, Catalonia, Spain. This is important for I have a really hard time picturing better place where to do it again if I had to. Such a lovely city.

Despite the doctorate being a collaboration, I did spent most of my time at the company office; PAL Robotics. A place crowded with robots, small ones, big ones, on wheels, on legs, sleeping or roaming around. A truly amazing place for whoever likes robots. When I was not doing plain research there, I was essentially learning software development.

I can't really say if I would recommend anyone to go down the path of an industrial PhD. I cannot compare it to a regular one but I have the feeling that it does multiply the amount of work. The research targets and those of a company does not always align well. It might have been better, in my case, to delay the beginning of the PhD by a couple years. But all in all, pursuing the doctoral study has been a formidable experience, both on a personal and professional level.

To conclude, really, not only did I learn a lot over the past years, I also found my path. At least for while. To put it simply: *I want to do software for robots*. But this is no place to poetize, get too sentimental. Or pay me a beer first.



# Appendix A

## A brief history of ROS

ROS started in 2006 as an academic project at Stanford University led by Keenan Wyrobek and Eric Berger. At that time ROS was yet another iteration of the most common problem in robotics at that time, almost each laboratory was implementing its own software infrastructure handling the low-level communication and orchestration between the many software bricks inside a given robot. This work duplication would sometimes happen within the same laboratory! Too much time spent on the low-level framework, too little on actual higher-level robotics programs. Originally called the 'Stanford Personal Robotics Program', the framework aimed at handling all the low-level communication in a flexible and modular fashion together with some tools allowing one to rapidly build complex intelligent robotics programs on top of it. Ironically, the project born as an attempt to break the circle of re-inventing the wheel in robotics by doing so (Fig. A.1). Indeed several other frameworks were launched or already existed at that time, Player/Stage [145], URBI [146], Open-R<sup>1</sup>, MRPT [128] or YARP [147] to name a few.

In 2008 Keenan and Eric met Scott Hassan, an investor and founder of Willow Garage, the startup incubator that would host the development of ROS for the following 6 years<sup>2</sup>. During that time ROS development was very fast paced and its popularity skyrocketed, boosted by many brilliant engineers and an army of interns from around the world. From 2010 to 2012, the ROS team produced no less than 6 releases,

- ROS Box Turtle
- ROS C Turtle
- ROS Diamond Back
- ROS Electric Emys

---

<sup>1</sup>[https://www.sony.net/SonyInfo/News/Press\\_Archive/200205/02-017E/](https://www.sony.net/SonyInfo/News/Press_Archive/200205/02-017E/)

<sup>2</sup>The original pitch deck of the 'Personal Robotics Program' is available online <https://tinyurl.com/ud4v2h1>



Figure A.1: Re-inventing the Wheel, by Jorge Cham.

- ROS Fuerte Turtle
- ROS Groovy Galapagos

It is also during that period that,

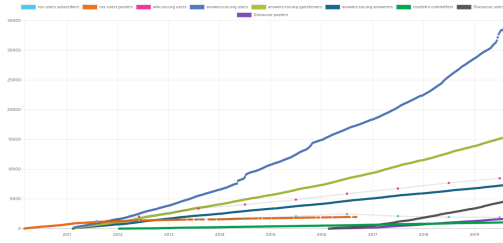
- Gazebo<sup>3</sup> [148] became the main 3D simulator of ROS.
- the ROS community gained in structure with the launch of the ROS Answers website<sup>4</sup> and the first edition of the ROSCon conference.
- the Personal Robot 2, also known as PR2, was built.

In 2013, the news that Willow Garage was closing its doors has cast worriedness in the robotics community as many had switched, partially or fully, to ROS. However everyone was quickly reassured when the creation of the *Open Source Robotics Foundation* (OSRF) was made public, a new legal structure that would take on the lead of ROS development. Followed years of growth and success both for the framework and its community (Fig. A.2). Followed also the maturity of the framework, with more and more commercial applications and robots delivered with ROS and a software release cycle called on the Ubuntu release cycle,

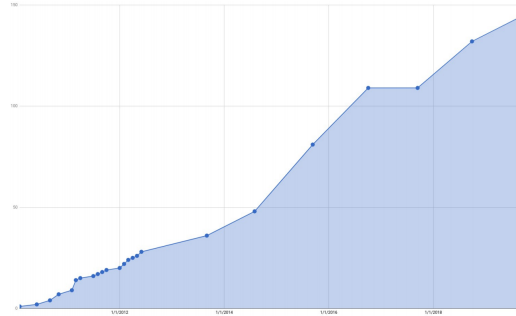
- ROS Hydro Medusa (2013)
- ROS Indigo Igloo (2014 - LTS)
- ROS Jade Turtle (2015)
- ROS Kinetic Kame (2016 - LTS)
- ROS Lunar Loggerhead (2017)

<sup>3</sup><http://gazebosim.org/>

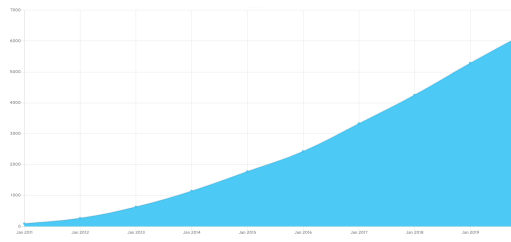
<sup>4</sup><https://answers.ros.org/>



(a) Growth of the ROS community as the number of users on the different community websites.



(b) Number of ROS-powered robots<sup>5</sup>.



(c) Citations of the original ROS paper [1].

Figure A.2: A collection of metrics showing the growth of the ROS community<sup>6</sup>.

- ROS Melodic Morenia (2018 - LTS)
- ROS Noetic Ninjemys (2020 - LTS)

where 'LTS' stands for Long Term Support. Releases tagged LTS are supported for five years.

Most modern robot manufacturers provide nowadays off-the-shelf ROS API (Fig. A.2(b)).

<sup>6</sup>Only accounts for robots registered on <https://robots.ros.org/>.

<sup>6</sup><https://metrics.ros.org/index.html>





# Appendix B

## Reminders

### B.1 Jacobians on vector spaces

For a multivariate function  $f(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^n$ , the Jacobian matrix is the  $n \times m$  matrix of all first-order partial derivatives,

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f_1}{\partial \mathbf{x}_m} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f_n}{\partial \mathbf{x}_m} \end{bmatrix} \in \mathbb{R}^{n \times m} . \quad (\text{B.1})$$

(B.1) shows that each column entry of the Jacobian matrix corresponds to the variation of the function  $f(\cdot)$  with respect to the perturbation of a single variable in the vector  $\mathbf{x}$ . For convenience we note  $\mathbf{J}_{\mathbf{x}}^{f(\mathbf{x})} \triangleq \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ , allowing the chain rule to be explicit as shown in Section B.2. Furthermore, one can write  $\mathbf{j}_i$ , the  $i$ -th column of  $\mathbf{J}_{\mathbf{x}}^{f(\mathbf{x})}$  as,

$$\mathbf{j}_i = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \triangleq \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h} \in \mathbb{R}^n , \quad (\text{B.2})$$

where  $\mathbf{e}_i$  is the  $i$ -th vector of the natural basis of  $\mathbb{R}^m$ .

By concatenating all columns in (B.2) and dropping the basis vectors  $\mathbf{e}_i$ , (B.1) can be written in a compact form,

$$\mathbf{J}_{\mathbf{x}}^{f(\mathbf{x})} \triangleq \lim_{\mathbf{h} \rightarrow 0} \frac{f(\mathbf{x} + \mathbf{h}) - f(\mathbf{x})}{\mathbf{h}} \in \mathbb{R}^{n \times m} . \quad (\text{B.3})$$

Note that (B.3) is only a notation convenience since the denominator  $\mathbf{h} \in \mathbb{R}^m$  and division by vector is undefined. Such notation then requires to be scattered to instances of (B.2) for computation.

