TESIS DE LA UNIVERSIDAD
DE ZARAGOZA

2018 77

Leonardo Javier Fermín León

Exploiting graph structure in Active SLAM

Departamento

Informática e Ingeniería de Sistemas

Director/es

CASTELLANOS GÓMEZ, JOSÉ ÁNGEL

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

EXPLOITING GRAPH STRUCTURE IN ACTIVE SLAM

Autor

Leonardo Javier Fermín León

Director/es

CASTELLANOS GÓMEZ, JOSÉ ÁNGEL

UNIVERSIDAD DE ZARAGOZA
Informática e Ingeniería de Sistemas

2018

Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Exploiting graph structure in Active SLAM

Leonardo Javier Fermín León

Ph.D. Dissertation

Advisor: José Ángel Castellanos

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza, Spain

April 2018

To Massy and Leo,
you are the light of my life

Acknowledgments

Foremost i want to thank God, without him there would be nothing.

I want to thank my wife Massiel and my son Leonardo for supporting me all of these years, they are the reason why i keep working and the reason I am not so crazy. I had to use some of our family time for this work and I only hope it is worthwhile.

I want to thank to my advisor José Ángel Castellanos for giving me the opportunity to be part of one of the most important robotics group in Spain and one of the more respected in the SLAM community worldwide. Without his experience and guidance this work would not be possible.

Thanks to the professor José Neira and the fruitful and frequent discussions. These discussions have seen the whole process from an idea into the full articles produced during this thesis. Even when many ideas have not yet become a publication I am sure the resulting insights will guide my next projects.

Along these years I have met many colleagues in the laboratory which make this time as PhD student memorable, they contributed directly or indirectly to this thesis, María Luisa Rodriguez, Rosario Araguez, Alejandro Perez-Yus, Jesús Bermúdez, Daniel Gutierrez, Alejo Concha, Raul Mur, Estibaliz Fraca, Yassir Latif, Henry Carrillo, Ana Cambra, Mayte Lazaro, Luis Riazuelo, Marta Salas, Sara Ruiz and the rest of the members of "Laboratorio de Becarios".

Research reported in this thesis was supported by the "Comisión Interministerial de Ciencia y Tecnología" (CICYT) of the "Ministerio de Economía y Competitividad" (MINECO) of the Government of Spain under project *Exploración Robótica Activa en Entornos 3D Densos* (DPI2015-68905-P). Also with funding from "Grupo DGA T04" and finally the funding of the "Fundación Carolina" for the PhD scholarship (Convocatoria C.2013).

Resumen

Aplicando análisis provenientes de la teoría de grafos, la teoría espectral de grafos, la exploración de grafos en línea, generamos un sistema de SLAM activo que incluye la planificación de rutas bajo incertidumbre, extracción de grafos topológicos de entornos y SLAM activo óptimo.

En la planificación de trayectorias bajo incertidumbre, incluimos el análisis de la probabilidad de asociación correcta de datos. Reconociendo la naturaleza estocástica de la incertidumbre, demostramos que planificar para minimizar su valor esperado es más fiable que los actuales algoritmos de planificación de trayectorias con incertidumbre.

Considerando el entorno como un conjunto de regiones convexas conectadas podemos tratar la exploración robótica como una exploración de grafos en línea. Se garantiza una cobertura total si el robot visita cada región. La mayoría de los métodos para segmentar el entorno están basados en píxeles y no garantizan que las regiones resultantes sean convexas, además pocos son algoritmos incrementales. En base a esto, modificamos un algoritmo basado en contornos en el que el entorno se representa como un conjunto de polígonos que debe segmentarse en un conjunto de polígonos pseudo convexos. El resultado es un algoritmo de segmentación que produjo regiones pseudo-convexas, robustas al ruido, estables y que obtienen un gran rendimiento en los conjuntos de datos de pruebas.

La calidad de un algoritmo se puede medir en términos de cuan cercano al óptimo est'a su rendimiento. Con esta motivación definimos la esencia de la tarea de exploración en SLAM activo donde las únicas variables son la distancia recorrida y la calidad de la reconstrucción. Restringiendo el dominio al grafo que representa el entorno y probando la relación entre la matriz asociada a la exploración y la asociada al grafo subyacente, podemos calcular la ruta de exploración óptima.

A diferencia de la mayoría de la literatura en SLAM activo, proponemos que la heurística para la exploración de grafos consiste en atravesar cada arco una vez. Demostramos que el tipo de grafos resultantes tiene un gran rendimiento con respecto a la trayectoria óptima, con resultados superiores al 97 % del óptimo en algunas medidas de calidad.

El algoritmo de SLAM activo *TIGRE* integra el algoritmo de extracción de grafos propuesto con nuestra versión del algoritmo de exploración incremental que atraviesa cada arco una vez. Nuestro algoritmo se basa en una modificación del algoritmo clásico de Tarry para la búsqueda en laberintos que logra el límite inferior en la aproximación para un algoritmo incremental. Probamos nuestro sistema incremental en un escenario de exploración típico y demostramos que logra un rendimiento similar a los métodos fuera de línea y también demostramos que incluso el método óptimo que visita todos los nodos calculado fuera de línea tiene un peor rendimiento que el nuestro.

Abstract

Applying analysis drawn from graph theory, spectral graph theory, on-line graph exploration we generate a pipeline for active SLAM that includes Path Planning under uncertainty, Topological Graph Extraction and Optimal Active SLAM.

In path planning under uncertainty we include the analysis of the probability of correct data association. Recognizing the stochastic nature of the uncertainty we demonstrate that planning to minimize its expected value is more reliable that current path planning under uncertainty algorithms.

Considering the environment as a set connected convex regions we can treat robotic exploration as an on-line graph exploration. Full coverage is guaranteed if the robot visit every region. Most of the methods to segment the environment are pixel based and do not guarantee that resulting regions are convex and few of them are incremental algorithms. Based on this we modify one contour-based algorithm in which the environment is represented as a set of polygons that needs to be segmented into a set of pseudo convex polygons. The result was a segmentation algorithm that produced pseudo-convex regions, robust to noise, stable and that obtained a great performance in public datasets.

The quality of one algorithm can be measured in terms of how close to the optimal is its performance. With this motivation we define the quintessential exploration task in active SLAM where the only variables are the traversed distance and the quality of the reconstruction. