Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

On the Uncertainty in Active SLAM: Representation, Propagation and Monotonicity

María Luisa Rodríguez Arévalo

Ph.D. Dissertation

Advisor: José Ángel Castellanos Gómez

Departamento de Informática e Ingeniería de Sistemas

Escuela de Ingeniería y Arquitectura

Universidad de Zaragoza, Spain

April 2018

Acknowledgments

The research work reported in this PhD thesis was supported by the "Ministerio de Economía, Industria y Competitividad" (MINECO) of the Government of Spain under projects, "Life-long Active Spatial Cooperative Mapping and Understanding" (DPI2012 – 36070) and "Active Robot Exploration for Dense 3D Mapping" (DPI2015 – 68905P).

Generous funding from the "Subprograma Estatal de Formación, Ayudas para contratos predoctorales para la formación de doctores" and "Subprograma Estatal de Movilidad, Ayudas a la movilidad predoctoral para la realización de estancias breves en centros de I+D" of the Government of Spain under grants BES2013-064129, EEBB-I-16-11543 and EEBB-I-17-12078 was also received.

Resumen

La localización y mapeo simultáneo activo (SLAM activo) ha recibido mucha atención por parte de la comunidad de robótica por su relevancia en aplicaciones de robot móviles. El objetivo de un algoritmo de SLAM activo es planificar la trayectoria del robot para maximizar el área explorada y minimizar la incertidumbre asociada con la estimación de la posición del robot. Durante la fase de exploración de un algoritmo de SLAM, donde el robot navega en una región previamente desconocida, la incertidumbre asociada con la localización del robot crece sin límites. Solo después de volver a visitar regiones previamente conocidas, se espera una reducción en la incertidumbre asociada con la localización del robot mediante la detección de cierres de bucle. Esta tesis doctoral se centra en la importancia de *representar* y *cuantificar* la incertidumbre para calcular correctamente la confianza asociada con la estimación de la localización del robot en cada paso de tiempo a lo largo de su recorrido y, por lo tanto, decidir la trayectoria correcta de acuerdo con el objetivo de SLAM activo.

En la literatura, se han propuesto fundamentalemente dos tipos de modelos de representación de la incertidumbre: *absoluta* y *diferencial*. En representación absoluta, la información sobre la incertidumbre asociada con la localización del robot está representada por una función de distribución de probabilidad, generalmente gausiana, sobre las variables de localización absoluta con respecto a una referencia base elegida. La estimación de la posición del robot está dada por la esperanza de las variables asociadas con la localización y la incertidumbre por su matriz de covarianza asociada. La representación diferencial utiliza una representación local de la incertidumbre, la posición estimada del robot se representa mediante la mejor aproximación de la posición absoluta y el error de estimación se representa localmente mediante un vector diferencial. Este vector generalmente también está representado por una función de distribución de probabilidad gausiana. Representaciones equivalentes al modelo diferencial han utilizado las herramientas de Grupos de Lie y Álgebras de Lie para representar la incertidumbre. Además de estos modelos, existen diferentes formas de representar la posición y orientación de la posición del robot, ángulos de Euler, cuaterniones y transformaciones homogéneas.

Los enfoques más comunes para cuantificar la incertidumbre en SLAM se basan en criterios de optimalidad con el objetivo de cuantificar el mapa y la incertidumbre de la posición del robot: *A-opt* (traza de la matriz de covarianza, o suma de sus autovalores), *D-opt* (determinante de la matriz de covarianza, o producto de sus autovalores) y *E-opt* (criterio del mayor autovalor). Alternativamente, otros algoritmos de SLAM activo, basados en la Teoría de la Información, se basan en el uso de la entropía de Shannon para seleccionar acciones que lleven al robot al objetivo seleccionado. En un escenario de SLAM activo, garantizar la monotonicidad de estos criterios en la toma de decisiones durante la exploración, es decir, cuantificar correctamente que la incertidumbre encapsulada en una matriz de covarianza está aumentando, es un paso esencial para tomar decisiones correctas. Como ya se ha mencionado, durante la fase de exploración la incertidumbre asociada con la localización del robot aumenta. Por lo tanto, si no se preserva la monotonicidad de los criterios considerados, el sistema puede seleccionar trayectorias o caminos que creen falsamente que conducen a una menor incertidumbre de la localización del robot.

En esta tesis, revisamos el trabajo relacionado sobre representación y propagación de la incertidumbre de la posición del robot en los diferentes modelos propuestos en la literatura. Además, se lleva a cabo un análisis de la incertidumbre representada localmente con un vector diferencial y la incertidumbre representada usando grupos de Lie. Investigamos la monotonicidad de diferentes criterios para la toma de decisiones, tanto en 2D como en 3D, dependiendo de la representación de la incertidumbre y de la representación de la orientación del robot. Nuestra conclusión fundamental es que la representación de la incertidumbre sobre grupos de Lie y usando un vector diferencial son similares e independientes de la representación utilizada para la parte rotacional de la posición del robot. Esto se debe a que la incertidumbre se representa localmente en el espacio de las transformaciones diferenciales que se corresponde con el álgebra de Lie del grupo euclidiano especial SE(n). Sin embargo, en el espacio tridimensional, la estimación de la localización del robot depende de las diferentes formas de representación de la parte rotacional. Por lo tanto, una forma adecuada de manipular conjuntamente la estimación y la incertidumbre del robot es utilizando la teoría de grupos de Lie debido a que es una representación que garantiza propiedades tales como una representación mínima y libre de singularidades en los ángulos de rotación. Analíticamente, demostramos que, utilizando representaciones diferenciales para la propagación de la incertidumbre, la monotonicidad se conserva para todos los criterios de optimalidad, *A-opt*, *D-opt* y *E-opt* y para la entropía de Shannon. También demostramos que la monotonicidad no se

cumple para ninguno de ellos en representaciones absolutas usando ángulos Roll-Pitch-Yaw y Euler. Finalmente, mostramos que al usar cuaterniones unitarios en representaciones absolutas, los únicos criterios que preservan la monotonicidad son *D-opt* y la entropía de Shannon.

Estos hallazgos pueden guiar a los investigadores de SLAM activo a seleccionar adecuadamente un modelo de representación de la incertidumbre, de modo que la planificación de trayectorias y los algoritmos de exploración puedan evaluar correctamente la evolución de la incertidumbre asociada a la posición del robot.

Abstract

Active Simultaneous Localization and Mapping (Active SLAM) has received a lot of attention from the robotics community for its relevance in mobile robotics applications. The objective of an active SLAM algorithm is to plan ahead the robot motion in order to maximize the area explored and minimize the uncertainty associated with the estimation, all within a time and computation budget. During the exploration phase of a SLAM algorithm, where the robot navigates in a previously unknown region, the uncertainty associated with the robot's localization grows unbounded. Only after revisiting previously known regions a reduction in the robot's localization uncertainty is expected by detecting loop-closures. This doctoral thesis focuses on the paramount importance of *representing* and *quantifying* uncertainty to correctly report the associated confidence of the robot's location estimate at each time step along its trajectory and therefore deciding the correct course of action in an active SLAM mission.

Two fundamental types of models of probabilistic representation of the uncertainty have been proposed in the literature: *absolute* and *differential*. In absolute representations, the information about the uncertainty in the location of the robot's pose is represented by a probability distribution function, usually Gaussian, over the variables of the absolute location with respect to a chosen base reference. The estimated location is given by the expected location variables and the uncertainty by its associated covariance matrix. Differential representations use a local representation of the uncertainty, the estimated location of the robot is represented by the best approximation of the absolute location and the estimation error is represented locally by a differential location vector. This vector is usually also represented by a Gaussian probability distribution function. Equivalent representations to differential models have used the tools of Lie groups and Lie algebras to represent uncertainties. In addition to uncertainty models, there are different ways to represent the position and orientation of the robot's pose, Euler angles, quaternions and homogeneous transformations.

The most common approaches to quantifying uncertainty in SLAM are based on optimality criteria which aim at quantifying the map and robot's pose uncertainty, namely *A-opt* (trace of the covariance matrix, or sum of its eigenvalues), *D-opt* (determinant of the covariance matrix, or product of

its eigenvalues) and *E-opt* (largest eigenvalue) criteria. Alternatively, other active SLAM algorithms, based on Information Theory, rely on the use of the Shannon's entropy to select courses of action for the robot to reach the commanded goal location. In an active SLAM scenario, guaranteeing monotonicity of these decision making criteria during exploration, i.e. quantifying correctly that the uncertainty encapsulated in a covariance matrix is increasing, is an essential step towards making correct decisions. As already mentioned, during exploration the uncertainty associated with the robot's localization increases. Therefore, if monotonicity of the criteria considered is not preserved, the system might select courses of action or paths that it falsely believes lead to less uncertainty in the robot.

In this thesis, we review related work about representation and propagation of the uncertainty of robot's pose and present a survey of different types of models proposed in the literature. Additionally, an analysis of the uncertainty represented with a differential uncertainty vector and the uncertainty represented on Lie groups is carried out. We investigate the monotonicity of different decision making criteria, both in 2D and 3D, depending on the representation of uncertainty and the orientation of the robot's pose. Our fundamental conclusion is that uncertainty representation over Lie groups and using differential location vectors are similar and independent of the representation used for rotational part of the robot's pose. This is due to the uncertainty is represented locally in the space of differential transformations for translation and rotation that correspond with the Lie algebra of special Euclidean group SE(n). However, in 3-dimensional space, the homogeneous transformation associated to the approximation of the real location depend on the different ways of representation the rotational part. Therefore, a proper way to jointly manipulating the estimation and uncertainty of the pose is to use the theory of Lie groups due to it is a representation to guarantee properties such as a minimal representation and free of singularities in rotation angles. We analytically show that, using differential representations to propagate spatial uncertainties, monotonicity is preserved for all optimality criteria, *A-opt*, *D-opt* and *E-opt* and for Shannon's entropy. We also show that monotonicity does not hold for any of them in absolute representations using Roll-Pitch-Yaw and Euler angles. Finally, we show that using unit quaternions in absolute representations, the only criteria that preserve monotonicity are *D-opt* and Shannon's entropy.

These findings can guide active SLAM researchers to adequately select a representation model for uncertainty, so that path planning and exploration algorithms can correctly assess the evolution of location uncertainty.

Contents

CHAPTER 1

Introduction

To perform a task, a robot needs an operative model of the environment. If the desired task has to be done without human intervention, i.e. full autonomy, the robot should at least be endowed with the capability of solving three problems: localization, mapping and motion control (Jefferies and Yeap, 2008), (Siegwart et al., 2011). Solving these three problems jointly in order to obtain the most accurate representation of the environment in a finite time is known as Active Simultaneous Localization and Mapping (Active SLAM) (Feder et al., 1999), (Leung et al., 2006), and it is the main theme of this doctoral thesis. This problem is considered as one of the most challenging problems of mobile robotics (Cadena et al., 2016).

Active SLAM refers to a solution of the robotic exploration problem in which the main concern of the robot performing the exploration task is to obtain the most accurate representation of the environment, i.e. uncertain as well as unknown parts of the environment are of the utmost interest. In other words, the objective of Active SLAM is to plan ahead the motion of the vehicle in order to maximize the explored areas and minimize the uncertainty associated with the estimation. In this setting, the robot must make a trade-off between exploring new area to complete the task and exploiting the existing information to maintain good localization, i.e. solving the so-called exploration-exploitation dilemma.

Selecting actions that decrease the map uncertainty whilst not significantly increasing the robot's localization uncertainty is challenging. Active SLAM algorithms select the next robot's action based on its utility to solve the exploration-exploitation dilemma. The exploration involves moving in previously unvisited parts with the objective of increasing the overall know-

ledge of the environment, while the latter is exploitation, i.e., it involves revisiting areas to maximize the information gain. Therefore, a definition of Active SLAM can be stated as a problem of controlling the movements of a robot performing SLAM so as to maximize the accuracy of its map representation and localization. This definition involves both Bajcsy's active perception (Bajcsy, 1988) and Thrun's robotic exploration (Thrun et al., 2005) paradigms.

An active SLAM algorithm usually starts by identifying and picking potential destinations that help a robot to solve the exploration-exploitation dilemma. Ideally a robot executing an active SLAM algorithm should evaluate every possible action in the robot and map space, but this has an exponential growth that proves to be computationally intractable in real applications (Burgard et al., 2005), (Martinez-Cantin et al., 2009). In concordance with the literature, an active SLAM algorithm can be divided in three steps (Bourgault et al., 2002), (Makarenko et al., 2002), (Smith et al., 2003):

- The robot identifies and select possible locations to explore or exploit, i.e. vantage locations, in its current estimate of the map.

- The robot computes the utility of taking every action available to it and selects the action with the highest utility.

- The robot carries out the selected action and decide if it is necessary to continue or to terminate the task.

In practice, a small subset of locations in the map is selected based on the information of its neighborhood, using techniques such as frontier-based exploration (Yamauchi, 1997) where robots seek known areas in the map that are adjacent to unexplored space. In the second step, the robot computes a utility or cost function to evaluate the effect of each candidate action. The two most common approaches to quantify utility are based in the Information Theory (Cover and Thomas, 2012) and the Theory of Optimal Experimental Design (Pukelsheim, 2006). The third step deals with the execution of the selected action and the decision of continuing with the active SLAM algorithm or terminate it.

1.1 Contributions

In this thesis the focus is on the paramount importance of *representing* and *quantifying* uncertainty to correctly report the associated confidence of the robot's location estimate at each time step along its trajectory and therefore deciding the correct course of action in an active SLAM mission. While the action set generation is critical since it determines which actions are finally executed, the evaluation of the actions is also important since a good evaluation is useless if the system select courses of action or paths that it falsely believes lead to less uncertainty in the robot.

Estimating uncertain spatial relationship is fundamentally important problem and its representation and estimation are complicated in practice because the location of one object relative to another may be known only indirectly through a sequence of relative frames of reference with uncertainty. Therefore, to reason the effects of uncertainties, uncertainties must be properly represented such as their manipulation can be performed. Two fundamental types of models of probabilistic representation of the uncertainty have been proposed in the literature: *absolute* and *differential*. In absolute representations (Bolle and Cooper, 1986), (Smith and Cheeseman, 1986), the information about the uncertainty in the location of the robot's pose is represented by a probability distribution function, usually Gaussian, over the variables of the absolute location with respect to a chosen base reference. The estimated location is given by the expected location variables and the uncertainty by its associated covariance matrix.

Differential representations use a local representation of the uncertainty (Durrant-Whyte, 1987), (Su and Lee, 1992), (Castellanos et al., 1999); the estimated location of the robot is represented by the best approximation of the absolute location and the estimation error is represented locally by a differential location vector. This vector is usually also represented by a Gaussian probability distribution function. Recently, equivalent representations on the Lie groups have also been considered in recent works (Smith et al., 2003), (Wang and Chirikjian, 2006), (Barfoot and Furgale, 2014) where transformations are represented by homogeneous matrices and uncertainties in a pose or motion correspond to probability density functions over Lie groups. How to propagate the local uncertainty has been studied by different arguments, such as differential homogeneous transforms (Su and Lee, 1992), representing probability density functions over Euclidean transformations (Smith et al., 2003), propagating error densities over Lie groups (Wang and Chirikjian,

2006) and using the representation of uncertain geometric information by means of the Symmetries and Perturbations Model (SPmodel) (Castellanos et al., 1999), reaching the same conclusion.

The most common approaches to quantifying uncertainty in SLAM are based on real scalar functions of the covariance matrix. Some active SLAM algorithms rely on optimality criteria which aim at quantifying the map and robot's pose uncertainty, namely *A-opt* (trace of the covariance matrix, or sum of its eigenvalues) (Chernoff, 1953), (Leung et al., 2006), (Kollar and Roy, 2008), (Meger et al., 2008), *D-opt* (determinant of the covariance matrix, or product of its eigenvalues) (Wald, 1943), (Vidal-Calleja et al., 2006), (Kim and Eustice, 2013) and *E-opt* (largest eigenvalue) (Ehrenfeld, 1955) criteria. Alternatively, other active SLAM algorithms, based on Information Theory (Stachniss et al., 2005), (Blanco et al., 2008), rely on the use of the Shannon's entropy to select courses of action for the robot to reach the commanded goal location.

In an active SLAM scenario, guaranteeing monotonicity of these decision making criteria during exploration, i.e. quantifying correctly that the uncertainty encapsulated in a covariance matrix is increasing, is an essential step towards making correct decisions. During the exploration phase of a SLAM algorithm, where the robot navigates in a previously unknown region, the uncertainty associated with the robot's localization grows unbounded (Kelly, 2004). Only after revisiting previously known regions a reduction in the robot's localization uncertainty is expected by detecting loop-closures (Durrant-Whyte and Bailey, 2006). Therefore, if monotonicity of the criteria considered is not preserved, the system might select courses of action or paths that it falsely believes lead to less uncertainty in the robot.

The results of this investigation are the utmost importance to adequately select a representation model for uncertainty with the aim to correctly assess the evolution of location uncertainty in an path planning or exploration algorithms.

In addition, in order to continue our future research on the active SLAM topic, other areas related such as Partially Observable Markov Decision Process (POMDPs) and Deep Learning have been studied to develop novel algorithms with real applications.

POMDPs have been studied because Active SLAM is an instance of a POMDPs framework. POMDP describes the process of making decisions when both the actions and the sensing are uncertain, i.e. the state of interest

is not directly observable. As future work, based on the knowledge acquired, the development of an active SLAM algorithm with restrictions is pursuit.

Currently, one of the new research lines is Deep Learning. Therefore, a review of the state of the art on the field of action of Deep Learning in SLAM was carried out and the study of Deep Learning and the techniques developed in this area for possible future research.

In summary, the main contributions of the thesis are encompassed under the topic of active SLAM, and in particular are:

- We review related work about representation and propagation of the uncertainty and present a survey of different types of models proposed in the literature.
- We show that uncertainty represented using differential representation and the uncertainty representation based on Lie groups are similar and independent of the representation used for rotational part of the robot's pose. The analysis is carried out taking into account the similarities between both models of representations. We conclude that, a proper way to jointly manipulation the estimation of the robot's and its uncertainty is to use the theory of Lie groups due to, in 3-dimensional space, it is a representation that guarantees relevant properties such as a minimal representation and absence of singularities in rotation angles.
- We investigate the monotonicity of different decision making criteria, both in 2-dimensional and 3-dimensional space, depending on the representation of uncertainty and orientation of the robot's pose. We analytically show that, using differential representations to propagate spatial uncertainties, monotonicity is preserved for all optimality criteria, *A-opt*, *D-opt* and *E-opt* and for Shannon's entropy. We also show that monotonicity does not hold for any of them in absolute representations using Roll-Pitch-Yaw and Euler angles. Finally, we show that using unit quaternions in absolute representations, the only criteria that preserve monotonicity are *D-opt* and Shannon's entropy.
- We present a review of the mathematical formulation about Partially Observable Markov Decision Process framework that has been studied during the research stay in The Robotics, Vision and Control Group of the department Ingeniería de Sistemas y Automática of Escuela Técnica Superior de Ingenieros (Universidad de Sevilla).

- We present a review of the state of the art on the field of Deep Learning in SLAM and an introduction of the basic definitions of Deep Learning framework and the techniques developed in this area for possible future research. This work was perform during a research stay at the Australian Center for Robotic Vision of the University of Adelaide.

1.2 Thesis Structure

The structure of the rest of this thesis is as follows:

In chapter 2, we present the active SLAM definition and general algorithm and review the related work in the literature. In chapter 3, we provide an overview of mathematical notation about the different ways to represent the robot's pose in 2-dimensional and 3-dimensional space used throughout the thesis. Among others, we review the representation of the position and orientation in homogeneous transformations matrices and location vectors and their fundamental properties. Additionally, we provide enough information about Lie group representing spatial transformations and derives useful formulae for working with the Lie groups that represent transformations in 2-dimensional and 3-dimensional space. In chapter 4, we review the different types of models proposed to uncertainty representation and present a survey of absolute, differential and over Lie groups uncertainty representation. Additionally, we look into the similarities and differences between the uncertainty represented with a differential error and the uncertainty represented over Lie groups theory. In chapter 5, we investigate the monotonicity of optimality criteria and the Shannon's entropy taking into account the different types of models proposed to uncertainty representation. We present the conclusions of this thesis in chapter 6. Finally, in Appendix A and Appendix B, we present the research carried out during research visit other groups.

1.3 Publications

The work described in this thesis resulted in the following publications:

- Carrillo, H., Latif, Y., Rodriguez-Arevalo, M. L., Neira, J., and Castellanos, J. A. (2015). On the monotonicity of optimality criteria during

exploration in active SLAM. In *IEEE International Conference In Robotics and Automation* (ICRA), pages 1476-1483. Nominated for the Best Student Paper Awards. Conference Rank: CORE2018 B.

- Rodriguez-Arevalo, M. L., Neira, J., and Castellanos, J. A. (2018). On the Importance of Uncertainty Representation in Active SLAM. In *IEEE Transactions on Robotics*. (Early Access)
  DOI: 10.1109/TRO.2018.2808902. Journal Rank: Q1.

- Rodriguez-Arevalo, M. L., Neira, J., and Castellanos, J. A. Uncertainty Representation and Propagation Review: Differential vs Lie algebra. *IEEE Transactions on Robotics* (In preparation). Journal Rank: Q1.

CHAPTER 2

Active SLAM

True autonomy and safe, intelligent, and purposeful behaviour of a mobile system in uncontrolled, time-changing scenarios requires a precise understanding of the environment where the system operates and the awareness of its place and the place of other entities of interest within it at all times. In mobile robotics, this problem is classically known as Simultaneous Location and Mapping (SLAM). SLAM initially referred to the problem of determining the position and heading of a moving sensor in an unknown environment and, concurrently, computing a map from the perceived surroundings, taking into account sensor and vehicle errors. A closely related and also classical problem in Computer Vision is Structure from Motion (SfM), the computation of the structure of a static object perceived from a camera in motion (or vice-versa).

Significant scientific contributions have populated both the robotics and computer vision literature and are helping to pave the way towards real applications. Currently, challenging open problems include the computation of truly dense geometric environment models of large environments, the inclusion of semantic information such as the identification of objects and places in the environment, the fusion of heterogeneous information sources such as a priori models obtained from large databases such as aerial photography or geographical information systems, the development of collaborative active exploration strategies that can consider diverse types of vehicles and other potentially independent mobile agents such as humans wearing sensors.

However, one of the most challenging problems of mobile robotics in an unknown environment is the decision making to improve the quality of the SLAM results and is referred as Active SLAM (Cadena et al., 2016). The

objective of an active SLAM algorithm is to plan ahead the robot motion in order to maximize the area explored and minimize the uncertainty associated with the estimation, all within a time and computation budget. Active SLAM is non-trivial due to the robot must trade-off the benefits of exploring new areas and exploiting visited areas to close loops (Yamauchi, 1997). New findings in active SLAM implies advances in real applications with autonomous operations under uncertainty that are essential in numerous problem domains, including autonomous navigation, object manipulation, multi-robot localization and tracking, and robotic surgery.

2.1 Active SLAM definition

The problem of active SLAM, often referred to as SPLAM (Simultaneous Planning Localisation and Mapping), focuses on designing robot trajectories to actively explore an environment and minimize the map error. The definition stems from the Bajcsy's active perception (Bajcsy, 1988) and Thrun's robotic exploration (Thrun et al., 2005) paradigms.

The concept of exploration addresses with the process of information gathering (Thrun et al., 2005). In robotics, the exploration refers to the autonomous construction of a model of the operative environment by a robot. To solve the problem of robotic exploration is necessary to solve three fundamental problems in mobile robot, namely localization, mapping and motion control. Localization is the problem of determining the position of the robot within a given map. Mapping refers to the problem of integrating robot's sensor information into a coherent representation. Motion control is the problem of how to steer the robot to a particular location, therefore giving the robot the capacity of performing active behavior.

Active perception is defined as a problem of an intelligent data acquisition process (Bajcsy, 1988) and can be applied to several important problems in mobile robotics. Active perception applied to localization address the problem to determine the movement of robot in order to localize itself with respect to a map, given that the robot knows the truth map of the environment. Specifically, this problem is called active localization and focuses on finding the joint solution of motion control and localization. Some studies on this topic are, for example, performed by Burgard et al. (1997) and Fox et al. (1998) who proposed an active localization approach based on Markov localization. In (Borghi and Caglioti, 1998), the self-localization of a mobile robot within

a known environment, by means of an orientable range finder, is considered. A probabilistic approach for mobile robot localization using an incomplete topological world model is presented by Jensfelt and Kristensen (2001).

Active perception applied to mapping refers to the problem of finding the sequence of movements that represents the most accurate representation of the environment. This problem is called active mapping and focuses on solving the motion control problem and the mapping problem. One of the first approaches of active mapping is the next best view problem (Reed et al., 1997), (Massios et al., 1998), (Pito, 1999). The next best view problem seeks a single additional sensor placement in order to improve an existing scene reconstruction derived from the current imaging configuration. Other authors used different techniques as Barratt (2017) who proposed an approach to learning agents for active robotic mapping based in reinforcement learning, where the goal is to map the environment as quickly as possible.

As it is well known, solving simultaneously the localization and mapping problem is the well known SLAM problem (Thrun et al., 2005). The joint solution of the three problems, active localization, active mapping and SLAM problem is named active SLAM and can be stated as a problem of controlling the movements of a robot performing SLAM so as to maximize the accuracy of its map representation and localization. The uncertainties of the robot, map and sensor measurements, and the dynamic and motion constraints need to be considered in the planning process. In this setting, the robot must make a trade-off between exploring new area to complete the task and exploiting the existing information to maintain good localization, i.e. solving the so-called exploration-exploitation dilemma.

2.2 Active SLAM related work

Several works have focused on the active SLAM problem. The first proposal and implementation of an active SLAM algorithm can be traced back to Feder et al. (1999). They addresses the problem of how to perform concurrent mapping and localization adaptively, i.e. active SLAM, using sonar data and an EKF-based SLAM algorithm. The algorithm proposed is a greedy approach where an action is chosen given the current knowledge to maximise the information gain in the next measurement.

Other similar approaches have been proposed by Bourgault et al. (2002)

and Makarenko et al. (2002). In (Bourgault et al., 2002), the problem of maximizing the accuracy of the map process during exploration by adaptively selecting control actions that maximize localisation accuracy is addressed. The contribution of this paper is the combination of simultaneously maximizing the information gain on the Occupancy Grid map and minimizing the uncertainty of the pose and map feature uncertainty in the SLAM process. Makarenko et al. (2002) considered an exploration strategy which balanced evaluation of alternative motion actions from the point of view of information gain, localization quality, and navigation cost as a form of utility. This factors are only considered at the destination.

There are authors who have done several works on active SLAM as Sim (2005b) who demonstrated that information optimal approaches to active exploration can be detrimental to map quality, particularly when coupled with approximating assumptions that are common in SLAM approaches. Also, the authors presented an alternative approach to exploration that optimizes a map's accuracy by taking a policy that emphasizes the conditioning of the map update step. Sim (2005a) examined the problem of information driven exploration for the purposes of SLAM and demonstrated that simple outlier removal does not significantly improve the performance of an information-driven exploration policy. Sim and Roy (2005) showed that conventional exploration algorithms for collecting map data are sub-optimal in both the objective function and choice of optimization procedure and that optimizing the A-optimal information measure results in a more accurate map than existing approaches, using a greedy, closed-loop strategy. Similarity to Feder et al. (1999), who used Fisher information as a metric in the objective function to construct an adaptive control action, Sim (2005b) and Sim and Roy (2005) used Fisher Information to improve exploration, reporting the need to consider the path in localization and mapping.

Other information metrics within a similar framework, such as the Cauchy-Schwarz quadratic mutual information (Charrow et al., 2015), the D-optimality criterion (Carrillo et al., 2012), and the Kullback-Leibler divergence (Carlone et al., 2010) have also been used. Recently, (Carrillo et al., 2018) presents a novel information theoretic utility function using Shannon and the Rényi entropy for selecting actions in a robot based autonomous exploration task.

Other studies as (Stachniss et al., 2004) conducted an active loop-closing with frontier based exploration in SLAM using occupancy grid and topological maps. The loop-closing approach is based in the entropy to select the best action. However, maximisation of information gain along the path is

not considered. Then, the same authors in (Stachniss et al., 2005) presented an integrated approach to exploration, mapping, and localization using a efficient Rao-Blackwellized particle filter to represent the poses and the maps. In contrast to the previous work (Stachniss et al., 2004), the approach presented in (Stachniss et al., 2005) is based on the expected uncertainty reduction about the trajectory of the robot as well as about maps.

The active SLAM problem is formulated as an optimal trajectory planning problem in (Leung et al., 2006) and (Leung et al., 2008). These works introduced an attractor combined with local planning strategies such as Model Predictive Control. The attractor is used to facilitate the local strategy in optimal trajectory planning. More information of Model Predictive Control can be found in (Morari and Lee, 1999) and (Rawlings, 2000). The active SLAM algorithm proposed in (Leung et al., 2006) and (Leung et al., 2008) is similar to the one of Feder et al. (1999) but with the difference of considering multiples steps in order to make the decision of where to go next. To solve the decision-making problem about efficient area coverage and good SLAM navigation, Kim and Eustice (2015) introduced perception-driven navigation algorithm that automatically balances between exploration and revisitation using a reward framework. Fairfield and Wettergreen (2010) investigated a realtime method for Active SLAM with an approach to SLAM problem that divides the environment up into segments, or submaps, using heuristic methods. In this work actions are selected in order to reduce uncertainty in both the local metric submap and the global topological map.

Recently, Carlone et al. (2014) addressed the problem of active SLAM and exploration with Rao-Blackwellized Particle Filters. They proposed an application of Kullback-Leibler divergence for the purpose of evaluating the particle-based SLAM posterior approximation. They applied this metric in the definition of the expected information from a policy, which allows the robot to autonomously decide between exploration and place revisiting actions. Mu et al. (2016) proposed one of the first active SLAM approach that plans robot paths to directly optimize a global feature based representation. The authors presented a graphical models which utilize independences between variables, and enables a unified quantification of exploration and exploitation gains with a single entropy metric to facilitate a balance between map exploration and refinement. Maurović et al. (2017) proposed a path planning algorithm that is able to continuously improve localization without interrupting the main task based on the modified D* path planning algorithm (see Stentz (1994) for details about D* path planning algorithm).

2.3 Active SLAM algorithm

Autonomous robots performing tasks such as monitoring, surveillance or exploration must be able to plan their future information-gathering actions in environments partially observable and stochastic where planning of the trajectory requires to reason over uncertain outcomes in the presence of sensor noise. Such problems are instances of Partially Observable Markov decision processes (Kaelbling et al., 1998), or POMDPs. Specifically, POMDP is the mathematical framework for modeling the active SLAM problem (Thrun et al., 2005). A POMDP models the process of making decisions when both the actions and the sensing are uncertain to achieve goal.

Formally, a POMDP is a tuple $(S, A, O, T, Z, R, \gamma)$, where S, A and O denote a robot's state space, action space and observation space, respectively. At each time step, the robot takes an action $a \in A$ to move from a state $s \in S$ to $s^{'} \in S$ and receives an observation $o \in O$. The model for the system dynamics is specified by a conditional probability function $T(s, a, s^{'}) = p(s^{'}|s, a)$, which accounts for uncertainty in robot control. The observation model is specified by a conditional probability function $Z(s^{'}, a, o) = p(o|s^{'}, a)$, which accounts for sensing uncertainty. The function $R(s, a)$ specifies a real-valued reward for the robot if it takes action a in state s. The robot's goal is to choose a sequence of actions that maximizes the expected total $E(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t))$, where s_t and a_t denote the system's state and action at time t. The discount factor $\gamma \in [0, 1)$ ensures that the total reward is finite, even when a planning task has an infinite horizon.

POMDP is a principled approach for planning and making decision under uncertainty but it is hard to solve. Some examples works related to autonomous robots are (Bai et al., 2014), (Seiler et al., 2015), (Lauri and Ritala, 2016), (Lauri and Ritala, 2016). Details about POMDPs and different approaches can be found in Appendix A.

In broad terms, an active SLAM algorithm usually starts by identifying and picking potential destinations that help a robot to solve the exploration-exploitation dilemma and then selecting actions based on its utility that decrease the map uncertainty whilst not significantly increasing the robot's localization uncertainty. An active SLAM algorithm can be divided in three steps (Bourgault et al., 2002), (Makarenko et al., 2002), (Smith et al., 2003):

- The robot identifies and select possible locations to explore or exploit,

i.e. vantage locations, in its current estimate of the map.

- The robot computes the utility of taking every action available to it and selects the action with the highest utility.

- The robot carries out the selected action and decide if it is necessary to continue or to terminate the task.

2.3.1 Step 1: Selecting vantage points

In an active SLAM algorithm, vantage points selection is the process to identifying and picking potential destinations that help the robot to solve the exploration-exploitation dilemma. A selected vantage points is a destination that should be reached by the robot after executing an action. Ideally, the robot should evaluate every possible destination in the map space, but this process has an exponential growth which proves to be computationally intractable in real applications (Burgard et al., 2005), (Martinez-Cantin et al., 2009). In practice, a small subset of locations in the map is selected based on the information of its neighborhood. In the following, different selection and identification techniques related to active SLAM algorithms are described.

One of the most widespread techniques in the active SLAM community to selecting vantage points is the frontier-based exploration technique. A frontier is the border between known and unknown area in the environment. This concept was developed by Yamauchi (1997). An advantage of using this technique is that selecting frontier points allows covering the entire environment quickly. However, using only frontiers is not generally a good option by itself because actions towards frontier points only perform the task of exploration and not exploitation, in addition, if the robot does not have a perfect location, the estimated map can become inconsistent (Sim, 2005b), (Sim, 2005a).

Surface-based exploration is one of the first techniques used in active SLAM. The idea underlying this method is look just the vicinity of the current pose of the robot at every decision time. This technique is equivalent to make a local optimization (Feder et al., 1999). The advantage of this vantage points selection scheme is that the horizon of prediction tends to be short, hence the select decision could be optimal but only locally. In contrast to frontier-based exploration, surface-based has the risk of not performing the exploration task.

Another techniques of vantage points selection is the random selection that consider all the destinations in the environment equally good but does not identify possible useful destinations to solve the exploration-exploitation dilemma. Another useful approach is to select as vantage points distinctive characteristic of the environment that have been visited previously. Doing only revisits leads to a slow converge in the area covered by a robot doing robotic exploration. A good idea for SLAM algorithms is the combination of the above techniques for identifying and selecting advantage points. A popular approach is to combine frontiers and revisits method for active SLAM (Stachniss et al., 2004), (Stachniss et al., 2005).

2.3.2 Step 2: Computing the utility of an action

In the second step, the robot computes the utility of performing each action available to it. An action in active SLAM implies navigating to one or several points of interest in the environment selected as explained in step 1. The process of assessing the payoff of a particular action to a vantage points is done through a so-called *utility function* and the reward's value or utility of an action is obtained by evaluating jointly the action under consideration and the state of the world.

The actions in active SLAM are robot motions and the state of the world is given by the environment representation and past robot locations.Therefore, the utility function in this case depend on the uncertainty of both the map representation and the robot's localization and its purpose is to help in achieving the active SLAM objectives, i.e. solving the exploration-exploitation dilemma in a robotic exploration scenario.

Again, ideally the active SLAM algorithm should to consider the complete universe of possibilities, i.e. search over the full joint distribution of the map and robot poses before and after taking the control action and receiving measurements. However, one major problem is that computing the aforementioned joint probability analytically is difficult and, in general, computationally intractable (Cadena et al., 2016). In practice, an approximation of the joint probability is computed. Previous work in active SLAM often consider the uncertainty of the map and robot to be independent (Valencia Carreño et al., 2012) or conditionally independent (Stachniss et al., 2005). Most of these approaches rely on a linear combination of the robot and map uncertainties (Bourgault et al., 2002), (Blanco et al., 2008), (Carlone et al., 2010), (Kim and Eustice, 2013). One drawback of this approach is that the scale

of the numerical values of the two uncertainties is not comparable, i.e. the uncertainty of the map is often orders of magnitude larger than the uncertainty of the robot, so manual tuning is required to correct for this (Blanco et al., 2008), (Carlone et al., 2014).

Given that the robot's pose and map representation are considered as random variables with an associated probability distribution in SLAM, a utility function for active SLAM must ultimately deal with the quantification of the uncertainty encompassed in a random variable product of an estimation process. The two most common approaches to quantify uncertainty in the estimation of random variables are the Information Theory (Cover and Thomas, 2012) and the Theory of Optimal Experimental Design (Pukelsheim, 2006).