## B.2 The chain rule

The chain rule is a formula for computing the derivative of the composition of functions. For  $\mathbf{y} = f(\mathbf{x})$  and  $\mathbf{z} = g(\mathbf{y})$  we have  $h = g \circ f$  the composed function which maps  $\mathbf{x} \rightarrow \mathbf{z}$ , or  $\mathbf{z} = g(f(\mathbf{x}))$ .

The derivative of the composition reads,

$$h' = (g' \circ f) \cdot f' \quad (\text{B.4a})$$

or equivalently in terms of the variables,

$$h'(\mathbf{x}) = g'(f(\mathbf{x})) \cdot f'(\mathbf{x}) \quad (\text{B.4b})$$

or equivalently in Leibniz's notation

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}}. \quad (\text{B.4c})$$

Replacing (B.4c) with the Jacobians notation defined in Section B.1 reads,

$$\mathbf{J}_{\mathbf{x}}^{\mathbf{z}} = \mathbf{J}_{\mathbf{y}}^{\mathbf{z}} \mathbf{J}_{\mathbf{x}}^{\mathbf{y}}. \quad (\text{B.5})$$

One may see how this notation helps in keeping tracks of the correctness of the chain. The super- and sub- scripts ( $\mathbf{y}$ ) of consecutive Jacobians must be the same ( $\mathbf{J}_{\mathbf{y}}^{\mathbf{z}} \mathbf{J}_{\mathbf{x}}^{\mathbf{y}}$ , reading from right to left). The subscript ( $\mathbf{x}$ ) of the element on the left-hand-side of the equal operator ( $\mathbf{J}_{\mathbf{x}}^{\mathbf{z}}$ ) is the same as the subscript ( $\mathbf{x}$ ) of the right-most element on the right-hand-side of the equal operator ( $\mathbf{J}_{\mathbf{x}}^{\mathbf{y}}$ ). Similarly the superscript ( $\mathbf{z}$ ) of the element on the left-hand-side of the equal operator ( $\mathbf{J}_{\mathbf{x}}^{\mathbf{z}}$ ) is the same as the superscript ( $\mathbf{z}$ ) of the left-most element on the right-hand-side of the equal operator ( $\mathbf{J}_{\mathbf{y}}^{\mathbf{z}}$ ).

# Appendix C

## Lie theory take away

This Appendix provides all of the necessary mathematical material for using the Lie theory with each of the typical Lie group used in robotics.

Operation	Inverse	Compose	Exp	Log	Right- $\oplus$	Right- $\ominus$
Right Jacobians	$\mathbf{J}_X^X = -\mathbf{Ad}_X$	$\mathbf{J}_X^{\circ\gamma} = \mathbf{Ad}_{\gamma^{-1}}$ $\mathbf{J}_\gamma^{\circ\gamma} = \mathbf{I}$	$\mathbf{J}_\tau^{\text{Exp}(\tau)} = \mathbf{J}_r(\tau)$	$\mathbf{J}_X^{\text{Log}(x)} = \mathbf{J}_r^{-1}(\tau)$	$\mathbf{J}_X^{\oplus\tau} = (\mathbf{Ad}_{\text{Exp}(\tau)})^{-1}$ $\mathbf{J}_\tau^{\oplus\tau} = \mathbf{J}_r(\tau)$	$\mathbf{J}_X^{\ominus x} = -\mathbf{J}_l^{-1}(\tau)$ $\mathbf{J}_\tau^{\ominus x} = \mathbf{J}_r^{-1}(\tau)$

Table C.1: Elementary Jacobian blocks

Lie group $\mathcal{M}, \circ$	size	dim	$\mathcal{X} \in \mathcal{M}$	Constraint	$\tau^\wedge \in \mathfrak{m}$	$\tau \in \mathbb{R}^m$	
Vector $n$ -D	$\mathbb{R}^n, +$	$n$	$\mathbf{v} \in \mathbb{R}^n$	$\mathbf{v} - \mathbf{v} = \mathbf{0}$	$\mathbf{v} \in \mathbb{R}^n$	$\mathbf{v} \in \mathbb{R}^n$	
Complex number	$S^1, \cdot$	2	1	$\mathbf{z} \in \mathbb{C}$	$\mathbf{z}^* \mathbf{z} = 1$	$i\theta \in i\mathbb{R}$	$\theta \in \mathbb{R}$
2D Rotation	$\text{SO}(2), \cdot$	4	1	$\mathbf{R}$	$\mathbf{R}^\top \mathbf{R} = \mathbf{I}$	$[\theta]_\times = \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix} \in \mathfrak{so}(2)$	$\theta \in \mathbb{R}$
2D Rigid Motion	$\text{SE}(2), \cdot$	9	3	$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$	$\mathbf{R}^\top \mathbf{R} = \mathbf{I}$	$\begin{bmatrix} [\theta]_\times & \boldsymbol{\rho} \\ 0 & 0 \end{bmatrix} \in \mathfrak{se}(2)$	$\begin{bmatrix} \boldsymbol{\rho} \\ \theta \end{bmatrix} \in \mathbb{R}^3$
Quaternion	$S^3, \cdot$	4	3	$\mathbf{q} \in \mathbb{H}$	$\mathbf{q}^* \mathbf{q} = 1$	$\boldsymbol{\theta}/2 \in \mathbb{H}_p$	$\boldsymbol{\theta} \in \mathbb{R}^3$
3D Rotation	$\text{SO}(3), \cdot$	9	3	$\mathbf{R}$	$\mathbf{R}^\top \mathbf{R} = \mathbf{I}$	$[\boldsymbol{\theta}]_\times = \begin{bmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{bmatrix} \in \mathfrak{so}(3)$	$\boldsymbol{\theta} \in \mathbb{R}^3$
3D Rigid Motion	$\text{SE}(3), \cdot$	16	6	$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$	$\mathbf{R}^\top \mathbf{R} = \mathbf{I}$	$\begin{bmatrix} [\boldsymbol{\theta}]_\times & \boldsymbol{\rho} \\ 0 & 0 \end{bmatrix} \in \mathfrak{se}(3)$	$\begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\theta} \end{bmatrix} \in \mathbb{R}^6$

Table C.2: Typical Lie groups used in robotics, including the trivial  $\mathbb{R}^n$ .

**Note:** All Jacobians in this document are **right Jacobians**, whose definition reads:  $\frac{\delta f(X)}{\delta X} \rightarrow \lim_{\varphi \rightarrow 0} \frac{f(X \oplus \varphi) \ominus f(X)}{\varphi}$ .

However, notice that one can relate the left- and right- Jacobians with the Ad-joint,  $\mathbf{Ad}_{\text{Exp}(\tau)} = \mathbf{J}_l(\tau) \mathbf{J}_r^{-1}(\tau)$ .

Op	Identity	Inverse	Compose	Act	Exp	Log
$\mathcal{M}_{1,0}$						
$\mathbb{R}^n, +$	$\mathbf{v} = [\mathbf{0}]$	$-\mathbf{v}$	$\mathbf{v}_1 + \mathbf{v}_2$	$\mathbf{v} + \mathbf{p}$	$\mathbf{v}$	$\mathbf{v}$
$S^1, \cdot$	$z = 1 + i0$	$z^*$	$z_1 z_2$		$z = \cos \theta + i \sin \theta$	$\theta = \arctan 2(\text{Im}(z), \text{Re}(z))$
$SO(2), \cdot$	$\mathbf{R} = \mathbf{I}$	$\mathbf{R}^{-1} = \mathbf{R}^T$	$\mathbf{R}_1 \mathbf{R}_2$	$\mathbf{R} \cdot \mathbf{v}$	$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$	$\theta = \arctan 2(r_{21}, r_{11})$
$SE(2), \cdot$	$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = \mathbf{I}$	$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$	$\mathbf{M}_1 \mathbf{M}_2 = \begin{bmatrix} \mathbf{R}_1 \mathbf{R}_2 & \mathbf{t}_1 + \mathbf{R}_1 \mathbf{t}_2 \\ \mathbf{0} & 1 \end{bmatrix}$	$\mathbf{M} \cdot \mathbf{p} = \mathbf{t} + \mathbf{R} \mathbf{p}$	$\mathbf{M} = \begin{bmatrix} \text{Exp}(\theta) & \mathbf{V}(\theta) \boldsymbol{\rho} \\ \mathbf{0} & 1 \end{bmatrix}$	$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\rho} \\ \theta \end{bmatrix} = \begin{bmatrix} \mathbf{V}^{-1}(\theta) \mathbf{p} \\ \text{Log}(\mathbf{R}) \end{bmatrix}^{(1)}$
$S^3, \cdot$	$\mathbf{q} = 1 + i0 + j0 + k0$	$\mathbf{q}^* = w - ix - jy - kz$	$\mathbf{q}_1 \mathbf{q}_2$	$\mathbf{q} \mathbf{v} \mathbf{q}^*$	$\mathbf{q} = \cos \frac{\theta}{2} + \mathbf{u} \sin \frac{\theta}{2}$	$\boldsymbol{\theta} = 2\mathbf{v} \frac{\arctan 2(\ \mathbf{u}\ , w)}{\ \mathbf{v}\ }$
$SO(3), \cdot$	$\mathbf{R} = \mathbf{I}$	$\mathbf{R}^{-1} = \mathbf{R}^T$	$\mathbf{R}_1 \mathbf{R}_2$	$\mathbf{R} \cdot \mathbf{v}$	$\mathbf{R} = \mathbf{I} + \sin \theta [\mathbf{u}]_{\times} + (1 - \cos \theta) [\mathbf{u}]_{\times}^2$	$\boldsymbol{\theta} = \frac{\theta(\mathbf{R} - \mathbf{R}^T)^{\wedge}}{2 \sin \theta}$
$SE(3), \cdot$	$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = \mathbf{I}$	$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$	$\mathbf{M}_1 \mathbf{M}_2 = \begin{bmatrix} \mathbf{R}_1 \mathbf{R}_2 & \mathbf{t}_1 + \mathbf{R}_1 \mathbf{t}_2 \\ \mathbf{0} & 1 \end{bmatrix}$	$\mathbf{M} \cdot \mathbf{p} = \mathbf{t} + \mathbf{R} \mathbf{p}$	$\mathbf{M} = \begin{bmatrix} \text{Exp}(\theta) & \mathbf{V}(\theta) \boldsymbol{\rho} \\ \mathbf{0} & 1 \end{bmatrix}$	$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\rho} \\ \theta \end{bmatrix} = \begin{bmatrix} \mathbf{V}^{-1}(\theta) \mathbf{p} \\ \text{Log}(\mathbf{R}) \end{bmatrix}^{(2)}$

Table C.3: Operating on typical Lie groups used in robotics.

$$^{(1)} \mathbf{V}(\theta) = \frac{\sin \theta}{\theta} \mathbf{I} + \frac{1 - \cos \theta}{\theta^2} [\mathbf{1}]_{\times}$$

$$^{(2)} \mathbf{V}(\theta) = \mathbf{I} + \frac{1 - \cos \theta}{\theta} [\mathbf{u}]_{\times} + \frac{\theta - \sin \theta}{\theta^3} [\mathbf{u}]_{\times}^2$$

$\mathcal{M}_o$	Ad/Jac	Ad	$\mathbf{J}_r$	$\mathbf{J}_l$	$\mathbf{J}_x^{\text{xp}}   \mathbf{J}_p^{\text{xp}}$ (Act)
$\mathbb{R}^n, +$	$\mathbf{I} \in \mathbb{R}^{n \times n}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{I}$
$S^1, \cdot$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{R}[1]_x \mathbf{v}$
$\text{SO}(2), \cdot$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{I}$	$\mathbf{R}[1]_x \mathbf{v}$
$\text{SE}(2), \cdot$	$\begin{bmatrix} \mathbf{R} & -[1]_x \mathbf{t} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$	$\begin{bmatrix} \sin \theta / \theta & (1 - \cos \theta) / \theta \\ (\cos \theta - 1) / \theta & \sin \theta / \theta \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \theta \rho_1 - \rho_2 + \rho_2 \cos \theta - \rho_1 \sin \theta / \theta^2 \\ \rho_1 + \theta \rho_2 - \rho_1 \cos \theta - \rho_2 \sin \theta / \theta^2 \\ 1 \end{bmatrix}$	$\begin{bmatrix} (\cos \theta - 1) / \theta & (\theta \rho_1 + \rho_2 - \rho_2 \cos \theta - \rho_1 \sin \theta) / \theta^2 \\ \sin \theta / \theta & (-\rho_1 + \theta \rho_2 + \rho_1 \cos \theta - \rho_2 \sin \theta) / \theta^2 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \mathbf{R} & \mathbf{R}[1]_x \mathbf{p} \\ \mathbf{R} & \mathbf{R}[1]_x \mathbf{p} \end{bmatrix}$
$S^3, \cdot$	$\mathbf{R}(\mathbf{q})$	$\mathbf{I} - \frac{1 - \cos \theta}{\theta^2} [\boldsymbol{\theta}]_x + \frac{\theta - \sin \theta}{\theta^3} [\boldsymbol{\theta}]_x^2$	$\mathbf{I} - \frac{1 - \cos \theta}{\theta^2} [\boldsymbol{\theta}]_x + \frac{\theta - \sin \theta}{\theta^3} [\boldsymbol{\theta}]_x^2$	$\mathbf{I} + \frac{1 - \cos \theta}{\theta^2} [\boldsymbol{\theta}]_x + \frac{\theta - \sin \theta}{\theta^3} [\boldsymbol{\theta}]_x^2$	$-\mathbf{R}(\mathbf{q})[\mathbf{v}]_x \stackrel{(3)}{\mathbf{R}(\mathbf{q})} \mathbf{R}(\mathbf{q}) \stackrel{(3)}$
$\text{SO}(3), \cdot$	$\mathbf{R}$	$\mathbf{I} - \frac{1 - \cos \theta}{\theta^2} [\boldsymbol{\theta}]_x + \frac{\theta - \sin \theta}{\theta^3} [\boldsymbol{\theta}]_x^2$	$\mathbf{I} + \frac{1 - \cos \theta}{\theta^2} [\boldsymbol{\theta}]_x + \frac{\theta - \sin \theta}{\theta^3} [\boldsymbol{\theta}]_x^2$	$-\mathbf{R}[\mathbf{v}]_x$	$\mathbf{R}$
$\text{SE}(3), \cdot$	$\begin{bmatrix} \mathbf{R} & [\mathbf{t}]_x \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}$	$\begin{bmatrix} \mathbf{J}_r(\boldsymbol{\theta}) & \mathbf{Q}(-\boldsymbol{\rho}, -\boldsymbol{\theta}) \\ \mathbf{0} & \mathbf{J}_r(\boldsymbol{\theta}) \end{bmatrix} \stackrel{(4)}$	$\begin{bmatrix} \mathbf{J}_r(\boldsymbol{\theta}) & \mathbf{Q}(\boldsymbol{\rho}, \boldsymbol{\theta}) \\ \mathbf{0} & \mathbf{J}_r(\boldsymbol{\theta}) \end{bmatrix} \stackrel{(4)}$	$\begin{bmatrix} \mathbf{R} & -\mathbf{R}[\mathbf{p}]_x \\ \mathbf{R} & -\mathbf{R}[\mathbf{p}]_x \end{bmatrix}$	$\mathbf{R}$

Table C.4: Adjoint and Jacobians of typical Lie groups used in robotics.