The two most common approaches to quantify uncertainty in the estimation of random variables are the Theory of Optimal Experimental Design (Pukelsheim, 2006) and the Information Theory (Cover and Thomas, 2012).

In the design of experiments, optimal designs are a class of experimental designs that are optimal with respect to some statistical criterion or some real-valued function which assesses the information gained through the design. Such a function is called a criterion function and are related to the variance matrix of the estimator. Specifically, this theory bases its process of quantifying the uncertainty on the second moment of the probability distribution associated to the estimated random variable, the covariance matrix. First papers of Theory of Optimal Experimental Design date back to the middle of the 19th century, as Wald (1943).

The traditional optimality-criteria commonly used to quantify the uncertainty, for a covariance matrix Σ with size $l \times l$ and eigenvalues λ_l, are,

- A-optimality criterion (*A-opt*) (Chernoff, 1953): This criterion targets the minimization of the average variance and it is defined as follows:

$$\text{trace}(\Sigma) = \sum_{k=1}^{l} \lambda_k \tag{2.1}$$

- D-optimality criterion (*D-opt*) (Wald, 1943): This criterion aims at capturing the full dimension of the covariance matrix and it can be defined as:

$$\det(\Sigma) = \prod_{k=1}^{l} \lambda_k \tag{2.2}$$

- E-optimality criterion (*E-opt*) (Ehrenfeld, 1955): This criterion intends to minimize the maximum eigenvalue of the covariance matrix and it is defined as follows:

$$\max_{1 \leq k \leq l}(\lambda_k) \tag{2.3}$$

Information theory is the branch of mathematics that describes how uncertainty should be quantified, manipulated and represented. Ever since the fundamental premises of information theory were laid down by Claude Shannon in 1949 (Shannon and Weaver, 1949). The most fundamental quantity in information theory is entropy (Shannon and Weaver, 1949) that measures the amount of uncertainty of an unknown or random quantity. The entropy of a random variable X is defined to be:

$$H(X) = -\sum_{\text{all}x} p(x) \log_2 p(x) \tag{2.4}$$

where the sum is over all values x that the variable X can take, and p(x) is the probability of each of these values occurring. Entropy is measured in bits and can be generalised to continuous variables as well, although care must be taken to specify the precision level at which we would like to represent the continuous variable.

In literature of active SLAM, the utility functions are classified in task-driven or information-driven, depending on if they are based on Theory of Optimal Experimental Design or Information Theory.

Task-driven utility functions are functionals of the eigenvalues of the inverse matrix of the variance-matrix called information matrix. In active SLAM these functions quantify the map and robot's pose uncertainty and allow the direct assessment of the task's improvement after an action is performed due to the uncertainty of the estimated parameters from the SLAM algorithm is encoded in the covariance matrix (Pázman, 1986), (Pukelsheim, 2006).

Active SLAM utility function based on the determinant of the covariance matrix of its state vector represents the n-dimensional volume of the n-dimensional parallelepiped spanned by its n column vectors. In particular for a covariance matrix from a Gaussian distribution, the determinant can be interpreted as the volume of the uncertainty encapsulated by the covariance matrix. Utility functions based in the trace is used a replacement of the determinant because the trace is computationally cheaper to compute. On the down side, it is an approximation of the volume of the uncertainty

encapsulated by the covariance matrix, i.e. an approximation of the use of the determinant.

The *A-opt*, *D-opt* and *E-opt* have been used in many active SLAM related works. In Kollar and Roy (2006), Leung et al. (2006), Kollar and Roy (2008), Meger et al. (2008) and Martinez-Cantin et al. (2009), the criteria used was *A-opt*. *D-opt* has been used as a measure of navigation uncertainty in Vidal-Calleja et al. (2006) and Kim and Eustice (2013).

Information-driven utility functions are designed to measure the reduction of entropy that is a general measure for the uncertainty of the posterior distribution of a random variable (Shannon and Weaver, 1949), (Rényi, 1961) (Kreucher et al., 2005). In active SLAM, these utility functions evaluate the improvement of the task indirectly by checking the so-called information flow of the measurements acquired.

2.3.3 Step 3: Executing actions or terminating exploration

The process of selecting the best action is an optimization problem (Nocedal and Wright, 2006) where the aim is maximize the given utility function subject to the set of actions available. There are many approaches to optimization and most of them could be applied in the context of active SLAM. A common approach is consider the set of actions discrete and apply a discrete optimization technique. Continuous optimization techniques can be used if the set of actions is considered continuous. For example, Indelman et al. (2014) or Van Den Berg et al. (2012) used least squares methods to select the next action, Jadidi et al. (2014), Maffei et al. (2014) or Vallvé and Andrade-Cetto (2014) used gradient-based techniques and Martinez-Cantin et al. (2009) or Souza et al. (2014) applied Bayesian Optimization methods.

Active SLAM is a computationally expensive task therefore the stopping criteria, i.e., the decision on whether or not the exploration task is complete, is a necessity. Uncertainty metrics from Theory of Optimal Experimental Design seem promising as stopping criteria, compared to information-theoretic metrics which are difficult to compare across systems. However, this decision is currently an open challenge (Cadena et al., 2016).

CHAPTER 3

Robot's Pose Representation

In computer vision and robotics, a typical task is to represent the position and orientation of objects in an environment. Such objects include robots, cameras, workpieces, obstacles and paths. This information can then be used, for example, to allow a robot to manipulate an object or to avoid moving into the object. The combination of position and orientation is referred to as the pose of an object and can be described by means of a rotation and translation transformation which brings the object from a reference pose to the observed pose.

In an attempt to make this document as self-contained as possible, and also to clarify notation, in this chapter we provide an overview of mathematical notation about the different ways to represent the robot's pose in 2-dimensional and 3-dimensional space used throughout this thesis.

3.1 Location vectors and homogeneous transformations

3.1.1 General definition

The pose of a robot, i.e. its position and orientation at any given time, can be represented for two alternative representations for transformations: homogeneous matrices and location vectors. In these section we summarize their fundamental properties and laws of transformation between the two

representations.

A homogeneous matrix $\mathbf{H}$ is $n \times n$ matrix, with $n = 3$ in the 2-dimensional space and $n = 4$ in 3-dimensional space, of the form,

$$\mathbf{H} = \begin{pmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.1}$$

where $\mathbf{R}$ is a $(n-1)\times(n-1)$ orthogonal rotation matrix and $\mathbf{p}$ is a translation vector of dimension $n-1$.

A location vector is composed of two Cartesian coordinates and one angle in 2-dimensional space and three Cartesian coordinates and three orientation angles in 3-dimensional space.

The conversions between location vectors and homogeneous matrices are given by the function $\mathrm{Loc}(\cdot)$ and $\mathrm{Hom}(\cdot)$. The closed-formulaes of these functions are described in following sections.

The composition and inversion of location vectors can be represented by means of operators $\oplus$ and $\ominus$, respectively, following Smith's notation (Smith et al., 1990),

$$\mathbf{x}_{AC} = \mathbf{x}_{AB} \oplus \mathbf{x}_{BC} = \mathrm{Loc}(\mathrm{Hom}(\mathbf{x}_{AB})\mathrm{Hom}(\mathbf{x}_{BC})) \tag{3.2}$$

$$\ominus\mathbf{x}_{AB} = \mathrm{Loc}(\mathrm{Hom}(\mathbf{x}_{AB})^{-1}) \tag{3.3}$$

To calculate them, location vector shall be converted to homogeneous matrices. Transformations composition and inversion is equivalent to homogeneous matrix product and inversion. Due to the special form of these matrices, these operations can be carried out as follows,

$$\mathbf{H}_{AC} = \mathbf{H}_{AB}\mathbf{H}_{BC} = \begin{pmatrix} \mathbf{R}_{AB}\mathbf{R}_{BC} & \mathbf{p}_{AB} + \mathbf{R}_{AB}\mathbf{p}_{BC} \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.4}$$

$$\mathbf{H}^{-1} = \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{p} \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.5}$$

Another useful expression to have at hand is the Jacobian matrix. The Jacobian is useful among others things to do linear approximation of func-

tions over a point of operation. The Jacobians of the composition of location vectors, with respect to the first and second operand are given by,

$$\mathbf{J}_{1\oplus}\{\mathbf{x}_{AB}, \mathbf{x}_{BC}\} = \left.\frac{\partial(\mathbf{y} \oplus \mathbf{z})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{x}_{AB}, \mathbf{z}=\mathbf{x}_{BC}} \tag{3.6}$$

$$\mathbf{J}_{2\oplus}\{\mathbf{x}_{AB}, \mathbf{x}_{BC}\} = \left.\frac{\partial(\mathbf{y} \oplus \mathbf{z})}{\partial \mathbf{z}}\right|_{\mathbf{y}=\mathbf{x}_{AB}, \mathbf{z}=\mathbf{x}_{BC}} \tag{3.7}$$

The calculation of the inverse of the Jacobians of the composition can also be carried out in a simple way, without the need of inverting matrices, making use of the following equalities,

$$\begin{aligned}
\mathbf{J}^{-1}_{1\oplus}\{\mathbf{x}_1, \mathbf{x}_2\} &= \left[\left.\frac{\partial(\mathbf{y} \oplus \mathbf{z})}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{x}_1, \mathbf{z}=\mathbf{x}_2}\right]^{-1} \\
&= \left.\frac{\partial \mathbf{y}}{\partial(\mathbf{y} \oplus \mathbf{z})}\right|_{\mathbf{y}=\mathbf{x}_1, \mathbf{z}=\mathbf{x}_2} \\
&= \left.\frac{\partial(\mathbf{y} \oplus \mathbf{z}) \oplus (\ominus \mathbf{z})}{\partial(\mathbf{y} \oplus \mathbf{z})}\right|_{\mathbf{y}\oplus\mathbf{z}=\mathbf{x}_1\oplus\mathbf{x}_2, \ominus\mathbf{z}=\mathbf{x}_2} \\
&= \mathbf{J}_{1\oplus}\{\mathbf{x}_1 \oplus \mathbf{x}_2, \ominus \mathbf{x}_2\}
\end{aligned} \tag{3.8}$$

$$\begin{aligned}
\mathbf{J}^{-1}_{2\oplus}\{\mathbf{x}_1, \mathbf{x}_2\} &= \left[\left.\frac{\partial(\mathbf{y} \oplus \mathbf{z})}{\partial \mathbf{z}}\right|_{\mathbf{y}=\mathbf{x}_1, \mathbf{z}=\mathbf{x}_2}\right]^{-1} \\
&= \left.\frac{\partial \mathbf{z}}{\partial(\mathbf{y} \oplus \mathbf{z})}\right|_{\mathbf{y}=\mathbf{x}_1, \mathbf{z}=\mathbf{x}_2} \\
&= \left.\frac{\partial(\ominus \mathbf{y}) \oplus (\mathbf{y} \oplus \mathbf{z})}{\partial(\mathbf{y} \oplus \mathbf{z})}\right|_{\ominus\mathbf{y}=\ominus\mathbf{x}_1, \mathbf{y}\oplus\mathbf{z}=\mathbf{x}_1\oplus\mathbf{x}_2} \\
&= \mathbf{J}_{2\oplus}\{\ominus \mathbf{x}_1, \mathbf{x}_1 \oplus \mathbf{x}_2\}
\end{aligned} \tag{3.9}$$

3.1.2 Transformation and Jacobian Matrices in 2D space

A location vector in 2-dimensional space is composed of two Cartesian coordinates (x, y) and one angle ϕ. The form of this vector and the transform-

ation are represented as follows,

$$\mathbf{x} = (x, y, \phi)^T \tag{3.10}$$
$$\mathbf{t} = \text{Trans}(x, y)\text{Rot}(Z, \phi) \tag{3.11}$$

where the representation of the elementary transformations using homogeneous matrices is the following,

$$\text{Trans}(x, y) = \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.12}$$

$$\text{Rot}(z, \phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.13}$$

The conversions between location vectors and homogeneous matrices are given by,

$$\mathbf{H} = \text{Hom}(\mathbf{x}) = \begin{pmatrix} \cos\phi & -\sin\phi & x \\ \sin\phi & \cos\phi & y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.14}$$

$$\mathbf{x} = \text{Loc}(\mathbf{H}) = \begin{pmatrix} x \\ y \\ \phi \end{pmatrix} = \begin{pmatrix} x \\ y \\ \text{atan2}(\sin\phi, \cos\phi) \end{pmatrix} \tag{3.15}$$

The Jacobians of the composition of location vectors $\mathbf{x}_{AB} = (x_{AB}, y_{AB}, \phi_{AB})$ and $\mathbf{x}_{BC} = (x_{BC}, y_{BC}, \phi_{BC})$, with respect to the first and second operand are given by (Smith et al., 1990),

$$\mathbf{J}_{1\oplus}\{\mathbf{x}_{AB}, \mathbf{x}_{BC}\} = \begin{pmatrix} 1 & 0 & y_{AB} - y_{AC} \\ 0 & 1 & x_{AC} - x_{AB} \\ 0 & 0 & 1 \end{pmatrix} \tag{3.16}$$

$$\mathbf{J}_{2\oplus}\{\mathbf{x}_{AB}, \mathbf{x}_{BC}\} = \begin{pmatrix} \cos\phi_{AB} & -\sin\phi_{AB} & 0 \\ \sin\phi_{AB} & \cos\phi_{AB} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.17}$$

3.1.3 Transformation and Jacobian Matrices in 3D space

A location vector in 3-dimensinal space is composed of three Cartesian coordinates (x, y, z) and three angles (ϕ, θ, ψ).

The representation of the elementary transformations using homogeneous matrices is the following,

$$\mathrm{Trans}(x, y, z) = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.18}$$

$$\mathrm{Rot}(X, \psi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi & 0 \\ 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.19}$$

$$\mathrm{Rot}(Y, \theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.20}$$

$$\mathrm{Rot}(Z, \phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.21}$$

Euler's rotation theorem states that any rotation can be represented by not more than three rotations about coordinate axes. This means that in general an arbitrary rotation can be decomposed into a sequence of three elementary rotations while guaranteeing that two successive rotations are not made about parallel axes. This implies that distinct set of angles are allowed out of all 27 possible combinations. In common usage all these sequences are called Euler angles. The particular angle sequence is often a convention within a particular technological field. In the following sections, two sets of Euler angles are analyzed, namely, ZYX (or Roll-Pitch-Yaw) angles and the ZYZ angles.

A fundamental problem with the three-angle representations just described is singularity. This occurs when the rotational axis of the middle term in the sequence becomes parallel to the rotation axis of the first or third term. Singularities are an unfortunate consequence of using a minimal representation. To eliminate this problem it is necessary to adopt different representations of orientation.

A non-minimal representation of orientation can be obtained by resorting to four parameters expressing a rotation of a given angle about an axis in space, called axis-angle representation. A simple way to encode this axis–angle representation in four numbers is unit quaternions. Unit quaternions, also known as versors, provide a convenient mathematical notation for representing orientations and rotations of objects in three dimensions. Compared to Euler angles they are simpler to compose and avoid the problem of gimbal lock.

Converting from angle and vector to a rotation matrix is achieved using Rodrigues'rotation formula. Details about this representation are provided in the Lie groups section.

To simplify the notation, a homogeneous transformation $\mathbf{H}$ is a 4×4 matrix of the form,

$$\mathbf{H} = \begin{pmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.22}$$

where $\mathbf{n}, \mathbf{o}, \mathbf{a}$ are three column vectors that form the rotation matrix $\mathbf{R}$.

Relationships using Roll-Pitch-Yaw Euler Angles

A widely used convention is the roll-pitch-yaw sequence angle which are intuitive when describing the attitude of vehicles such as ships, aircraft and cars. Roll, pitch and yaw (also called bank, attitude and heading) refer to rotations about the Z-, Y-, X-axes, respectively.

An Roll-Pitch-Yaw location vector is composed of three cartesian coordinates and three Roll-Pitch-Yaw angles. The form of this vector is,

$$\mathbf{x} = (x, y, z, \psi, \theta, \phi)^T \tag{3.23}$$
$$\phi : \text{Roll}, \theta : \text{Pitch}, \psi : \text{Yaw} \tag{3.24}$$
$$\mathbf{t} = \text{Trans}(x, y, z)\text{Rot}(Z, \phi)\text{Rot}(Y, \theta)\text{Rot}(X, \psi) \tag{3.25}$$

The conversion between location vector and homogeneous matrices are given by,

$$\mathbf{H} = \text{Hom}(\mathbf{x}) = \begin{pmatrix} \cos\phi\cos\theta & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & x \\ \sin\phi\cos\theta & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & y \\ -\sin\phi & \cos\theta\sin\psi & \cos\theta\cos\psi & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.26}$$

$$\mathbf{x} = \text{Loc}(\mathbf{H}) = \begin{pmatrix} x \\ y \\ z \\ \psi \\ \theta \\ \phi \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ p_z \\ \text{atan2}(o_z, a_z) \\ \text{atan2}(-n_z, n_x\cos\phi + n_y\sin\phi) \\ \text{atan2}(n_y, n_x) \end{pmatrix} \tag{3.27}$$

The submatrix of $\text{Hom}(\mathbf{x})$ corresponding to the rotation matrix will be represented by $\mathbf{R} = Mrot(\mathbf{x})$.

The Jacobians of the composition of location vectors, with respect to the first and second operand are given by (Smith et al., 1990),

$$\mathbf{J}_{1\oplus}\{\mathbf{x}_1, \mathbf{x}_2\} = \begin{pmatrix} \mathbf{I}_{3\times3} & \mathbf{M} \\ \mathbf{0}_{3\times3} & \mathbf{K}_1 \end{pmatrix} \tag{3.28}$$
$$\mathbf{J}_{1\oplus}\{\mathbf{x}_1, \mathbf{x}_2\} = \begin{pmatrix} \mathbf{R}_1 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{K}_2 \end{pmatrix} \tag{3.29}$$

where, taking $\mathbf{x}_3 = \mathbf{x}_1 \oplus \mathbf{x}_2$,

$$\mathbf{M} = \begin{pmatrix} y_2 a_{z_1} - z_2 o_{z_1} & -x_2 \cos\theta_1 - y_2 \sin\theta_1 \sin\psi_1 - z_2 \sin\theta_1 \cos\psi_1 & 0 \\ y_2 a_{y_1} - z_2 o_{y_1} & (z_3 - z_1)\sin\phi_1 & x_3 - x_1 \\ y_2 a_{x_1} - z_2 o_{x_1} & (z_3 - z_1)\cos\phi_1 & y_1 - y_3 \end{pmatrix} \tag{3.30}$$

$$\mathbf{K}_1 = \begin{pmatrix} \cos\theta_1 \cos(\phi_3 - \phi_1)/\cos\theta_3 & \sin(\phi_3 - \phi_1)/\cos\theta_3 & 0 \\ -\cos\theta_1 \sin(\phi_3 - \phi_1) & \cos(\phi_3 - \phi_1) & 0 \\ (o_{x_2}\sin\psi_3 + a_{x_2}\cos\psi_3)/\cos\theta_3 & \sin\theta_3 \sin(\phi_3 - \phi_1)/\cos\theta_3 & 1 \end{pmatrix} \tag{3.31}$$

$$\mathbf{K}_2 = \begin{pmatrix} 1 & \sin\theta_3 \sin(\psi_3 - \psi_2)/\cos\theta_3 & (a_{x_1}\cos\phi_3 + a_{y_1}\sin\phi_3)/\cos\theta_3 \\ 0 & \cos(\psi_3 - \psi_2) & -\cos\theta_2 \sin(\psi_3 - \psi_2) \\ 0 & \sin(\psi_3 - \psi_2)/\cos\theta_3 & \cos\theta_2 \cos(\psi_3 - \psi_2)/\cos\theta_3 \end{pmatrix} \tag{3.32}$$

$$\mathbf{R}_1 = Mrot(\mathbf{x}_1) \tag{3.33}$$

It can be seen that the expression of the formulas given above has been simplified using terms of $\mathbf{x}_3 = \mathbf{x}_1 \oplus \mathbf{x}_2$ and terms of the rotation matrix corresponding to $\mathbf{x}_1$ and $\mathbf{x}_2$. In the case where the value of θ_3 be $-\pi/2$ or $\pi/2$, the Jacobians would be undefined. These values correspond to singular configurations of Roll-Pitch-Yaw angles.

Relationships using ZYZ Euler Angles

The ZYZ sequence is commonly used in aeronautics and mechanical dynamics and it refers to rotations about the Z-, Y-, Z-axes, respectively.

A location vector using ZYZ Euler angles is defined by,

$$\mathbf{x} = (x, y, z, \psi, \theta, \phi)^T \tag{3.34}$$

$$\mathbf{t} = \mathrm{Trans}(x, y, z)\mathrm{Rot}(z, \phi)\mathrm{Rot}(y, \theta)\mathrm{Rot}(z, \psi) \tag{3.35}$$

The conversion between location vector and homogeneous matrices are given by,

$$\mathbf{H} = \mathrm{Hom}(\mathbf{x}) = \begin{pmatrix} \cos\phi\cos\theta\cos\psi - \sin\phi\sin\psi & -\cos\phi\cos\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\sin\theta & x \\ \sin\phi\cos\theta\cos\psi + \cos\phi\sin\psi & -\sin\phi\cos\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\sin\theta & y \\ -\sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.36}$$

$$\mathbf{x} = \mathrm{Loc}(\mathbf{H}) = \begin{pmatrix} x \\ y \\ z \\ \psi \\ \theta \\ \phi \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ p_z \\ \mathrm{atan2}(-n_x\sin\phi + n_y\cos\phi, -o_x\sin\phi + o_y\cos\phi) \\ \mathrm{atan2}(a_x\cos\theta + a_y\sin\theta, a_z) \\ \mathrm{atan2}(a_y, a_x) \end{pmatrix} \tag{3.37}$$

The submatrix of $\mathrm{Hom}(\mathbf{x})$ corresponding to the rotation matrix will be represented by $\mathbf{R} = Mrot(\mathbf{x})$.

The Jacobians of the composition of location vectors, with respect to the first and second operand are given by (Smith et al., 1990),

$$\begin{aligned} \mathbf{J}_{1\oplus}\{\mathbf{x}_1, \mathbf{x}_2\} &= \begin{pmatrix} \mathbf{I}_{3\times3} & \mathbf{M} \\ \mathbf{0}_{3\times3} & \mathbf{K}_1 \end{pmatrix} \\ \mathbf{J}_{1\oplus}\{\mathbf{x}_1, \mathbf{x}_2\} &= \begin{pmatrix} \mathbf{R}_1 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{K}_2 \end{pmatrix} \end{aligned} \tag{3.38}$$

where, taking $\mathbf{x}_3 = \mathbf{x}_1 \oplus \mathbf{x}_2$, where $\mathbf{x}_1 = (x_1, y_1, z_1, \psi_1, \theta_1, \phi_1)^T$ and $\mathbf{x}_2 = (x_2, y_2, z_2, \psi_2, \theta_2, \phi_2)^T$,

$$\mathbf{M} = \begin{pmatrix} x_2 o_{z_1} - y_2 n_{z_1} & -x_2\cos\theta_1\cos\psi + y_2\sin\theta_1\sin\psi_1 - z_2\sin\theta_1 & 0 \\ x_2 o_{y_1} - y_2 n_{y_1} & (z_3 - z_1)\sin\phi_1 & x_3 - x_1 \\ x_2 o_{x_1} - y_2 n_{x_1} & (z_3 - z_1)\cos\phi_1 & y_1 - y_3 \end{pmatrix} \tag{3.39}$$

$$\mathbf{K}_1 = \begin{pmatrix} \sin\theta_1 \cos(\phi_3 - \phi_1)/\sin\theta_3 & \sin(\psi_3 - \psi_1)/\sin\theta_3 & 0 \\ \sin\theta_2 \sin(\psi_3 - \psi_2) & \cos(\phi_3 - \phi_1) & 0 \\ \sin\theta_2 \cos(\psi_3 - \psi_2)/\sin\theta_3 & \cos\theta_3 \sin(\psi_3 - \psi_1)/\sin\theta_3 & 1 \end{pmatrix} \tag{3.40}$$

$$\mathbf{K}_2 = \begin{pmatrix} 1 & (\cos\theta_3 \sin(\psi_3 - \psi_2))/\sin\theta_3 & (\sin\theta_1 \sin(\phi_3 - \phi_1))/\sin\theta_3 \\ 0 & \cos(\psi_3 - \psi_2) & \sin\theta_2 \sin(\psi_3 - \psi_2) \\ 0 & \sin(\psi_3 - \psi_2)/\sin\theta_3 & (\sin\theta_1 \sin(\phi_3 - \phi_1))/\sin\theta_3 \end{pmatrix} \tag{3.41}$$

$$\mathbf{R}_1 = Mrot(\mathbf{x}_1) \tag{3.42}$$

Again, it can be seen that the expression of the formulas given above has been simplified using terms of $\mathbf{x}_3 = \mathbf{x}_1 \oplus \mathbf{x}_2$ and terms of the rotation matrix corresponding to $\mathbf{x}_1$ and $\mathbf{x}_2$.

Relationships using unit quaternions

The quaternion is an extension of the complex number – a hyper-complex number – and is written as a scalar plus a vector. Quaternions are elegant, powerful and computationally straightforward and widely used for robotics, computer vision, computer graphics and aerospace inertial navigation applications.

A pose can be also described with a displacement in 3-dimensional space plus a rotation defined by a unit quaternion as,

$$\mathbf{x} = (x, y, z, q_r, q_x, q_y, q_z)^T \tag{3.43}$$

where the quaternion elements are $[q_r, (q_x, q_y, q_z)]$.

The transformation matrix associated to a pose is given by,

$$\mathbf{H} = \mathrm{Hom}(\mathbf{x}) = \tag{3.44}$$

$$\begin{pmatrix} q_r^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_r q_z) & 2(q_z q_x - q_r q_y) & x \\ 2(q_x q_y + q_r q_z) & q_r^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_r q_x) & y \\ 2(q_z q_x - q_r q_y) & 2(q_y q_z + q_r q_x) & q_r^2 - q_x^2 - q_y^2 + q_z^2 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.45}$$

A numerically stable method to convert a 4×4 matrix into a unit quaternions pose is described in (Bar-Itzhack, 2000).

The Jacobians of the composition of location vectors, with respect to the first and second operand are given by (Blanco, 2010),

$$\mathbf{J}_{1\oplus}\{\mathbf{x}_1, \mathbf{x}_2\} = \begin{pmatrix} \mathbf{I}_{3\times 3} & \mathbf{M} \\ \mathbf{0}_{4\times 3} & \mathbf{K}_1 \end{pmatrix} \tag{3.46}$$

$$\mathbf{J}_{2\oplus}\{\mathbf{x}_1, \mathbf{x}_2\} = \begin{pmatrix} \mathbf{R}_1 & \mathbf{0}_{3\times 4} \\ \mathbf{0}_{4\times 3} & \mathbf{K}_2 \end{pmatrix} \tag{3.47}$$

where,

$$\mathbf{M} = 2\mathbf{M}_1 \frac{1}{(q_{r_1}^2 + q_{x_1}^2 + q_{y_1}^2 + q_{z_1}^2)^{3/2}} \mathbf{M}_2 \tag{3.48}$$

with,

$$\mathbf{M}_1 = \begin{pmatrix} -q_{z_1} y_2 + q_{y_1} z_2 & q_{y_1} y_2 + q_{z_1} z_2 & -2q_{y_1} x_2 + q_{x_1} y_2 + q_{r_1} z_2 & -2q_{z_1} x_2 - q_{r_1} y_2 + q_{x_1} z_2 \\ q_{z_1} x_2 - q_{x_1} z_2 & q_{y_1} x_2 - 2q_{x_1} y_2 - q_{r_1} z_2 & q_{x_1} x_2 + q_{z_1} z_2 & q_{r_1} x_2 - 2q_{z_1} y_2 + q_{y_1} z_2 \\ -q_{y_1} x_2 + q_{x_1} y_2 & q_{z_1} x_2 + q_{r_1} y_2 - 2q_{x_1} z_2 & -q_{r_1} x_2 + q_{z_1} y_2 - 2q_{y_1} z_2 & q_{x_1} x_2 + q_{y_1} y_2 \end{pmatrix} \tag{3.49}$$

$$\mathbf{M}_2 = \begin{pmatrix} q_{x_1}^2 + q_{y_1}^2 + q_{z_1}^2 & -q_{r_1} q_{x_1} & -q_{r_1} q_{y_1} & -q_{r_1} q_{z_1} \\ -q_{x_1} q_{r_1} & q_{r_1}^2 + q_{y_1}^2 + q_{z_1}^2 & -q_{x_1} q_{y_1} & -q_{x_1} q_{z_1} \\ -q_{y_1} q_{r_1} & -q_{y_1} q_{x_1} & q_{r_1}^2 + q_{x_1}^2 + q_{z_1}^2 & -q_{y_1} q_{z_1} \\ -q_{z_1} q_{r_1} & -q_{z_1} q_{x_1} & -q_{z_1} q_{y_1} & q_{r_1}^2 + q_{x_1}^2 + q_{y_1}^2 \end{pmatrix} \tag{3.50}$$

$$\mathbf{K}_1 = \begin{pmatrix} q_{r_2} & -q_{x_2} & -q_{y_2} & -q_{z_2} \\ q_{x_2} & q_{r_2} & q_{z_2} & -q_{y_2} \\ q_{y_2} & -q_{z_2} & q_{r_2} & q_{x_2} \\ q_{z_2} & q_{y_2} & -q_{x_2} & q_{r_2} \end{pmatrix} \tag{3.51}$$

$$\mathbf{K}_2 = \begin{pmatrix} q_{r_1} & -q_{x_1} & -q_{y_1} & -q_{z_1} \\ q_{x_1} & q_{r_1} & -q_{z_1} & q_{y_1} \\ q_{y_1} & q_{z_1} & q_{r_1} & -q_{x_1} \\ q_{z_1} & -q_{y_1} & q_{x_1} & q_{r_1} \end{pmatrix} \tag{3.52}$$

$$\mathbf{R}_1 = Mrot(\mathbf{x}_1) \tag{3.53}$$

3.2 Lie Groups

Lie groups theory is a coherent and robust framework for representing and working with homogeneous transformations. Lie groups and their associated machinery address with operations related to transformations as composed, inverted, differentiated and interpolated.

This section to provide enough information about Lie groups representing spatial transformations can be employed usefully in robotics and computer vision and derives useful formulae for working with the Lie groups that represent transformations in 2-dimensional and 3-dimensional space.

3.2.1 General definitions

Lie Groups

A Lie group is a group that is also a differentiable manifold, with the property that the group operations are compatible with the smooth structure. More information about Lie groups and the concepts covered in this chapter can be found in (Gallier, 2011), (Varadarajan, 2013) and (Hall, 2015).

A robot's pose can be defined with Lie groups using the special Euclidean group that represents rotation and translation, for 2D and 3D cases ($n = 2, 3$, respectively) as,

$$SE(n) = \left\{ \begin{pmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)} | \mathbf{R} \in SO(n), \mathbf{p} \in \mathbb{R}^n \right\} \tag{3.54}$$

where $SO(n)$ is the special orthogonal group defined as,

$$SO(n) = \{\mathbf{C} \in \mathbb{R}^{n\times n} | \mathbf{C}\mathbf{C}^T = \mathbf{I}, \det(\mathbf{C}) = 1\} \tag{3.55}$$

Lie algebra

To every Lie group can be associated a Lie algebra whose underlying vector space is the space of differential transformation (tangent space) around the identity element of the Lie group and which completely captures the local structure of the group. Informally, we can think of elements of the Lie algebra as elements of the group that are "infinitesimally close" to the identity.

Consider a Lie group G represented in $\mathbb{R}^{n\times n}$, with k degrees of freedom. Then, the Lie algebra $\mathfrak{g}$ is a k-dimensional vector space with basis elements $\{\mathbf{G}_1, \cdots, \mathbf{G}_k\}$ called generator. Elements of $\mathfrak{g}$ are represented as matrices in $\mathbb{R}^{n\times n}$.

For such Lie algebra $\mathfrak{g}$, a vector representation is associated with the *hat*-operator $\widehat{\cdot}$,

$$\begin{aligned} \widehat{\cdot} &: \mathbb{R}^k \to \mathbb{R}^{n\times n} \\ \widehat{\mathbf{x}} &= \sum_{i=1}^{k} \mathbf{x}_i \mathbf{G}_i \end{aligned} \tag{3.56}$$

which maps a k-vector tangent representation onto an $(n \times n)$ matrix representation.

Exponential Map

A way to associate elements of the Lie algebra to elements of the underlying Lie group is using the exponential map. Let us consider the formal definition of the standard exponential function,

$$e^x : \mathbb{R} \to \mathbb{R}^+$$
$$e^x = \sum_{k=0}^{\infty} \frac{x^n}{n!} \tag{3.57}$$

This expression can be generalized for squared matrices as follows,

$$\exp(\mathbf{X}) : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n} \tag{3.58}$$
$$\exp(\mathbf{X}) = \sum_{k=0}^{\infty} \frac{\mathbf{X}^n}{n!} \tag{3.59}$$

with $\mathbf{X}^0 = I$. Not without reason, this function is called exponential map since it has similar properties to the exponential function including,

$$\frac{\partial}{\partial t} \exp(t\mathbf{X}) = \mathbf{X} \exp(t\mathbf{X}) = \exp(t\mathbf{X})\mathbf{X} \tag{3.60}$$

$$\mathbf{XY} = \mathbf{YX} \Rightarrow \exp(\mathbf{X}) \exp(\mathbf{Y}) = \exp(\mathbf{X} + \mathbf{Y}) \tag{3.61}$$

Furthermore, it can be shown that,

$$\exp(\mathbf{AXA}^{-1}) = \mathbf{A} \exp(\mathbf{X})\mathbf{A}^{-1} \tag{3.62}$$

$$\exp(\mathbf{X})^{-1} = \exp(-\mathbf{X}) \tag{3.63}$$

The inverse of the exponential map is the logarithm,

$$\exp\left(\log \mathbf{X}\right) = \mathbf{X} \tag{3.64}$$

The logarithm is usually not continuous everywhere, but is always continuous near the identity.

Adjoint Map

Other concept of the Lie groups theory is the adjoint representation (or adjoint action) of a Lie group. The adjoint representation is a way of representing the elements of the group as linear transformations of the group's Lie algebra, i.e., transforms a tangent vector from the tangent space around one element to tangent space of another.

$$\begin{aligned} Adj_{\mathbf{X}} &: \mathfrak{g} \to \mathfrak{g} \\ Adj_{\mathbf{X}}(\mathbf{A}) &= \mathbf{X} \cdot \mathbf{A} \cdot \mathbf{X}^{-1} \end{aligned} \tag{3.65}$$

Elements of the adjoint representation are usually written as $k \times k$ matrices acting on the coefficient vectors of elements in the Lie algebra $\mathfrak{g}$ by multiplication.

The adjoint representation preserves the group structure of the Lie group $\mathbf{G}$,

$$\mathbf{X}, \mathbf{Y} \in \mathbf{G} \tag{3.66}$$

$$Adj_{\mathbf{X}\cdot\mathbf{Y}} = Adj_{\mathbf{X}} \cdot Adj_{\mathbf{Y}} \tag{3.67}$$

$$Adj_{\mathbf{X}^{-1}} = Adj_{\mathbf{X}}^{-1} \tag{3.68}$$

To get a better understanding of the usefulness of the adjoint, let us to consider,

$$\mathbf{X} \in G, \mathbf{A} \in \mathfrak{g} \tag{3.69}$$

$$\exp(Adj_{\mathbf{X}}(\mathbf{A})) = \mathbf{X} \cdot \exp(\mathbf{A}) \cdot \mathbf{X}^{-1} \tag{3.70}$$

$$\exp(Adj_{\mathbf{X}}(\mathbf{A})) \cdot \mathbf{X} = \mathbf{X} \cdot \exp(\mathbf{A}) \tag{3.71}$$

Therefore, the adjoint allows us to move the matrix exponential from the right-hand side to the left-hand side of A. Thus, the adjoint is the Jacobian of the transformation of tangent vectors through elements of the group.

3.2.2 SO(2): Rotations in 2D space

Lie group definition

$SO(2)$ is the group of rotations in the 2-dimensional plane. Elements of the rotation group in two dimensions are represented by 2D rotation matrices with one degree of freedom: angle of rotation.

$$SO(2) = \left\{ \mathbf{R} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \in \mathbb{R}^{2\times 2} | \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det(\mathbf{R}) = 1 \right\} \quad (3.72)$$

Composition and inversion in the group correspond to matrix multiplication and inversion. Because rotation matrices are orthogonal, inversion is equivalent to transposition.