$$\begin{aligned}
\stackrel{(3)}{\mathbf{R}}(\mathbf{q}) &= \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & w^2 - x^2 + y^2 - z^2 & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & w^2 - x^2 - y^2 + z^2 \end{bmatrix} \\
\stackrel{(4)}{\mathbf{Q}}(\boldsymbol{\rho}, \boldsymbol{\theta}) &= \frac{\theta - \sin \theta}{\theta^3} ([\boldsymbol{\theta}]_x [\boldsymbol{\rho}]_x + [\boldsymbol{\rho}]_x [\boldsymbol{\theta}]_x) + \frac{\theta - \sin \theta}{\theta^3} ([\boldsymbol{\theta}]_x [\boldsymbol{\rho}]_x + [\boldsymbol{\rho}]_x [\boldsymbol{\theta}]_x) - \frac{1 - \frac{\theta^2}{2} - \cos \theta}{\theta^4} ([\boldsymbol{\theta}]_x^2 [\boldsymbol{\rho}]_x + [\boldsymbol{\rho}]_x [\boldsymbol{\theta}]_x^2 - 3 [\boldsymbol{\theta}]_x [\boldsymbol{\rho}]_x [\boldsymbol{\theta}]_x) \\
&\quad - \frac{1}{2} \left( \frac{\theta^2 - \cos \theta}{\theta^4} - 3 \frac{\theta - \sin \theta - \frac{\theta^3}{6}}{\theta^5} \right) ([\boldsymbol{\theta}]_x [\boldsymbol{\rho}]_x [\boldsymbol{\theta}]_x^2 + [\boldsymbol{\theta}]_x^2 [\boldsymbol{\rho}]_x [\boldsymbol{\theta}]_x)
\end{aligned}$$



# Appendix D

## The manif library

### D.1 Implementation details

Listing D.1: Inheritance hierarchy of SE2.

```
/**
 * The class LieGroupBase declares
 * the interface common to all groups classes
 * and defines some functions as composition of other.
 */
template <typename DerivedLieGroup>
struct LieGroupBase
{
    // Must be implemented by DerivedLieGroup

    DerivedTangent log(){
        // Call the implementation from the derived class
        derived().log();
    }

    Vector act(Vector v) { ... }
    Jacobian adj() { ... }
    DerivedLieGroup compose(DerivedLieGroup o) { ... }
    DerivedLieGroup inverse() { ... }

    // Useful, deduced functions

    DerivedLieGroup rplus(DerivedTangent t) {
        // Use of the basic functions
    }
}
```

```

    return compose(t.exp())
}
DerivedLieGroup lplus(DerivedTangent t) { ... }
DerivedTangent rminus(DerivedLieGroup o) { ... }
DerivedTangent lminus(DerivedLieGroup o) { ... }

// Some specific helpers

void setIdentity() { ... }
void setRandom() { ... }

// Some operators for syntactic sugar
DerivedLieGroup operator +(DerivedTangent t) {
    return this->rplus(t);
}
DerivedLieGroup operator -(DerivedTangent t) { ... }
}

/**
 * [SE2Base description]
 * @type {[type]}
 */
template <typename DerivedS02>
struct SE2Base : LieGroupBase<DerivedS02>
{
    // Define the necessary functions

    DerivedTangent log() {
        // Compute Log
    }

    Vector act(Vector) { ... }
    Jacobian adj() { ... }
    DerivedLieGroup compose(DerivedLieGroup) { ... }
    DerivedLieGroup inverse() { ... }

    // SE2-specific functions

    Scalar x() { ... }
    Scalar y() { ... }
    Scalar theta() { ... }
}

/**
 * [SE2 description]
 * @type {[type]}
 */
template <typename Scalar>

```



```

struct SE2 : SE2Base<SE2<Scalar>>
{
    // SE2 constructors

    SE2() = default;
    SE2(Scalar x, Scalar y, Scalar theta) { ... };

    // Actual data member
    Eigen::Vector4d data;
}

```

Listing D.2: ‘TangentBase’ class pseudo-code

```

/**
 *
 */
template <typename DerivedTangent>
struct TangentBase
{
    // Must be implemented by DerivedTangent
    LieAlg generator();
    LieAlg hat();
    DerivedLieGroup exp();

    Jacobian rjac();
    Jacobian ljac();

    // Useful, deduced functions
    void setZero();
    void setRandom();

    Jacobian rjacinv();
    Jacobian ljacinv();
    Jacobian smallAdj();
}

```

## D.2 Writing generic code

In this example we tackle the problem of writing a Lie group abstract de Casteljau algorithm [149]. This algorithm, developed by Paul de Faget de Casteljau, is a recursive geometric algorithm which approximates polynomials in Bernstein form or Bézier curves. In Appendix D.2, the algorithm is used to compute  $N$  elements of a discrete smooth spline  $\in \mathcal{G}$  that fits a user-provided collection of points lying on a manifold  $\mathcal{G}$ .

Listing D.3: A group-abstract ‘DeCasteljau’ algorithm

```

/**
 * @brief Curve fitting using the DeCasteljau algorithm

```

```

* on Lie groups.
*
* @param trajectory, a discretized trajectory.
* @param degree, the degree of smoothness of the fitted curve.
* @param k_interp, the number of points to interpolate
* between two consecutive points of the trajectory.
* interpolate k_interp for t in ]0,1].
* @param closed_curve Whether the input trajectory is closed or not.
* If true, the first and the last points of the input trajectory are
  used
* to interpolate points inbetween. Default false.
* @return The interpolated smooth trajectory
*
* @note A naive implementation of the DeCasteljau algorithm
* on Lie groups.
*
* @link https://www.wikiwand.com/en/De\_Casteljau%27s\_algorithm
*/
template <typename LieGroup>
std::vector<typename LieGroup::LieGroup>
decasteljau(const std::vector<LieGroup>& trajectory,
            const unsigned int degree,
            const unsigned int k_interp,
            const bool closed_curve = false)
{
  MANIF_CHECK(trajectory.size() > 2,
    "Input trajectory must have more than two points!");
  MANIF_CHECK(degree <= trajectory.size(),
    "Degree must be less or equal to the number of input points!");
  MANIF_CHECK(k_interp > 0,
    "k_interp must be greater than zero!");

  // Number of connected, non-overlapping segments
  const unsigned int n_segments = static_cast<unsigned int>(
    std::floor(double(trajectory.size()-degree)/double((degree-1)
      +1))
  );

  std::vector<std::vector<const LieGroup*>> segments_control_points;
  for (unsigned int t=0; t<n_segments; ++t)
  {
    segments_control_points.emplace_back(std::vector<const LieGroup
      *>());

    // Retrieve control points of the current segment
    for (unsigned int n=0; n<degree; ++n)
    {
      segments_control_points.back().push_back(

```

```

        &trajectory[t*(degree-1)+n
    );
    }
}

// Close the curve if there are left-over points
if (closed_curve && (n_segments*(degree-1)) <= trajectory.size()-1)
{
    const unsigned int last_pts_idx = n_segments*(degree-1);
    const unsigned int left_over = trajectory.size()-1-last_pts_idx;
    segments_control_points.emplace_back(std::vector<const LieGroup
        *>());

    // Get the left-over points
    for (unsigned int p=last_pts_idx; p<trajectory.size(); ++p)
    {
        segments_control_points.back().push_back( &trajectory[p] );
    }
    // Add a extra points from the beginning of the trajectory
    for (unsigned int p=0; p<degree-left_over-1; ++p)
    {
        segments_control_points.back().push_back( &trajectory[p] );
    }
}

const unsigned int segment_k_interp = (degree == 2) ?
    k_interp : k_interp * degree;

// Actual curve fitting
std::vector<LieGroup> curve;
for (unsigned int s=0; s<segments_control_points.size(); ++s)
{
    for (unsigned int t=1; t<=segment_k_interp; ++t)
    {
        // t in [0,1]
        const double t_01 = static_cast<double>(t)/(segment_k_interp);

        std::vector<LieGroup> Qs, Qs_tmp;

        for (const auto m : segments_control_points[s])
            Qs.emplace_back(*m);

        // recursive chunk of the algo,
        // compute tmp control points.
        for (unsigned int i=0; i<degree-1; ++i)
        {
            for (unsigned int q=0; q<Qs.size()-1; ++q)
            {

```

```

        Qs_tmp.push_back( Qs[q].rplus(Qs[q+1].rminus(Qs[q]) * t_01)
        );
    }

    Qs = Qs_tmp;
    Qs_tmp.clear();
}

curve.push_back(Qs[0]);
}
}

return curve;
}

```

## D.3 On-manifold optimization using Ceres and manif

### Group-abstract ‘LocalParameterization’

Benefiting for the possibility of writing generic code with `manif`, one can write a single wrapper-class to implement the local-parameterization required by `Ceres` for any Lie group implemented in `manif`. Following Section 6.2 and Appendix D.2, we define a group-abstract ‘LocalParameterization’ as follows,

Listing D.4: Group-abstract ‘LocalParameterization’

```

template <typename _LieGroup>
class LocalParameterizationFunctor
{
    using LieGroup = _LieGroup;
    using Tangent = typename _LieGroup::Tangent;

    template <typename _Scalar>
    using LieGroupT = typename LieGroup::template LieGroupTemplate<
        _Scalar>;

    template <typename _Scalar>
    using TangentT = typename Tangent::template TangentTemplate<_Scalar
        >;

public:

    LocalParameterizationFunctor() = default;
    virtual ~LocalParameterizationFunctor() = default;

    template<typename T>
    bool operator()(const T* state_raw,

```

```

        const T* delta_raw,
        T* state_plus_delta_raw) const
{
    const Eigen::Map<const LieGroupT<T>> state(state_raw);
    const Eigen::Map<const TangentT<T>> delta(delta_raw);
    Eigen::Map<LieGroupT<T>> state_plus_delta(state_plus_delta_raw);
    state_plus_delta = state + delta;
    return true;
}
};
//
...

// Some typedef helpers
using LocalParameterizationS02 = LocalParameterizationFunctor<S02d>;
using LocalParameterizationSE2 = LocalParameterizationFunctor<SE2d>;
using LocalParameterizationS03 = LocalParameterizationFunctor<S03d>;
using LocalParameterizationSE3 = LocalParameterizationFunctor<SE3d>;

// Helper function to create a Ceres
// autodiff local parameterization wrapper.
template <typename _LieGroup>
std::shared_ptr<
    ceres::AutoDiffLocalParameterization<
        LocalParameterizationFunctor<_LieGroup>,
        _LieGroup::RepSize, _LieGroup::DoF>
    >
make_local_parameterization_autodiff()
{
    return std::make_shared<
        ceres::AutoDiffLocalParameterization<
            LocalParameterizationFunctor<_LieGroup>,
                _LieGroup::RepSize,
                _LieGroup::DoF>>();
}

```

Its use is shown below in Appendix D.3.

## A Ceres problem example

This example highlights the use of the predefined Ceres helper classes available with `manif`. In this example, we compute some average point from 4 points in  $SE(2)$ .

Listing D.5: Ceres problem

```

// Tell Ceres not to take ownership of the raw pointers
Ceres::Problem::Options problem_options;
problem_options.cost_function_ownership = Ceres::
    DO_NOT_TAKE_OWNERSHIP;
problem_options.local_parameterization_ownership =

```

```

Ceres::DO_NOT_TAKE_OWNERSHIP;

Ceres::Problem problem(problem_options);

// We use a first manif helper that creates a Ceres cost-function.
// The cost function computes the distance between
// the desired state and the current state

// Create 4 objectives which are 'close' in SE2.
std::shared_ptr<Ceres::CostFunction> obj_pi_over_4 =
    manif::make_objective_autodiff<SE2d>(3, 3, M_PI/4.);

std::shared_ptr<Ceres::CostFunction> obj_3_pi_over_8 =
    manif::make_objective_autodiff<SE2d>(3, 1, 3.*M_PI/8.);

std::shared_ptr<Ceres::CostFunction> obj_5_pi_over_8 =
    manif::make_objective_autodiff<SE2d>(1, 1, 5.*M_PI/8.);

std::shared_ptr<Ceres::CostFunction> obj_3_pi_over_4 =
    manif::make_objective_autodiff<SE2d>(1, 3, 3.*M_PI/4.);

SE2d average_state(0,0,0);

////////////////////////////////////

// Add residual blocks to Ceres problem
problem.AddResidualBlock( obj_pi_over_4.get(),
                          nullptr,
                          average_state.data() );

problem.AddResidualBlock( obj_3_pi_over_8.get(),
                          nullptr,
                          average_state.data() );

problem.AddResidualBlock( obj_5_pi_over_8.get(),
                          nullptr,
                          average_state.data() );

problem.AddResidualBlock( obj_3_pi_over_4.get(),
                          nullptr,
                          average_state.data() );

// We use a second manif helper that creates
// a Ceres local parameterization
// for our optimized state block.

std::shared_ptr<Ceres::LocalParameterization> local_parameterization
=

```

```
    manif::make_local_parametrization_autodiff<SE2d>();

problem.SetParameterization( average_state.data(),
                             local_parameterization.get() );

// Run the solver!
Ceres::Solver::Options options;
options.minimizer_progress_to_stdout = true;

Ceres::Solver::Summary summary;
Ceres::Solve(options, &problem, &summary);

std::cout << "summary:\n" << summary.FullReport() << "\n";

std::cout << "Average state:\nx:" << average_state.x()
          << "\ny:" << average_state.y()
          << "\nt:" << average_state.angle()
          << "\n\n";
```





# Bibliography

- [1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: An open-source robot operating system”, in *ICRA Workshop on Open Source Software*, vol. 3, Kobe, 2009, p. 5.
- [2] H. Durrant-Whyte and T. Bailey, “Simultaneous localisation and mapping (SLAM): Part I the essential algorithms”, *IEEE Robotics Autom. Mag.*, vol. 13, no. 2, pp. 99–110, 2006.
- [3] T. Bailey and H. Durrant-Whyte, “Simultaneous localisation and mapping (SLAM): Part II state of the art”, *IEEE Robotics Autom. Mag.*, vol. 13, no. 3, pp. 108–117, 2006.
- [4] G. Dissanayake, S. Huang, Z. Wang, and R. Ranasinghe, “A review of recent developments in simultaneous localization and mapping”, in *Proc. Int. Conf. Industrial Inf. Syst.*, Kandi, Sri Lanka, 2011, pp. 477–482.
- [5] S. Huang and G. Dissanayake, “A critique of current developments in simultaneous localization and mapping”, *Int. J. Adv. Robotic Syst.*, vol. 13, no. 5, 2016.
- [6] 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. Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.
- [7] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM”, in *Proc. 13th Eur. Conf. Comput. Vis*, ser. Lect. Notes Comput. Sci. Vol. 8689, Zurich, 2014, pp. 834–849.
- [8] R. Mur-Artal, J. Montiel, and J. D. Tardós, “ORB-SLAM: A versatile and accurate monocular SLAM system”, *IEEE Trans. Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [9] M. W.M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, “A solution to the simultaneous localization and map building (SLAM) problem”, *IEEE Trans. Robotics Autom.*, vol. 17, no. 3, pp. 229–241, 2001.
- [10] R. M. Eustice, H. Singh, and J. J. Leonard, “Exactly sparse delayed-state filters for view-based SLAM”, *IEEE Trans. Robotics*, vol. 22, no. 6, pp. 1100–1114, 2006.