Lie Algebra

The Lie algebra $\mathfrak{so}(2)$ is the set of 2×2 skew-symmetric matrices and is generated by one antisymmetric element, corresponding to a differential rotation in 2D,

$$\mathbf{G} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad (3.73)$$

Exponential Map

The exponential map from $\mathbf{so}(2)$ to $SO(2)$ is simply a 2D rotation,

$$\exp(\hat{\theta}) = \exp \begin{pmatrix} 0 & -\theta \\ \theta & 0 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \quad (3.74)$$

Adjoint Representation

The adjoint representation of $SO(2)$ is,

$$\mathbf{X} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \in SO(2), \quad a^2 + b^2 = 1 \tag{3.75}$$

$$Adj_{\mathbf{X}}(\hat{\theta}) = \mathbf{X}\hat{\theta}\mathbf{X}^{-1} = \begin{pmatrix} 0 & -\theta \\ \theta & 0 \end{pmatrix} = \hat{\theta} \tag{3.76}$$

$$Adj_{\mathbf{X}} = \mathbf{I} \tag{3.77}$$

3.2.3 SE(2): Rigid transformations in 2D space

Lie group definition

$SE(2)$ is the group of rigid transformations in the 2D plane. The group has three dimensions, corresponding to translation and rotation in the plane.

$$SE(2) = \left\{ \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}, \mathbf{R} \in SO(2), \mathbf{t} \in \mathbb{R}^2 \right\} \tag{3.78}$$

Transformation composition and inversion are coincident with matrix multiplication and inversion,

$$\mathbf{X}_1, \mathbf{X}_2 \in SE(2) \tag{3.79}$$

$$\mathbf{X}_1\mathbf{X}_2 = \begin{pmatrix} \mathbf{R}_1 & \mathbf{t}_1 \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}_2 & \mathbf{t}_2 \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{t}_2 + \mathbf{t}_1 \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.80}$$

$$\mathbf{X}^{-1} = \begin{pmatrix} \mathbf{R}_1^T & -\mathbf{R}_1^T\mathbf{t}_1 \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.81}$$

Lie Algebra

The Lie algebra $\mathfrak{se}(2)$ is the set of 3×3 matrices corresponding to differential translations and rotation around the identity. The three generators of the algebra are,

$$\mathbf{G}_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} ; \ \mathbf{G}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} ; \ \mathbf{G}_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{3.82}$$

Exponential Map

The exponential map from $\mathfrak{se}(2)$ to $SE(2)$ is the matrix exponential on a linear combination of the generators,

$$\mathbf{x} = (x, y, \theta)^T \tag{3.83}$$

$$\exp(\hat{\mathbf{x}}) = \exp \left(\begin{array}{c|c} \exp(\hat{\theta}) & \begin{matrix} x \\ y \end{matrix} \\ \hline \mathbf{0} & 0 \end{array} \right) = \left(\begin{array}{c|c} \mathbf{R} & \mathbf{V} \begin{pmatrix} x \\ y \end{pmatrix} \\ \hline \mathbf{0} & 1 \end{array} \right) \tag{3.84}$$

where $\mathbf{R} \in SO(2)$ and the elements of $\mathbf{V}$ is calculated with Taylor series when θ is small,

$$\mathbf{V} = \begin{pmatrix} \frac{\sin\theta}{\theta} & -\frac{1-\cos\theta}{\theta} \\ \frac{1-\cos\theta}{\theta} & \frac{\sin\theta}{\theta} \end{pmatrix} \tag{3.85}$$

Adjoint representation

The adjoint in $SE(2)$ has a closed form,

$$\mathbf{X} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \in SE(2) \tag{3.86}$$

$$Adj_{\mathbf{X}} = \left(\begin{array}{c|c} \mathbf{R} & \begin{matrix} y \\ -x \end{matrix} \\ \hline \mathbf{0} & 1 \end{array} \right) \tag{3.87}$$

3.2.4 SO(3): Rotations in 3D space

Lie group definition

SO(3) is the group of rotations in 3-dimensional space, represented by 3×3 orthogonal matrices with unit determinant. It has three degree of freedom, one for each differential rotation axis.

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det(\mathbf{R}) = 1\} \tag{3.88}$$

Composition and inversion in the group correspond to a matrix multiplication and inversion.

Lie Algebra

The Lie algebra $\mathfrak{so}(3)$ is the set of antisymmetric 3×3 matrices, generated by the differential rotations about each axis,

$$\mathbf{G}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}; \ \mathbf{G}_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}; \ \mathbf{G}_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{3.89}$$

The mapping $\widehat{\cdot}: \mathbb{R}^3 \to \mathfrak{so}(3)$ sends 3-vectors to their skew matrix,

$$\widehat{\boldsymbol{\omega}} = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}, \boldsymbol{\omega} = (a, b, c)^T \tag{3.90}$$

Exponential map

The exponential map from $\mathfrak{so}(3)$ to $SO(3)$ has a closed form called the Rodrigues formula. The tangent vector $\boldsymbol{\omega} \in \mathbb{R}^3$ can be interpreted as an axis-angles representation of rotation, its exponential is the rotation around the axis $\boldsymbol{\omega}/||\boldsymbol{\omega}||$ by $||\boldsymbol{\omega}||$ radians,

$$exp\left(\widehat{\boldsymbol{\omega}}_{\mathfrak{so}(3)}\right) = exp([\boldsymbol{\omega}]_x) = I + \frac{\sin\theta}{\theta}[\boldsymbol{\omega}]_x + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\omega}]_x^2 \tag{3.91}$$

where $\theta = \sqrt{\boldsymbol{\omega}^T\boldsymbol{\omega}}$.

Adjoint representation

The adjoint representation of $SO(3)$ is actually identical to the rotation matrix representation due to properties of the cross product,

$$\mathbf{R} \in SO(3), \mathbf{a}, \mathbf{b} \in \mathbb{R}^3 \tag{3.92}$$

$$\begin{aligned}(\mathbf{R}\cdot\mathbf{a}_\times\cdot\mathbf{R}^T)\cdot\mathbf{b} =& \mathbf{R}\cdot(\mathbf{a}\times\mathbf{R}^T)\cdot\mathbf{b})\\ =&(\mathbf{R}\cdot\mathbf{a})\times\mathbf{b}\\ =&(\mathbf{R}\cdot\mathbf{a})_\times\mathbf{b}\end{aligned} \tag{3.93}$$

$$\mathbf{R}\cdot\mathbf{a}_\times\cdot\mathbf{R}^T = (\mathbf{R}\cdot\mathbf{a})_\times \tag{3.94}$$

$$Adj_{\mathbf{R}} = \mathbf{R} \tag{3.95}$$

3.2.5 SE(3): Rigid transformations in 3D space

Lie group definition

$SE(3)$ is the group of rigid transformations in the 3D plane and is well represented by linear transformations on homogeneous four-vectors. The group has six dimensions, corresponding three for translation and three for rotation.

$$SE(3) = \left\{ \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}, \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\} \tag{3.96}$$

Transformation composition and inversion are coincident with matrix multiplication and inversion,

$$\mathbf{X}_1, \mathbf{X}_2 \in SE(3) \tag{3.97}$$

$$\mathbf{X}_1\mathbf{X}_2 = \begin{pmatrix} \mathbf{R}_1 & \mathbf{t}_1 \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}_2 & \mathbf{t}_2 \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{t}_2 + \mathbf{t}_1 \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.98}$$

$$\mathbf{X}^{-1} = \begin{pmatrix} \mathbf{R}_1^T & -\mathbf{R}_1^T\mathbf{t}_1 \\ \mathbf{0} & 1 \end{pmatrix} \tag{3.99}$$

Lie Algebra

The Lie algebra $\mathfrak{se}(3)$ is the set of 4×4 matrices corresponding to differential translations and rotations around the identity. The six generators of the algebra are,

$$\mathbf{G}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}; \mathbf{G}_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}; \quad \mathbf{G}_3 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.100}$$

$$\mathbf{G}_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}; \mathbf{G}_5 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}; \qquad \mathbf{G}_6 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.101}$$

These generators represent the derivatives of x, y and z translations and x, y and z axis rotations respectively.

The mapping $\widehat{\cdot}: \mathbb{R}^6 \to \mathfrak{se}(3)$ is,

$$\widehat{\begin{pmatrix} v \\ w \end{pmatrix}}_{\mathfrak{se}(3)} = \begin{pmatrix} [w]_x & v \\ 0_{1\times 3} & 0 \end{pmatrix} \tag{3.102}$$

where $\mathbf{v}, \boldsymbol{\omega} \in \mathbb{R}^3$ and $[w]_x$ is the hat-operator of $\mathfrak{so}(3)$.

Exponential map

The exponential map $\mathfrak{se}(3)$ to $SE(3)$ has closed form,

$$\exp : \mathfrak{se}(3) \to SE(3) \tag{3.103}$$

$$\exp\left(\widehat{\begin{pmatrix} v \\ w \end{pmatrix}}\right) = \begin{pmatrix} \exp([w]_x) & Vu \\ 0 & 1 \end{pmatrix} \tag{3.104}$$

with $\theta = \|w\|_2$ and $V = I$ if $\theta \to 0$ and $V = I + \frac{1-\cos(\theta)}{\theta^2}[w]_x + \frac{\theta - sin(\theta)}{\theta^3}[w]_x^2$ in other case.

Adjoint representation

The adjoint in $SE(3)$ has a closed form:

$$\mathbf{X} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \in SE(3) \tag{3.105}$$

$$Adj_{\mathbf{X}} = \left(\begin{array}{c|c} \mathbf{R} & \mathbf{t}_\times \mathbf{R} \\ \hline \mathbf{0} & \mathbf{R} \end{array}\right) \tag{3.106}$$

CHAPTER 4

Uncertainty Representation and its Propagation

In robotics, it has been well established that estimating uncertain spatial relationship is fundamentally important problem and its representation and estimation has received attention for years. In many applications dealing with the real-world assembly task, such as industrial automation, and autonomous mobility, there is a need to represent and reason about spatial uncertainty arising from object representation, robot motion and sensory information. This problem is complicated in practice because the location of one object relative to another may be known only indirectly through a sequence of relative frames of reference with uncertainty. Therefore, to reason the effects of uncertainties, uncertainties must be properly represented such as their manipulation can be performed. Different types of models to represent uncertainty and its propagation have been proposed in the literature.

Two fundamental types of models of probabilistic representation of the uncertainty have been proposed: *absolute* and *differential*. In absolute representations (Bolle and Cooper, 1986), (Smith and Cheeseman, 1986), the information about the uncertainty in the location of the robot's pose is represented by a probability distribution function, usually Gaussian, over the variables of the absolute location with respect to a chosen base reference. The estimated location is given by the expected location variables and the uncertainty by its associated covariance matrix. Differential representations use a local representation of the uncertainty (Durrant-Whyte, 1987), (Su and Lee, 1992), (Castellanos et al., 1999). The estimated location of the robot is represented by the best approximation of the absolute location and

the estimation error is represented locally by a differential location vector. This vector is usually also represented by a Gaussian probability distribution function.

Equivalent representations on the Lie group have also been considered in recent works (Smith et al., 2003), (Wang and Chirikjian, 2006), (Barfoot and Furgale, 2014) where transformations are represented by homogeneous matrices and uncertainties in a pose or motion correspond to probability density functions over Lie groups. By adopting this geometric framework, many of the ad hoc definitions and notational conventions found in existing dynamics algorithms can provide other advantages as they avoid singularities when representing state spaces with either redundant degrees of freedom or constraint issues (Barfoot and Furgale, 2014), (Hertzberg et al., 2013).

In this chapter, we review related work about representation and propagation of the uncertainty and present a survey of different types of models proposed in the literature. Additionally, an analysis of the uncertainty represented with a differential error and the uncertainty propagation on Lie groups is carried out taking into account the similarities between both models of representations.

4.1 Literature review

In the literature, two competing approaches for representing spatial uncertainty have been studied, the set-oriented representation and the probabilistic representation. The set-oriented representation basically uses the worst-case bounds which include all possible errors to represent the real location of an object (Taylor, 1976), (Brooks, 1982), (Brooks, 1984), (Erdmann, 1984), (Grimson and Lozano-Perez, 1984), (Pertin-Troccaz and Puget, 1988), (Taylor and Rajan, 1988), (Xiao et al., 1989). The probabilistic representation, however, assigns probabilities to all possible locations of an object, thus the real location of an object becomes a probability distribution over the space (Chatila and Laumond, 1985), (Smith and Cheeseman, 1986), (Durrant-Whyte, 1987), (Durrant-Whyte, 1988), (Mazon and Alami, 1989), (Su and Lee, 1991), (Su and Lee, 1992).

Most of the previous work on uncertainties focused on the fine-motion planning problem (Lozano-Perez et al., 1984), (Juan and Paul, 1986), (Erdmann, 1986), (Donald, 1986), (Natarajan, 1988), (Xiao et al., 1989), (Desai

et al., 1989), (Donald, 1990), and the task-level planning problem (Taylor, 1976), (Brooks, 1982), (Pertin-Troccaz and Puget, 1988) using set-oriented representation. In the fine-motion planning problem, the existing techniques reduce uncertainties from the local point of view by combining sensory activities with incremental robot motions aiming at progressively achieving the desired goal. In the task-level planning problem, Taylor (1976) estimated the compound error from the bounds of each individual object involved in a task. Brooks (1982) considered symbolic constraints in computing errors. By manipulating the constraint dependencies between successive plan steps, his system can be used both to provide error estimates and to identify plan variables whose values should be constrained for successful task execution.

Because of using worst-case bounds, the set-oriented representation usually results in conservative estimates that limit their use in the decision-making process and lead to highly restricted planning strategies. Hence, research work on the uncertainty representation focused more on the probabilistic representation. This representation is used to represent spatial uncertainties that arise from object modeling, robot motion, and sensory information. Furthermore, the probabilistic representation can provide more systematic propagation methodologies than the set-oriented representation, and it can be easily transformed to the set-oriented representation with a small percentage of error.

Smith and Cheeseman (1986) characterized the uncertainties of motion parameters by a mean vector and a covariance matrix, and developed formulas for compounding the uncertainties and for merging two estimates of the same location of an object. One of the drawbacks of this models is due to the singularities that suffer some angles for representing rotation matrix. A appropriate way to avoid this inconvenient is describe the rigid motion using the twist theory for describing rigid body kinematics because it does not suffer from singularities due to the use of local coordinates.

Paul (1981) presented a way of manipulation of differential relationships and how to transform differential changes in one coordinate frame into changes in another. He showed that differential rotations is independent of the order of rotation and are equivalent to a differential rotation made about a unit magnitude vector. Durrant-Whyte (1987) represented spatial uncertainties by differential transforms in the homogeneous transformation representation. However, Durrant-Whyte only proposed error models for homogeneous transforms and did not present any manipulation formula in terms of homogeneous transforms and in a probabilistic sense. Since it is in-

sufficient to only consider uncertainties locally, methodologies for uncertainty propagation for various actions and for verifying the success of an action or a plan are required.

Su and Lee (1991) and Su and Lee (1992) presented a systematic methodology of manipulation and propagating spatial uncertainties in the task-level planning verifying the applicability of actions in an assembly task using differential transform. The spatial uncertainties of motion parameters are characterized by mean vectors and covariance matrices and expressed in the form of homogeneous transforms. The manipulations of uncertainties focuses on compounding the uncertainties and fusing the estimates of the same location of an object. They proposed an approach of handling the problem of uncertainty fusion based on the first-order approximation, which do not require to extract six motion parameters from the homogeneous transforms in 3-dimensional space.

Papers aforementioned provide approaches for representing spatial uncertainty using the differential transformation of homogeneous transformation with Euler angles for representing rotation matrix, but as already mentioned, these approaches have difficulties because the solid angle swept out by a given change in Euler angles depends on the current pose. Euler angles are commonly used to constrain the rotation matrix, but they suffer from singularities and lead not to a simple formulation in the optimization procedure (for example Basu et al. (1996) propose a 3D ellipsoidal tracker based on Euler angles).

An appropriate way to avoid this inconvenient is describe the rigid motion using the twist and product of exponential map formalism for kinematic chains. There are two main advantages to using screws, twists, and wrenches for describing rigid body kinematics. The first is that they allow a global description of rigid body motion which does not suffer from singularities due to the use of local coordinates (such singularities are inevitable when one chooses to represent rotation via Euler angles, for example). The second advantage is that screw theory provides a very geometric description of rigid motion which greatly simplifies the analysis of mechanisms (Murray et al., 1994). This theory is built using the tools of matrix Lie groups and Lie algebras. Several distinct research fields relate to this theory. These include the theory of Lie groups, probability and statistics, robot kinematics, methods for describing spatial uncertainty, and state estimation.

The Lie-group-theoretic notation and terminology to the robotics community, which has now become standard vocabulary, is presented by Murray

et al. (1994) and Selig (1996). Blackmore and Leu (1992) showed that problems in manufacturing associated with swept volumes can be cast within a Lie-group setting. Park and Brockett (1994) showed how dexterity measures can be viewed in a Lie-group setting, and how this coordinate-free approach can be used in robot design. Kyatkin and Chirikjian (1998) showed that many problems in robot kinematics and motion planning can be formulated as the convolution of functions on the Euclidean group. Wang and Chirikjian (2004) showed that the workspace densities of manipulators with many degrees of freedom can be generated by solving a diffusion equation on the Euclidean group.

Recently, equivalent representations to differential representations of uncertainty have used the tools of Lie groups and Lie algebras to represent uncertainties using the twist and product of exponential map formalism for kinematic chains. How errors propagate on the Euclidean motion group, specifically in the Lie group SE(3), is shown in Smith et al. (2003) and Wang and Chirikjian (2006). In theses papers, poses and motions are represented as coordinate frame transformations. This transformations are represented by matrices which form the Lie Group and hence uncertainties in a pose or motion correspond to probability density functions over Lie group. Smith et al. (2003) showed how the exponential map can be used to represent probability distributions over a group, and how these can be correctly propagated along a trajectory.

In all previous studies about representing uncertainties with probabilities, errors are assumed to be small enough that covariances can be propagated by Jacobian-based methods or first-order error propagation theories. However, Wang and Chirikjian (2008) and Barfoot and Furgale (2014) addressed the problem of the propagation of large errors in rigid-body poses. Wang and Chirikjian (2008) showed how errors propagated by convolution on the Euclidean motion group, SE(3), can be approximated to second order using the theory of Lie algebras and Lie groups. Barfoot and Furgale (2014) propagating uncertainty through a compounding of two transformation matrices fusing multiple uncertain measurements of a pose into a single estimate and propagating uncertainty through a nonlinear model to second order in the associated covariances.

4.2 Absolute Representation of Uncertainty

In classical probabilistic models (Bolle and Cooper, 1986), (Smith and Cheeseman, 1986), the information about the location of the element B with respect to A is represented by a probability distribution function assumed to be Gaussian. The estimated location is given by the expected value of the vector $\mathbf{x}_{AB}$, and the uncertainty by its associated covariance matrix,

$$\begin{aligned} \bar{\mathbf{x}}_{AB} &= E[\mathbf{x}_{AB}] \\ \mathbf{\Sigma}_{AB} &= E[(\mathbf{x}_{AB} - \bar{\mathbf{x}}_{AB})(\mathbf{x}_{AB} - \bar{\mathbf{x}}_{AB})^T] \end{aligned} \tag{4.1}$$

where, in the 2D case, $\mathbf{x}_{AB} = (x_{AB}, y_{AB}, \theta_{AB})^T$ whilst in the 3D case, $\mathbf{x}_{AB} = (x_{AB}, y_{AB}, z_{AB}, \boldsymbol{\omega}_{AB})^T$ where the components of $\boldsymbol{\omega}_{AB}$ depend on the rotation's representation as detailed in chapter 3.

Let us consider two spatial relations $\mathbf{x}_{AB}$ and $\mathbf{x}_{BC}$ with their associated uncertainties $\mathbf{\Sigma}_{AB}$ and $\mathbf{\Sigma}_{BC}$ as is showed in Figure 4.1.

The compounding pose $\mathbf{x}_{AC}$ is computed by the nonlinear transformation,

$$\mathbf{x}_{AC} = \mathbf{x}_{AB} \oplus \mathbf{x}_{BC} \tag{4.2}$$

The uncertainty propagation is obtained by means of a first order Taylor expansion Smith and Cheeseman (1986) of Equation 4.2, thus,

$$\mathbf{x}_{AC} \simeq \bar{\mathbf{x}}_{AC} + \mathbf{J}_{1\oplus}(\mathbf{x}_{AB} - \bar{\mathbf{x}}_{AB}) + \mathbf{J}_{2\oplus}(\mathbf{x}_{BC} - \bar{\mathbf{x}}_{BC}) \tag{4.3}$$

where $\mathbf{J}_{1\oplus}$ and $\mathbf{J}_{2\oplus}$ are the Jacobians of the composition of locations vectors with respect to the first and second operand respectively, expressed in the 2D case by,

$$\mathbf{J}_{1\oplus} = \frac{\partial(\mathbf{x}_{AB} \oplus \mathbf{x}_{BC})}{\partial \mathbf{x}_{AB}} = \begin{pmatrix} 1 & 0 & y_{AB} - y_{AC} \\ 0 & 1 & x_{AC} - x_{AB} \\ 0 & 0 & 1 \end{pmatrix} \tag{4.4}$$

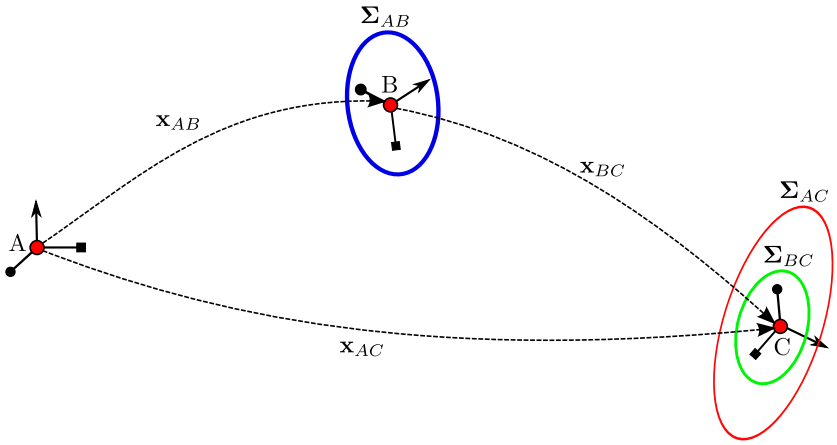

Figure 4.1: Composition using absolute representation.

$$\mathbf{J}_{2\oplus} = \frac{\partial(\mathbf{x}_{AB} \oplus \mathbf{x}_{BC})}{\partial \mathbf{x}_{AB}} = \begin{pmatrix} \cos\theta_{AB} & -\sin\theta_{AB} & 0 \\ \sin\theta_{AB} & \cos\theta_{AB} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.5}$$

and, in the 3D case, by,

$$\mathbf{J}_{1\oplus} = \frac{\partial(\mathbf{x}_{AB} \oplus \mathbf{x}_{BC})}{\partial \mathbf{x}_{AB}} = \begin{pmatrix} \mathbf{I} & \mathbf{M} \\ \mathbf{0} & \mathbf{K}_1 \end{pmatrix} \tag{4.6}$$

$$\mathbf{J}_{2\oplus} = \frac{\partial(\mathbf{x}_{AB} \oplus \mathbf{x}_{BC})}{\partial \mathbf{x}_{BC}} = \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 \end{pmatrix} \tag{4.7}$$

where $\mathbf{M}$, $\mathbf{R}$, $\mathbf{K}_1$ and $\mathbf{K}_2$ are defined in Smith et al. (1990) for relationships using Roll-Pitch-Yaw and Euler angles and in Blanco (2010) for relationships using unit quaternions. In chapter 3 expressions for the components of the Jacobians, required in subsequent sections, are detailed.

Finally, the estimated covariance matrix of $\mathbf{x}_{AC}$, assuming uncorrelated noise sequences, is computed from Equation 4.3 as,

$$\mathbf{\Sigma}_{AC} \simeq \begin{pmatrix} \mathbf{J}_{1\oplus} & \mathbf{J}_{2\oplus} \end{pmatrix} \begin{pmatrix} \mathbf{\Sigma}_{AB} & 0 \\ 0 & \mathbf{\Sigma}_{BC} \end{pmatrix} \begin{pmatrix} \mathbf{J}_{1\oplus}^T \\ \mathbf{J}_{2\oplus}^T \end{pmatrix} \tag{4.8}$$

Therefore, we can obtain,

$$\mathbf{\Sigma}_{AC} \simeq \mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T + \mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T \tag{4.9}$$

If the two relationships being compounded are not independent, then the covariance matrix of the compounding poses is,

$$\mathbf{\Sigma}_{AC} \simeq \begin{pmatrix}\mathbf{J}_{1\oplus} & \mathbf{J}_{2\oplus}\end{pmatrix} \begin{pmatrix}\mathbf{\Sigma}_{AB} & \mathbf{\Sigma}_{AB,BC} \\ \mathbf{\Sigma}_{BC,AB} & \mathbf{\Sigma}_{BC}\end{pmatrix} \begin{pmatrix}\mathbf{J}_{1\oplus}^T \\ \mathbf{J}_{2\oplus}^T\end{pmatrix} \tag{4.10}$$

In this case, we can obtained,

$$\mathbf{\Sigma}_{AC} \simeq \mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T + \mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T + \mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC,AB}\mathbf{J}_{1\oplus}^T + \mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB,BC}\mathbf{J}_{2\oplus}^T \tag{4.11}$$

where $\mathbf{\Sigma}_{AB,BC}$ is the correlation between the spatial relations $\mathbf{x}_{AB}$ and $\mathbf{x}_{BC}$.

4.3 Differential Representation of Uncertainty

Differential models use a local representation of uncertainty (Durrant-Whyte, 1987), (Paul, 1981), (Castellanos et al., 1999). The estimated location of the robot with respect to a base reference A is denoted by $\bar{\mathbf{x}}_{AB}$ representing the best approximation of the real location. Additionally, the estimation error is represented locally by a differential location vector $\mathbf{d}_B$ with respect to the reference frame B,

$$\mathbf{x}_{AB} = \bar{\mathbf{x}}_{AB} \oplus \mathbf{d}_B \tag{4.12}$$

where $\mathbf{d}_B = (dx, dy, d\theta)^T$ in 2D case and $\mathbf{d}_B = (dx, dy, dz, d\omega_x, d\omega_y, d\omega_z)^T$ in 3D case, is a differential location vectors being normally distributed with mean and covariance matrix given by,

$$\begin{aligned}\bar{\mathbf{d}}_B &= E[\mathbf{d}_B] \\ \mathbf{\Sigma}_B &= E[(\mathbf{d}_B - \bar{\mathbf{d}}_B)(\mathbf{d}_B - \bar{\mathbf{d}}_B)^T]\end{aligned} \tag{4.13}$$

that represents location uncertainty. The elements of $\mathbf{d}_B$ represents the differential translational and rotational along the axis, independently from the base reference frame A.

Additionally, the differential location vector can be represented with respect to the reference frame A,

$$\mathbf{x}_{AB'} = \mathbf{d}_A \oplus \mathbf{x}_{AB} \tag{4.14}$$

where $\mathbf{d}_A = (dx, dy, d\theta)^T$ in 2D case and $\mathbf{d}_A = (dx, dy, dz, d\omega_x, d\omega_y, d\omega_z)^T$ in 3D case, is a differential location vectors being normally distributed with mean and covariance matrix given by,

$$\begin{aligned} \bar{\mathbf{d}}_A &= E[\mathbf{d}_A] \\ \mathbf{\Sigma}_A &= E[(\mathbf{d}_A - \bar{\mathbf{d}}_A)(\mathbf{d}_A - \bar{\mathbf{d}}_A)^T] \end{aligned} \tag{4.15}$$

The differential location vectors expressed in reference A and B are different, in general, and are related by the Jacobian of the relative transformation (Paul, 1981),

$$\mathbf{d}_A = \mathbf{J}_{AB}\mathbf{d}_B \tag{4.16}$$

$$\mathbf{d}_B = \mathbf{J}_{BA}\mathbf{d}_A = \mathbf{J}_{AB}^{-1}\mathbf{d}_A \tag{4.17}$$

where $\mathbf{J}_{AB}$ is the Jacobian (Paul, 1981) of the relative transformation $\bar{\mathbf{x}}_{AB}$, expressed in the 2D case as,

$$\mathbf{J}_{AB} = \left(\begin{array}{c|c} \mathbf{R}_{AB} & \mathbf{R}_{AB}\begin{pmatrix} y_{AB} \\ -x_{AB} \end{pmatrix} \\ \hline 0 & 1 \end{array}\right) \tag{4.18}$$

and, in the 3D case, as,

$$\mathbf{J}_{AB} = \begin{pmatrix} \mathbf{R}_{AB} & \mathbf{D}_{AB}\mathbf{R}_{AB} \\ \mathbf{0} & \mathbf{R}_{AB} \end{pmatrix} \tag{4.19}$$

where $\mathbf{R}_{AB}$ is an orthogonal rotation matrix and $\mathbf{D}_{AB}$ is the 3×3 skew symmetric matrix related to the translational part of $\bar{\mathbf{x}}_{AB}$,

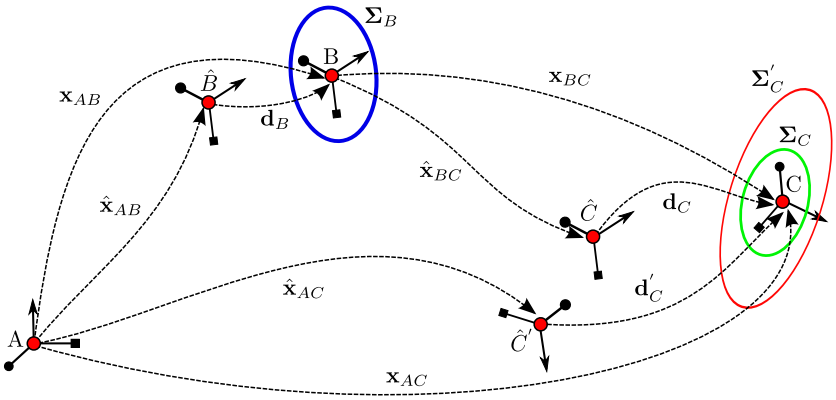

Figure 4.2: Composition using differential representation.

$$\mathbf{D}_{AB} = \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix} \tag{4.20}$$

In Figure 4.2, the composition of two spatial relations $\mathbf{x}_{AB}$ and $\mathbf{x}_{BC}$ with their corresponding differential location vectors $\mathbf{d}_B$ and $\mathbf{d}_C$ with associated covariance matrices $\mathbf{\Sigma}_B$ and $\mathbf{\Sigma}_C$ is illustrated.

The compounding pose $\mathbf{x}_{AC}$ is obtained by composition of two spatial relations $\mathbf{x}_{AB}$ and $\mathbf{x}_{BC}$ as,

$$\begin{aligned} \mathbf{x}_{AC} =& \mathbf{x}_{AB} \oplus \mathbf{x}_{BC} \\ =& \bar{\mathbf{x}}_{AB} \oplus \mathbf{d}_B \oplus \bar{\mathbf{x}}_{BC} \oplus \mathbf{d}_C \\ \simeq& \bar{\mathbf{x}}_{AB} \oplus \bar{\mathbf{x}}_{BC} \oplus \mathbf{J}_{CB}\mathbf{d}_B \oplus \mathbf{d}_C \\ =& \bar{\mathbf{x}}_{AC} \oplus \mathbf{d}'_C \end{aligned} \tag{4.21}$$

Assuming small errors, we can obtain,

$$\mathbf{d}'_C \simeq \mathbf{J}_{CB}\mathbf{d}_B + \mathbf{d}_C \tag{4.22}$$

Assuming uncorrelated noise sequences, i.e., if $\mathbf{d}_B$ and $\mathbf{d}_C$ are uncorrelated variable, the covariance matrix of $\mathbf{d}'_C$ with respect to the coordinate frame C is given by,

$$\mathbf{\Sigma}'_C \simeq \mathbf{J}_{CB}\mathbf{\Sigma}_B\mathbf{J}_{CB}^T + \mathbf{\Sigma}_C \tag{4.23}$$

where $\mathbf{J}_{CB}$ is the Jacobian, Paul (1981), of the relative transformation $\bar{\mathbf{x}}_{CB}$, expressed in the 2D case as,

$$\mathbf{J}_{CB} = \left(\begin{array}{c|c} \mathbf{R}_{BC}^T & \mathbf{R}_{BC}^T \begin{pmatrix} -y_{BC} \\ x_{BC} \end{pmatrix} \\ \hline 0 & 1 \end{array}\right) \tag{4.24}$$

and, in the 3D case, as,

$$\mathbf{J}_{CB} = \begin{pmatrix} \mathbf{R}_{BC}^T & \mathbf{R}_{BC}^T\mathbf{D}_{BC}^T \\ \mathbf{0} & \mathbf{R}_{BC}^T \end{pmatrix} \tag{4.25}$$

where $\mathbf{R}_{BC}$ is an orthogonal rotation matrix and $\mathbf{D}_{BC}$ is the 3×3 skew symmetric matrix related to the translational part of $\bar{\mathbf{x}}_{BC}$,

$$\mathbf{D}_{BC} = \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix} \tag{4.26}$$

If the independence assumption between the error vectors is not true, then the covariance matrix of the compounding poses is,

$$\mathbf{\Sigma}'_C \simeq \mathbf{J}_{CB}\mathbf{\Sigma}_B\mathbf{J}_{CB}^T + \mathbf{\Sigma}_C + \mathbf{J}_{CB}\mathbf{\Sigma}_{B,C} + \mathbf{\Sigma}_{C,B}\mathbf{J}_{CB}^T \tag{4.27}$$

where $\mathbf{\Sigma}_{B,C}$ is the correlation between the differential error $\mathbf{d}_B$ and $\mathbf{d}_C$.

4.4 Uncertainty Representation and Propagation over Lie Groups

Uncertainty robot's pose can be defined with Lie groups using the special Euclidean group. As in the model of differential representation, in this model the estimated location of the robot is represented by a 'large', noise free

value and the estimation error is represented locally by a 'small' uncertainty perturbation. However, Equation 4.12 does not directly apply to members of the special Euclidean group. For that, one notable concept used for applying error perturbation to transformation matrix is relationship between special Euclidean Group and its corresponding Lie algebra.

Uncertainty robot's pose can be defined with Lie groups using the special Euclidean group that represents rotation and translation, for 2D and 3D cases ($n = 2, 3$, respectively) as,

$$SE(n) = \left\{ \begin{pmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)} | \mathbf{R} \in SO(n), \mathbf{p} \in \mathbb{R}^n \right\} \tag{4.28}$$

where $SO(n)$ is the special orthogonal group defined as,

$$SO(n) = \{\mathbf{C} \in \mathbb{R}^{n\times n} | \mathbf{C}\mathbf{C}^T = \mathbf{I}, \det(\mathbf{C}) = 1\} \tag{4.29}$$

Every Lie group has an associated Lie algebra, which is the tangent space around the identity element of the group and they are related via exponential map Varadarajan (2013), Bregler and Malik (1998). Importantly, the tangent space associated with a Lie group provides an optimal space in which to represent differential quantities related to the group such as covariances of transformations. The Lie algebra corresponding to $SE(n)$ and $SO(n)$ is denoted by $\mathfrak{se}(n)$ and $\mathfrak{so}(n)$, respectively.