- [11] J. Nieto, T. Bailey, and E. Nebot, “Scan-SLAM: Combining EKF-SLAM and scan correlation”, in *Proc. 5th Int. Conf. Field and Service Robotics*, ser. Springer Tracts Adv. Robotics, vol. 25, Port Douglas: Springer, 2005, pp. 167–178.
- [12] K. Konolige and M. Agrawal, “FrameSLAM: From bundle adjustment to real-time visual mapping”, *IEEE Trans. Robotics*, vol. 24, no. 5, pp. 1066–1077, 2008.
- [13] V. Ila, J. M. Porta, and J. Andrade-Cetto, “Information-based compact pose SLAM”, *IEEE Trans. Robotics*, vol. 26, no. 1, pp. 78–93, 2010.
- [14] R. Smith, M. Self, and P. Cheeseman, “A stochastic map for uncertain spatial relationships”, in *Proc. 4th Int. Sym. Robotics Res.*, Santa Cruz, 1987, pp. 467–474.
- [15] P. Moutarlier and R. Chatila, “Stochastic multisensory data fusion for mobile robot location and environment modeling”, in *Proc. 5th Int. Sym. Robotics Res.*, Tokyo, 1989.
- [16] J. J. Leonard, H. F. Durrant-Whyte, and I. J. Cox, “Dynamic map building for an autonomous mobile robot”, *Int. J. Robotics Res.*, vol. 11, no. 4, pp. 286–292, 1992.
- [17] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: A factored solution to the simultaneous localization and mapping problem”, in *Proc. 18th Nat. Conf. Artif. Intell.*, Edmonton, 2002, pp. 593–598.
- [18] J. L. Blanco, J. A. Fernández-Madrigal, and J. Gonzalez, “A novel measure of uncertainty for mobile robot SLAM with Rao-Blackwellized particle filters”, *Int. J. Robotics Res.*, vol. 27, no. 1, pp. 73–89, 2008.
- [19] F. Dellaert and M. Kaess, “Square root SAM: Simultaneous localization and mapping via square root information smoothing”, *Int. J. Robotics Res.*, vol. 25, no. 12, pp. 1181–1204, 2006.
- [20] L. Polok, V. Ila, M. Solony, P. Smrz, and P. Zemčík, “Incremental block cholesky factorization for nonlinear least squares in robotics”, in *Robotics: Science and Systems*, 2013.
- [21] V. Ila, L. Polok, M. Solony, and P. Svoboda, “Highly efficient compact pose SLAM with SLAM++”, *Comput. Res. Repos.*, vol. abs/1608.03037, 2016.
- [22] M. Kaess, A. Ranganathan, and F. Dellaert, “iSAM: Incremental smoothing and mapping”, *IEEE Trans. Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.
- [23] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, “iSAM2: Incremental smoothing and mapping using the bayes tree”, *Int. J. Robotics Res.*, vol. 31, no. 2, pp. 216–235, 2011.
- [24] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. MIT press, 2011.
- [25] P. J. Besl and N. D. McKay, “A method for registration of 3-D shapes”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.
- [26] F. Lu and E. Milios, “Robot pose estimation in unknown environments by matching 2D range scans”, *Journal of Intelligent & Robotic Systems*, vol. 18, no. 3, pp. 249–275, 1997.

- [27] J. Minguez, F. Lamiroux, and L. Montesano, “Metric-based scan matching algorithms for mobile robot displacement estimation”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Barcelona, 2005, pp. 3568–3574.
- [28] A. Censi, “An ICP variant using a point-to-line metric”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Pasadena, 2008, pp. 19–25.
- [29] A. A. Aghamohammadi, H. D. Taghirad, A. H. Tamjidi, and E. Mihankhah, “Feature-based laser scan matching for accurate and high speed mobile robot localization”, in *Proc. Eur. Conf. Mobile Robots*, Freiburg, 2007, pp. 1–6.
- [30] A. Diosi and L. Kleeman, “Fast laser scan matching using polar coordinates”, *Int. J. Robotics Res.*, vol. 26, no. 10, pp. 1125–1153, 2007.
- [31] G. Huo, L. Zhao, K. Wang, R. Li, and J. Li, “Polar metric-weighted norm-based scan matching for robot pose estimation”, *Discrete Dynamics in Nature and Society*, vol. 2016, 2016.
- [32] E. B. Olson, “Real-time correlative scan matching”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Kobe, 2009, pp. 4387–4393.
- [33] M. Jaimez, J. G. Monroy, and J. González-Jiménez, “Planar odometry from a radial laser scanner. A range flow-based approach”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Stockholm, 2016, pp. 4479–4485.
- [34] M. Jaimez, J. Monroy, M. Lopez-Antequera, and J. Gonzalez-Jimenez, “Robust planar odometry based on symmetric range flow and multi-scan alignment”, *IEEE Trans. Robotics*, vol. 34, pp. 1623–1635, 2018.
- [35] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with Rao-Blackwellized particle filters”, *IEEE Trans. Robotics*, vol. 23, no. 1, pp. 34–46, 2007.
- [36] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2D lidar SLAM”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Stockholm, 2016, pp. 1271–1278.
- [37] A. Oliva and A. Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope”, *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.
- [38] N. Sünderhauf and P. Protzel, “BRIEF-Gist - closing the loop by simple means”, in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Francisco, 2011, pp. 1234–1241.
- [39] C. McManus, B. Upcroft, and P. Newmann, “Scene signatures: Localised and point-less features for localisation”, in *Robotics: Science and Systems*, Berkeley, 2014.
- [40] H. Friedrich, D. Dederscheck, K. Krajsek, and R. Mester, “View-based robot localization using spherical harmonics: Concept and first experimental results”, in *Pattern Recognition*, ser. Lect. Notes Comput. Sci. Vol. 4713, Heidelberg, 2007, pp. 21–31.
- [41] S. Se, D. Lowe, and J. Little, “Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks”, *Int. J. Robotics Res.*, vol. 21, pp. 735–758, 2002.

- [42] P. Newman and K. L. Ho, “SLAM loop closing with visually salient features”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Barcelona, 2005, pp. 635–642.
- [43] J. Collier, S. Se, and V. Kotamraju, “Multi-sensor appearance-based place recognition”, in *Proc. Int. Conf. Comput. and Robot Vis.*, Regina, 2013, pp. 128–135.
- [44] J. Sivic and A. Zisserman, “Video Google: A text retrieval approach to object matching in videos”, in *Proc. IEEE Int. Conf. Comput. Vis.*, Nice, 2003, 1470–1477 vol.2.
- [45] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints”, in *Proc. ECCV Workshop Stat. Learn. Comput. Vis.*, Prague, 2004, pp. 1–22.
- [46] D. Nister and H. Stewenius, “Scalable recognition with a vocabulary tree”, in *Proc. 20th IEEE Conf. Comput. Vis. Pattern Recognit.*, New York, 2006, pp. 2161–2168.
- [47] J. Callmer, K. Granström, J. Nieto, and F. Ramos, “Tree of words for visual loop closure detection in urban SLAM”, in *Proc. Australasian Conf. Robotics Autom.*, Canberra, 2008, pp. 1–8.
- [48] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, “View-based maps”, *Int. J. Robotics Res.*, vol. 29, no. 8, pp. 941–957, 2010.
- [49] D. Galvez-Lopez and J. Tardos, “Bags of binary words for fast place recognition in image sequences”, *IEEE Trans. Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.
- [50] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth, “OpenFABMAP: An open source toolbox for appearance-based loop closure detection”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Saint Paul, 2012, pp. 4730–4735.
- [51] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, “Visual place recognition: A survey”, *IEEE Trans. Robotics Autom.*, vol. 32, no. 1, pp. 1–19, 2016.
- [52] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, ser. The Springer International Series in Engineering and Computer Science. Springer, 1992, vol. 159.
- [53] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd. Cambridge: Cambridge University Press, 2004.
- [54] V. Lepetit, F. Moreno-Noguer, and P. Fua, “EPnP: An accurate  $O(n)$  solution to the PnP problem”, *Int. J. Comput. Vis.*, vol. 81, no. 2, p. 155, 2009.
- [55] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots”, in *Proc. IEEE Int. Conf. Robotics Autom.*, vol. 2, St. Louis, 1985, pp. 500–505.
- [56] S. LaValle and J. Kuffner, “Randomized kinodynamic planning”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Detroit, 1999, pp. 473–479.

- [57] J. Kuffner and S. LaValle, “RRT-connect: An efficient approach to single-query path planning”, in *Proc. IEEE Int. Conf. Robotics Autom.*, vol. 2, San Francisco, 2000, pp. 995–1001.
- [58] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”, *IEEE Trans. Robotics Autom.*, vol. 12, no. 4, pp. 566–580, 1996.
- [59] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning”, *Int. J. Robotics Res.*, vol. 30, no. 7, pp. 846–894, 2011.
- [60] D. J. Webb and J. v. d. Berg, “Kinodynamic RRT\*: Optimal motion planning for systems with linear differential constraints”, *Comput. Res. Repos.*, vol. abs/1205.5088, 2012. arXiv: 1205.5088.
- [61] S. Quinlan and O. Khatib, “Elastic bands: Connecting path planning and control”, in *Proc. IEEE Int. Conf. Robotics Autom.*, vol. 2, Atlanta, 1993, pp. 802–807.
- [62] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, “Efficient trajectory optimization using a sparse model”, in *Proc. Eur. Conf. Mobile Robots*, Barcelona, 2013, pp. 138–143.
- [63] C. Rösmann, F. Hoffmann, and T. Bertram, “Kinodynamic trajectory optimization and control for car-like robots”, in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vancouver, 2017, pp. 5681–5686.
- [64] G. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*, ser. Applied and Numerical Harmonic Analysis. Birkhäuser Boston, 2011.
- [65] R. Howe, “Very basic Lie theory”, *The American Mathematical Monthly*, vol. 90, pp. 600–623, 1983.
- [66] H. Abbaspour and M. Moskowitz, *Basic Lie Theory*. World Scientific, 2007.
- [67] J. Stillwell, *Naive Lie Theory*. Springer-Verlag New York, 2008.
- [68] E. Eade, “Lie groups for 2D and 3D transformations”, Cambridge University, Tech. Rep., 2013.
- [69] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.
- [70] J. Solà, J. Deray, and D. Atchuthan, “A micro Lie theory for state estimation in robotics”, Institut de Robòtica i Informàtica Industrial, Barcelona, Tech. Rep. IRI-TR-18-01, 2018. arXiv: 1812.01537.
- [71] M. Himstedt, J. Frost, S. Hellbach, H. J. Bohme, and E. Maehle, “Large scale place recognition in 2D LIDAR scans using geometrical landmark relations”, in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Chicago, 2014, pp. 5030–5035.
- [72] M. Himstedt and E. Maehle, “Geometry matters: Place recognition in 2D range scans using geometrical surface relations”, in *Proc. Eur. Conf. Mobile Robots*, Lincoln, 2015.
- [73] D. Lodi Rizzini, F. Galasso, and S. Caselli, “Geometric relation distribution for place recognition”, *IEEE Robotics Autom. Lett.*, vol. 4, no. 2, pp. 523–529, 2019.

- [74] G. Tipaldi and K. Arras, “FLIRT - interest regions for 2D range data”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Anchorage, 2010, pp. 3616–3622.
- [75] G. D. Tipaldi, L. Spinello, and W. Burgard, “Geometrical FLIRT phrases for large scale place recognition in 2D range data”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Karlsruhe, 2013, pp. 2693–2698.
- [76] F. Kallasi, D. Lodi Rizzini, and S. Caselli, “Fast keypoint features from laser scanner for robot localization and mapping”, *IEEE Robotics Autom. Lett.*, vol. 1, no. 1, pp. 176–183, 2016.
- [77] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset”, *Int. J. Robotics Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [78] G. Bradski, “Open source computer vision library”, *Dr. Dobb’s Journal of Software Tools*, 2000.
- [79] P. Hansen and B. Browning, “Visual place recognition using HMM sequence matching”, in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Chicago, 2014, pp. 4549–4555.
- [80] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”, *IEEE Trans. Inform. Theory*, vol. 13, no. 2, pp. 260–269, 1967.
- [81] L. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition”, *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [82] R. Shinghal and G. T. Toussaint, “Experiments in text recognition with the modified Viterbi algorithm”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 1, no. 2, pp. 184–193, 1979.
- [83] J. Philbin and A. Zisserman, “Object mining using a matching graph on very large image collections”, in *Proc. 6th Indian Conf. Comput. Vis. Graphics Image Process.*, Bhubanswar, 2008, pp. 738–745.
- [84] P. Turcot and D. Lowe, “Better matching with fewer features: The selection of useful features in large database recognition problems”, in *Proc. ICCV Workshop Emergent Issues Large Amount. Vis. Data*, Kyoto, 2009, pp. 2109–2116.
- [85] A. Howard and N. Roy, *The robotics data set repository (Radish)*, <http://radish.sourceforge.net>, 2003.
- [86] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg, “Hierarchical optimization on manifolds for online 2D and 3D mapping”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Anchorage, 2010, pp. 273–287.
- [87] J. Borenstein, “UMBmark: A benchmark test for measuring odometry errors in mobile robots”, *Proc. SPIE*, vol. 2591, pp. 113–124, 1995.
- [88] A. Kelly, “Fast and easy systematic and stochastic odometry calibration”, in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 4, Sendai, 2004, pp. 3188–3194.
- [89] A. Martinelli and R. Siegwart, “Estimating the odometry error of a mobile robot during navigation”, in *Proc. 1st Eur. Conf. Mobile Robots*, Radziejowice, 2003.

- [90] A. Martinelli, N. Tomatis, and R. Siegwart, “Simultaneous localization and odometry self calibration for mobile robot”, *Auton. Robots*, vol. 22, no. 1, pp. 75–85, 2007.
- [91] A. Censi, A. Franchi, L. Marchionni, and G. Oriolo, “Simultaneous calibration of odometry and sensor parameters for mobile robots”, *IEEE Trans. Robotics*, vol. 29, no. 2, pp. 475–492, 2013.
- [92] R. Kummerle, G. Grisetti, C. Stachniss, and W. Burgard, “Simultaneous parameter calibration, localization, and mapping for robust service robotics”, in *Proc. IEEE Workshop Adv. Robotics Soc. Impacts*, Half-Moon Bay, CA, 2011, pp. 76–79.
- [93] M. D. Cicco, B. D. Corte, and G. Grisetti, “Unsupervised calibration of wheeled mobile platforms”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Stockholm, 2016, pp. 4328–4334.
- [94] T Lupton and S Sukkarieh, “Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions”, *IEEE Trans. Robotics*, vol. 28, no. 1, pp. 61–76, 2012.
- [95] C Forster, L Carlone, F Dellaert, and D Scaramuzza, “On-manifold preintegration for real-time visual-inertial odometry”, *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 1–21, 2017.
- [96] D. Atchuthan, A. Santamaria-Navarro, N. Mansard, O. Stasse, and J. Solà, “Odometry based on auto-calibrating inertial measurement unit attached to the feet”, in *Proc. Eur. Control. Conf.*, Limassol, 2018, pp. 3031–3037.
- [97] *Vicon*, <http://www.vicon.com>.
- [98] M. Zefran, V. Kumar, and C. B. Croke, “On the generation of smooth three-dimensional rigid body motions”, *IEEE Trans. Robotics Autom.*, vol. 14, no. 4, pp. 576–589, 1998.
- [99] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Shanghai, 2011, pp. 2520–2525.
- [100] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, “Continuous-time trajectory optimization for online UAV replanning”, in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Daejeon, 2016, pp. 5332–5339.
- [101] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, “Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments”, *IEEE Robotics Autom. Lett.*, vol. 2, no. 3, pp. 1688–1695, 2017.
- [102] S. Lovegrove, A. Patron-Perez, and G. Sibley, “Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras”, in *British Mach. Vis. Conf.*, BMVA Press, 2014, pp. 93.1–93.11.
- [103] T. Mercy, R. V. Parys, and G. Pipeleers, “Spline-based motion planning for autonomous guided vehicles in a dynamic environment”, *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 6, pp. 2182–2189, 2017.