In this model, uncertainty robot's pose can be defined using homogeneous matrices by,

$$\mathbf{T}_{AB} = \bar{\mathbf{T}}_{AB} \exp(\hat{\mathbf{d}}_B) \tag{4.30}$$

where $\bar{\mathbf{T}}_{AB} \in SE(n)$ is a 'large', noise free value and, $\mathbf{d}_B = (dx, dy, d\theta)^T$ in 2D case and $\mathbf{d}_B = (dx, dy, dz, d\omega_x, d\omega_y, d\omega_z)^T$ in 3D case, is a 'small' uncertainty perturbation variable with respect to the reference frame B, being assumed usually to be Gaussian,

$$\begin{aligned} \bar{\mathbf{d}}_B &= E[\mathbf{d}_B] \\ \mathbf{\Sigma}_B &= E[(\mathbf{d}_B - \bar{\mathbf{d}}_B)(\mathbf{d}_B - \bar{\mathbf{d}}_B)^T] \end{aligned} \tag{4.31}$$

For notational convenience, the hat-operator $\hat{\cdot}$ in Equation 4.30 turns the vector $\mathbf{d}_B$ in a member of the Lie algebra $\mathfrak{se}(n)$ according to, in the 2D case,

$$\hat{\mathbf{d}}_B = \left(\begin{array}{c|c} [d\theta]_\times & \begin{matrix} dx \\ dy \end{matrix} \\ \hline \mathbf{0} & 0 \end{array}\right) \tag{4.32}$$

being $[d\theta]_\times$ the 2×2 skew symmetric matrix corresponding to the hat-operator in $\mathfrak{so}(2)$,

$$[d\theta]_\times = \begin{pmatrix} 0 & -d\theta \\ d\theta & 0 \end{pmatrix} \tag{4.33}$$

and in the 3D case,

$$\hat{\mathbf{d}}_B = \left(\begin{array}{c|c} & \begin{matrix} dx \\ dy \\ dz \end{matrix} \\ [\boldsymbol{d\omega}]_\times & \\ \hline \mathbf{0} & 0 \end{array}\right) \tag{4.34}$$

being $[\boldsymbol{d\omega}]_\times$ is the 3×3 skew symmetric matrix corresponding to the hat-operator in $\mathfrak{so}(3)$,

$$[\boldsymbol{d\omega}]_\times = \begin{pmatrix} 0 & -d\omega_z & d\omega_y \\ d\omega_z & 0 & -d\omega_x \\ -d\omega_y & d\omega_x & 0 \end{pmatrix} \tag{4.35}$$

The exponential map in Equation 4.30 is a way to associate elements of the tangent space $\mathfrak{se}(n)$ to elements of the underlying Lie group $SE(n)$.

The exponential map $\mathfrak{se}(2)$ to $SE(2)$ has closed form,

$$\exp(\hat{\mathbf{d}}_B) = \left(\begin{array}{c|c} \exp([d\theta]_\times) & \mathbf{V} \begin{pmatrix} dx \\ dy \end{pmatrix} \\ \hline \mathbf{0} & 1 \end{array}\right) \tag{4.36}$$

where $\exp([d\theta]_\times)$ is the exponential map $\mathfrak{so}(2)$ to $SO(2)$ defined by,

$$\exp([d\theta]_\times) = \exp\begin{pmatrix} 0 & -d\theta \\ d\theta & 0 \end{pmatrix} = \begin{pmatrix} \cos(d\theta) & -\sin(d\theta) \\ \sin(d\theta) & \cos(d\theta) \end{pmatrix} \tag{4.37}$$

and,

$$\mathbf{V} = \begin{pmatrix} \frac{\sin(d\theta)}{d\theta} & -\frac{1-\cos(d\theta)}{d\theta} \\ \frac{1-\cos(d\theta)}{d\theta} & \frac{\sin(d\theta)}{d\theta} \end{pmatrix} \tag{4.38}$$

The exponential map $\mathfrak{se}(3)$ to $SE(3)$ has closed form:

$$\exp(\hat{\mathbf{d}}_B) = \left(\begin{array}{c|c} \exp([\boldsymbol{d\omega}]_\times) & \mathbf{V}\begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} \\ \hline \mathbf{0} & 1 \end{array}\right) \tag{4.39}$$

where $\exp([\boldsymbol{d\omega}]_\times)$ is the exponential map $\mathfrak{so}(3)$ to $SO(3)$ also called the Rodrigues formula defined by,

$$\exp([\boldsymbol{d\omega}]_\times) = \mathbf{I} + \frac{\sin\theta}{\theta}[\boldsymbol{d\omega}]_\times + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{d\omega}]_\times^2 \tag{4.40}$$

where $d\theta = \sqrt{d\omega_x^2 + d\omega_y^2 + d\omega_z^2}$ and,

$$\mathbf{V} = \mathbf{I} + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{d\omega}]_\times + \frac{\theta - sin\theta}{\theta^3}[\boldsymbol{d\omega}]_\times^2 \tag{4.41}$$

More information about the mathematical formulation in Lie groups can be found in chapter 3.

As in differential representation, the uncertainty perturbation variable can be represented with respect to the reference frame A, then uncertainty robot's pose is defined using homogeneous matrices as,

$$\mathbf{T}_{AB} = \exp(\hat{\mathbf{d}}_A)\bar{\mathbf{T}}_{AB} \tag{4.42}$$

where $\mathbf{d}_A = (dx, dy, d\theta)^T$ in 2D case and $\mathbf{d}_A = (dx, dy, dz, d\omega_x, d\omega_y, d\omega_z)^T$ in 3D case, is assumed usually to be Gaussian,

$$\begin{aligned}\bar{\mathbf{d}}_A &= E[\mathbf{d}_A] \\ \mathbf{\Sigma}_A &= E[(\mathbf{d}_A - \bar{\mathbf{d}}_A)(\mathbf{d}_A - \bar{\mathbf{d}}_A)^T]\end{aligned} \tag{4.43}$$

The uncertainty perturbation variable with respect reference A and B are related by the adjoint representation (or adjoint action) of a Lie group. The adjoint representation is a way of representing the elements of the group as linear transformations of the group's Lie algebra, i.e., transforms a tangent vector from the tangent space around one element to tangent space of another. Therefore, $\mathbf{d}_A$ and $\mathbf{d}_B$ are related as,

$$\mathbf{d}_A = Ad_{\mathbf{T}_{AB}}\mathbf{d}_B \tag{4.44}$$

$$\mathbf{d}_B = Ad_{\mathbf{T}_{BA}}\mathbf{d}_A \tag{4.45}$$

where $Ad_{\mathbf{T}_{AB}}$ is the the adjoint of $\mathbf{T}_{AB}$ in $SE(2)$ and $SE(3)$ is defined, respectively, as,

$$Ad_{\mathbf{T}_{AB}} = \left(\begin{array}{c|c} \mathbf{R}_{AB} & \mathbf{R}_{AB}\begin{pmatrix} y_{AB} \\ -x_{AB} \end{pmatrix} \\ \hline 0 & 1 \end{array}\right) \tag{4.46}$$

$$Ad_{\mathbf{T}_{AB}} = \begin{pmatrix} \mathbf{R}_{AB} & \mathbf{D}_{AB}\mathbf{R}_{AB} \\ 0 & \mathbf{R}_{AB} \end{pmatrix} \tag{4.47}$$

where $\mathbf{R}_{AB}$ is an orthogonal rotation matrix and $\mathbf{D}_{AB}$ is the 3×3 skew symmetric matrix related to the translational part of $\bar{\mathbf{T}}_{AB}$.

Let us consider two noisy poses, $\mathbf{T}_{AB}$ and $\mathbf{T}_{BC}$, with perturbation error associated $\mathbf{d}_B$ and $\mathbf{d}_C$. The uncertainty at $\mathbf{T}_{AC}$ can easily be estimated by propagating forwards,

$$\mathbf{T}_{AC} = \mathbf{T}_{AB}\mathbf{T}_{BC} = \bar{\mathbf{T}}_{AB}\exp(\hat{\mathbf{d}}_B)\bar{\mathbf{T}}_{BC}\exp(\hat{\mathbf{d}}_B) \tag{4.48}$$

A useful property of the adjoint representation to $\mathbf{Y} \in \mathfrak{se}(n)$ and $\mathbf{X} \in SE(n)$ is,

$$\exp\left(Ad_{\mathbf{X}}\mathbf{Y}\right) = \mathbf{X}\exp\left(\mathbf{Y}\right)\mathbf{X}^{-1} \tag{4.49}$$

or equivalently,

$$\exp\left(\mathbf{Y}\right) = \mathbf{X}^{-1}\exp\left(Ad_{\mathbf{X}}\mathbf{Y}\right)\mathbf{X} \tag{4.50}$$

Then, using $\mathbf{Y} = \hat{\mathbf{d}}_B$, $\mathbf{X} = \bar{\mathbf{T}}_{BC}^{-1}$ and the adjoint to correctly gather all of uncertainty to the right-hand side, we can obtain,

$$\begin{aligned}
\mathbf{T}_{AC} =& \bar{\mathbf{T}}_{AB}\exp\left(\hat{\mathbf{d}}_B\right)\bar{\mathbf{T}}_{BC}\exp\left(\hat{\mathbf{d}}_C\right) \\
=& \bar{\mathbf{T}}_{AB}\bar{\mathbf{T}}_{BC}\exp\left(Ad_{\mathbf{T}_{BC}^{-1}}\hat{\mathbf{d}}_B\right)\bar{\mathbf{T}}_{BC}^{-1}\bar{\mathbf{T}}_{BC}\exp\left(\hat{\mathbf{d}}_C\right) \\
=& \bar{\mathbf{T}}_{AB}\bar{\mathbf{T}}_{BC}\exp\left(Ad_{\mathbf{T}_{BC}^{-1}}\hat{\mathbf{d}}_B\right)\exp\left(\hat{\mathbf{d}}_C\right) \\
=& \bar{\mathbf{T}}_{AB}\bar{\mathbf{T}}_{BC}\exp\left(Ad_{\mathbf{T}_{BC}^{-1}}\hat{\mathbf{d}}_B + \hat{\mathbf{d}}_C\right) \\
=& \bar{\mathbf{T}}_{AC}\exp\left(\hat{\mathbf{d}}'_C\right)
\end{aligned} \tag{4.51}$$

where $Ad_{\mathbf{T}_{BC}^{-1}}$ is the adjoint representation of transformation $\bar{\mathbf{T}}_{BC}^{-1}$.

Therefore, the uncertainty perturbation matrix of $\mathbf{T}_{AC}$ is,

$$\hat{\mathbf{d}}'_C = Ad_{\mathbf{T}_{BC}^{-1}}\hat{\mathbf{d}}_B + \hat{\mathbf{d}}_C \tag{4.52}$$

Assuming that the perturbation error $\mathbf{d}_B$ and $\mathbf{d}_C$ are uncorrelated with one another, the uncertainty of $\mathbf{T}_{AC}$ is,

$$\mathbf{\Sigma}'_C \simeq Ad_{\mathbf{T}_{BC}^{-1}}\mathbf{\Sigma}_B Ad_{\mathbf{T}_{BC}^{-1}}^T + \mathbf{\Sigma}_C \tag{4.53}$$

where the adjoint of $\mathbf{T}_{BC}^{-1}$ in $SE(2)$ and $SE(3)$ is defined, respectively, as,

$$Ad_{\mathbf{T}_{BC}^{-1}} = \left(\begin{array}{c|c} \mathbf{R}_{BC}^T & \mathbf{R}_{BC}^T\begin{pmatrix}-y_{BC}\\ x_{BC}\end{pmatrix} \\ \hline 0 & 1 \end{array}\right) \tag{4.54}$$

$$Ad_{\mathbf{T}_{BC}^{-1}} = \begin{pmatrix} \mathbf{R}_{BC}^T & \mathbf{R}_{BC}^T\mathbf{D}_{BC} \\ 0 & \mathbf{R}_{BC}^T \end{pmatrix} \tag{4.55}$$

where $\mathbf{R}_{BC}$ is an orthogonal rotation matrix and $\mathbf{D}_{BC}$ is the 3×3 skew symmetric matrix related to the translational part of $\bar{\mathbf{T}}_{BC}$.

This result is one which can be obtained from the theory of extended Kalman filtering Smith et al. (2003), and also has been obtained using propagate error densities over Lie groups in Wang and Chirikjian (2006).

If the independence assumption between the perturbation error $\mathbf{d}_B$ and $\mathbf{d}_C$ are correlated, then the covariance matrix of the compounding poses is,

$$\mathbf{\Sigma}'_C \simeq Ad_{\mathbf{T}^{-1}_{BC}} \mathbf{\Sigma}_B Ad^T_{\mathbf{T}^{-1}_{BC}} + \mathbf{\Sigma}_C + Ad_{\mathbf{T}^{-1}_{BC}} \mathbf{\Sigma}_{B,C} + \mathbf{\Sigma}_{C,B} Ad_{\mathbf{T}^{-1}_{BC}} \tag{4.56}$$

where $\mathbf{\Sigma}_{B,C}$ is the correlation between $\mathbf{d}_B$ and $\mathbf{d}_C$.

4.5 Differential Uncertainty vs. Lie Algebra Uncertainty

In the previous section we have presented several ways to represent the uncertainty of the robot's pose. Observing Equation 4.23 and Equation 4.53 a great similarity between both is observed. Comparatively the expressions of the Jacobian of the differential transformations in Equation 4.24 and Equation 4.25 and the adjoint representation of the Lie group Equation 4.54 and Equation 4.55 are similar, respectively.

This situation leads us to think about what are the similarities and differences between representation of the uncertainty of robot's pose over Lie groups and using differential location vectors. For that, in this section we are going to analyze the correspondence between the representation of the uncertainty of robot's pose in both models.

In differential models, a location vector $\mathbf{x}_{AB}$ is represented locally by Equation 4.12,

$$\mathbf{x}_{AB} = \bar{\mathbf{x}}_{AB} \oplus \mathbf{d}_B \tag{4.57}$$

The above expression can be represented by a homogeneous transforms as follows,

$$\mathbf{T}_{AB} = \bar{\mathbf{T}}_{AB}\mathbf{T}_B \tag{4.58}$$

On the other hand, the homogeneous transforms $\mathbf{T}_{AB}$ assotiated to the location vector $\mathbf{x}_{AB}$ can be represented over the Lie groups as in Equation 4.30,

$$\mathbf{T}_{AB} = \bar{\mathbf{T}}_{AB} \exp(\hat{\mathbf{d}}_B) \tag{4.59}$$

In the following we analyze if the homogeneous transform of uncertainty representation, $\mathbf{T}_B$ and $\exp(\hat{\mathbf{d}}_B)$, are similar in both models for 2D and 3D cases.

4.5.1 2-dimensional space

A differential location vector $\mathbf{d}_B = (dx, dy, d\theta)^T$ can be represented by homogeneous matrix as,

$$\mathbf{H} = \mathrm{Hom}(\mathbf{d}_B) = \begin{pmatrix} \cos(d\theta) & -\sin(d\theta) & dx \\ \sin(d\theta) & \cos(d\theta) & dy \\ 0 & 0 & 1 \end{pmatrix} \tag{4.60}$$

For a differential change $d\theta$ the corresponding trigonometric functions become,

$$\lim_{d\theta \to 0} \sin(d\theta) \to d\theta \tag{4.61}$$

$$\lim_{d\theta \to 0} \cos(d\theta) \to 1 \tag{4.62}$$

Therefore,

$$\mathbf{H} = \mathrm{Hom}(\mathbf{d}_B) = \begin{pmatrix} 1 & -d\theta & dx \\ d\theta & 1 & dy \\ 0 & 0 & 1 \end{pmatrix} \tag{4.63}$$

In Lie groups theory, $\exp(\hat{\mathbf{d}}_B)$ is defined as,

$$\exp(\hat{\mathbf{d}}_B) = \left(\begin{array}{c|c} \exp([d\theta]_\times) & \mathbf{V}\begin{pmatrix} dx \\ dy \end{pmatrix} \\ \hline \mathbf{0} & 1 \end{array}\right) = \tag{4.64}$$

$$= \left(\begin{array}{cc|cc} \cos(d\theta) & -\sin(d\theta) & \frac{\sin(d\theta)}{d\theta} & -\frac{1-\cos(d\theta)}{d\theta} \\ \sin(d\theta) & \cos(d\theta) & \frac{1-\cos(d\theta)}{d\theta} & \frac{\sin(d\theta)}{d\theta} \end{array} \begin{pmatrix} dx \\ dy \end{pmatrix} \right) \tag{4.65}$$

$$\begin{array}{cc|c} & & \\ \multicolumn{2}{c|}{\mathbf{0}} & 1 \end{array}$$

Taking into account that $\cos(d\theta) \simeq 1$ and $\sin(d\theta) \simeq d\theta$,

$$\exp([d\theta]_\times) = \begin{pmatrix} 1 & -d\phi_z & dx \\ d\phi_z & 1 & dy \\ 0 & 0 & 1 \end{pmatrix} \tag{4.66}$$

It can be seen that Equation 4.63 and Equation 4.66 are similar. Therefore, the homogeneous transformation of uncertainty are the same for both models in 2-dimensional space.

4.5.2 3-dimensional space

In 3D case, the homogeneous matrix corresponding to differential vector $\mathbf{d}_B = (dx, dy, dz, d\omega_x, d\omega_y, d\omega_z)^T$ can be represented as homogeneous transform in different ways depend on the common alternatives to representing the rotational part: Euler angles (Roll-Pitch-Yaw), orthogonal rotation matrices from SO(3) and unit quaternions (see chapter 3).

Using Roll-Pitch-Yaw representation, the homogeneous matrix associated to $\mathbf{d}_B$ is defined as,

$$\mathbf{H} = \mathrm{Hom}(\mathbf{d}_B) = \begin{pmatrix} h_{11} & h_{12} & h_{13} & dx \\ h_{21} & h_{22} & h_{23} & dy \\ h_{31} & h_{32} & h_{33} & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.67}$$

with,

$$
\begin{aligned}
h_{11} &= \cos(d\omega_z)\cos(d\omega_y) \\
h_{12} &= \cos(d\omega_z)\sin(d\omega_y)\sin(d\omega_x) - \sin(d\omega_z)\cos(d\omega_x) \\
h_{13} &= \cos(d\omega_z)\sin(d\omega_y)\cos(d\psi) + \sin(d\phi)\sin(d\psi) \\
h_{21} &= \sin(d\omega_z)\cos(d\omega_y) \\
h_{22} &= \sin(d\omega_z)\sin(d\omega_y)\sin(d\omega_x) + \cos(d\omega_z)\cos(d\psi) \\
h_{23} &= \sin(d\omega_z)\sin(d\omega_y)\cos(d\omega_x) - \cos(d\omega_z)\sin(d\psi) \\
h_{31} &= -\sin(d\omega_z) \\
h_{32} &= \cos(d\omega_y)\sin(d\omega_x) \\
h_{33} &= \cos(d\omega_y)\cos(d\omega_x)
\end{aligned}
$$

As in 2-dimensional case, taking into account that $\cos(d\theta) \simeq 1$ and $\sin(d\theta) \simeq d\theta$,

$$
\mathbf{H} = \mathrm{Hom}(\mathbf{d}_B) = \begin{pmatrix} 1 & -d\omega_z & d\omega_y & dx \\ d\omega_z & 1 & -d\omega_x & dy \\ -d\omega_y & d\omega_x & 1 & dy \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.68}
$$

In Lie groups theory, the rotational part is represented by an orthogonal rotation matrix from SO(3) and the homogeneous transformation associated to differential vector $\mathbf{d}_B$ is $\exp(\hat{\mathbf{d}}_B)$ is defined as,

$$
\exp(\hat{\mathbf{d}}_B) = \left(\begin{array}{c|c} \exp([\boldsymbol{d\omega}]_\times) & \mathbf{V}\begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} \\ \hline \mathbf{0} & 1 \end{array}\right) \tag{4.69}
$$

where $\exp([\boldsymbol{d\omega}]_\times)$ is the exponential map $\mathfrak{so}(3)$ to $SO(3)$ defined by,

$$
\exp([\boldsymbol{d\omega}]_\times) = \mathbf{I} + \frac{\sin(d\theta)}{d\theta}[\boldsymbol{d\omega}]_\times + \frac{1-\cos(d\theta)}{d\theta^2}[\boldsymbol{d\omega}]_\times^2 \tag{4.70}
$$

where $d\theta = \sqrt{d\omega_x^2 + d\omega_y^2 + d\omega_z^2}$, $\mathbf{dk} = \boldsymbol{d\omega}$ and,

$$
\mathbf{V} = \mathbf{I} + \frac{1-\cos(d\theta)}{d\theta^2}[\boldsymbol{d\omega}]_\times + \frac{d\theta - sin(d\theta)}{d\theta^3}[\boldsymbol{d\omega}]_\times^2 \tag{4.71}
$$

Taking into account that $\cos(d\theta) \simeq 1$ and $\sin(d\theta) \simeq d\theta$:

$$\exp(\hat{\mathbf{d}}_B) = \begin{pmatrix} 1 & -d\omega_z & d\omega_y & dx \\ d\omega_z & 1 & -d\omega_x & dy \\ -d\omega_y & d\omega_x & 1 & dy \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.72}$$

The unit-quaternion representation $(q_r, q_x, q_y, q_z)^T$ has the special property that it can be considered as a rotation of θ about the unit vector $\mathbf{k} = (k_x, k_y, k_z)$ which are related to the quaternion components by,

$$q_r = \cos\frac{\theta}{2} \tag{4.73}$$

$$q_x = \left(\sin\frac{\theta}{2}\right) k_x \tag{4.74}$$

$$q_y = \left(\sin\frac{\theta}{2}\right) k_y \tag{4.75}$$

$$q_z = \left(\sin\frac{\theta}{2}\right) k_z \tag{4.76}$$

and is similar to the angle-axis representation.

Transformations representing translation and rotation θ about a vector $\mathbf{k}$ associated to the differential vector $\mathbf{d}_B = (dx, dy, dz, d\omega_x, d\omega_y, d\omega_z)^T$ can be written as,

$$\mathbf{H} = \left(\begin{array}{c|c} \multirow{3}{*}{$\mathbf{R}$} & dx \\ & dy \\ & dz \\ \hline \mathbf{0} & 1 \end{array}\right) \tag{4.77}$$

where $d\theta = \sqrt{d\omega_x^2 + d\omega_y^2 + d\omega_z^2}$, $\mathbf{dk} = \boldsymbol{d\omega}$ and $\mathbf{R}$ is the rotation matrix associated to $\mathbf{d}_B$ defined by the Rodrigues formula in Equation 4.40,

$$\mathbf{R} = \mathbf{I} + \frac{\sin(d\theta)}{d\theta}[\boldsymbol{dk}]_\times + \frac{1 - \cos(d\theta)}{d\theta^2}[\boldsymbol{dk}]_\times^2 \tag{4.78}$$

Taking into account that $\cos(d\theta) \simeq 1$ and $\sin(d\theta) \simeq d\theta$,

$$\mathbf{H} = \begin{pmatrix} 1 & -dk_z & dk_y & dx \\ dk_z & 1 & -dk_x & dy \\ -dk_y & dk_x & 1 & dy \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.79}$$

In this case Equation 4.68, Equation 4.72 and Equation 4.79 are similar. Additionally, in (Paul, 1981), it was shown that, differential rotations transformations about the x, y and z axis are independent of the order of rotation and are equivalent to a differential rotation made about a unit magnitude vector.

Therefore, the representation of the uncertainty of robot's pose over Lie groups and using differential location vectors are similar independent of the representation used for the rotational part associated to robot's pose $\mathbf{x}_{AB}$. This situation is due to in differential representations, the underlying idea is that differential transformations are transformations whose elements are the derivatives of the original transformation elements. Specifically, for transformations which represent translation and rotation, the derivative can be expressed as a differential translation and rotation. Specifically, the space of differential transformations for translation and rotation is the Lie algebra $\mathfrak{se}(3)$.

However, the homogeneous transformation associated to the approximation of the real location $\bar{\mathbf{x}}_{AB}$ is different depend on the different ways of representation the rotational part. In contrast to the Euler-angle based representation and unit quaternions, representing the pose and its associated uncertainty over Lie groups guarantee properties such as a minimal representation and absence of singularities.

4.6 Conclusions

Representation, estimation and propagation of the uncertainty of robot's pose is a fundamentally important problem that has received attention for years. Different types of models of probabilistic representation of the uncertainty have been proposed in the literature.

In absolute representation the estimated location of the robot is given by the expected value and the uncertainty by its associated covariance matrix,

while the differential representation uses a local representation of the uncertainty where the estimated location of robot is the best approximation of the real location and the estimation error is represented locally by a differential location vector. Equivalent representations over the Lie group have also been considered where uncertainties of the location of the robot are represented in the tangent space associated to the Lie group.

We review related work about representation and propagation of the uncertainty of robot's pose and present a survey of different types of models proposed in the literature. Additionally, an analysis of the uncertainty represented with a differential uncertainty vector and the uncertainty propagation on Lie groups is carried out.

Our fundamental conclusion is that uncertainty representation over Lie groups and using differential location vectors are similar and are independent of the representation used for rotational part of the robot's pose. This is due to the uncertainty is represented locally in the space of differential transformations for translation and rotation that correspond with the Lie algebra of special Euclidean group SE(n).

However, in 3-dimensional space, the homogeneous transformation associated to the approximation of the real location because it depends on the different ways of representation the rotational part. Therefore, a proper way to jointly manipulation the estimation and uncertainty of the pose is to use the theory of Lie groups due to is a representation to guarantee properties such as a minimal representation and absence of singularities in rotation angles.

CHAPTER 5

Utility Function Monotonicity: the Importance of Uncertainty Representation

This chapter focuses on the paramount importance of *quantifying* uncertainty to correctly report the associated confidence of the robot's location estimate at each time step along its trajectory and therefore deciding the correct course of action in an active SLAM mission.

Remembering the definition of active SLAM, the objective of an active SLAM algorithm is to plan ahead the robot motion in order to maximize the area explored and minimize the uncertainty associated with the estimation, all within a time and computation budget. During the exploration phase of a SLAM algorithm, where the robot navigates in a previously unknown region, the uncertainty associated with the robot's localization grows unbounded (Kelly, 2004). Only after revisiting previously known regions a reduction in the robot's localization uncertainty is expected by detecting loop-closures (Durrant-Whyte and Bailey, 2006). In an active SLAM scenario, guaranteeing monotonicity of these decision making criteria during exploration is utmost importance to making correct decisions.

The most common approaches to quantifying uncertainty in SLAM are based on real scalar functions of the covariance matrix. Some active SLAM algorithms rely on optimality criteria which aim at quantifying the map and robot's pose uncertainty, namely *A-opt* (trace of the covariance matrix, or sum of its eigenvalues (Chernoff, 1953), (Leung et al., 2006), (Kollar and Roy, 2008), (Meger et al., 2008)), *D-opt* (determinant of the covariance matrix,

or product of its eigenvalues (Wald, 1943), (Vidal-Calleja et al., 2006), (Kim and Eustice, 2013)) and *E-opt* (largest eigenvalue (Ehrenfeld, 1955)) criteria. Alternatively, other active SLAM algorithms, based on Information Theory (Stachniss et al., 2005), (Blanco et al., 2008), rely on the use of the Shannon's entropy to select courses of action for the robot to reach the commanded goal location.

We investigate the monotonicity of different decision making criteria, both in 2-dimensional and 3-dimensional space, depending on the representation of uncertainty and of the orientation of the robot's pose. We analytically show that, using differential representations to propagate spatial uncertainties, monotonicity is preserved for all optimality criteria, *A-opt*, *D-opt* and *E-opt* and for Shannon's entropy. We also show that monotonicity does not hold for any of them in absolute representations using Roll-Pitch-Yaw and Euler angles. Finally, we show that using unit quaternions in absolute representations, the only criteria that preserve monotonicity are *D-opt* and Shannon's entropy.

5.1 Introduction

Simultaneous Localization and Mapping (SLAM) algorithm estimate the pose of a robot as well as the associated uncertainty and the map of the environment at the same time. This algorithm can be divided into two phases: exploration and loop closure. During the exploration phase the robot moves through unknown environment and during loop closure phase the robot revisits previously explored regions of the environment.

Kelly (2004) shows that the uncertainty in the robot's pose estimate grows while the robot is performing dead-reckoning, i.e., during the exploration phase, the uncertainty associated with the robot's localization grows with time as a complete model of the robot's motion may not be available or the sensors' readings may be noisy (Kelly, 2004), (LaValle, 2006). This uncertainty grows unbounded until the robot performs a loopclosure, in which case, a reduction in the robot's localization uncertainty is expected (Frese, 2006).

In an active SLAM scenario, guaranteeing monotonicity of these decision making criteria during exploration, i.e. quantifying correctly that the uncertainty encapsulated in a covariance matrix is increasing, is an essential

step towards making correct decisions. As already mentioned, during exploration the uncertainty associated with the robot's localization increases (Feder et al., 1999). Therefore, if monotonicity of the criteria considered is not preserved, the system might select courses of action or paths that it falsely believes lead to less uncertainty in the robot.

As mentioned in chapter 2, the most common approaches to quantifying uncertainty in SLAM are based on real scalar functions of the covariance matrix. Some active SLAM algorithms rely on optimality criteria which aim at quantifying the map and robot's pose uncertainty, namely *A-opt* (trace of the covariance matrix, or sum of its eigenvalues) (Chernoff, 1953), (Leung et al., 2006), (Kollar and Roy, 2008), (Meger et al., 2008), *D-opt* (determinant of the covariance matrix, or product of its eigenvalues) (Wald, 1943), (Vidal-Calleja et al., 2006), (Kim and Eustice, 2013) and *E-opt* (largest eigenvalue) (Ehrenfeld, 1955) criteria.

In the literature, *D-opt* has been disregarded as unfruitful criterion for it does not allow checking task completion as *A-opt* does. Furthermore, it may not provide fruitful information as it can be driven quickly to zero (Mihaylova et al., 2003), (Sim and Roy, 2005), (Lefebvre et al., 2005). Mihaylova et al. (2003) have done comparisons between uncertainty criteria in order to determine if there is a criterion that converges faster to a desired solution. Sim and Roy (2005) and Lefebvre et al. (2005) reported that optimizing the *A-opt* criterion results in a more accurate map than existing approaches such as *D-opt*. However, Kiefer (1974) and Pukelsheim (2006) state that *D-opt* has more appealing characteristics than *A-opt* and *E-opt* because *D-opt* is the criterion that captures the global uncertainty due to all the elements of a covariance matrix. Moreover, Kiefer (1974) demonstrated that *A-opt*, *D-opt* and *E-opt* belong to a general family of uncertainty criteria. Recently, Carrillo et al. (2012) presented a comparison of *A-opt* and *D-opt*. They demonstrate that *D-opt* is capable of giving fruitful information as a metric for the uncertainty and performs comparably to *A-opt*. Through various experiments they show that the use of *D-opt* has desirable effects in various SLAM related tasks such as active mapping and exploration.

Alternatively, other active SLAM algorithms, based on Information Theory (Stachniss et al., 2005), (Blanco et al., 2008), rely on the use of the Shannon's entropy to select courses of action for the robot to reach the commanded goal location. Recently, Carrillo et al. (2018) presented a novel information- theoretic utility function for selecting actions in a robot based autonomous exploration task. The authors simultaneously considered uncer-

tainty in both the robot pose and the map computing the difference between the Shannon and the Rényi entropy of the current distribution over maps.

5.2 Monotonicity in Absolute Representations of Uncertainty

The following analysis about the monotonicity of the optimality criteria using absolute representation for the propagation of the uncertainty is presented for 2-dimensional and 3-dimensional space. We analyze theoretically the monotonicity of each optimality criteria. The reported analysis is valid for common interpretations of the orientation angles: Roll-Pitch-Yaw, Euler angles and unit quaternions.

5.2.1 A-optimality

Monotonicity for the *A-opt* criterium requires that the trace of the covariance matrix of the compounding pose be larger than or equal to the trace of the covariance matrix of the initial pose, that is,

$$\mathrm{tr}(\boldsymbol{\Sigma}_{AC}) - \mathrm{tr}(\boldsymbol{\Sigma}_{AB}) \geq 0 \tag{5.1}$$

In chapter 4, we present the uncertainty propagation of compounding two poses for absolute representation in Equation 4.9 as,

$$\boldsymbol{\Sigma}_{AC} \simeq \mathbf{J}_{1\oplus}\boldsymbol{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T + \mathbf{J}_{2\oplus}\boldsymbol{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T \tag{5.2}$$

Computing the trace in both sides of above equation and substituting gives the difference between $\mathrm{tr}(\boldsymbol{\Sigma}_{AC})$ and $\mathrm{tr}(\boldsymbol{\Sigma}_{AB})$ as,

$$\mathrm{tr}(\boldsymbol{\Sigma}_{AC}) - \mathrm{tr}(\boldsymbol{\Sigma}_{AB}) = \mathrm{tr}(\mathbf{J}_{1\oplus}\boldsymbol{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T) + \mathrm{tr}(\mathbf{J}_{2\oplus}\boldsymbol{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T) - \mathrm{tr}(\boldsymbol{\Sigma}_{AB}) \tag{5.3}$$

In order to evaluate the previous expression we will analyze each of the terms for the 2D and 3D cases.

2-dimensional space

The term $\mathrm{tr}(\mathbf{\Sigma}_{AB}) \geq 0$ by definition because $\mathbf{\Sigma}_{AB}$ is a covariance matrix. To analyze the other terms, we use the following useful property of positive semidefinite matrices: if P is a non-singular matrix (i.e. $\det(P) \neq 0$) and if A is positive semidefinite matrix then $P^{'}AP$ is positive definite (positive semidefinite) matrix.

In the two dimensional space, the jacobian of the transformation of compounding $\mathbf{x}_{AC} = \mathbf{x}_{AB} \oplus \mathbf{x}_{BC}$ is defined as in chapter 3. Therefore,

$$\det(\mathbf{J}_{1\oplus}) = \det \begin{pmatrix} 1 & 0 & y_{AB} - y_{AC} \\ 0 & 1 & x_{AC} - x_{AB} \\ 0 & 0 & 1 \end{pmatrix} = 1 \tag{5.4}$$

$$\det(\mathbf{J}_{2\oplus}) = \det \begin{pmatrix} \cos\theta_{AB} & -\sin\theta_{AB} & 0 \\ \sin\theta_{AB} & \cos\theta_{AB} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \cos^2\theta_{AB} + \sin^2\theta_{AB} = 1 \tag{5.5}$$

Then, $\mathbf{J}_{1\oplus}$ and $\mathbf{J}_{2\oplus}$ are non-singular matrices and $\mathbf{\Sigma}_{AB}$ and $\mathbf{\Sigma}_{BC}$ are positive semi-definite matrices by definition, thus it follows that $\mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T$ and $\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T$ are positive semi-definite matrices. Therefore,

$$\mathrm{tr}(\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T) \geq 0 \tag{5.6}$$

$$\mathrm{tr}(\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T) \geq 0 \tag{5.7}$$

However, condition in Equation 5.1 can not be affirmed to guarantee the monotony of the A-optimality criterion because Equation 5.3 is,

$$\mathrm{tr}(\mathbf{\Sigma}_{AC}) - \mathrm{tr}(\mathbf{\Sigma}_{AB}) = \underbrace{\mathrm{tr}(\mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T)}_{\geq 0} + \underbrace{\mathrm{tr}(\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T)}_{\geq 0} \underbrace{-\mathrm{tr}(\mathbf{\Sigma}_{AB})}_{\leq 0} \tag{5.8}$$

To analyze which variables do not allow monotonicity to be guaranteed, $\mathbf{\Sigma}_{AB}$ and $\mathbf{\Sigma}_{BC}$ are written as,

$$\mathbf{\Sigma}_{AB} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{5.9}$$

$$\mathbf{\Sigma}_{BC} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} \tag{5.10}$$

Noting that $\mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T$ and $\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T$ matrix is as follows, we can analyze which variables do not allow monotonicity to be guaranteed,

$$\begin{aligned} \mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T &= \begin{pmatrix} 1 & 0 & \alpha \\ 0 & 1 & \beta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha & \beta & 1 \end{pmatrix} \\ &= \begin{pmatrix} a_{11} + 2\alpha a_{13} + \alpha^2 a_{33} & \star & \star \\ \star & a_{22} + 2\beta a_{33} + \beta^2 a_{33} & \star \\ \star & \star & a_{33} \end{pmatrix} \end{aligned} \tag{5.11}$$

where,

$$\alpha = y_{AB} - y_{AC} \tag{5.12}$$

$$\beta = x_{AC} - x_{AB} \tag{5.13}$$

$$\begin{aligned} \mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T &= \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} m_{11} & \star & \star \\ \star & m_{22} & \star \\ \star & \star & m_{33} \end{pmatrix} \end{aligned} \tag{5.14}$$

where,

$$m_{11} = b_{11}\cos^2\theta - b_{21}\cos\theta\sin\theta - b_{12}\sin\theta\cos\theta + b_{22}\sin^2\theta \tag{5.15}$$
$$m_{22} = b_{11}\sin^2\theta + b_{21}\sin\theta\cos\theta + b_{12}\cos\theta\sin\theta + b_{22}\cos^2\theta \tag{5.16}$$
$$m_{33} = b_{33} \tag{5.17}$$

Then,

$$\begin{aligned}\mathrm{tr}(\mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T) =& a_{11} + a_{22} + a_{33} + 2\alpha a_{13} + \alpha^2 a_{33} + 2\beta a_{23} + \beta^2 a_{33} \\ =& \mathrm{tr}(\mathbf{\Sigma}_{AB}) + 2\alpha a_{13} + \alpha^2 a_{33} + 2\beta a_{23} + \beta^2 a_{33}\end{aligned} \tag{5.18}$$

$$\begin{aligned}\mathrm{tr}(\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T) =& b_{11}\cos^2\theta + b_{22}\sin^2\theta + b_{11}\sin^2\theta + b_{22}\cos^2\theta + b_{33} \\ =& b_{11} + b_{22} + b_{33}\end{aligned} \tag{5.19}$$

Substituting the above equation in Equation 5.3, we obtain that,

$$\begin{aligned}\mathrm{tr}(\mathbf{\Sigma}_{AC}) - \mathrm{tr}(\mathbf{\Sigma}_{AB}) =& \mathrm{tr}(\mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T) + \mathrm{tr}(\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T) - \mathrm{tr}(\mathbf{\Sigma}_{AB}) \\ =& \mathrm{tr}(\mathbf{\Sigma}_{AB}) + 2\alpha a_{13} + \alpha^2 a_{33} + 2\beta a_{23} + \beta^2 a_{33} \\ &+ b_{11} + b_{22} + b_{33} - \mathrm{tr}(\mathbf{\Sigma}_{AB}) \\ =& 2\alpha a_{13} + \alpha^2 a_{33} + 2\beta a_{23} + \beta^2 a_{33} + b_{11} + b_{22} + b_{33}\end{aligned} \tag{5.20}$$

By definition, α and β are values unbounded therefore we can not guarantee that the following expression is non-negative,

$$\mathrm{tr}(\mathbf{\Sigma}_{AC}) - \mathrm{tr}(\mathbf{\Sigma}_{AB}) = \underbrace{2\alpha a_{13} + 2\beta a_{23}}_{?} + \underbrace{\alpha^2 a_{33}}_{\geq 0} + \underbrace{\beta^2 a_{33}}_{\geq 0} + \underbrace{b_{11} + b_{22} + b_{33}}_{\geq 0} \underbrace{\geq}_{?} 0 \tag{5.21}$$

This implies that A-opt criteria breaks the monotonicity of covariance matrix propagation in a dead-reckoning scenario in 2D case.