- [104] S. Jalel, P. Marthon, and A. Hamouda, “NURBS based multi-objective path planning”, in *Mex. Conf. on Pattern Recognit.*, vol. 9116, 2015, pp. 190–199.
- [105] ———, “A new path generation algorithm based on accurate NURBS curves”, *Int. J. Adv. Robotic Syst.*, vol. 13, no. 2, p. 75, 2016.
- [106] F. C. Park and B. Ravani, “Bézier curves on Riemannian manifolds and Lie groups with kinematics applications”, *J. Mech. Design*, vol. 117, no. 1, pp. 36–40, 1995.
- [107] T. Popiel and L. Noakes, “Bézier curves and  $C^2$  interpolation in Riemannian manifolds”, *J. of Approx. Theory*, vol. 148, no. 2, pp. 111–127, 2007.
- [108] M. Elbanhawi, M. Simic, and R. Jazar, “In the passenger seat: Investigating ride comfort measures in autonomous cars”, *IEEE Trans. Intell. Transport. Syst.*, vol. 7, no. 3, pp. 4–17, 2015.
- [109] J. Jakubiak, F. S. Leite, and R. C. Rodrigues, “A two-step algorithm of smooth spline generation on riemannian manifolds”, *J. Comput. and Appl. Math.*, vol. 194, no. 2, pp. 177–191, 2006.
- [110] C. Richter, A. Bry, and N. Roy, “Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments”, in *Proc. 17th Int. Sym. Robotics Res. Ser. Springer Tracts in Advanced Robotics*, vol. 114, Sestri Levante: Springer, 2015, pp. 649–666.
- [111] P. F. Felzenszwalb and D. P. Huttenlocher, “Distance transforms of sampled functions”, *Theory of Comput.*, vol. 8, no. 1, pp. 415–428, 2012.
- [112] S. Agarwal, K. Mierle, and Others, *Ceres Solver*, <http://ceres-solver.org>, Google, 2016.
- [113] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. R. Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lütcke, *et al.*, “Ros\_control: A generic and simple control framework for ROS”, *J. Open Source Softw.*, vol. 2, p. 456, 2017.
- [114] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, “CHOMP: Gradient optimization techniques for efficient motion planning”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Kobe, 2009, pp. 489–494.
- [115] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox, “Space-time functional gradient optimization for motion planning”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Hong Kong, 2014, pp. 6499–6506.
- [116] K. He, E. Martin, and M. Zucker, “Multigrid CHOMP with local smoothing”, in *Proc. 11th IEEE-RAS Int. Conf. Humanoid Robots*, Atlanta, 2013, pp. 315–322.
- [117] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “STOMP: Stochastic trajectory optimization for motion planning”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Shanghai, 2011, pp. 4569–4574.
- [118] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, “Finding locally optimal, collision-free trajectories with sequential convex optimization.”, in *Robotics: Science and Systems*, Citeseer, vol. 9, 2013, pp. 1–10.



- [119] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, “Motion planning with sequential convex optimization and convex collision checking”, *Int. J. Robotics Res.*, vol. 33, no. 9, pp. 1251–1270, 2014.
- [120] Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa, “Functional gradient motion planning in reproducing kernel hilbert spaces”, *Robotics: Science and Systems XII*, 2016.
- [121] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, “Motion planning as probabilistic inference using gaussian processes and factor graphs.”, in *Robotics: Science and Systems*, vol. 12, 2016.
- [122] M.-J. Tsai, “Workspace geometric characterization and manipulability of industrial robots”, PhD thesis, The Ohio State University, 1986.
- [123] J. Pan, S. Chitta, and D. Manocha, “FCL: A general purpose library for collision and proximity queries”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Saint Paul, 2012, pp. 3859–3866.
- [124] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “G2o: A general framework for graph optimization”, in *Proc. IEEE Int. Conf. Robotics Autom.*, Shanghai, 2011, pp. 3607–3613.
- [125] I. A. Sucan and S. Chitta, *Moveit!*, [online] <http://moveit.ros.org>, 2013.
- [126] J. Deray and J. Solà, “Manif: A micro lie theory library for state estimation in robotics applications”, *J. Open Source Softw.*, vol. 5, no. 46, p. 1371, 2020.
- [127] F. Dellaert, “Factor graphs and GTSAM: A hands-on introduction”, Georgia Institute of Technology, Tech. Rep. GT-RIM-CP&R-2012-002, 2012.
- [128] J. L. Blanco, *Development of Scientific Applications with the Mobile Robot Programming Toolkit*. University of Malaga, 2008.
- [129] C. Gehring, *Kinematics and dynamics for robotics*, [online] <https://github.com/ANYbotics/kindr>, 2018.
- [130] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, “Manopt, a Matlab toolbox for optimization on manifolds”, *J. Mach. Learning Res.*, vol. 15, pp. 1455–1459, 2014.
- [131] R. Wagner, O. Birbach, and U. Frese, “Rapid development of manifold-based graph optimization systems for multi-sensor calibration and slam”, in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Francisco, 2011, pp. 3305–3312.
- [132] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, “SymPy: Symbolic computing in python”, *PeerJ Computer Science*, vol. 3, e103, 2017.
- [133] J. Townsend, N. Koep, and S. Weichwald, “Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation”, *J. Mach. Learning Res.*, vol. 17, no. 137, pp. 1–5, 2016.

- [134] A. Censi, *Pygeometry*, [online] <https://andreacensi.github.io/geometry>, 2011.
- [135] C. Hertzberg, *Manifold toolkit*, [online] <https://openslam-org.github.io/MTK.html>, 2008.
- [136] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, “Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds”, *J. Inform. Fusion*, vol. 14, no. 1, pp. 57–77, 2013.
- [137] E. Eade, *Liegroups*, [online] <https://bitbucket.org/ethaneade/liegroups/src/master/>, 2013.
- [138] H. Strasdat, *Sophus*, [online] <https://github.com/strasdat/Sophus>, 2018.
- [139] L. Koppel and S. L. Waslander, “Manifold geometry with fast automatic derivatives and coordinate frame semantics checking in C++”, in *Proc. 15th Conf. Comput. Robot Vision*, 2018.
- [140] —, “Manifold geometry with fast automatic derivatives and coordinate frame semantics checking in C++”, in *Int. Conf. Comput. and Robot Vis.*, Toronto: IEEE Computer Society, 2018, pp. 126–133.
- [141] S. Koranne, “Boost c++ libraries”, in *Handbook of Open Source Tools*. Boston, MA: Springer US, 2011, pp. 127–143.
- [142] G. Guennebaud, B. Jacob, *et al.*, *Eigen v3*, [online] <http://eigen.tuxfamily.org>, 2010.
- [143] D. Vandevorde, N. M. Josuttis, and D. Gregor, *C++ Templates: The Complete Guide*, 2nd. Addison-Wesley Professional, 2017.
- [144] M. Biel and M. Norrlöf, “Efficient trajectory reshaping in a dynamic environment”, in *Proc. 15th IEEE Int. Workshop on Advanced Motion Control*, 2018, pp. 54–59.
- [145] B. Gerkey, R. Vaughan, and A. Howard, “The player/stage project: Tools for multi-robot and distributed sensor systems”, in *Proc. Intl. Conf. on Adv. Robotics*, 2003.
- [146] J.-C. Baillie, A. Demaille, Q. Hocquet, M. Nottale, and S. Tardieu, “The urbi universal platform for robotics”, in *Proc 1st Int. Workshop on Stand. and Common Platf. for Robotics*, 2008.
- [147] G. Metta, P. Fitzpatrick, and L. Natale, “Yarp: Yet another robot platform”, *Int. J. Adv. Robotic Syst.*, vol. 3, no. 1, p. 8, 2006. eprint: <https://doi.org/10.5772/5761>.
- [148] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator”, in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sendai, 2004, pp. 2149–2154.
- [149] W. Boehm and A. Müller, “On de casteljau’s algorithm”, *Comput. Aided Geom. Design*, vol. 16, no. 7, pp. 587–605, 1999.
- [150] *Proc. IEEE Int. Conf. Robotics Autom.* Barcelona, 2005.
- [151] *Proc. IEEE Int. Conf. Robotics Autom.* Kobe, 2009.
- [152] *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* Sendai, 2004.

- [153] *Proc. IEEE Int. Conf. Robotics Autom.* Anchorage, 2010.
- [154] *Proc. IEEE Int. Conf. Robotics Autom.* Stockholm, 2016.
- [155] *Proc. IEEE Int. Conf. Robotics Autom.* Saint Paul, 2012.
- [156] *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* Chicago, 2014.
- [157] *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* San Francisco, 2011.
- [158] *Proc. IEEE Int. Conf. Robotics Autom.* Shanghai, 2011.