3-dimensional space

In the 3 dimesional case, a similar analysis is carried out to study the terms of Equation 5.3. Taking into account the definition of the jacobian of the transformation of compounding $\mathbf{x}_{AC} = \mathbf{x}_{AB} \oplus \mathbf{x}_{BC}$ defined in chapter 3, we can obtain,

$$\det(\mathbf{J}_{1\oplus}) = \det \begin{pmatrix} \mathbf{I} & \mathbf{M} \\ \mathbf{0} & \mathbf{K}_1 \end{pmatrix} = \det(\mathbf{K}_1) \neq 0 \tag{5.22}$$

$$\det(\mathbf{J}_{2\oplus}) = \det \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 \end{pmatrix} = \det(\mathbf{R}) \det(\mathbf{K}_2) \neq 0 \tag{5.23}$$

Then, $\mathbf{J}_{1\oplus}$ and $\mathbf{J}_{2\oplus}$ are non-singular matrices and $\mathbf{\Sigma}_{AB}$ and $\mathbf{\Sigma}_{BC}$ are positive semi-definite matrices by definition, thus it follows that $\mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T$ and $\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T$ are positive semi-definite matrices. Therefore,

$$\mathrm{tr}(\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T) \geq 0 \tag{5.24}$$

$$\mathrm{tr}(\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T) \geq 0 \tag{5.25}$$

However, as in the previous case, condition in Equation 5.1 can not be affirmed to guarantee the monotony of the A-optimality criterion because Equation 5.3 is,

$$\mathrm{tr}(\mathbf{\Sigma}_{AC}) - \mathrm{tr}(\mathbf{\Sigma}_{AB}) = \underbrace{\mathrm{tr}(\mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T)}_{\geq 0} + \underbrace{\mathrm{tr}(\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T)}_{\geq 0} \underbrace{-\mathrm{tr}(\mathbf{\Sigma}_{AB})}_{\leq 0} \tag{5.26}$$

To analyze which variables do not allow monotonicity to be guaranteed, $\mathbf{\Sigma}_{AB}$ and $\mathbf{\Sigma}_{BC}$ are written as,

$$\mathbf{\Sigma}_{AB} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \tag{5.27}$$

$$\mathbf{\Sigma}_{BC} = \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{pmatrix} \tag{5.28}$$

and using Equation 4.6 and Equation 4.7 result in,

$$\begin{aligned}\mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^{T} &= \begin{pmatrix}\mathbf{I} & \mathbf{M}\\ \mathbf{0} & \mathbf{K}_1\end{pmatrix}\begin{pmatrix}\mathbf{A}_{11} & \mathbf{A}_{12}\\ \mathbf{A}_{21} & \mathbf{A}_{22}\end{pmatrix}\begin{pmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{M}^T & \mathbf{K}_1^T\end{pmatrix} = \\ &= \begin{pmatrix}\mathbf{A}_{11}+\mathbf{M}\mathbf{A}_{21}+\mathbf{A}_{12}\mathbf{M}^T+\mathbf{M}\mathbf{A}_{22}\mathbf{M}^T & \mathbf{A}_{12}\mathbf{K}_1^T+\mathbf{M}\mathbf{A}_{22}\mathbf{K}_1^T\\ \mathbf{K}_1\mathbf{A}_{21}+\mathbf{K}_1\mathbf{A}_{22}\mathbf{M}^T & \mathbf{K}_1\mathbf{A}_{22}\mathbf{K}_1^T\end{pmatrix}\end{aligned} \tag{5.29}$$

$$\begin{aligned}\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^{T} &= \begin{pmatrix}\mathbf{R} & \mathbf{0}\\ \mathbf{0} & \mathbf{K}_2\end{pmatrix}\begin{pmatrix}\mathbf{B}_{11} & \mathbf{B}_{12}\\ \mathbf{B}_{21} & \mathbf{B}_{22}\end{pmatrix}\begin{pmatrix}\mathbf{R}^T & \mathbf{0}\\ \mathbf{0} & \mathbf{K}_2^T\end{pmatrix} = \\ &= \begin{pmatrix}\mathbf{R}\mathbf{B}_{11}^T & \mathbf{R}\mathbf{B}_{12}\mathbf{K}_2^T\\ \mathbf{K}_2\mathbf{B}_{21}^T\mathbf{R}^T & \mathbf{K}_2\mathbf{B}_{22}\mathbf{K}_2^T\end{pmatrix}\end{aligned} \tag{5.30}$$

Then,

$$\begin{aligned}\mathrm{tr}(\mathbf{\Sigma}_{AC}) - \mathrm{tr}(\mathbf{\Sigma}_{AB}) = {} & \underbrace{\mathrm{tr}(\mathbf{M}\mathbf{A}_{21})}_{?} + \underbrace{2\mathrm{tr}(\mathbf{A}_{12}\mathbf{M}^T)}_{?} + \underbrace{\mathrm{tr}(\mathbf{M}\mathbf{A}_{22}\mathbf{M}^T)}_{?} + \\ & + \underbrace{\mathrm{tr}(\mathbf{K}_1\mathbf{A}_{22}\mathbf{K}_1^T)}_{\geq 0} + \underbrace{\mathrm{tr}(\mathbf{R}\mathbf{B}_{11}\mathbf{R}^T)}_{\geq 0} + \underbrace{\mathrm{tr}(\mathbf{K}_2\mathbf{B}_{22}\mathbf{K}_2^T)}_{\geq 0} - \underbrace{\mathrm{tr}(\mathbf{A}_{22})}_{\geq 0}\end{aligned} \tag{5.31}$$

because $\mathbf{A}_{22}$, $\mathbf{B}_{11}$ and $\mathbf{B}_{22}$ are positive semi-definite matrices, and $\mathbf{R}$, $\mathbf{K}_1$ and $\mathbf{K}_2$ are non-singular matrices. However, the definition of $\mathbf{M}$ involves unbounded values (see chapter 3) for the cartesian coordinates and therefore, expressions involving the values of $\mathbf{M}$ could either be negative or non-negative.

Therefore the monotonicity of *A-opt* in 3D case, cannot be guarantied for the absolute representation of uncertainty, independently of the chosen representation for the orientation terms of the location vectors.

5.2.2 D-optimality

Monotonicity for the *D-opt* criterium requires that the determinant of the covariance matrix of the compounding pose would be larger than or equal to the determinant of the covariance matrix of the initial pose, that is,

$$\det(\mathbf{\Sigma}_{AC}) - \det(\mathbf{\Sigma}_{AB}) \geq 0 \tag{5.32}$$

Computing the determinant in both sides of Equation 4.9,

$$\mathbf{\Sigma}_{AC} \simeq \mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^{T} + \mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^{T} \tag{5.33}$$

leads to,

$$\det(\mathbf{\Sigma}_{AC}) = \det(\mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^{T} + \mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^{T}) \tag{5.34}$$

As the determinant is not a linear operator, the determinant of each term can be analyse separately. However, a property of positive semidefinite matrices can be applied. The Minkowski inequality for matrices (Seber, 2008) states that if A and B are symmetric and positive semi-definite matrices of dimension n, then,

$$\det(A + B)^{1/n} \geq \det(A)^{1/n} + \det(B)^{1/n} \tag{5.35}$$

where n is de dimension of matrix.

Therefore, given that the Jacobians of the composition are non-singular matrices, and both $\mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^{T}$ and $\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^{T}$ are symmetric and positive semi-definite matrices, from the Minkowski's inequality for matrices results in,

$$\det(\mathbf{\Sigma}_{AC})^{\frac{1}{n}} \geq \det(\mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^{T})^{\frac{1}{n}} + \det(\mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^{T})^{\frac{1}{n}} \tag{5.36}$$

Thus, using basic determinant properties,

$$\det(\mathbf{A}^{T}) = \det(\mathbf{A}) \tag{5.37}$$

$$\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B}) \tag{5.38}$$

we have,

$$\begin{aligned}\det(\mathbf{\Sigma}_{AC})^{\frac{1}{n}} \geq \det(\mathbf{J}_{1\oplus})^{\frac{1}{n}} \det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}} \det(\mathbf{J}_{1\oplus}^T)^{\frac{1}{n}} + \\ + \det(\mathbf{J}_{2\oplus})^{\frac{1}{n}} \det(\mathbf{\Sigma}_{BC})^{\frac{1}{n}} \det(\mathbf{J}_{2\oplus}^T)^{\frac{1}{n}} = \\ = \det(\mathbf{J}_{1\oplus})^{\frac{1}{n}} \det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}} \det(\mathbf{J}_{1\oplus})^{\frac{1}{n}} + \\ + det(\mathbf{J}_{2\oplus})^{\frac{1}{n}} \det(\mathbf{\Sigma}_{BC})^{\frac{1}{n}} \det(\mathbf{J}_{2\oplus})^{\frac{1}{n}} = \\ = \det(\mathbf{J}_{1\oplus})^{\frac{2}{n}} \det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}} + \det(\mathbf{J}_{2\oplus})^{\frac{2}{n}} \det(\mathbf{\Sigma}_{BC})^{\frac{1}{n}} \end{aligned} \tag{5.39}$$

2-dimensional space

In two dimensional space, the determinant of the jacobian transformation is,

$$\det(\mathbf{J}_{1\oplus}) = 1 \tag{5.40}$$

$$\det(\mathbf{J}_{2\oplus}) = 1 \tag{5.41}$$

Then, subtituting Equation 5.40 and Equation 5.41 in Equation 5.39, we have,

$$\det(\mathbf{\Sigma}_{AC})^{\frac{1}{n}} \geq \det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}} + \det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}} \tag{5.42}$$

As the determinant of covariance matrix $\mathbf{\Sigma}_{AB}$ and $\mathbf{\Sigma}_{BC}$ are non-negative, we have that,

$$\det(\mathbf{\Sigma}_{AC})^{\frac{1}{n}} \geq \det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}} + \det(\mathbf{\Sigma}_{BC})^{\frac{1}{n}} \geq \det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}} \geq 0 \tag{5.43}$$

Taking into account that $\det(\mathbf{\Sigma}_{AC}) \geq det(\mathbf{\Sigma}_{AB})$ is equivalent to the condition $\det(\mathbf{\Sigma}_{AC})^{\frac{1}{n}} \geq \det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}}$,

$$\det(\mathbf{\Sigma}_{AC}) - \det(\mathbf{\Sigma}_{AB}) \geq 0 \tag{5.44}$$

Therefore, the above inequality guarantee the monotonicity of D-opt criteria in 2D space.

3-dimensional space

In the 3-dimensional space, $\det(\mathbf{J}_{1\oplus})$ and $\det(\mathbf{J}_{2\oplus})$ depends on its values so the condition $\det(\mathbf{\Sigma}_{AC})^{\frac{1}{n}} \geq \det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}}$ cannot be guarantee.

Nevertheless, a sufficient condition for the monotonicity of the *D-opt* criterium is given by,

$$\begin{aligned} \det(\mathbf{\Sigma}_{AC})^{\frac{1}{n}} &\geq \det(\mathbf{J}_{1\oplus})^{\frac{2}{n}} \det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}} + \\ &+ \det(\mathbf{J}_{2\oplus})^{\frac{2}{n}} \det(\mathbf{\Sigma}_{BC})^{\frac{1}{n}} \\ &\geq \det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}} \end{aligned} \tag{5.45}$$

that results in,

$$\det(\mathbf{J}_{1\oplus})^{\frac{2}{n}} \geq 1 - \frac{\det(\mathbf{J}_{2\oplus})^{\frac{2}{n}} \det(\mathbf{\Sigma}_{BC})^{\frac{1}{n}}}{\det(\mathbf{\Sigma}_{AB})^{\frac{1}{n}}} \tag{5.46}$$

In the case of unit quaternions are used for representing orientation, *D-opt* preserves monotonicity because Equation 5.46 is verified for all cases due to $\det(\mathbf{J}_{1\oplus}) = \det(\mathbf{J}_{2\oplus}) = 1$. However, condition Equation 5.46 is not satisfied in the absolute representation of uncertainty when Roll-Pitch-Yaw or Euler angles are used.

5.2.3 E-optimality

Finally, monotonicity for the *E-opt* criterium requires that the largest eigenvalue of the covariance matrix of the compounding pose would be larger than or equal to the largest eigenvalue of the covariance matrix of the initial pose, that is,

$$max_{\lambda_i}(\mathbf{\Sigma}_{AC}) - max_{\lambda_j}(\mathbf{\Sigma}_{AB}) \geq 0 \tag{5.47}$$

A property of positive semidefinite matrices in relation to their eigenvalues is the Weyl's inequality (Bernstein, 2009) that states that if $\mathbf{A}$, $\mathbf{B}$ are

symmetric matrices and $\mathbf{A} \leq \mathbf{B}$, i.e. $\mathbf{B}-\mathbf{A}$ is a positive semi-definite matrix, then,

$$\lambda_i(\mathbf{A}) \leq \lambda_i(\mathbf{B}) \tag{5.48}$$

with λ_i the i-th largest eigenvalue of $\mathbf{A}$ and $\mathbf{B}$, respectively.

To apply the previous inequality, we have to check if $\mathbf{\Sigma}_{AC} - \mathbf{\Sigma}_{AB} \geq 0$. For that, we notice that a property of the semi-positive definite matrices is that its principal minor are non-negative. Then, $\mathbf{\Sigma}_{AC} - \mathbf{\Sigma}_{AB}$ is semi-definite positive matrix if its principal minor are non-negative.

2-dimensional space

Subsituting Equation 4.9,

$$\mathbf{\Sigma}_{AC} \simeq \mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T + \mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T \tag{5.49}$$

in the difference between $\mathbf{\Sigma}_{AC}$ and $\mathbf{\Sigma}_{AB}$, we have,

$$\mathbf{\Sigma}_{AC} - \mathbf{\Sigma}_{AB} = \mathbf{J}_{1\oplus}\mathbf{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T + \mathbf{J}_{2\oplus}\mathbf{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T - \mathbf{\Sigma}_{AB} \tag{5.50}$$

Using Equation 5.9, Equation 5.10, Equation 5.11 and Equation 5.14, the first principal minor of $\mathbf{\Sigma}_{AC} - \mathbf{\Sigma}_{AB}$ writes as follows,

$$\underbrace{2\alpha a_{13}}_{?} + \underbrace{\alpha^2 a_{33}}_{\geq 0} + \underbrace{b_{11}\cos^2\theta}_{\geq 0} - \underbrace{b_{21}\cos\theta\sin\theta}_{?} - \underbrace{b_{12}\sin\theta\cos\theta}_{?} + \underbrace{b_{22}\sin^2\theta}_{\geq 0} \tag{5.51}$$

It can not be said that the above expression is always non-negative since α, a_{13}, b_{21} and b_{12} can be values positive or negative as it happened in the A-opt case. Then it can not claim that the E-optimality criteria is monotone increasing.

3-dimensional space

Subsituting Equation 4.9 in the difference between $\mathbf{\Sigma}_{AC}$ and $\mathbf{\Sigma}_{AB}$, we have,

$$\boldsymbol{\Sigma}_{AC} - \boldsymbol{\Sigma}_{AB} = \mathbf{J}_{1\oplus}\boldsymbol{\Sigma}_{AB}\mathbf{J}_{1\oplus}^T + \mathbf{J}_{2\oplus}\boldsymbol{\Sigma}_{BC}\mathbf{J}_{2\oplus}^T - \boldsymbol{\Sigma}_{AB} \tag{5.52}$$

Using Equation 5.27, Equation 5.28, Equation 5.29 and Equation 5.30, the first principal minor of $\boldsymbol{\Sigma}_{AC} - \boldsymbol{\Sigma}_{AB}$ writes as follows,

$$|\mathbf{M}\mathbf{A}_{21} + \mathbf{A}_{12}\mathbf{M}^T + \mathbf{M}\mathbf{A}_{22}\mathbf{M}^T + \mathbf{R}_1\mathbf{B}_{11}\mathbf{R}_1^T| \underbrace{\geq}_{?} 0 \tag{5.53}$$

The above expression is negative or non-negative depending on the unbounded values of $\mathbf{M}$ (see chapter 3) as in the *A-opt* case. Therefore, using an absolute representations *E-opt* may not preserve monotonicity independently of the chosen representation of the orientation terms of the location vectors.

5.3 Monnotonicity in Differential Representations of Uncertainty

Comparing the uncertainty associated to the differencial location vectors $\mathbf{d}_B$, related to the previous robot's pose in Equation 4.12,

$$\mathbf{x}_{AB} = \bar{\mathbf{x}}_{AB} \oplus \mathbf{d}_B \tag{5.54}$$

and $\mathbf{d}'_C$, related to the compounding pose of the robot in Equation 4.21,

$$\mathbf{x}_{AC} = \bar{\mathbf{x}}_{AC} \oplus \mathbf{d}'_C \tag{5.55}$$

requires them to be expressed in a common reference frame. Let A (equivalently both B and C could have been used) be such a reference, then,

$$\begin{aligned} \mathbf{d}_{B|A} &= \mathbf{J}_{AB}\mathbf{d}_B \\ \mathbf{d}'_{C|A} &= \mathbf{J}_{AC}(\mathbf{J}_{CB}\mathbf{d}_B \oplus \mathbf{d}_C) \\ &= \mathbf{J}_{AB}\mathbf{d}_B \oplus \mathbf{J}_{AC}\mathbf{d}_C \end{aligned} \tag{5.56}$$

Assuming small errors,

$$\mathbf{d}'_{C|A} \simeq \mathbf{J}_{AB}\mathbf{d}_B + \mathbf{J}_{AC}\mathbf{d}_C \tag{5.57}$$

and, assuming uncorrelated noise sequences,

$$\mathbf{\Sigma}_{B|A} \simeq \mathbf{J}_{AB}\mathbf{\Sigma}_B\mathbf{J}_{AB}^T \tag{5.58}$$

$$\mathbf{\Sigma}'_{C|A} \simeq \mathbf{J}_{AB}\mathbf{\Sigma}_B\mathbf{J}_{AB}^T + \mathbf{J}_{AC}\mathbf{\Sigma}_C\mathbf{J}_{AC}^T \tag{5.59}$$

In this case, we analyse the monotonicity of *A-opt*, *D-opt* and *E-opt* criteria taking into account the following properties from (Seber, 2008):

If $\mathbf{A}$ and $\mathbf{B}$ are $n \times n$ symmetric matrices then,

- If $\mathbf{A} \geq \mathbf{B}$, then $\text{tr}(\mathbf{A}) \geq \text{tr}(\mathbf{B})$.
- If $\mathbf{A} \geq \mathbf{B} \geq 0$, then $\det(\mathbf{A}) \geq \det(\mathbf{B})$.
- If $\mathbf{A} \geq \mathbf{B}$, then for all *i = 1,..., n*, $\lambda_i(\mathbf{A}) \geq \lambda_i(\mathbf{B})$ with λ_i the i-th largest eigenvalue of $\mathbf{A}$ and $\mathbf{B}$, respectively.

Thus, we should evaluate if,

$$\mathbf{\Sigma}'_{C|A} - \mathbf{\Sigma}_{B|A} \geq 0 \tag{5.60}$$

Subtituting Equation 5.58 and Equation 5.59 in Equation 5.60, we have,

$$\mathbf{\Sigma}'_{C|A} - \mathbf{\Sigma}_{B|A} \simeq \mathbf{J}_{AC}\mathbf{\Sigma}_C\mathbf{J}_{AC}^T \tag{5.61}$$

In 2-dimensional space, the jacobian of the relative transformation $\mathbf{J}_{AC}$ is defined by,

$$\mathbf{J}_{AC} = \left(\begin{array}{c|c} \mathbf{R}_{AC} & \mathbf{R}_{AC}\begin{pmatrix} -y_{AC} \\ x_{AC} \end{pmatrix} \\ \hline 0 & 1 \end{array}\right) \tag{5.62}$$

where $\mathbf{R}_{AC}$ is an orthogonal rotation matrix. $\mathbf{J}_{AC}$ is a non-singular matrix because $\det(\mathbf{J}_{AC}) = \det(\mathbf{R}_{AC}) = 1$.

In 3-dimensional space, the jacobian of the relative transformation $\mathbf{J}_{AC}$ is defined by,

$$\mathbf{J}_{AC} = \begin{pmatrix} \mathbf{R}_{AC} & \mathbf{R}_{AC}\mathbf{D}_{AC} \\ \mathbf{0} & \mathbf{R}_{AC} \end{pmatrix} \tag{5.63}$$

where $\mathbf{R}_{AC}$ is an orthogonal rotation matrix and $\mathbf{D}_{AC}$ is the 3×3 skew symmetric matrix related to the translational part of $\bar{\mathbf{x}}_{AC}$,

$$\mathbf{D}_{AC} = [\bar{\boldsymbol{x}}_{AC}]_{\times} = \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix} \tag{5.64}$$

Then, $\mathbf{J}_{AC}$ is a non-singular matrix because $\det(\mathbf{J}_{AC}) = \det(\mathbf{R}_{AC}) \det(\mathbf{R}_{AC}) = 1$.

As $\mathbf{\Sigma}_C$ is a positive semi-definite matrix and $\mathbf{J}_{AC}$ is a non-singular matrix then $\mathbf{J}_{AC}\mathbf{\Sigma}_C\mathbf{J}_{AC}^T$ is a positive semi-definite matrix. Then we have,

$$\mathbf{\Sigma}'_{C|A} - \mathbf{\Sigma}_{B|A} \simeq \mathbf{J}_{AC}\mathbf{\Sigma}_C\mathbf{J}_{AC}^T \geq 0 \tag{5.65}$$

and,

- $\mathbf{\Sigma}'_{C|A} \geq \mathbf{\Sigma}_{B|A} \geq 0$, then $\mathrm{tr}(\mathbf{\Sigma}'_{C|A}) \geq \mathrm{tr}(\mathbf{\Sigma}_{B|A})$.
- $\mathbf{\Sigma}'_{C|A} \geq \mathbf{\Sigma}_{B|A} \geq 0$, then $\det(\mathbf{\Sigma}'_{C|A}) \geq \det(\mathbf{\Sigma}_{B|A})$.
- $\mathbf{\Sigma}'_{C|A} \geq \mathbf{\Sigma}_{B|A} \geq 0$, then for all *i = 1, ..., n*, $\lambda_i(\mathbf{\Sigma}'_{C|A}) \geq \lambda_i(\mathbf{\Sigma}_{B|A})$ with λ_i the i-th largest eigenvalue of $\mathbf{\Sigma}'_{C|A}$ and $\mathbf{\Sigma}_{B|A}$, respectively.

Therefore, the three optimality criteria *A-opt*, *D-opt* and *E-opt* would satisfy monotonicity independently of the representation of the orientation of the location vectors.

5.4 Monotonicity of the Shannon's Entropy

In this section we study, from the perspective of Information Theory, whether the monotonicity of Shannon's entropy holds when the robot is also performing an exploration phase within an active SLAM system.

A common approach to quantify uncertainty in the estimation of random variables are the Information Theory (IT) framework (Cover and Thomas, 2012). The IT framework relies on the definition of entropy to quantify the uncertainty of the probability distribution associated to the estimated random variable.

It is standard to assume that the robot's pose is represented using a multivariate Gaussian distribution. In this case the (differential) Shannon entropy of the distribution is given by the following expression Stachniss et al. (2005),

$$\mathbb{H}[\mathcal{P}(\mathbf{x})] = \frac{n}{2}(1 + \log(2\pi)) + \frac{1}{2}\log(\det(\mathbf{\Sigma})) \tag{5.66}$$

where n is the dimension of the robot's pose and Σ is its $n \times n$ covariance matrix.

Monotonicity for the entropy requires that the definition of Shannon entropy of the compounding pose would be larger than or equal to the expression of Shannon entropy of the initial pose, that is,

$$\mathbb{H}[\mathcal{P}(\mathbf{x}_{AC})] \geq \mathbb{H}[\mathcal{P}(\mathbf{x}_{AB})] \tag{5.67}$$

Using Equation 5.66, the Shannon entropy associated to the robot's poses $\mathbf{x}_{AC}$ and $\mathbf{x}_{AB}$ are defined as follows,

$$\mathbb{H}[\mathcal{P}(\mathbf{x}_{AC})] = \frac{n}{2}(1 + \log(2\pi)) + \frac{1}{2}\log(\det(\mathbf{\Sigma}_{AC})) \tag{5.68}$$

$$\mathbb{H}[\mathcal{P}(\mathbf{x}_{AB})] = \frac{n}{2}(1 + \log(2\pi)) + \frac{1}{2}\log(\det(\mathbf{\Sigma}_{AB})) \tag{5.69}$$

To analyze the monotonicity we will study the cases in which the following expression is maintained,

$$\mathbb{H}[\mathcal{P}(\mathbf{x}_{AC})] \geq \mathbb{H}[\mathcal{P}(\mathbf{x}_{AB})] \Leftrightarrow \mathbb{H}[\mathcal{P}(\mathbf{x}_{AC})] - \mathbb{H}[\mathcal{P}(\mathbf{x}_{AB})] \geq 0 \tag{5.70}$$

Substituting the expressions in Equation 5.68 and Equation 5.69 in Equation 5.70, leads to,

$$\begin{aligned}
\mathbb{H}[\mathcal{P}(\mathbf{x}_{AC})] - \mathbb{H}[\mathcal{P}(\mathbf{x}_{AB})] =& \frac{n}{2}(1 + \log(2\pi)) + \frac{1}{2}\log(\det(\mathbf{\Sigma}_{AC})) \\
&- \frac{n}{2}(1 + \log(2\pi)) - \frac{1}{2}\log(\det(\mathbf{\Sigma}_{AB})) \\
=& \frac{1}{2}\log(\det(\mathbf{\Sigma}_{AC})) - \frac{1}{2}\log(\det(\mathbf{\Sigma}_{AB})) \\
=& \frac{1}{2}\left[\log(\det(\mathbf{\Sigma}_{AC})) - \log(\det(\mathbf{\Sigma}_{AB}))\right] \\
=& \frac{1}{2}\log\left(\frac{\det(\mathbf{\Sigma}_{AC})}{\det(\mathbf{\Sigma}_{AB})}\right)
\end{aligned} \tag{5.71}$$

Case 1:

If $\det(\mathbf{\Sigma}_{AC}) > \det(\mathbf{\Sigma}_{AB}) \Rightarrow \mathbb{H}[\mathcal{P}(\mathbf{x}_{AC})] > \mathbb{H}[\mathcal{P}(\mathbf{x}_{AB})]$

Proof:

By hypothesis,

$$\det(\mathbf{\Sigma}_{AC}) > \det(\mathbf{\Sigma}_{AB}) \Rightarrow \frac{\det(\mathbf{\Sigma}_{AC})}{\det(\mathbf{\Sigma}_{AB})} > 1 \tag{5.72}$$

Therefore, we have,

$$\mathbb{H}[\mathcal{P}(\mathbf{x}_{AC})] - \mathbb{H}[\mathcal{P}(\mathbf{x}_{AB})] = \underbrace{\frac{1}{2}\log\underbrace{\left(\frac{\det(\mathbf{\Sigma}_{AC})}{\det(\mathbf{\Sigma}_{AB})}\right)}_{>1}}_{>0} > 0 \tag{5.73}$$

Case 2:

If $\det(\mathbf{\Sigma}_{AC}) < \det(\mathbf{\Sigma}_{AB}) \Rightarrow \mathbb{H}[\mathcal{P}(\mathbf{x}_{AC})] < \mathbb{H}[\mathcal{P}(\mathbf{x}_{AB})]$

Proof:

By hypothesis,

$$\det(\mathbf{\Sigma}_{AC}) < \det(\mathbf{\Sigma}_{AB}) \Rightarrow \frac{\det(\mathbf{\Sigma}_{AC})}{\det(\mathbf{\Sigma}_{AB})} < 1 \tag{5.74}$$

Therefore, we have,

$$\mathbb{H}[\mathcal{P}(\mathbf{x}_{AC})] - \mathbb{H}[\mathcal{P}(\mathbf{x}_{AB})] = \underbrace{\frac{1}{2}\log\underbrace{\left(\frac{\det(\mathbf{\Sigma}_{AC})}{\det(\mathbf{\Sigma}_{AB})}\right)}_{<1}}_{<0} < 0 \tag{5.75}$$

Case 3:

If $\det(\mathbf{\Sigma}_{AC}) = \det(\mathbf{\Sigma}_{AB}) \Rightarrow \mathbb{H}[\mathcal{P}(\mathbf{x}_{AC})] = \mathbb{H}[\mathcal{P}(\mathbf{x}_{AB})]$

Proof:

By hypothesis,

$$\det(\mathbf{\Sigma}_{AC}) = \det(\mathbf{\Sigma}_{AB}) \Rightarrow \frac{\det(\mathbf{\Sigma}_{AC})}{\det(\mathbf{\Sigma}_{AB})} = 1 \tag{5.76}$$

Therefore, we have,

$$\mathbb{H}[\mathcal{P}(\mathbf{x}_{AC})] - \mathbb{H}[\mathcal{P}(\mathbf{x}_{AB})] = \underbrace{\frac{1}{2}\log\underbrace{\left(\frac{\det(\mathbf{\Sigma}_{AC})}{\det(\mathbf{\Sigma}_{AB})}\right)}_{=1}}_{=0} = 0 \tag{5.77}$$

Using the analysis performed for *D-opt* criteria, we have that,

- In 2D, the *D-opt* criteria maintains the monotonicity, then this is the case 1 and 3. Therefore the entropy maintains the monotonicity.
- In 3D when the pose's robot is represented using unit quaternions the monotonicity of the *D-opt* criteria is maintained. This is the case 1 and

3 therefore the entropy maintains the monotonicity with this representation.

- In 3D when the pose's robot is represented using Roll-Pitch-Yaw and Euler angles the monotonicity of the *D-opt* criteria is not maintained. This is the case 2 then the entropy does not maintain the monotonicity with this representation.

- When the pose's robot is represented using differential representation in 2D and 3D the monotonicity of the *D-opt* criteria is maintained. This is the case 1 and 3 therefore the entropy maintains the monotonicity with this representation.

Therefore, we can conclude that in 2D the entropy maintains the monotonicity. In 3D, differential representations maintain the monotonicity of the entropy and in absolute representation the only representation that maintains monotonicity is unit quaternions.

5.5 Simulations

In this section, we present several experiments to empirically confirm and illustrate the analytical results about monotonicity using the uncertainty representations analysed.

5.5.1 2-dimensional experiment

In 2D, we simulate an open loop trajectory followed a differential driven robot. The displacement of the mobile robot $\mathbf{x}_{R_{k-1}R_k} = (x, y, \theta)^T$ is represented in polar coordinates as,

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} \rho\cos\theta \\ \rho\sin\theta \\ \theta \end{pmatrix} \tag{5.78}$$

with (ρ, θ) a vector being normally distributed with mean and covariance matrix given by,

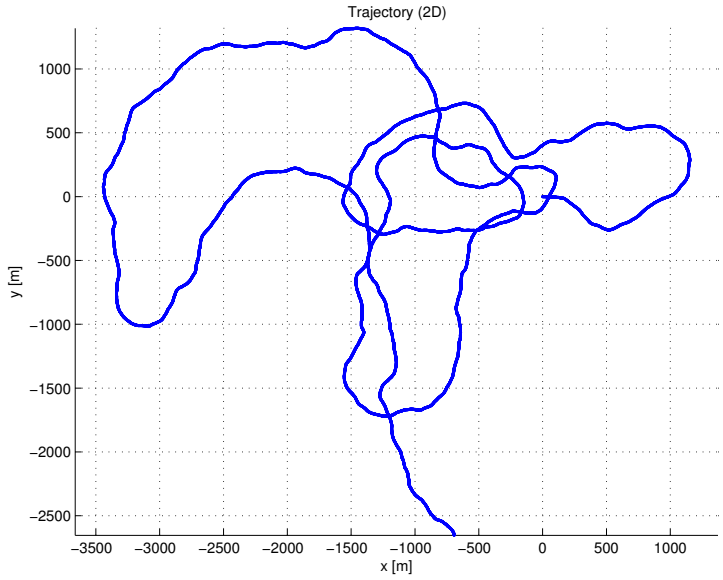

Figure 5.1: Pictorial representation of the trajectory used in 2-dimensional experiment.

$$\begin{pmatrix}\rho\\ \theta\end{pmatrix} \sim \mathcal{N}\left\{\begin{pmatrix}\bar{\rho}\\ \bar{\theta}\end{pmatrix}; \begin{pmatrix}\sigma_\rho^2 & 0\\ 0 & \sigma_\theta^2\end{pmatrix}\right\} \tag{5.79}$$

The propagation of the associated error of a robot's pose transformation in a dead-reckoning scenario is based on the approximation of the nonlinear transformation presented in the compounding of the poses by a Taylor series and then selecting the first degree terms. Therefore, the covariance matrix associated with the transformation $\mathbf{x}_{R_{k-1}R_k}$ is given by,

$$\mathbf{C}_{R_{k-1}R_k} \simeq \mathbf{J}_{R_{k-1}R_k} \begin{pmatrix}\sigma_\rho^2 & 0\\ 0 & \sigma_\theta^2\end{pmatrix} \mathbf{J}_{R_{k-1}R_k}^T \tag{5.80}$$

with,

$$\mathbf{J}_{R_{k-1}R_k} = \begin{pmatrix}\cos\theta & -\rho\sin\theta\\ \sin\theta & \rho\cos\theta\\ 0 & 1\end{pmatrix} \tag{5.81}$$

Figure 5.1 shows the path followed by the mobile robot for the parameters $\bar{\rho} = 1\text{m}$, $\bar{\theta} = 0.001\text{rad}$, $\sigma_\rho = 0.1\text{m}$ and $\sigma_\theta = 0.01\text{rad}$.

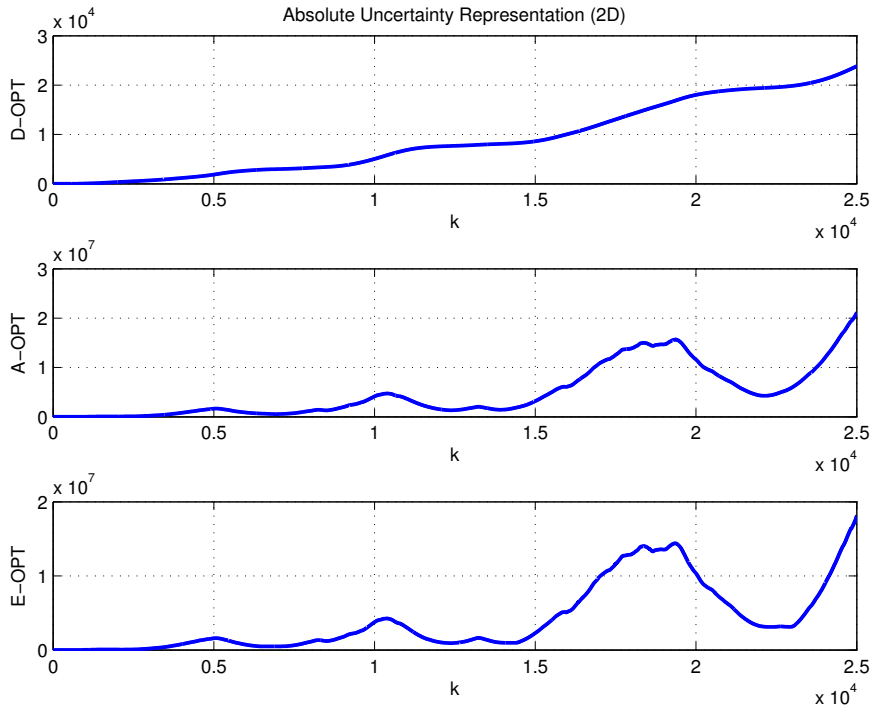

Figure 5.2: Evolution of the optimality criteria *D-opt*, *A-opt* and *E-opt* for the trajectory for 2-dimensional experiment using absolute representation of uncertainty.

For each step of the trajectory, we compute the uncertainty propagation using absolute and differential representations as given in chapter 5. Then, we analyze the evolution of the *A-opt*, *D-opt* and *E-opt* criteria shown in Figure 5.2 for absolute representation of uncertainty and Figure 5.3 for differential representation of uncertainty.

As shown in Figure 5.2 the monotonicity of *A-opt* and *E-opt* does not hold in absolute representation. However, breaking of monotonicity is not observed for *D-opt*. Figure 5.3 shows that the monotonicity of the optimality criteria holds using a differential representation for the location uncertainty. This experiment confirms the theoretical demonstrations shown in the previous sections.

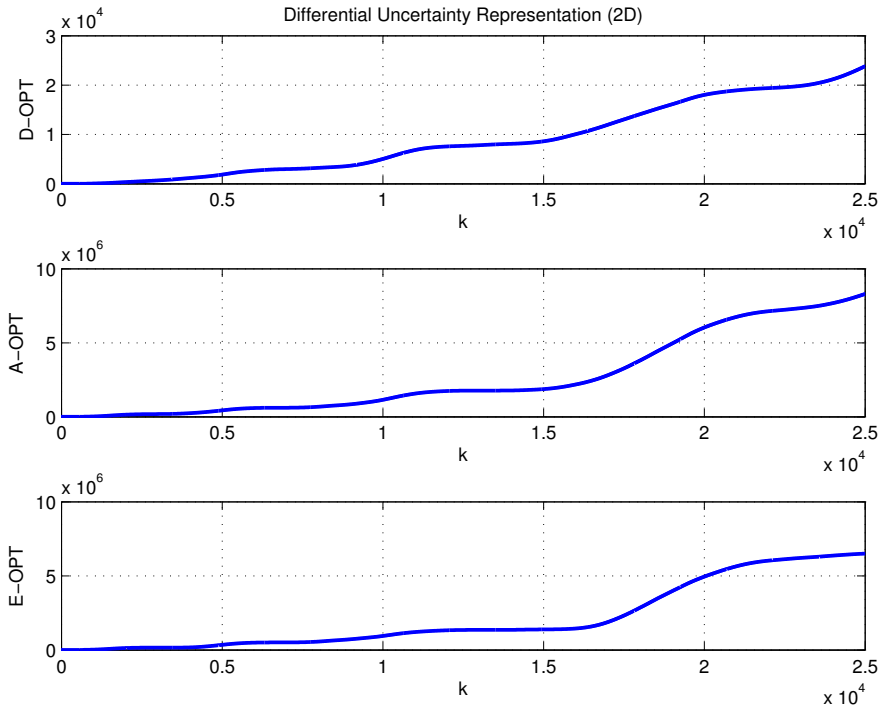

Figure 5.3: Evolution of the optimality criteria *D-opt*, *A-opt* and *E-opt* for the trajectory for 2-dimensional experiment using differential representation of uncertainty.

5.5.2 3-dimensional experiment

In the following, we show an experiment to test the monotonicity of optimality criteria in a dead-reckoning scenario in 3D. As in the previous case, we estimate the propagation of the uncertainty associated to the displacement of the mobile robot and then, we analyze the evolution of optimality criteria using absolute and differential representation of uncertainty.

In 3-dimensional case, the displacement of the mobile robot is represented by $\mathbf{x}_{R_{k-1}R_k} = (x, y, z, \phi, \theta, \psi)^T$ where ϕ, θ and ψ are the angles of rotation on the axis Z, Y, X, respectively. This displacement is represented in spherical coordinates as,

$$\begin{pmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{pmatrix} = \begin{pmatrix} \rho\cos\phi\cos\theta \\ \rho\sin\phi\cos\theta \\ \rho\sin\theta \\ \phi \\ \theta \\ \psi \end{pmatrix} \tag{5.82}$$

with $(\rho, \phi, \theta, \psi)$ a vector being normally distributed with mean and covariance matrix given by,

$$\begin{pmatrix} \rho \\ \phi \\ \theta \\ \psi \end{pmatrix} \sim \mathcal{N}\left\{ \begin{pmatrix} \bar{\rho} \\ \bar{\phi} \\ \bar{\theta} \\ \bar{\psi} \end{pmatrix} ; \begin{pmatrix} \sigma_\rho^2 & 0 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 & 0 \\ 0 & 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & 0 & \sigma_\psi^2 \end{pmatrix} \right\} \tag{5.83}$$

Again, the propagation of the associated error of a robot's pose transformation in a dead-reckoning scenario is based on the approximation of the nonlinear transformation presented in the compounding of the poses by a Taylor series and then selecting the first degree terms. Therefore, the covariance matrix associated with the transformation $\mathbf{x}_{R_{k-1}R_k}$ is given by,

$$\mathbf{C}_{R_{k-1}R_k} \simeq \mathbf{J}_{R_{k-1}R_k} \begin{pmatrix} \sigma_\rho^2 & 0 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 & 0 \\ 0 & 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & 0 & \sigma_\psi^2 \end{pmatrix} \mathbf{J}_{R_{k-1}R_k}^T \tag{5.84}$$

with,

$$\mathbf{J}_{R_{k-1}R_k} = \begin{pmatrix} \cos\phi\cos\theta & -\rho\sin\phi\cos\theta & -\rho\cos\phi\cos\theta & 0 \\ \sin\phi\cos\theta & \rho\cos\phi\cos\theta & -\rho\sin\phi\sin\theta & 0 \\ \sin\theta & 0 & \rho\cos\theta & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.85}$$

Figure 5.4 shows the XY-projection of the path followed by the mobile robot for the parameters $\bar{\rho} = 1\text{m}$, $\bar{\phi} = 0.01\text{rad}$, $\bar{\theta} = 0.001\text{rad}$, $\bar{\phi} = 0.001\text{rad}$, $\sigma_\rho = 0.1\text{m}$, $\sigma_\phi = 0.01\text{rad}$, $\sigma_\theta = 0.005\text{rad}$ and $\sigma_\psi = 0.005\text{rad}$.

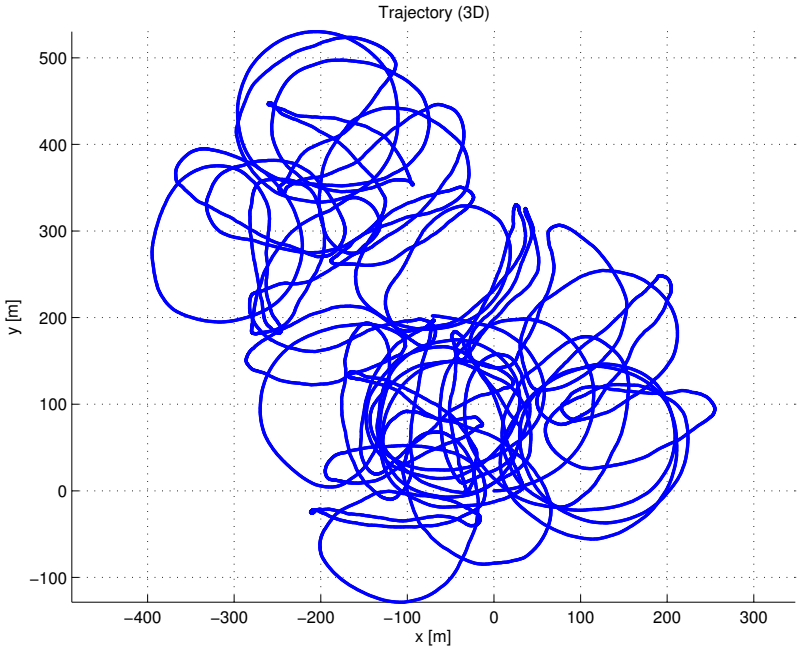

Figure 5.4: Pictorial representation of the trajectory used in 3-dimensional experiment.

In order to analyze the evolution of the *A-opt*, *D-opt* and *E-opt* criteria, uncertainty propagation using absolute and differential representations as given in chapter 5 is computed. In absolute representations, the orientation of uncertainty location vector is represented depending on the representation of the rotation angles. In this simulation, we use Roll-Pitch-Yaw angles and unit quaternions. The evolution of the *A-opt*, *D-opt* and *E-opt* criteria using absolute representation of uncertainty for Roll-Pitch-Yaw angles is shown in Figure 5.5 and for unit quaternions in Figure 5.6. Figure 5.3 shows the evolution of the optimality criteria using differential representation of uncertainty.

In Figure 5.5 the loss of monotonicity of the *A-opt* and *E-opt* criteria can be observed when the covariance is propagated using absolute representation as proved in chapter 5. As explained before, in this case, the only criterion that does not break the monotonicity is the *D-opt* using unit quaternions as is shown in Figure 5.6. Using either Roll-Pitch-Yaw, the monotonicity of *D-opt* does not hold. However, as in the previous experiment in 2D, the monotonicity holds for all optimality criteria when using a differential representation.

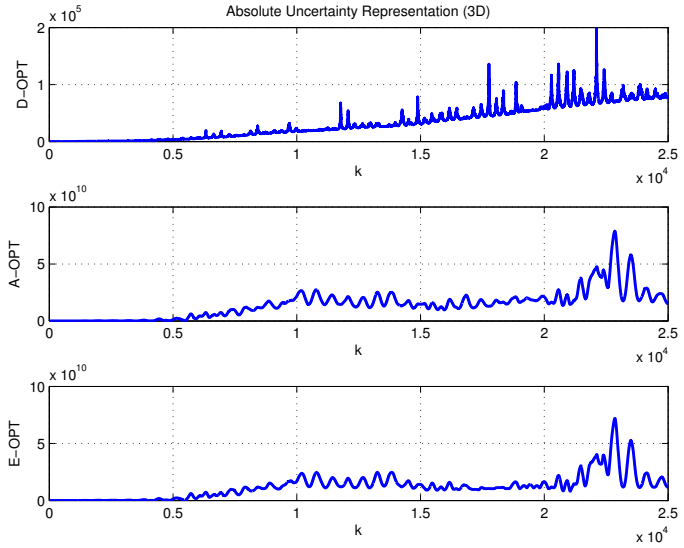

Figure 5.5: Evolution of the optimality criteria *D-opt*, *A-opt* and *E-opt* for the trajectory for 2-dimensional experiment using absolute representation of uncertainty for Roll-Pitch-Yaw angles.

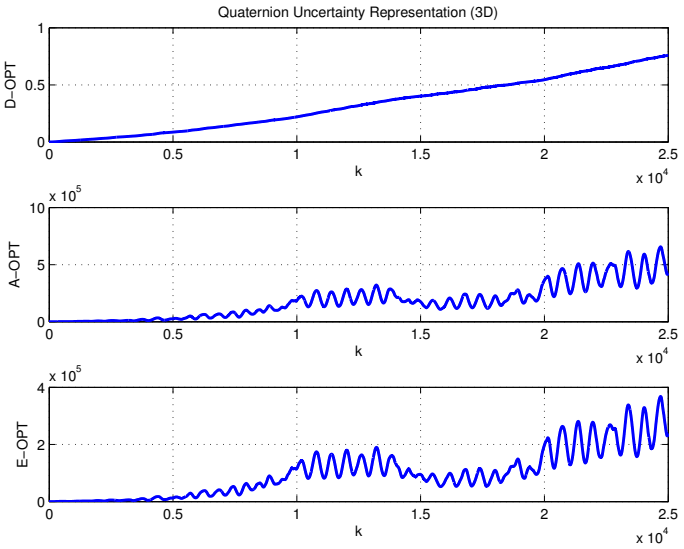

Figure 5.6: Evolution of the optimality criteria *D-opt*, *A-opt* and *E-opt* for the trajectory for 2-dimensional experiment using absolute representation of uncertainty for unit quaternions.

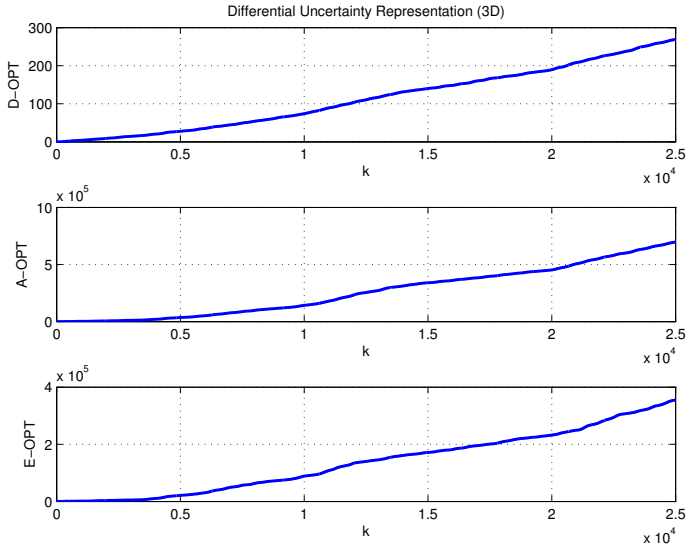

Figure 5.7: Evolution of the optimality criteria *D-opt*, *A-opt* and *E-opt* for the trajectory for 3-dimensional experiment using differential representation of uncertainty.

5.6 Conclusions

Given that assessing if the uncertainty increases or decreases is of utmost importance in an active SLAM algorithm, in this paper we have shown that the monotonicity of decision making criteria to quantify the uncertain localization strongly depends on the representation of the uncertainty of the robot's pose.

Mainly two types of models of probabilistic representation of the uncertainty have been considered. In absolute representation the estimated location of the robot is given by the expected value and the uncertainty by its associated covariance matrix, while the differential representation uses a local representation of the uncertainty where the estimated location of robot is the best approximation of the real location and the estimation error is represented locally by a differential location vector.

We have used both representations in a linearized framework to propagate the covariance matrix of the robot's pose in a dead-reckoning scenario using different ways to represent rotations: orthonormal rotation matrices, Roll-Pitch-Yaw and Euler angles, rotation axis and angle, and unit quaternions.

The two most common approaches to quantify uncertainty in the estimation of random variables are the Information Theory and the Theory of Optimal Experimental Design. The most fundamental quantity in information theory is entropy that measures the amount of uncertainty of an unknown or random quantity. In Theory of Optimal Experimental Design bases its process of quantifying the uncertainty on the second moment of the probability distribution associated to the estimated random variable, the covariance matrix. The common optimality criteria used in the literature are *A-opt* (trace of the covariance matrix, or sum of its eigenvalues), *D-opt* (determinant of the covariance matrix, or product of its eigenvalues) and *E-opt* (largest eigenvalue) criteria.

Our fundamental conclusion is that differential representations maintain the monotonicity of the decision making criteria derived from the theory of optimal design of experiments and from information theory usually employed in active SLAM missions. In absolute representation the only criterion that maintains monotonicity is the *D-opt* if unit quaternions are used. Also, we report that guarantying the monotonicity of the Shannon's entropy is equivalent to guarantying the monotonicity of the *D-opt* criterion.

CHAPTER 6

Conclusions

Active Simultaneous Localization and Mapping (Active SLAM) is the central topic of this thesis. This problem has received a lot of attention from the robotics community for its relevance in mobile robotics applications. New findings in active SLAM implies advances in real applications with autonomous operations under uncertainty that are essential in numerous problem domains, including autonomous navigation, object manipulation, multi-robot localization and tracking, and robotic surgery. The objective of Active SLAM is to design robot trajectories to actively explore an environment and minimize the map error. In this setting an SLAM algorithm must solve the so-called exploration-exploitation dilemma. The exploration involves moving in previously unvisited parts with the objective of increasing the overall knowledge of the environment, while the latter is exploitation, i.e., it involves revisiting areas to maximize the information gain. In this thesis, the focus is on the exploration part of Active SLAM.

Representation, estimation and propagation of the uncertainty in the estimation of the mobile robot is of utmost importance to correctly report the associated confidence of the robot's location estimate at each time step along its trajectory and therefore deciding the correct course of action in an active SLAM mission. Different types of models of probabilistic representation of the uncertainty have been proposed in the literature. In absolute representation the estimated location of the robot is given by the expected value and the uncertainty by its associated covariance matrix, while the differential representation uses a local representation of the uncertainty where the estimated location of robot is the best approximation of the real location and the estimation error is represented locally by a differential location vector. Equi-

valent representations over the Lie group have also been considered where uncertainties of the location of the robot are represented in the tangent space associated to the Lie group.

In this thesis, we review related work about representation and propagation of the uncertainty and present a survey of different types of models proposed in the literature to propagate the uncertainty representation. Additionally, an analysis of the uncertainty represented with a differential error and over Lie groups is carried out taking into account the similarities between both models of representation. We show that the representation of uncertainty over Lie groups and using differential location vectors are similar and are independent of the representation used for rotational part of the robot's pose. This is due to the uncertainty is represented locally in the space of differential transformations for translation and rotation that correspond with the Lie algebra of special Euclidean group SE(n). However, in 3-dimensional space, the homogeneous transformation associated to the approximation of the real location depend on the different ways of representation the rotational part. Therefore, a proper way to jointly manipulation the estimation and uncertainty of the pose is to use the theory of Lie groups due to is a representation to guarantee properties such as a minimal representation and free of singularities in rotation angles.

A second problem addressed in this thesis is the quantification of uncertainty in an active SLAM scenario because quantifying correctly the uncertainty associated to the estimation of the mobile robot is an essential step towards making correct decisions in an active SLAM algorithm. During the exploration phase, where the robot navigates in a previously unknown region, the uncertainty associated with the robot's localization grows unbounded. Therefore, guaranteeing monotonicity of the decision making criteria is essential, if monotonicity of the criteria considered is not preserved, the system might select courses of action or paths that it falsely believes lead to less uncertainty in the robot.

The two most common approaches to quantify uncertainty in the estimation of random variables are the Information Theory and the Theory of Optimal Experimental Design. The most fundamental quantity in information theory is entropy that measures the amount of uncertainty of an unknown or random quantity. In Theory of Optimal Experimental Design bases its process of quantifying the uncertainty on the second moment of the probability distribution associated to the estimated random variable, the covariance matrix. The common optimality criteria used in the literature are

A-opt (trace of the covariance matrix, or sum of its eigenvalues), *D-opt* (determinant of the covariance matrix, or product of its eigenvalues) and *E-opt* (largest eigenvalue) criteria.

In this thesis, we investigate the monotonicity of different decision making criteria, both in 2-dimensional and 3-dimensional space, taking into account the different models of representation of uncertainty of the robot's pose. We have used both absolute and differential representations in a linearized framework to propagate the covariance matrix of the robot's pose in a dead-reckoning scenario using different ways to represent rotations: orthonormal rotation matrices, Roll-Pitch-Yaw and Euler angles, rotation axis and angle, and unit quaternions. We analytically show that, using differential representations to propagate spatial uncertainties, monotonicity is preserved for all optimality criteria, *A-opt*, *D-opt* and *E-opt* and for Shannon's entropy. We also show that monotonicity does not hold for any of them in absolute representations using Roll-Pitch-Yaw and Euler angles. Finally, we show that using unit quaternions in absolute representations, the only criteria that preserve monotonicity are *D-opt* and Shannon's entropy.

These findings can guide active SLAM researchers to adequately select a representation model for uncertainty, so that path planning and exploration algorithms can correctly assess the evolution of location uncertainty.

APPENDIX A

A review of Partially Observable Markov Decision Processes

A.1 Introduction

As in mentioned in chapter 2, active SLAM is an instance of a more general framework known as Partially Observable Markov Decision Processes (POMDP) (Thrun et al., 2005). POMDP describes the process of making decisions when both the actions and the sensing are uncertain, i.e. the state of interest is not directly observable.

In this appendix, we present the mathematical formulation of POMDP framework that has been studied during the research visit to The Robotics, Vision and Control Group of the department Ingeniería de Sistemas y Automática (Escuela Técnica Superior de Ingenieros, Universidad de Sevilla). This survey is based mainly on (White, 1991) and (Braziunas, 2003).

As future work, based on the knowledge acquired, the development of an active SLAM algorithm with restrictions is pursuit. Specifically, a decision-making algorithm with uncertainty for active perception in systems with multiple unmanned aerial vehicles will be worked on. In decision-making, restrictions related to time of flight, battery recharge, minimization of energy consumed, environmental factors (such as wind) and communications link will be raised.

A.2 Sequential decision processes

A.2.1 Markov decision process framework

The most commonly used formal model of fully-observable sequential decision processes is the Markov decision process (MDP) model. An MDP can be viewed as an extension of Markov chains with a set of decisions (actions) and a state-based reward or cost structure. For each possible state of the process, a decision has to be made regarding which action should be executed in that state. The chosen action affects both the transition probabilities and the costs (or rewards) incurred. The goal is to choose an optimal action in every state to increase some predefined measure of performance. The decision process for doing this is referred to as the Markov decision process.

A state is a description of the environment at a particular point in time. It can generally be assumed that the environment can be in a finite number of states, and the agent can choose from a finite set of actions. Let $S = \{s_0, s_1, \ldots, s_N\}$ be a finite set of states. Since the process is stochastic, a particular state at some discrete stage, or time step $t \in T$, can be viewed as a random variable S^t whose domain is the state space S.

For a process to be Markovian, the state has to contain enough information to predict the next state. This means that the past history of system states is irrelevant to predicting the future,

$$Pr(S^{t+1}|S^0, S^1, \ldots, S^t) = Pr(S^{t+1}|S^t). \tag{A.1}$$

At each stage, the agent can affect the state transition probabilities by executing one of the available actions. The set of all actions will be denoted by A. Thus, each action $a \in A$ is fully described by a $|S| \times |S|$ state transition matrix, whose entry in an i-th row and j-th column is the probability that the system will move from state s_i to state s_j if action a gets executed,

$$p_{ij}^a = Pr(S^{t+1} = s_j | S^t = s_i, A^t = a) \tag{A.2}$$

It is usually assumed that the processes are stationary, i.e., that the transition probabilities do not depend on the current time step.

The transition function $T(\cdot)$ summarizes the effects of actions on systems states. $T : S \times A \to \Delta(S)$ is a function that for each state and action

associates a probability distribution over the possible successor states ($\Delta(S)$ denotes the set of all probability distributions over S). Thus, for each $s, s^{'} \in S$ and $a \in A$, the function T determines the probability of a transition from state s to state $s^{'}$ after executing action a, i.e.,

$$T(s, a, s^{'}) = Pr(S^{t+1} = s^{'} | S^{t} = s, A^{t} = a) \tag{A.3}$$

$R : S \times A \rightarrow \mathbb{R}$ is a reward function that for each state and action assigns a numeric reward (or cost, if the value is negative). $R(s, a)$ is an immediate reward that an agent would receive for being in state s and executing action a.

A.2.2 Partially observable Markov decision processes framework

Partially observable Markov decision processes (POMDPs) provide a natural framework for sequential decision making under uncertainty. This model augments a well-researched framework of Markov decision processes (MDPs) (Howard, 1960; Puterman, 1994) to situations where an agent cannot reliably identify the underlying environment state. The goal is to find a policy that maximizes the expected total return specified by a reward function. The POMDP formalism is very general and powerful, extending the application of MDPs to many realistic problems. Unfortunately, the generality of POMDPs entails high computational cost.

A POMDP is a generalization of MDPs to situations in which system states are not fully observable. This realistic extension of MDPs dramatically increases the complexity of POMDPs, making exact solutions virtually intractable. In order to act optimally, an agent might need to take into account all the previous history of observations and actions, rather than just the current state it is in. A POMDP is comprised of an underlying MDP, extended with an observation space O and observation function $Z(\cdot)$.

Let O be a set of observations an agent can receive. In MDPs, the agent has full knowledge of the system state; therefore, $O \equiv S$. In partially observable environments, observations are only probabilistically dependent on the underlying environment state. Determining which state the agent is in becomes problematic, because the same observation can be observed in different states.

$Z : S \times A \to \Delta(O)$ is an observation function that specifies the relationship between system states and observations. $Z(s^{'}, a, o^{'})$ is the probability that observation $o^{'}$ will be recorded after an agent performs action a and lands in state $s^{'}$,

$$Z(s^{'}, a, o^{'}) = Pr(O^{t+1} = o^{'} | S^{t+1} = s^{'}, A^{t} = a) \tag{A.4}$$

Formally, a POMDP is a tuple $\langle S, A, T, R, O, Z \rangle$, consisting of the state space S, action space A, transition function $T(\cdot)$, reward function $R(\cdot)$, observation space O, and observation function $Z(\cdot)$.

A.3 Basic concepts

A.3.1 History

A history is a record of everything that happened during the execution of the process. For POMDPs, a complete system history from the beginning till time t is a sequence of state, observation, and action triples,

$$\langle S^0, O^0, A^0 \rangle, \langle S^1, O^1, A^1 \rangle, \ldots, \langle S^t, O^t, A^t \rangle \tag{A.5}$$

The set of all complete histories (or trajectories) will be denoted as H.

Since rewards depend only on visited states and executed actions, a system history is enough to evaluate an agent's performance. Thus, a system history is just a sequence of state and action pairs,

$$\langle S^0, A^0 \rangle, \langle S^1, A^1 \rangle, \ldots, \langle S^t, A^t \rangle. \tag{A.6}$$

A system history h from the set of all system histories H_s provides an external, objective view about the process; therefore, value functions will be defined on the set H_s in the next subsection.

In a partially observable environment, an agent cannot fully observe the underlying world state; therefore, it can only base its decisions on the observable history. Let's assume that at the outset, the agent has prior beliefs about the world that are summarized by the probability distribution b_0 over the system states; the agent starts by executing some action a_0 based solely

on b_0. The observable history until time step t is then a sequence of action and observation pairs,

$$\langle A^0, O^1\rangle, \langle A^1, O^2\rangle, \ldots, \langle A^{t-1}, O^t\rangle \tag{A.7}$$

The set of all possible observable histories will be denoted as H_o. Different ways of structuring and representing H_o have resulted in different POMDP solution and policy execution algorithms.

A.3.2 Value function

At each step in a sequential decision process, an agent has to decide what action to perform based on its observable history. A policy $\pi : H_o \rightarrow A$ is a rule that maps observable trajectories into actions. A given policy induces a probability distribution over all possible sequences of states and actions, for an initial distribution b_0. Therefore, an agent has control over the likelihood of particular system trajectories. Its goal is to choose a policy that maximizes some objective function that is defined on the set of system histories H_s.

Such objective function is called a value function $V(\cdot)$; it essentially ranks system trajectories by assigning a real number to each $h \in H_s$; a system history h is preferred to $h^{'}$ if and only if $V(h) > V(h^{'})$. Formally, a value function is a mapping from the set of system histories into real numbers,

$$V : H_s \rightarrow \mathbb{R} \tag{A.8}$$

A common assumption is that $V(\cdot)$ is additive, the value of a particular system history is simply a sum of rewards accrued at each time step. If the decision process stops after a finite number of steps H, the problem is a finite horizon problem. In such problems, it is common to maximize the total expected reward. The value function for a system trajectory h of length H is simply the sum of rewards attained at each stage Bellman (1957),

$$V(h) = \sum_{t=0}^{t=H} R(s^t, a^t) \tag{A.9}$$

The sum of rewards over an infinite trajectory may be unbounded. A mathematically elegant way to address this problem is to introduce a discount factor γ; the rewards received later get discounted, and contribute less than

current rewards. The value function for a total discounted reward problem is (**?**),

$$V(h) = \sum_{t=0}^{\infty} \gamma^t R(s^t, a^t), 0 \leq \gamma < 1 \tag{A.10}$$

This formulation is very common in current MDP and POMDP literature, including the key papers concerning policy-based search in POMDPs (Hansen, 1997, 1998b; Meuleau et al., 1999a,b). Another popular value function is the average reward per stage, e.g., in (Aberdeen and Baxter, 2002).

A.3.3 Policy representations

Generally, an agent's task is to calculate the optimal course of action in an uncertain environment and then execute its plan contingent on the history of its sensory inputs. The agent's behavior is therefore determined by its policy π, which in its most general form is a mapping from the set of observable histories to actions,

$$\pi : H_o \rightarrow A \tag{A.11}$$

Given a history,

$$h^t = \langle a^0, o^1 \rangle, \langle a^1, o^2 \rangle, \ldots, \langle a^{t-1}, o^t \rangle \tag{A.12}$$

the action prescribed by the policy π at time t would be $a^t = \pi(h^t)$; a^0 is the agent's initial action, and o^t is the latest observation.

One of the more important concepts is that of an expected policy value. Taking into account a prior belief distribution over the system states b_0, a policy induces a probability distribution $Pr(h|\pi, b_0)$ over the set of system histories H_s. The expected policy value is simply the expected value of system trajectories induced by the policy π,

$$EV(\pi) \equiv V^\pi = \sum_{h \in H_s} V(h) Pr(h|\pi, b_0) \tag{A.13}$$

The value of the policy π at a given starting state s_0 will be denoted $V^\pi(s_0)$. Then,

$$EV(\pi) = \sum_{h \in H_s} b_0(s) V^\pi(s) \tag{A.14}$$

The agent's goal is to find a policy $\pi^* \in \Pi$ with the maximal expected value from the set Π of all possible policies.

The general form of a policy as a mapping from arbitrary observation histories to actions is very impractical. Existing POMDP solution algorithms exploit structure in value and observation functions to calculate optimal policies that have much more tractable representations. For example, observable histories can be represented as probability distributions over system states, or grouped into a finite set of distinguishable classes using finite-suffix trees or finite-state controllers.

MDP policies

A POMDP where an agent can fully observe the underlying system state reduces to an MDP. Since the sequence of states forms a Markov chain, the next state depends only on the current state; the history of the previous states is therefore rendered irrelevant.

For finite horizon MDP problems, the knowledge of the current state and stage is sufficient to represent the whole observable trajectory for the purposes of maximizing total reward (discounted or not). Therefore, a policy π can be reduced to a mapping from states and stages to actions,

$$\pi : S \times T \rightarrow A \tag{A.15}$$

Let $\pi(s,t)$ be the action prescribed by the policy at state s with t stages remaining till the end of the process. The expected value of a policy at any state can then be computed by the following recurrence **?**,

$$\begin{aligned} V_0^\pi(s) &= R(s, \pi(s,0)) \\ V_t^\pi(s) &= R(s, \pi(s,t)) + \gamma \sum_{s' \in S} T(s, \pi(s,t), s') V_{t-1}^\pi(s') \end{aligned} \tag{A.16}$$

The value functions in the set $\{V_t^\pi\}_{0 \leq t \leq H}$ are called t-horizon, or t-step, value functions; H is the horizon length, a predetermined number of stages the process goes through.

A policy π^* is optimal if $V_h^{pi^*}(s) \geq V_h^{pi'}(s)$ for all H-horizon policies π' and all states $s \in S$. The optimal value function is a value function of an optimal policy: $V_H^* \equiv V_H^{\pi^*}$. A key result, called Bellman's principle of optimality **?**

allows to calculate the optimal t-step value function from the $(t-1)$-step value function,

$$V_t^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_{t-1}^*(s') \right] \tag{A.17}$$

This equation has served as a basis for value-iteration MDP solution algorithms and inspired analogous POMDP solution methods.

For infinite horizon MDP problems, optimal decisions can be calculated based only on the current system state, since at any stage, there is still an infinite number of time steps remaining.Without loss of optimality, infinite horizon policies can be represented as mappings from states to actions Howard (1960),

$$\pi : S \to A \tag{A.18}$$

Policies that do not depend on stages are called stationary policies.

The value of a stationary policy π can be determined by a recurrence analogous to the finite horizon case,

$$V^\pi(s) = R(s,\pi(s)) + \gamma \sum_{s' \in S} T(s,\pi(s),s') V^\pi(s') \tag{A.19}$$

The agent's goal is to find a policy π^* that would maximize the value function $V(\cdot)$ for all states $s \in S$. The optimal value function is,

$$V^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \right] \tag{A.20}$$

Equation A.16 and Equation A.19 show how to find the value of a given policy π and provide the basis for policy-iteration algorithms. The calculation is straightforward and amounts to solving a system of linear equations of size $|S| \times |S|$.

On the other hand, value-iteration methods employ Equation A.16 to calculate optimal value functions directly. Optimal policies can then be defined implicitly by value functions. First, we introduce a notion of a Q-function,

or Q-value: $Q(s,a)$ is the value of executing action a at state s, and then following the optimal policy,

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s') \tag{A.21}$$

The optimal infinite horizon policy is a greedy policy with respect to the optimal value function $V^*(\cdot)$,

$$\pi^*(s) = \arg\max_a Q(s,a) \tag{A.22}$$

A stochastic infinite horizon MDP policy is a generalization of a deterministic policy; instead of prescribing a single action to a state, it assigns a distribution over all actions to a state. That is, a stochastic policy,

$$\psi : S \to \Delta(A) \tag{A.23}$$

maps a state to a probability distribution over actions; $\psi(s,a)$ is the probability that action a will be executed at state s. By incorporating expectation over actions, Equation A.19 is rewritten for stochastic policies in a straightforward manner,

$$V^{\psi}(s) = \sum_{a \in A} R(s,a) + \gamma \sum_{s',a} \psi(s,a)T(s,\pi(s),s')V^{\psi}(s') \tag{A.24}$$

While stochastic policies have no advantage for infinite horizon MDPs, we will use them in solving partially observable MDPs. Making policies stochastic allows to convert the discrete action space into a continuous space of distributions over actions. We can then optimize the value function using continuous optimization techniques.

POMDP policy trees

In partially observable environments, an agent can only base its decisions on the history of its actions and observations. Instead of a simple mapping from system states to actions, a generic POMDP policy assumes a more complicated form.

As for MDPs, we will first consider finite horizon policies. With one stage left, all an agent can do is to execute an action; with two stages left, it can

execute an action, receive an observation, and then execute the final action. For a finite horizon of length H, a policy is a tree of height H. Since the number of actions and observations is finite, the set of all policies for horizon H can be represented by a finite set of policy trees.

Each node prescribes an action to be taken at a particular stage; then, an observation received determines the branch to follow. A policy tree for a horizon of length H contains,

$$\sum_{t=0}^{t=H-1} |O|^t = \frac{|O|^H - 1}{|O| - 1} \tag{A.25}$$

nodes. At each node, there are $|A|$ choices of actions. Therefore, the size of the set of all possible H-horizon policy trees is,

$$|A|^{\frac{|O|^H - 1}{|O| - 1}} \tag{A.26}$$

We will now present a recursive definition of policy trees using an important notion of conditional plans. A conditional plan $\sigma \in \Gamma$ is a pair $\langle a, v \rangle$ where $a \in A$ is an action, and $v : O \to \Gamma$ is an observation strategy. The set of all observation strategies will be denoted as Γ^O; obviously, its size is $|\Gamma|^{|O|}$.

A particular conditional plan tells an agent what action to perform, and what to do next contingent on an observation received. Let Γ_t be the set of all conditional plans available to an agent with t stages left,

$$\Gamma_t = \{\langle a, v_t \rangle | a \in A, v_t \in \Gamma_{t-1}^O\} \tag{A.27}$$

In this case, $v_t : O \to \Gamma_{t-1}$ is a stage-dependent observation strategy. As a tree of height t can be defined recursively in terms of its subtrees of height $t - 1$, so the conditional plans of horizon t can be defined in terms of conditional plans of horizon $t - 1$. At the last time step, a conditional plan simply returns an action. A policy tree therefore directly corresponds to a conditional plan. We will use the set Γ_t to denote both the set of t-step policy trees and the equivalent set of conditional plans.

Representing policy trees as conditional plans allows us to write down a recursive expression for their value function. The value function of a non-stationary policy π_t represented by a t-horizon conditional plan $\sigma_t = \langle a, v_t \rangle$

is,

$$V_0^{\pi}(s) = R(s, \sigma_0(s))$$
$$V_t^{\pi}(s) = V_t^{\sigma_t}(s) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \sum_{o \in O} Z(s',a,o) V_{t-1}^{v_t(o)}(s') \quad \text{(A.28)}$$

where $\sigma_0(s)$ is the action to be executed at the last stage.

Since the actual system state is not fully known, we need to calculate the value of a particular policy tree with respect to a (initial) belief state b. Such value is just an expectation of executing the conditional plan σ_t at each state $s \in S$,

$$V_t^{\pi}(b) = V_t^{\sigma_t}(b) = \sum_{s \in S} b(s) V_t^{\sigma_t}(s) \quad \text{(A.29)}$$

The optimal t-step value function for the belief state b can be found simply by enumerating all the possible policy trees in the set Γ_t,

$$V_t^*(b) = \max_{\sigma \in \Gamma_t} \sum_{s \in S} b(s) V_t^{\sigma}(s) \quad \text{(A.30)}$$

Thus, the t-step value function for the continuous belief simplex B can in principle be represented by a finite (although doubly exponential in t!) set of conditional plans and a max operator. The next section discusses some ways of making such a representation more tractable.

α-vectors and belied state MDPs

Equation A.30 illustrates the fact that the optimal t-step POMDP value function is piecewise linear and convex (Sondik, 1971, 1978). From equation Equation A.29 we can see that the value of any policy tree V_t^{σ} is linear in b; hence, from Equation A.30, V_t^* is simply the upper surface of the collection of value functions of policies in Γ_t.

Let α^{σ} be a vector of size $|S|$ whose entries are the values of the conditional plan σ (or, values of a policy tre corresponding to σ) for each state s,

$$\alpha^{\sigma} = [V^{\sigma}(s_0), V^{\sigma}(s_1), \ldots, V^{\sigma}(s_N)] \quad \text{(A.31)}$$

Equation A.30 can then be rewritten in terms of α-vectors,

$$V_t^*(b) = \max_{\sigma \in \Gamma_t} \sum_{s \in S} b(s) \alpha^{\sigma}(s) = \max_{\alpha \in V_t} \sum_{s \in S} b(s) \alpha(s) \quad \text{(A.32)}$$

Here, the set V_t contains all t-step α-vectors; these vectors correspond to t-step policy trees and are sufficient to define the optimal t-horizon value function.

The optimal value function V_t is represented by the upper surface of the α-vectors in V_t. Although in the worst case any policy in Γ_t might be superior for some belief region, this rarely happens in practice. Many vectors in the set V_t might be dominated by other vectors, and therefore not needed to represent the optimal value function. Given the set of all α-vectors V_t, it is possible to prune it down to a parsimonious subset V_t^- that represents the same optimal value function,

$$V_t^*(b) = \max_{\alpha \in V_t} \sum_{s \in S} b(s)\alpha(s) = \max_{\alpha \in V_t^-} \sum_{s \in S} b(s)\alpha(s) \tag{A.33}$$

A vector α is useful if there is a non-empty belief region $R(\alpha, V)$ over which it dominates all other vectors, where

$$R(\alpha, V) = \{b | b \cdot \alpha \rangle b \cdot \alpha', \alpha' \in V - \{\alpha\}, b \in B\} \tag{A.34}$$

The existence of such region can be easily determined using linear programming. Various value-based POMDP solution algorithms differ in their methods of pruning the set of all α-vectors V_t to a parsimonious subset V_t^-.

Implicit POMDP policies

As we already know, an explicit t-step POMDP policy can be represented by a policy tree or a recursive conditional plan. Given an initial belief state b_0, the optimal t-step policy can be found by going through the set of all useful policy trees and finding the one whose value function is maximal with respect to b_0. Then, executing the finite horizon policy is straightforward: an agent only needs to perform actions at the nodes, and follow the observation links to policy subtrees.

Instead of keeping all policy trees, it is enough to maintain the set of useful α-vectors V_t^- for each stage t. As for MDPs, an implicit t-step policy can be defined by doing a greedy one-step look ahead. First, we will define the Q-value function $Q_t(b, a)$ as a value of taking action a at belief state b and continuing optimally for the remaining $t - 1$ stages,

$$Q_t(b, a) = \sum_{s \in S} b(s)R(s, a) + \gamma \sum_{o \in O} Pr(o|a, b)V_{t-1}^*(b_o^a) \tag{A.35}$$

where b_o^a is the belief state that results from b after taking action a and receiving observation o. It can be calculated using POMDP model and Bayes's theorem.

The optimal action to take at b with t stages remaining simply,

$$\pi^*(b,t) = \arg\max_{a \in A} Q_t(b,a) \tag{A.36}$$

Belief state MDPs

A finite horizon POMDP policy now becomes a mapping from belief states and stages to actions,

$$\pi : B \times T \to A \tag{A.37}$$

Astrom (1965) has shown that properly updated probability distribution over the state space S is sufficient to summarize all the observable history of a POMDP agent without loss of optimality.

Therefore, a POMDP can be cast into a framework of a fully observable MDP where belief states comprise the continuous, but fully observable, MDP state space. A belief state MDP is threfore a quadruple $\langle B, A, T^b, R^b \rangle$, where

- $B = \Delta(S)$ is the continuous state space.
- A is the action space, which is the same as in the original POMDP.
- $T^b : B \times A \to B$ is the belief transition function:

$$\begin{aligned} T^b(b,a,b') &= Pr(b'|b,a) \\ &= \sum_{o \in O} Pr(b'|a,b,o) Pr(o|a,b) \\ &= \sum_{o \in O} Pr(b'|a,b,o) \sum_{s' \in S} Z(s',a,o) \sum_{s \in S} T(s,a,s') b(s) \end{aligned} \tag{A.38}$$

  where

$$Pr(b'|a,b,o) = \begin{cases} 1 & \text{if } b_o^a = b' \\ 0 & \text{otherwise} \end{cases} \tag{A.39}$$

  After action a and observation o, the update belief b_o^a can be calculated from the previous belief b:

$$b_o^a(s') = \frac{Z(s',a,o) \sum_{s \in S} T(s,a,s') b(s)}{Pr(o|a,b)} \tag{A.40}$$

- $R^b : B \times A \rightarrow \mathbb{R}$ is the reward function:

$$R^b(b,a) = \sum_{s \in S} b(s) R(s,a) \tag{A.41}$$

To follow the policy that maps from belief states to actions, the agent simply has to execute the action prescribed by the policy, and then update its probability distribution over the system states according to Equation A.40.

The infinite horizon optimal value function remains convex, but not necessarily piecewise linear, although it can be approximated arbitrarily closely by a piecewise linear and convex function (Sondik, 1978). The optimal policy for infinite horizon problems is then just a stationary mapping from belief space to actions,

$$\pi : B \rightarrow A \tag{A.42}$$

It can be extracted by performing a greedy one-step lookahead with respect to the optimal value function V^*,

$$\begin{aligned} Q(b,a) &= \sum_{s \in S} b(s) R(s,a) + \gamma \sum_{o \in O} Pr(o|a,b) V^*(b_o^a) \\ \pi^*(b) &= \arg\max_{a \in A} Q(b,a) \end{aligned} \tag{A.43}$$

Finite-state controllers

The optimal infinite horizon value function V^* can be approximated arbitrarily closely by successive finite horizon value functions $V_0, V_1, \ldots, V_t$, as $t \rightarrow \infty$ (Sondik, 1978). While all optimal t-horizon policies are piecewise-linear and convex, this is not always true for infinite horizon value functions. They remain convex (White and Harrington, 1980), but may contain infinitely many facets.

Some optimal value functions do remain piecewise linear; therefore, at some horizon t, the two successive value functions V_t and V_{t+1} are equal, and therefore, optimal,

$$V^* = V_t = V_{t+1} \tag{A.44}$$

Each vector α in a parsimonious set V^* that represents the optimal infinite horizon value function V^* has an associated belief space region $R(\alpha, V^*)$ over

which it dominates all other vectors,

$$R(\alpha, V^*) = \{b | b \cdot \alpha > b \cdot \alpha^{'}, \alpha^{'} \in V^* - \{\alpha\}, b \in B\} \tag{A.45}$$

Thus, α-vectors define a partition of the belief space. In addition, it has been shown that for each partition there is an optimal action (Smallwood and Sondik, 1973). When an optimal value function V^* can be represented by a finite set of vectors, all belief states within one region get transformed to new belief states within the same single belief partition, given the optimal action and a resulting observation. The set of partitions and belief transitions constitute a policy graph, where nodes correspond to belief space partitions with optimal actions attached, and transitions are guided by observations (Cassandra et al., 1994). Another way of understanding the concept of policy graphs is illustrated in an article by Kaelbling et al. (1998). If the finite horizon value functions V_t and V_{t+1} become equal, at every level above t the corresponding conditional plans have the same value. Then, it is possible to redraw the observation links from one level to itself as if it were the succeeding level.

Essentially, we can convert non-stationary t-step policy trees (which are non-cyclic policy graphs) into stationary cyclic policy graphs. Such policy graphs enable an agent to execute policies simply by doing actions prescribed at the nodes, and following observation links to successor nodes. The nodes partition the belief space in a way that, for a given action and observation, all belief states in a particular region map to a single region (represented by another graph node). Therefore, an agent does not have to explicitly maintain its belief state and perform expensive operations of updating its beliefs and finding the best *alpha*-vector for the belief state. The starting node is optimized for the initial belief state. Of course, not all POMDP problems allow for optimal infinite horizon policies to be represented by a finite policy graph. Since such a graph cannot be extracted from a suboptimal value function, a policy in such cases is usually defined implicitly by a value function.

However, limiting the size of a policy provides a tractable way of solving POMDPs approximately. Although generally the optimal policy depends on the whole history of observations and actions, one way of facilitating the solution of POMDPs is to assume that an agent has a finite memory. We can represent this finite memory by a set of internal states N. The internal states are fully observable; therefore an agent can execute a policy that maps from internal states to actions.

The action selection function determines what action to execute at each internal memory state $n \in N$. In addition to the mapping from internal states to actions, we also need to specify the dynamics of the internal process, i.e., describe the transitions from one internal state to another. The internal memory states can be viewed as nodes, and the transitions between nodes will depend on observations received. Together, the set of nodes and the transition function constitute a policy graph, or a finite-state controller (FSC).

A FSC model can be defined by the following model. A deterministic policy graph is a triple $\langle N, \gamma, \mu \rangle$, where

- N is a set of controller nodes n, also known as internal memory states.
- $\psi : N \rightarrow A$ is the action selection function that for each node n prescribes an action $\psi(n)$.
- $\mu : N \times O \rightarrow N$ is the node transition function that for each node and observation assigns a successor node $n^{'}$. $\mu(n, \cdot)$ is essentially an observation strategy for the node n, described above when discussing policy trees and conditional plans

In a stochastic FSC, the action selection function ψ and the internal transition function μ are stochastic. Here,

- $\psi : N \rightarrow \Delta(A)$ is the stochastic action selection function that for each node n prescribes a distribution over actions:

$$\psi(n, a) = Pr(A^t = a | N^t = n) \tag{A.46}$$

- $\mu : N \times O \rightarrow \Delta(N)$ is the stochastic node transition function that for each node and observation assigns a probability distribution over successor nodes $n^{'}$; $\mu(n, o, n^{'})$ is the probability of transition from node n to node $n^{'}$ after observing $o^{'} \in O$:

$$\mu(n, o^{'}, n^{'}) = Pr(N^{t+1} = n | N^t = n, O^{t+1} = o^{'}) \tag{A.47}$$

Cross-product MDP

In the way that an MDP policy $\pi : S \rightarrow \Delta(A)$ gives rise to a Markov chain defined by the transition matrix T, a POMDP policy, represented by a finite

graph, is also sufficient to render the dynamics of a POMDP Markovian. The cross-product between the POMDP and the finite policy graph is itself a finite MDP, which will be referred to as the cross-product MDP. The structure of both the POMDP and the policy graph can be represented in the cross-productMDP.

Given a POMDP $\langle S, A, T, R, O, Z\rangle$ and a policy graph with the node set N, the new cross-product MDP $\bar{S}, \bar{A}, \bar{T}, \bar{R}$ can be described as follows (Meuleau et al., 1999a),

- The state space $\bar{S} = N \times S$ is the Cartesian product of external system states and internal memory nodes; it consists of pairs $\langle n, s\rangle, n \in N, s \in S$.

- At each state $\langle n, s\rangle$, there is a choice of action $a \in A$, and a conditional observation strategy $v : O \rightarrow N$, which determines the next internal node for each possible observation. The new action space $\bar{A} = A \times N^O$ is therefore a cross product between A and the space of observation mappings N^O. A pair $\langle a, v\rangle$ is a conditional plan, where $a \in A$ is an action and $v \in N^O$ is a deterministic observation strategy.

- $\bar{T} : \bar{S} \times \bar{A} \rightarrow \bar{S}$ is the transition function:

$$\bar{T}(\langle n, s\rangle, \langle a, v\rangle, \langle n^{'}, s^{'}\rangle) = T(s, a, s^{'}) \sum_{o|v(o)=n^{'}} Z(s^{'}, a, o) \tag{A.48}$$

- The reward function $\bar{R} : \bar{S} \times \bar{A} \rightarrow \mathbb{R}$ becomes:

$$\bar{R}(\langle n, s\rangle, \langle a, v\rangle) = R(s, a) \tag{A.49}$$

Given a (stochastic) policy graph $\pi = (N, \psi, \mu)$ and a POMDP decribed by (S, A, T, R, O, Z), the generated sequence of node-state pairs (N^t, S^t) constitutes a Markov chain (Hansen, 1997, 1998b; Meuleau et al., 1999a). The value of a given policy graph can be calculated using Bellman's equations,

$$\bar{V}^{\pi}(\bar{s}) = \bar{R}^{\pi}(\bar{s}) + \gamma \sum_{\bar{s}^{'}} \bar{T}^{\pi}(\bar{s}, \bar{s}^{'}) \bar{V}^{\pi}(\bar{s}^{'}) \tag{A.50}$$

where $\bar{s}, \bar{s}^{'}$ are node-state pairs in $\bar{S}$, and

- $\bar{T}^{\pi}$ is the transition matrix. Given stochastic function $psi(\cdot)$ and $\mu(\cdot)$, the transition matrix is:

$$\bar{T}^{\pi}(\langle n, s\rangle, \langle n^{'}, s^{'}\rangle) = \sum_{a,o} \psi(n,a)\mu(n,o,n^{'})T(s,a,s^{'})Z(s^{'},a,o) \quad \text{(A.51)}$$

- $\bar{R}^{\pi}$ is the reward vector:

$$\bar{R}^{\pi}(\langle n, s\rangle) = \sum_{a} \psi(n,a)R(s,a) \quad \text{(A.52)}$$

A.4 Exact solution algorithms

A.4.1 Value iteration

MDP value iteration

Value iteration for MDPs is a standard method of finding the optimal finite horizon policy π^* using a sequence of optimal finite horizon value functions $V_0^*, V_1^*, \ldots, V_t^*$ (Howard, 1960). The difference between the optimal value function and the optimal t-horizon value function goes to zero as t goes to infinity,

$$\lim_{t\to\infty} \max_{s\in S} |V^*(s) - V_t^*(s)| = 0 \quad \text{(A.53)}$$

It turns out that the optimal value function can be calculated in finite number of steps given the *Bellman error*, ϵ, which is the maximum difference (for all states) between two successive finite horizon value functions. Using Equation A.17, the value iteration algorithm for MDPs can be summarized as follows,

- Initialize $t = 0$ and $V_0(s) = 0$ for all $s \in S$.
- While $\max_{s\in S} |V_{t+1}(s) - V_t(s)| > \epsilon$, calculated $V_{t+1}(s)$ for all states $s \in S$ according to the following equation, and then increment t:

$$V_{t+1}(s) = \max_{a\in A} \left[R(s,a) + \gamma \sum_{s^{'}\in S} T(s,a,s^{'})V_t(s^{'}) \right] \quad \text{(A.54)}$$

POMDP value iteration

As described in previous sections, any POMDP can be reduced to a continuous belief-state MDP. Therefore, value iteration can also be used to calculated optimal infinite horizon POMDP policies as follows,

- Initialize $t = 0$ and $V_0(b) = 0$ for all $b \in B$.
- While $\sup_{b \in B} |V_{t+1}(b) - V_t(b)| > \epsilon$, calculate $V_{t+1}(b)$ for all states $b \in B$ according to the following equation, and then increment t:

$$V_{t+1}(b) = \max_{a \in A} \left[R^b(s,a) + \gamma \sum_{b' \in B} T^b(b,a,b') V_t(b') \right] \tag{A.55}$$

The previous equation can be rewritten in terms of the original POMDP formulation as,

$$V_{t+1}(b) = \max_{a \in A} \left[\sum_{s \in S} b(s) R(s,a) + \gamma \sum_{o \in O} Pr(o|a,b) V_t(b_o^a) \right] \tag{A.56}$$

where $Pr(o|a,b)$ is,

$$Pr(o|a,b) = \sum_{s' \in S} Z(s',a,o) \sum_{s \in S} T(s,a,s') b(s) \tag{A.57}$$

A.4.2 Policy iteration

Policy iteration algorithms proceed by iteratively improving the policies themselves. The sequence $\pi_0, \pi_1, \ldots, \pi_t$ then converges to the optimal infinite horizon π^*, as $t \to \infty$. Policy iteration algorithms usually consist of two stages: policy evaluation and policy improvement (Braziunas, 2003).

MDP policy iteration

First, we summarize the policy iteration method for MDPs (Howard, 1960),

- Initialize $\pi_0(s) = a$, for all $s \in S$; $a \in A$ is an arbitrary action. Then, repeat the following policy iteration and improvement steps until the policy does not change anymore, i.e., $\pi_{t+1}(s) = \pi_t(s)$ for all states $s \in S$.

- Policy evaluation: calculate the value of policy π_t (using Equation A.19):

$$V^{\pi_t}(s) = R(s, \pi_t(s)) + \gamma \sum_{s' \in S} T(s, \pi_t(s), s') V^{\pi_t(s')} \tag{A.58}$$

- Policy improvement: for each $s \in S$ and $a \in A$, compute the Q-function $Q_t(s, a)$:

$$Q_{t+1}(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^{\pi_t}(s') \tag{A.59}$$

  Then, improve the policy π_{t+1}:

$$\pi_{t+1}(s) = \arg\max_{a \in S} Q_{t+1}(s, a) \text{ for all } s \in S \tag{A.60}$$

Policy iteration tends to converge much faster than value iteration in practice. However, it performs more computation at each step; policy evaluation step requires a solution of a $|S| \times |S|$ linear system.

POMDP policy iteration

For value iteration, it is important to be able to extract a policy from a value function. For policy iteration, it is important to be able to represent a policy so that its value function can be calculated easily. Here, we will describe a POMDP policy iteration method that uses an FSC to represent the policy explicitly and independently of the value function.

The first POMDP policy iteration algorithm was described by Sondik (1971, 1978). It used a cumbersome representation of a policy as a mapping from a finite number of polyhedral belief space regions to actions, and then converted it to an FSC in order to calculate the policy's value. Because the conversion between the two representations is extremely complicated and difficult to implement, Sondik's policy iteration is not used in practice.

Hansen (1997, 1998b) proposed a similar approach, where a policy is directly represented by a finite state controller. His policy iteration algorithm is

analogous to the policy iteration in MDPs. The policy is initially represented by a deterministic finite-state controller π_0. The algorithm then performs the usual policy iteration steps: evaluation and improvement. The evaluation of the controller π is straightforward; during the improvement step, a dynamic programming update transforms the current controller into an improved one. The sequence of finite-state controllers $\pi_0, \pi_1, \ldots, \pi_t$ converges to the optimal policy π^* as $t \to \infty$.

Policy evaluation

In exact policy iteration, each controller node corresponds to an α-vector in a piecewise-linear and convex value function representation. Since our policy graph is deterministic, $\psi(n)$ outputs the action associated with the node n, and $\mu(n, o)$ is the successor node of n after receiving observation o. The α-vector representation of a value function can be calculated using the cross-product MDP evaluation formula,

$$\bar{V}^{\pi}(\langle n, s\rangle) = R(s, \psi(n)) + \gamma \sum_{s^{'},o} T(s, \psi(n), s^{'}) Z(s^{'}, \psi(n), o) \bar{V}^{\pi}(\mu(n, o), s^{'}) \tag{A.61}$$

$\bar{V}^{\pi}(\langle n, s\rangle)$ is the value of state s of an α-vector corresponding to the node n,

$$\bar{V}^{\pi}(\langle n, s\rangle) \equiv \alpha_i(s) \tag{A.62}$$

Thus, evaluating the cross-product MDP for all states $\bar{s} \in S$ is equivalent to computing a set of α-vectors $\mathcal{V}^{\pi}$. Therefore, policy evaluation step is fairly straightforward and its running time is proportional to $|N \times S|^2$.

Policy improvement

Policy improvement step simply performs a standard dynamic programming backup during which the value function V^{π}, represented by a finite set of α-vectors $\mathcal{V}^{\pi}$, gets transformed into a improved value function $V^{'}$, represented by another finite set of α-vectors $\mathcal{V}^{'}$. Although in the worst case the size of $\mathcal{V}^{'}$ can be proportional to $|A||\mathcal{V}^{\pi}|^{|o|} = |A||N|^{|O|}$ (where $|N|$ is the number of controller nodes at the current iteration), many exact algorithms, such as (Cassandra et al., 1994), fare better in practise.

In the policy evaluation step, a set of α-vectors $\mathcal{V}^{\pi}$ is calculated from the finite-state controller π. Then, the set $\mathcal{V}^{'}$ is computed using dynamic programming backup on the set $\mathcal{V}^{\pi}$.The key insight in Hansen's policy iteration algorithm is observation that the new improved controller $\pi^{'}$ can be constructed from the new set $\mathcal{V}^{'}$ and the current controller π by following three simple rules,

- For each vector $\alpha^{'} \in \mathcal{V}^{'}$:
  - If the action and successor links of $\alpha^{'}$ are identical to the action and conditional plan of some node that is already in π, then the same node will remain unchanged in $\pi^{'}$.
  - If $\alpha^{'}$ pointwise dominates some nodes in π, replace those nodes by a node corresponding to $\alpha^{'}$, i.e., change the action and successor links to those of the vector $\alpha^{'}$.
  - Else, add a node to $\pi^{'}$ that has the action and observation strategy associated with $\alpha^{'}$.

  item Prune any node in π that has no corresponding α-vector in $\mathcal{V}^{'}$ as long as that node is not reachable from a node with an associated vector in $\mathcal{V}^{'}$.

If the policy improvement step does not change the FSC, the controller must be optimal. Of course, this can happen only if the optimal infinite horizon value function does have a finite representation. Otherwise, a succession of FSCs will approximate the optimal value function arbitrarily closely; an ϵ-optimal FSC can be found in a finite number of iterations (Hansen, 1998a). Like MDP policy iteration, POMDP policy iteration in practice requires fewer steps to converge.

Since policy evaluation complexity is negligible compared to the worst-case exponential complexity of the dynamic-programming improvement step, policy iteration appears to have a clearer advantage over value iteration for POMDPs (Hansen, 1998b). Controllers found by Hansen's policy iteration are optimized for all possible initial belief states. The convexity of the value function is preserved because the starting node maximizes the value for the initial belief state. From the next section onward, we will usually assume that an initial belief state is known beforehand, and our solutions will take computational advantage of this fact. Optimal controllers can be much smaller if they do not need to be optimized for all possible belief states (Kaelbling et al., 1998; Hansen, 1998b).

A.5 POMDP algorithms

As it mentioned above, POMDP is a principled approach for planning and making decision under uncertainty but it is hard to solve. In the literature, there are significant efforts in developing approximation algorithms. Traditional planning and control problems in POMDP target to find a policy that maximizes the expectation of accumulated rewards. Since belief state is sufficient statistics for history, POMDP can be viewed as MDP with a continuous state space formed by belief states (Astrom, 1965). This inspires solving POMDP planning by finding the optimal control policy over the continuous belief state space (Zhang et al., 2017). Exact planning of POMDP can be intractable with the size of state space and planning horizon exploding quickly (Sondik, 1978). Therefore, approximation methods are proposed to approximate the value function or limit the policy search space to alleviate the computational complexity.

One of the most popular approaches is the point-based value iteration (PBVI). This approach optimizes the value function only over a selected finite set of belief states and provides the optimization result with a bounded error (Zhang and Zhang, 2001; Pineau et al., 2006; Kurniawati et al., 2008). Point based algorithms have the computational advantage of approximating the value function at a finite set of belief points. This permits much faster updates of the value function compared to exact methods. PBVI provides good results when points are representative and well-separated in the belief space Thrun et al. (2005). An approximate POMDP solution algorithm is the Heuristic Search Value Iteration (HSVI). This method combines techniques for heuristic search with piecewise linear convex value function representations (Smith and Simmons, 2004). Another method is the so-called Successive Approximations of the Reachable Space under Optimal Policies (SARSOP) that improves point-based algorithms by computing successive approximations of R, the subset of belief points reachable from the initial point, through sampling and converging to it iteratively (Kurniawati et al., 2008).

Compared to the point-based approach that solves POMDP on the continuous state space of belief states, the controller based approach (Amato et al., 2010) finds the optimal policy represented by a finite state controller (FSC) with finite memory. There are two types of approaches to find an FSC: policy iteration and gradient search. The policy iteration tends to find the optimal controller, but the size of the controller can grow expo-

nentially fast and turns intractable. The gradient search usually leads to a suboptimal solution that often traps in local optimum Aberdeen and Baxter (2002); Meuleau et al. (1999a). Poupart and Boutilier (2003) proposed bounded policy iteration to combine the advantages from gradient ascent and policy iteration.

A.6 POMDP related frameworks

A.6.1 Constrained POMDPs framework

In many applications, there are several objectives and it may be desirable to ensure that some bounds are respected for some secondary objectives. To that effect, POMDPs have been extended to constrained POMDPs (CPOMDPs) by allowing secondary objectives to be specified with corresponding bounds on the expectation of those objectives (Isom et al., 2008).

A partially observable Markov decision process (POMDP) is defined formally by a tuple, $\langle S, A, O, T, Z, R, \gamma, b_0 \rangle$ where S is the set of states s, A is the set of actions a, O is the set of observations o, $T(s^{'}, s, a) = Pr(s^{'}|s, a)$ is the transition distribution, $Z(o, s^{'}, a) = Pr(o|s^{'}, a)$ is the observation distribution, $R(s, a) \in \mathbb{R}$ is the reward function, $\gamma \in (0, 1)$ is the discount factor and $b_0(s_0) = Pr(s_0)$ is the initial belief at time step 0. Since the underlying state of the system is not directly observable, the decision maker can maintain a belief $b(s_t) = Pr(s_t)$ about the current state s_t at each time step t. The belief can be updated as time progresses based on Bayes' theorem. For instance, the belief b_a^o obtained after executing a in b and observing o is,

$$B^{ao}(s^{'}) = \sum_s b(s) Pr(s^{'}|s, a) Pr(o|s^{'}, a), \forall S^{'} \tag{A.63}$$

A policy $\pi : B \rightarrow A$ is a mapping from beliefs to actions. In POMDP, there is a single objective consisting of the reward function. The goal is to find a policy π^* that maximizes the expected discounted sum of rewards,

$$\pi^* = \arg\max_{\pi} E\left[\sum_t \gamma^t R(s_t, a_t)|\pi\right] \tag{A.64}$$

A constrained POMDP (Isom et al., 2008) allows multiple objectives to be considered. One of the objectives is optimized, while the remaining object-

ives are bounded. without loss of generality, we will assume that the primary objective is maximized and therefore we will denote it by the reward function, while the secondary objectives are assumed to be upper bounded and therefore we will denoted them by cost functions.

Hence, a CPOMDP can be defined by a tuple,

$$\langle S, A, O, T, Z, R, \{C_k\}_{1\cdots K}, \{c_k\}_{1\cdots K}, \gamma, b_0 \rangle \tag{A.65}$$

where $C_k(s, a) \in \mathbb{R}$ is the cost function with upper bound c_k (among K cost functions).

The goal is to find a policy π^* that maximizes the expected discounted sum of rewards while ensuring that the expected discounted sum of costs remains bounded,

$$\pi^* = \arg\max_{\pi} E\left[\sum_t \gamma^t R(s_t, a_t)|\pi\right] \tag{A.66}$$

$$\text{subject to } E\left[\sum_t \gamma^t C_k(s_t, a_t)|\pi\right] \leq c_k \forall k \tag{A.67}$$

The optimization of CPOMDP policies is notoriously difficult. We cannot restrict ourselves to deterministic policies since the optimal policies may all be stochastic (Altman, 1999; Kim et al., 2011). This is due to the presence of multiple objectives, where it may be necessary to randomize over multiple actions in order to trade off between costs and rewards. An exact solution can be obtained by solving a minimax quadratically constrained optimization problem (Kim et al., 2011), however this approach is intractable. A suboptimal dynamic programming technique (Isom et al., 2008) as well as a constrained version of point-based value iteration (Kim et al., 2011) have also been proposed. However, the non-convex nature of the optimal value function complicates dynamic programming and point-based value iteration. Alternatively, it is possible to think of the CPOMDP as a constrained belief MDP (Poupart et al., 2015).

A.6.2 Mixed observability MDPs framework

For robotic systems, uncertainty arises from two main sources: robot control and sensing. It is, important to note that robotic systems often have

mixed observability: even when a robot's state is not fully observable, some components of the state may still be so. For example, consider a mobile robot equipped with a compass, but not a global positioning system (GPS). Its orientation is fully observable, although its position may only be partially observable. Such problems can be called mixed observability MDPs (MOMDPs), a special class of POMDPs (Ong et al., 2010).

In a MOMDP, the fully observable state components are represented as a single state variable x, while the partially observable components are represented as another state variable y. Thus, (x, y) specifies the complete system state, and the state space is factored as $S = X \times Y$, where X is the space of all values for x and Y is the space of all values for y.

Formally, a MOMDP model is specified as a tuple,

$$(X, Y, A, O, T_X, T_Y, Z, R, \gamma) \tag{A.68}$$

The transition function $T_X(x, y, a, x^{'}) = Pr(x^{'}|x, y, a)$ gives the probability that the fully observable state variable has value $x^{'}$ if the robot takes action a in state (x, y), and $T_Y(x, y, a, x^{'}, y^{'}) = Pr(y^{'}|x, y, a, x^{'})$ gives the probability that the partially observable state variable has value $y^{'}$ if the robot takes action a in state (x, y) and the fully observable state variable has resulting value $x^{'}$.

Compared with the POMDP model, the MOMDP model has a factored state space $X \times Y$, with transition functions T_X and T_Y, while other aspects remain the same.

APPENDIX B

Deep Learning SLAM

Currently, one of the new research lines is Deep Learning. Therefore, during the research stay at the Australian Center for Robotic Vision of the University of Adelaide (Australia), a review of the state of the art on the field of action of Deep Learning in SLAM was carried out and the study of Deep Learning and the techniques developed in this area for possible future research.

B.1 Literature review

Location and Simultaneous Mapping, or SLAM, is one of the most important problems in the field of Robotics, with pioneering work done by the community of researchers in computer vision and robotics. At present, algorithms based on Visual SLAM, that is, algorithms that require visual sensors, for example cameras, to perform SLAM, are capable of simultaneously constructing 3D maps while tracking the location and orientation of the camera.

The SLAM algorithms are complementary to those of Deep Learning: SLAM focuses on geometric problems and Deep Leaning on problems of perception (recognition). However, Deep Learning applications in autonomous robots are not yet widespread. Next, a review of the literature of some Deep Learning techniques applied to the main problem in perception, SLAM, is carried out.

Obtaining a good estimate of the location of the robot is crucial in mobile robots, navigation and augmented reality. Recently, there are several authors

who have treated this topic applying Deep Learning. In the work of Agrawal et al. (2015) and Konda and Memisevic (2015) the objective is to learn how to perform visual Odometry from a couple of nearby images.

Kendall et al. (2015) presented a new location system called PoseNet. The main contribution of this work is an estimation of the position of the camera using a Convolutional Neural Network (CNN) from a single RGB image. This system eliminates several existing problems in the SLAM algorithms of the state of the art, such as the need to store keyframes in a dense way, the need to maintain separate mechanisms for estimating the position of the location and the landmarks, or the need to perform feature mapping between images. The authors extended their system to a Bayesian model that is capable of determining the location uncertainty in (Kendall and Cipolla, 2015).

It's also an important issue, as correct loop closures guarantee the consistency of the SLAM map and improve all-around accuracy. Computational efficiency and robustness to false positives are the most important characteristics of a successful loop closure subsystem. Hou et al. (2015) focussed one specific problem that can benefit from the recent development of the CNN technology, i.e., they focussed on using a pre-trained CNN model as a method of generating an image representation appropriate for visual loop closure detection in SLAM. They performed a comprehensive evaluation of the outputs at the intermediate layers of a CNN as image descriptors, in comparison with state-of-the-art image descriptors, in terms of their ability to match images for detecting loop closures Chen et al. (2014) have also worked along the same lines, by leveraging the representations embedded in an existing CNN.

Since SLAM systems are very useful for autonomous navigation as mentioned above, it is possible to ask if there is still a need to explicitly SLAM in many of these applications. For instance, the following latest work by DeepMind is obviously able to understand the environment and navigate within it, without needing any explicit SLAM module. Rather it just learns to navigate in the virtual environment, using a deep reinforcement learning objective.

Kober et al. (2013) attempted to make a strong link between the two research communities by providing a study in reinforced learning (Reinforcement Learning) for the behavior of a generation of robots. They discuss how contributions in the study of algorithms or representation models can achieve great improvements in both areas. In particular, the study focuses on the choice between different models and methods for the search of object-

ive policies and functions. Kollar and Roy (2008) addressed the problem of how a robot should make a plan to explore an unknown environment and collect data in order to maximize the accuracy of the resulting map. They pose exploration as an optimization problem using Reinforcement Learning to find trajectories that maintain the accuracy of the map. Zhang et al. (2015) proposed a new learning algorithm, called Geometric Reinforcement Learning (GRL), to plan trajectories in unmanned aerial vehicles.

Pomerleau (1989) presented a system called ALVINN (Autonomous Land Vehicle in a Neural Network). In this study, a neural network is trained under various conditions, suggesting the possibility of a novel autonomous navigation system capable of adapting to environmental conditions. Lu et al. (2016) designed a CNN with multiple layers to solve the problem of nano robot trajectory planning.

Recently, Li et al. (2017) and Xia et al. (2017) analyzed the loop closure detection problem using deep learning framework. In (Li et al., 2017), the authors solved loop closure detection and relative pose transformation using 2D LiDAR within an end-to-end Deep Learning framework. In (Xia et al., 2017), a comparison and analysis of several popular deep neural networks and traditional methods for loop closure detection is performed. The authors evaluated their performance on two open datasets in terms of accuracy and processing time and concluded that deep neural network is suitable for loop closure detection. Other authors as Ma et al. (2017) analyzed the semantic mapping problem and proposed a novel approach to object-class segmentation from multiple RGB-D views using deep learning.

Using reinforcement learning framework, Bruce et al. (2017) presented a method for learning to navigate, to a fixed goal and in a known environment, on a mobile robot. In Zhu et al. (2017) the objective was to show the use of reinforcement learning for navigation in indoor environments. A navigation framework for autonomous mobile robots in dynamic environments using a reinforcement learning algorithm is proposed in (Zhu et al., 2017).

After carrying out this review of the literature, it is worth noting that both algorithms in Deep Learning and in Reinforcement Learning are of great interest for studies to be applied to SLAM problems. Therefore, a study of the main concepts in both areas has been carried out. More information and details about formulation of deep learning and reinforcement learning can be found in (Bengio, 2009), (Deng et al., 2014), (Schmidhuber, 2015), (Kaelbling et al., 1996), (Kober et al., 2013) and (Wirth et al., 2017).

B.2 Deep Learning

Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence. See these course notes for a brief introduction to Machine Learning for AI and an introduction to Deep Learning algorithms. Specifically, Deep Learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound, and text. For more about deep learning algorithms, see for example:

Below, we summarize the main ideas about the concepts studied in neuron networks and their training, and how they can be used to carry out Deep Learning.

B.2.1 Motivation to study Deep Learning

One of the keys to advanced artificial intelligence is learning. It is increasingly common to ask machines to learn by themselves without the need to pre-program each of the situations that appear in the real world. Instead of doing that, the machines are required to be able to program themselves, that is, to learn from their own experience. The Machine Learning discipline deals with this challenge. More specifically, in computer science, machine learning (machine learning) is a branch of artificial intelligence whose objective is to develop programs capable of generalizing behaviors from unstructured information. Provided in the form of examples, it is, therefore, a process of knowledge induction.The discipline of machine learning is in full swing thanks to its wide range of applications.

At present, there are many advances and improvements of the most traditional algorithms in artificial intelligence, from the ensemble of classifiers (ensemble learning) to the Deep Learning, which is very fashionable nowadays due to its ability to get closer and closer to the human perceptive power. Deep Learning is the use of a set of algorithms to make abstract representations of information and facilitate machine learning. In this approach, logical structures are used that more closely resemble the organization of the nervous system of mammals, having layers of process units (artificial neurons) that specialize in detecting certain characteristics existing in the perceived objects. Artificial vision is one of the areas where Deep Learning provides a

considerable improvement compared to more traditional algorithms. Deep Learning represents a more intimate approach to the mode of operation of the human nervous system.

B.2.2 Introduction

An idea that has been around researchers for some time is to decompose problems into sub-problems at different levels of abstraction. The goal of deep learning or Deep Learning is to automatically discover those abstractions between low-level attributes and high-level concepts. The depth of the architecture refers to the number of levels. Most learning systems have shallow architectures. For decades, neural network researchers wanted to train deep multi-layered networks with little success. The real change came in 2006 with an article by Hinton and his colleagues at the University of Toronto (Hinton et al., 2006). Soon after, many other schemes were developed with the same general idea: to guide learning by levels using unsupervised learning at each level.

For a long time, many authors have tried to train deep multi-layer networks. The results of his research suggest that using gradient-based training is trapped in valleys of apparent local minimums, and this worsens with deeper architectures. In conclusion, it was obtained that better results could be obtained if the network was pre-trained with unsupervised algorithms, in one layer at a time. Almost all the schemes follow the idea of learning by layers in an unsupervised way, giving rise to a proper initialization of parameters for supervised learning. The output of a layer serves as an input to the upper layer.

The training can be done using some of the multiple variants of backpropagation (conjugate gradient, steepest descent, etc.). The problem of backpropagation with many layers is that by propagating the errors to the first layers they become very small and therefore ineffective. This causes the network to end up learning the average of all the training examples.

There are three classes of Deep Learning architectures:

- Generative architectures: in general an unsupervised training is done as pre-training, where you learn in a "greedy" way layer by layer in a "bottom-up" way. In this category are the energy-based models, which include the self-encoders and the deep Boltzmann machines (DBM).

- Discriminative architectures: Conditional Random Fields deep and convolutional neural networks (Convolutional Neural Network) or CNN.
- Hybrid Architectures: They use the two schemes.

B.2.3 Neural networks

Although there are several ways to implement Deep Learning, one of the most common is to use neural networks. A neural network is a mathematical modeling tool, a very simplified way, the functioning of neurons in the brain. A neural network is composed of units called neurons. Each neuron receives a series of inputs through interconnections and issues an output. This output is given by three functions:

- A propagation function (also known as an excitation function), which usually consists of the sum of each input multiplied by the weight of its interconnection (net value). If the weight is positive, the connection is called excitatory; if it is negative, it is called inhibitory.
- An activation function, which modifies the previous one. It may not exist, being in this case the output the same propagation function.
- A transfer function, which is applied to the value returned by the activation function. It is used to limit the output of the neuron and is usually given by the interpretation we want to give these outputs. Some of the most used are the sigmoid function (to obtain values in the interval (0,1)) and the hyperbolic tangent (to obtain values in the interval (-1,1)).

In other words, normally all the neurons in each layer have a connection to each neuron in the next layer. These connections are associated with a number, which is called weight. The main operation performed by the network of neurons is to multiply the values of a neuron by the weights of its outgoing connections. Each neuron in the next layer receives numbers from several incoming connections, and the first thing it does is add them all together.

Another operation performed by all the layers except the input layer, before continuing to multiply its values by the outgoing connections, is the activation function mentioned above. This function receives as input the sum

of all the numbers arriving through the incoming connections, transforms the value by means of a formula, and produces a new number. There are several options, but one of the most common functions is the sigmoid function. One of the objectives of the activation function is to keep the numbers produced by each neuron within a reasonable range (for example, real numbers between 0 and 1). The sigmoid function is one of the most used in networks of neurons because it is non-linear. This is very important, because if the activation function that we choose is linear, the network will be limited to solving linear problems (very simple).

A simple way to implement networks of neurons is to store the weights in matrices. It is easy to see that if you save the values of all the neurons of a layer in a vector, the product of the vector and the matrix of output weights, gives us the input values of each neuron in the next layer. Then the activation function that has been chosen is applied to each element of that second vector, and the process is repeated. Without going into detail, it is interesting to know that there are architectures of different neuron networks that are also sometimes used to implement Deep Learning. For example, networks of recurrent neurons do not have a layer structure, but allow arbitrary connections between all neurons, even creating cycles. This allows the concept of temporality to be incorporated into the network, and allows the network to have memory, because the numbers that we introduce at a given moment in the input neurons are transformed and continue to circulate through the network even after changing the input numbers by others different.

Another interesting architecture is the networks of convolutional neurons (Convolutional neural networks). In this case the concept of layers is maintained, but each neuron in a layer does not receive incoming connections from all the neurons in the previous layer, but only in some. This favors a neuron to specialize in a region of the list of numbers in the previous layer, and drastically reduces the number of weights and multiplications needed. It is usual for two consecutive neurons in an intermediate layer to specialize in overlapping regions of the anterior layer.

B.2.4 Learning neural networks

Assuming that the number of neurons that are needed at the entrance and exit has already been decided, it is still necessary to decide several things to have a network that works: how many hidden layers will be included; how

many neurons to put in each hidden layer; what concrete weights will be used in the connections between each pair of layers. Usually the first two points are decided by hand, and sometimes by trial and error. The more neurons you have in hidden layers, the more complex the network is, and you can solve more complex problems in turn. On the other hand, the more hidden neurons you have, the more it will cost to make all the products and sums.

It is important to mention that if a linear activation function is chosen, then it is not worth using hidden layers because it is possible to verify that the power of the network will be the same for many layers that you have. The power of the network only increases with the number of layers for non-linear activation functions, such as the sigmoid.

The third point can be resolved automatically, through a process called training. If the weights of the intermediate layers are initialized with random weights, a deep network does not learn well. This can be remedied if initial (non-random) weights are used that approximate the final or pre-training solution. One way is by training pairs of successive layers such as a restricted Boltzmann machine (RBM) or Auto-Encoders and then using backpropagation to fine-tune the weights. To train a network of neurons, you need to first collect some examples of inputs and the output that you want for each example. This training process is known as supervised learning, because the system needs a supervisor to explain what it has to do (through examples of inputs and outputs).

Backpropagation

The backpropagation is a supervised learning algorithm used to train artificial neural networks. The algorithm employs a propagation-adaptation cycle of two phases. Once a pattern has been applied to the input of the network as a stimulus, it propagates from the first layer through the upper layers of the network, to generate an output. The output signal is compared with the desired output and an error signal is calculated for each of the outputs.

The error outputs are propagated backwards, starting from the output layer, to all the neurons in the hidden layer that contribute directly to the output. However, the neurons of the hidden layer only receive a fraction of the total signal of the error, based approximately on the relative contribution that each neuron has contributed to the original output. This process is repeated, layer by layer, until all the neurons in the network have received

an error signal that describes their relative contribution to the total error.

The importance of this process is that, as the network is trained, the neurons of the intermediate layers organize themselves in such a way that the different neurons learn to recognize different characteristics of the total input space. After training, when presented with an arbitrary input pattern that contains noise or is incomplete, the neurons in the hidden layer of the network will respond with an active output if the new input contains a pattern that resembles that feature that individual neurons have learned to recognize during their training. In other words, once the set of examples has been configured, it is easy to evaluate the network with each example and check the error between the desired output and the output that the network is producing. To calculate the error, in each neuron of the output layer, the produced and expected value is subtracted. Knowing what the error is for a given example, you can try to correct it.

The idea to correct errors is to look for culprits. Among the output neurons it is easy to know who are the culprits: it is known exactly how much error occurs in each neuron, as explained before. But each one of those neurons is connected to the previous neurons by means of some weights. You can use those weights to determine how much the neurons before the error contribute, simply by propagating the errors backwards in the same way that values are propagated forward, multiplying and adding. In this way, we can know how much each neuron in the entire network contributes to the error.

Knowing how much each neuron contributes to the error, p can try to update the weights to reduce that error. First you need to find out how the changes to the weights affect the error. That is, you need to determine the speed with which the error changes with respect to the weights. Knowing this, the next step is to change the weights to just the right amount so that the error is reduced as fast as possible. The speed with which the error changes with respect to the weights is calculated with partial derivatives, and implies that it must be able to derive the activation function.

When using an existing library of neuron networks, the most usual thing is that when training you get information about how the process goes in the form of mean square error (MSE). The idea is that for each example, the network evaluates the error in all its output neurons, elevates each one of those numbers squared, and finally calculates the average. By raising each error squared, the errors are always positive, so the errors of some neurons do not cancel out those of others. To decide which is the minimum error from which you can stop training, a simple way is to decide the maximum

error that is willing to accept for each output neuron, calculate the squares and then the average.

A serious problem with backward propagation algorithms is that the error dilutes exponentially as it traverses layers on its way to the beginning of the network. This is a problem because in a very deep network (with many hidden layers), only the last layers are trained, while the former hardly suffer any changes. That is why it sometimes makes up for using networks with few hidden layers that contain many neurons, instead of networks with many hidden layers that contain few neurons. This problem was corrected thanks to Deep Learning.

B.2.5 Autoencoders

Another tool commonly used to implement Deep Learning is the autoencoders. Normally they are implemented as networks of neurons with three layers (only one hidden layer). A self-encoder is an artificial neural network used to perform an unsupervised learning system. The goal of a self-encoder is to learn a representation for a set of data, usually to achieve a reduction in dimensionality. A self-coder learns to output exactly the same information it receives at the entrance. Therefore, the input and output layers must always have the same number of neurons. The idea is the following:

- Use an input layer, a hidden layer with a minor (although it can be larger) number of nodes and train to get an output layer equal to the input.
- Use the information learned in the hidden layer (abstraction) as the new input layer for the next level.
- Continue with the levels determined by the user.
- Finish, using the last hidden layer as input to a classifier.

Stacked Autocoders

A single auto-encoder can find fundamental characteristics in the input information, the most primitive and simple features that can be extracted from that information. However, for machines to detect more complex concepts

more power is needed. The idea of Deep Learning using stacked auto-coders is to use several coders, and train them one by one, using each coder trained to train the next. A deep network created in this way has two very important characteristics:

- Learn without supervision, you only need data and find alone frequent features with which to tag the data.
- The training of very deep networks is possible. As discussed in the section of backward propagation, when training very deep networks there was a problem since the error is diluted and the first layers are almost not trained. In this way that problem is solved.

Convolution and pooling

In particular, in order to solve the identification of the same displaced characteristic, Deep Learning uses a technique known as convolution, which is based on how neurons are actually structured in our visual system. For example, training an auto-coder with image patches is possible, and then moving the coder over the entire image, as if it were a scanner, looking for features. This process transforms the pixel array into another array of characteristics (lists of numbers) produced by the encoder. It is essentially the same image, but each pixel is much richer in information, and contains information on the region in which the pixel is located, not just the isolated pixel.

The next step, called pooling, consists of grouping the characteristics (lists of numbers) of several contiguous coordinates of the image, with some function of grouping (the average, or the maximum). This stage reduces the resolution of the image. Then you can continue doing convolution and pooling until there is a 1×1 pixel image with a lot of information, or you can include intermediate auto-encoders that process the data to look for higher level features.

B.2.6 Supervised learning and Deep Learning

Since all composite encoders are a conventional network of neurons, the user can continue with conventional supervised training (backward propagation). When using backward propagation, there is the problem of the error that is

diluted and, therefore, the first layers will suffer less training than the last ones. But precisely the first layers are those that need less changes, because they focus on fundamental characteristics (of very low level) necessary to detect any complex object. The closer a layer of the exit is, the more interested it is in changes in supervised training and the focus is on detecting the complex objects of interest.

Therefore, a very common technique in Deep Learning is to train unsupervised a stack of auto-coders, and then compose the coders and continue with a supervised training. That is, supervised training, instead of starting with weights at random, starts with useful weights, especially for the first layers. It is usually much easier to find untagged information (for which we do not have a "desired output") than tagged information. And suddenly, all that untagged information is useful for the unsupervised training phase.

B.2.7 Alternatives

Auto-coders are not the only mechanism to perform Deep Learning. There are other alternatives, such as the Deep Belief Networks. These also consist of a series of layers trained one by one, from the most specific to the most generic, but each layer instead of a self-encoder uses a Restricted Boltzmann Machine.

B.3 Reinforcement Learning

Reinforcement Learning is generally used to solve Markov decision problems (MDP). The theory of reinforced learning is based on dynamic programming and artificial intelligence. Details about mathematical formulation in MDP are shown in Appendix A.

In recent years, Reinforcement Learning, also called as approximate dynamic programming, has taken an interest as a powerful tool to solve complex decision problems in control theory. Although the seminal research work in this area was done by the artificial intelligence community, it has recently attracted attention in control theory. The power of Reinforcement Learning becomes more obvious in dynamic optimization problems, particularly in Markov decision problems and their variants. For many years, the problem of dimensionality and modeling as problems of classical dynamics have not

been effective in large-scale MDPs. Reinforcement Learning allows solving these problems that had been considered intractable. The success of Reinforcement Learning is due to its strong mathematical basis on the principles of dynamic programming, Monte Carlo simulation, approach function and artificial intelligence.

Reinforcement Learning is a problem that deals with learning from interaction, that is, how to behave to achieve an objective. The agent, for example a robot, and its environment interact on a discrete sequence of steps in time. This specification of the problem defines a particular task: the actions are the choices made by the agent; states are the basis for making choices: and rewards are the basis for the evaluation of elections. The environment of the agent is not controlled by him and may or may not be completely known. A policy is a stochastic rule by which the agent selects actions, as a function of states. The objective of the agent is to maximize the amount of reward he receives over time.

The value functions of the policies assign to each state, or pair state-action, the expected reward from that state, given the policy used by the agent. The optimal value function assigns to each state, or pair state-action, the highest expected reward achieved by any policy. A policy whose value function is optimal is an optimal policy.

A problem of Reinforcement Learning can be posed in a great variety of different ways depending on the hypothesis about the level of initial knowledge available to the agent. In problems of complete knowledge, the agent has a complete and accurate model of the dynamics of the environment. If the environment is a Markov decision problem (MDP), then such a model consists of one-step transition probabilities and expected reward for all states and their allowed actions. In incomplete knowledge problems, a complete and perfect model of the environment is not available.

More details about this topic can be found in the book: Simulation-based optimization: 'Parametric Optimization techniques and reinforcement learning' (Zilinskas, 2005). Other book with detail on Reinforcement Learning is 'Reinforcement Learning: An Introduction' (Sutton and Barto, 1998) and a recent tutorial on this topic is 'Reinforcement learning: A tutorial survey and recent advances' (Gosavi, 2009).

Bibliography

Aberdeen, D. and Baxter, J. (2002). Scaling internal-state policy-gradient methods for POMDPs. In *Proceedings of the International Conference on Machine Learning*, pages 3–10.

Agrawal, P., Carreira, J., and Malik, J. (2015). Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45.

Altman, E. (1999). *Constrained Markov decision processes*, volume 7. CRC Press.

Amato, C., Bonet, B., and Zilberstein, S. (2010). Finite-state controllers based on mealy machines for centralized and decentralized POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1052–1058.

Astrom, K. J. (1965). Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205.

Bai, H., Hsu, D., and Lee, W. S. (2014). Integrated perception and planning in the continuous space: A POMDP approach. *The International Journal of Robotics Research*, 33(9):1288–1302.

Bajcsy, R. (1988). Active perception. *Proceedings of the IEEE*, 76(8):966–1005.

Bar-Itzhack, I. Y. (2000). New method for extracting the quaternion from a rotation matrix. *Journal of Guidance, Control, and Dynamics*, 23(6):1085–1087.

Barfoot, T. D. and Furgale, P. T. (2014). Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics*, 30(3):679–693.

Barratt, S. (2017). Active Robotic Mapping through Deep Reinforcement Learning. *arXiv preprint arXiv:1712.10069*.

Basu, S., Essa, I., and Pentland, A. (1996). Motion regularization for model-based head tracking. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume 3, pages 611–616.

Bellman, R. (1957). *Dynamic Programming*. P (Rand Corporation). Princeton University Press.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127.

Bernstein, D. S. (2009). *Matrix mathematics: theory, facts, and formulas*. Princeton University Press.

Blackmore, D. and Leu, M. C. (1992). Analysis of swept volume via Lie groups and differential equations. *The International Journal of Robotics Research*, 11(6):516–537.

Blanco, J. L. (2010). A tutorial on SE(3) transformation parameterizations and on-manifold optimization. Technical report, University of Malaga.

Blanco, J. L., Fernandez-Madrigal, J.-A., and González, J. (2008). A novel measure of uncertainty for mobile robot slam with rao—blackwellized particle filters. *The International Journal of Robotics Research*, 27(1):73–89.

Bolle, R. M. and Cooper, D. B. (1986). On optimally combining pieces of information with application to estimating 3D complex-object position from range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (5):619–638.

Borghi, G. and Caglioti, V. (1998). Minimum uncertainty explorations in the self-localization of mobile robots. *IEEE Transactions on Robotics and Automation*, 14(6):902–911.

Bourgault, F., Makarenko, A. A., Williams, S. B., Grocholsky, B., and Durrant-Whyte, H. F. (2002). Information based adaptive robotic exploration. In *Procceding of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 540–545.

Braziunas, D. (2003). POMDP solution methods. Technical report, University of Toronto.

Bregler, C. and Malik, J. (1998). Tracking people with twists and exponential maps. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8–15.

Brooks, R. A. (1982). Symbolic error analysis and robot planning. *The International Journal of Robotics Research*, 1(4):29–78.

Brooks, R. A. (1984). Aspects of mobile robot visual map making. *Robotics Research*, 2:369–375.

Bruce, J., Sünderhauf, N., Mirowski, P., Hadsell, R., and Milford, M. (2017). One-shot reinforcement learning for robot navigation with interactive replay. *arXiv preprint arXiv:1711.10137*.

Burgard, W., Fox, D., and Thrun, S. (1997). Active mobile robot localization by entropy minimization. In *Proceedings of the Second Euromicro workshop on Advanced Mobile Robots*, pages 155–162.

Burgard, W., Moors, M., Stachniss, C., and Schneider, F. E. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386.

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

Carlone, L., Du, J., Ng, M. K., Bona, B., and Indri, M. (2010). An application of Kullback-Leibler divergence to active slam and exploration with particle filters. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 287–293.

Carlone, L., Du, J., Ng, M. K., Bona, B., and Indri, M. (2014). Active SLAM and exploration with particle filters using Kullback-Leibler divergence. *Journal of Intelligent & Robotic Systems*, 75(2):291–311.

Carrillo, H., Dames, P., Kumar, V., and Castellanos, J. A. (2018). Autonomous robotic exploration using a utility function based on Rényi's general theory of entropy. *Autonomous Robots*, 42(2):235–256.

Carrillo, H., Reid, I., Castellanos, J., et al. (2012). On the comparison of uncertainty criteria for active slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2080–2087.

Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. (1994). Acting optimally in partially observable stochastic domains. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 94, pages 1023–1028.

Castellanos, J. A., Montiel, J., Neira, J., and Tardós, J. D. (1999). The SPmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–952.

Charrow, B., Liu, S., Kumar, V., and Michael, N. (2015). Information-theoretic mapping using cauchy-schwarz quadratic mutual information. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4791–4798.

Chatila, R. and Laumond, J.-P. (1985). Position referencing and consistent world modeling for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 138–145.

Chen, Z., Lam, O., Jacobson, A., and Milford, M. (2014). Convolutional neural network-based place recognition. *arXiv preprint arXiv:1411.1509*.

Chernoff, H. (1953). Locally optimal designs for estimating parameters. *The Annals of Mathematical Statistics*, pages 586–602.

Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

Deng, L., Yu, D., et al. (2014). Deep learning: methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387.

Desai, R. S., Volz, R., et al. (1989). Identification and verification of termination conditions in fine motion in presence of sensor errors and geometric uncertainties. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 800–807.

Donald, B. R. (1986). Robot motion planning with uncertainty in the geometric models of the robot and environment: A formal framework for error detection and recovery. In *Proceedings of the IEEE International Conference on Robotics and Automation. Proceedings*, volume 3, pages 1588–1593.

Donald, B. R. (1990). Planning multi-step error detection and recovery strategies. *The International Journal of Robotics Research*, 9(1):3–60.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Magazine*, 13(2):99–108.

Durrant-Whyte, H. F. (1987). Consistent integration and propagation of disparate sensor observations. *The International Journal of Robotics Research*, 6(3):3–24.

Durrant-Whyte, H. F. (1988). Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, 4(1):23–31.

Ehrenfeld, S. (1955). On the efficiency of experimental designs. *The Annals of Mathematical Statistics*, 26(2):247–255.

Erdmann, M. (1986). Using backprojections for fine motion planning with uncertainty. *The International Journal of Robotics Research*, 5(1):19–45.

Erdmann, M. A. (1984). On motion planning with uncertainty. *AI Technical Reports (1964 - 2004)*.

Fairfield, N. and Wettergreen, D. (2010). Active SLAM and loop prediction with the segmented map using simplified models. In *Field and Service Robotics*, pages 173–182. Springer.

Feder, H. J. S., Leonard, J. J., and Smith, C. M. (1999). Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18(7):650–668.

Fox, D., Burgard, W., and Thrun, S. (1998). Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3-4):195–207.

Frese, U. (2006). A discussion of simultaneous localization and mapping. *Autonomous Robots*, 20(1):25–42.

Gallier, J. (2011). *Geometric methods and applications: for computer science and engineering*, volume 38. Springer Science & Business Media.

Gosavi, A. (2009). Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, 21(2):178–192.

Grimson, W. E. L. and Lozano-Perez, T. (1984). Model-based recognition and localization from sparse range or tactile data. *The International Journal of Robotics Research*, 3(3):3–35.

Hall, B. (2015). *Lie groups, Lie algebras, and representations: an elementary introduction*, volume 222. Springer.

Hansen, E. A. (1997). An improved policy iteratioll algorithm for partially observable MDPs. In *Proceedings of the 10th International Conference on Neural Information Processing Systems*, pages 1015–1021. MIT Press.

Hansen, E. A. (1998a). *Finite-memory control of partially observable systems*. PhD thesis, University of Massachusetts Amherst.

Hansen, E. A. (1998b). Solving POMDPs by searching in policy space. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 211–219. Morgan Kaufmann Publishers Inc.

Hertzberg, C., Wagner, R., Frese, U., and Schröder, L. (2013). Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 14(1):57–77.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hou, Y., Zhang, H., and Zhou, S. (2015). Convolutional neural network-based image representation for visual loop closure detection. In *Proceedings of the IEEE International Conference on Information and Automation (ICRA)*, pages 2238–2245.

Howard, R. A. (1960). *Dynamic Programming and Markov processes*. MIT Press.

Indelman, V., Carlone, L., and Dellaert, F. (2014). Planning under uncertainty in the continuous domain: a generalized belief space approach. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6763–6770.

Isom, J. D., Meyn, S. P., and Braatz, R. D. (2008). Piecewise linear dynamic programming for constrained pomdps. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 291–296.

Jadidi, M. G., Miro, J. V., Valencia, R., and Andrade-Cetto, J. (2014). Exploration on continuous gaussian process frontier maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6077–6082.

Jefferies, M. E. and Yeap, W.-K. (2008). *Robotics and cognitive approaches to spatial mapping*. Springer.

Jensfelt, P. and Kristensen, S. (2001). Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation*, 17(5):748–760.

Juan, J. and Paul, R. P. (1986). Automatic programming of fine-motion for assembly. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 1582–1587.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Kelly, A. (2004). Linearized error propagation in odometry. *The International Journal of Robotics Research*, 23(2):179–218.

Kendall, A. and Cipolla, R. (2015). Modelling uncertainty in deep learning for camera relocalization. *arXiv preprint arXiv:1509.05909*.

Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2938–2946.

Kiefer, J. (1974). General equivalence theory for optimum designs (approximate theory). *The annals of Statistics*, pages 849–879.

Kim, A. and Eustice, R. M. (2013). Perception-driven navigation: Active visual SLAM for robotic area coverage. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3196–3203.

Kim, A. and Eustice, R. M. (2015). Active visual SLAM for robotic area coverage: Theory and experiment. *The International Journal of Robotics Research*, 34(4-5):457–475.

Kim, D., Lee, J., Kim, K.-E., and Poupart, P. (2011). Point-based value iteration for constrained POMDPs. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

Kollar, T. and Roy, N. (2006). Using reinforcement learning to improve exploration trajectories for error minimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3338–3343.

Kollar, T. and Roy, N. (2008). Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research*, 27(2):175–196.

Konda, K. and Memisevic, R. (2015). Learning visual odometry with a convolutional network. In *Proceedings of the International Conference on Computer Vision Theory and Applications*, pages 486–490.

Kreucher, C., Hero, A. O., and Kastella, K. (2005). A comparison of task driven and information driven sensor management for target tracking. In *Proceedings of the IEEE Conference on Decision and Control and European Control Conference*, pages 4004–4009.

Kurniawati, H., Hsu, D., and Lee, W. S. (2008). SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Proceedings of the Robotics: Science and Systems*, volume 2008. Zurich, Switzerland.

Kyatkin, A. and Chirikjian, G. (1998). Applications of noncommutative harmonic analysis in robotics. *Courses and lectures / International Centre for Mechanical Sciences*, pages 119–126.

Lauri, M. and Ritala, R. (2016). Planning for robotic exploration based on forward simulation. *Robotics and Autonomous Systems*, 83:15–31.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge University Press.

Lefebvre, T., Bruyninckx, H., and De Schutter, J. (2005). Task planning with active sensing for autonomous compliant motion. *The International Journal of Robotics Research*, 24(1):61–81.

Leung, C., Huang, S., and Dissanayake, G. (2006). Active SLAM using model predictive control and attractor based exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5031.

Leung, C., Huang, S., and Dissanayake, G. (2008). Active SLAM in structured environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1898–1903.

Li, J., Zhan, H., Chen, B. M., Reid, I., and Lee, G. H. (2017). Deep learning for 2D scan matching and loop closure. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 763–768.

Lozano-Perez, T., Mason, M. T., and Taylor, R. H. (1984). Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–24.

Lu, Y., Yi, S., Liu, Y., and Ji, Y. (2016). A novel path planning method for biomimetic robot based on deep learning. *Assembly Automation*, 36(2):186–191.

Ma, L., Stückler, J., Kerl, C., and Cremers, D. (2017). Multi-view deep learning for consistent semantic mapping with RGB-D cameras. *arXiv preprint arXiv:1703.08866*.

Maffei, R., Jorge, V. A., Prestes, E., and Kolberg, M. (2014). Integrated exploration using time-based potential rails. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3694–3699.

Makarenko, A. A., Williams, S. B., Bourgault, F., and Durrant-Whyte, H. F. (2002). An experiment in integrated exploration. In *Procceding of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 534–539.

Martinez-Cantin, R., de Freitas, N., Brochu, E., Castellanos, J., and Doucet, A. (2009). A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2):93–103.

Massios, N. A., Fisher, R. B., et al. (1998). A best next view selection algorithm incorporating a quality criterion. In *Proceedings of the British Machine Vision Conference*, pages 780–789.

Maurović, I., Seder, M., Lenac, K., and Petrović, I. (2017). Path planning for active SLAM based on the D* algorithm with negative edge weights. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Mazon, I. and Alami, R. (1989). Representation and propagation of positioning uncertainties through manipulation robot programs-integration into a task-level programming system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 646–652.

Meger, D., Rekleitis, I., and Dudek, G. (2008). Heuristic search planning to reduce exploration uncertainty. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3392–3399.

Meuleau, N., Kim, K.-E., Kaelbling, L. P., and Cassandra, A. R. (1999a). Solving POMDPs by searching the space of finite policies. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 417–426. Morgan Kaufmann Publishers Inc.

Meuleau, N., Peshkin, L., Kim, K.-E., and Kaelbling, L. P. (1999b). Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 427–436. Morgan Kaufmann Publishers Inc.

Mihaylova, L., Lefebvre, T., Bruyninckx, H., Gadeyne, K., and De Schutter, J. (2003). A comparison of decision making criteria and optimization methods for active robotic sensing. In *Numerical Methods and Applications*, pages 316–324. Springer.

Morari, M. and Lee, J. H. (1999). Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682.

Mu, B., Giamou, M., Paull, L., Agha-mohammadi, A.-a., Leonard, J., and How, J. (2016). Information-based active SLAM via topological feature graphs. In *Proceedings of the IEEE 55th Conference on Decision and Control (CDC)*, pages 5583–5590.

Murray, R. M., Li, Z., Sastry, S. S., and Sastry, S. S. (1994). *A mathematical introduction to robotic manipulation*. CRC press.

Natarajan, B. (1988). The complexity of fine motion planning. *The International Journal of Robotics Research*, 7(2):36–42.

Nocedal, J. and Wright, S. J. (2006). *Numerical optimization 2nd*. Springer Science & Business Media.

Ong, S. C., Png, S. W., Hsu, D., and Lee, W. S. (2010). Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8):1053–1068.

Park, F. C. and Brockett, R. W. (1994). Kinematic dexterity of robotic mechanisms. *The International Journal of Robotics Research*, 13(1):1–15.

Paul, R. P. (1981). *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. MIT Press series in artificial intelligence.

Pázman, A. (1986). *Foundations of optimum experimental design*, volume 14. Springer.

Pertin-Troccaz, J. and Puget, P. (1988). Delaying with uncertainty in robot planning using program proving techniques. In *Proceedings of the 4th international symposium on Robotics Research*, pages 455–446. MIT Press.

Pineau, J., Gordon, G., and Thrun, S. (2006). Anytime point-based approximations for large pomdps. *Journal of Artificial Intelligence Research*, 27:335–380.

Pito, R. (1999). A solution to the next best view problem for automated surface acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1016–1030.

Pomerleau, D. A. (1989). Alvinn: An autonomous land vehicle in a neural network. Technical report, DTIC Document.

Poupart, P. and Boutilier, C. (2003). Bounded finite state controllers. In *NIPS*, pages 823–830.

Poupart, P., Malhotra, A., Pei, P., Kim, K.-E., Goh, B., and Bowling, M. (2015). Approximate linear programming for constrained partially observable markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3342–3348.

Pukelsheim, F. (2006). *Optimal Design of Experiments*. SIAM.

Puterman, M. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics.

Rawlings, J. B. (2000). Tutorial overview of model predictive control. *IEEE Control Systems*, 20(3):38–52.

Reed, M. K., Allen, P. K., and Stamos, I. (1997). Automated model acquisition from range images with view planning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 72–77.

Rényi, A. (1961). On measures of entropy and information. Technical report, Hungarian academy of science, Budapest, Hungary.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Seber, G. A. (2008). *A matrix handbook for statisticians*, volume 15. John Wiley & Sons.

Seiler, K. M., Kurniawati, H., and Singh, S. P. (2015). An online and approximate solver for POMDPs with continuous action space. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2290–2297.

Selig, J. (1996). *Geometrical Methods in Robotics*. Springer-Verlag New York, Inc.

Shannon, C. E. and Weaver, W. (1949). *The Mathematical Theory of Communication*. University of Illinois Press.

Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.

Sim, R. (2005a). Stabilizing information-driven exploration for bearings-only SLAM using range gating. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3396–3401.

Sim, R. (2005b). Stable exploration for bearings-only SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2411–2416.

Sim, R. and Roy, N. (2005). Global a-optimal robot exploration in SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 661–666.

Smallwood, R. D. and Sondik, E. J. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088.

Smith, P., Drummond, T., and Roussopoulos, K. (2003). Computing map trajectories by representing, propagating and combining pdfs over groups. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1275–1282.

Smith, R., Self, M., and Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer.

Smith, R. C. and Cheeseman, P. (1986). On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68.

Smith, T. and Simmons, R. (2004). Heuristic search value iteration for POMDPs. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 520–527. AUAI Press.

Sondik, E. J. (1971). *The Optimal Control of Partially Observable Markov Decision Processes*. PhD thesis, Stanford University.

Sondik, E. J. (1978). The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304.

Souza, J. R., Marchant, R., Ott, L., Wolf, D. F., and Ramos, F. (2014). Bayesian optimisation for active perception and smooth navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4081–4087.

Stachniss, C., Grisetti, G., and Burgard, W. (2005). Information Gain-based Exploration using Rao-Blackwellized Particle Filters. In *Proceedings of the Robotics: Science and Systems*, volume 2, pages 65–72.

Stachniss, C., Hahnel, D., and Burgard, W. (2004). Exploration with active loop-closing for FastSLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1505–1510.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3310–3317.

Su, S. and Lee, C. (1991). Uncertainty manipulation and propagation and verification of applicability of actions in assembly tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2471–2476.

Su, S.-F. and Lee, C. G. (1992). Manipulation and propagation of uncertainty and verification of applicability of actions in assembly tasks. *IEEE Transactions on Systems, Man and Cybernetics*, 22(6):1376–1389.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Taylor, R. and Rajan, V. (1988). The efficient computation of uncertainty spaces for sensor-based robot planning. *Transformation*, 7:1.

Taylor, R. H. (1976). *The Synthesis of Manipulator Control Programs from Task-level Specifications*. PhD thesis, Stanford University.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Valencia Carreño, R., Valls Miró, J., Dissanayake, G., and Andrade-Cetto, J. (2012). Active pose SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1885–1891.

Vallvé, J. and Andrade-Cetto, J. (2014). Dense entropy decrease estimation for mobile robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6083–6089.

Van Den Berg, J., Patil, S., and Alterovitz, R. (2012). Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278.

Varadarajan, V. S. (2013). *Lie groups, Lie algebras, and their representations*, volume 102. Springer Science & Business Media.

Vidal-Calleja, T., Davison, A. J., Andrade-Cetto, J., and Murray, D. W. (2006). Active control for single camera SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1930–1936.

Wald, A. (1943). On the efficient design of statistical investigations. *The annals of mathematical statistics*, 14(2):134–140.

Wang, Y. and Chirikjian, G. S. (2004). Workspace generation of hyper-redundant manipulators as a diffusion process on se (n). *IEEE Transactions on Robotics and Automation*, 20(3):399–408.

Wang, Y. and Chirikjian, G. S. (2006). Error propagation on the euclidean group with applications to manipulator kinematics. *IEEE Transactions on Robotics*, 22(4):591–602.

Wang, Y. and Chirikjian, G. S. (2008). Nonparametric second-order theory of error propagation on motion groups. *The International Journal of Robotics Research*, 27(11-12):1258–1273.

White, C. and Harrington, D. P. (1980). Application of Jensen's inequality to adaptive suboptimal design. *Journal of Optimization Theory and Applications*, 32(1):89–99.

White, C. C. (1991). A survey of solution techniques for the partially observed markov decision process. *Annals of Operations Research*, 32(1):215–230.

Wirth, C., Akrour, R., Neumann, G., Fürnkranz, J., et al. (2017). A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46.

Xia, Y., Li, J., Qi, L., Yu, H., and Dong, J. (2017). An evaluation of deep learning in loop closure detection for visual SLAM. In *, 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 85–91.

Xiao, J., Volz, R., et al. (1989). On replanning for assembly tasks using robots in the presence of uncertainties. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 638–645.

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151.

Zhang, B., Mao, Z., Liu, W., and Liu, J. (2015). Geometric reinforcement learning for path planning of uavs. *Journal of Intelligent & Robotic Systems*, 77(2):391–409.

Zhang, N. L. and Zhang, W. (2001). Speeding up the convergence of value iteration in partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 14:29–51.

Zhang, X., Wu, B., and Lin, H. (2017). Supervisor synthesis of POMDP based on automata learning. *arXiv preprint arXiv:1703.08262*.

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3357–3364.

Zilinskas, A. (2005). *Simulation-based optimization: Parametric optimization techniques and reinforcement learning*. Springer US.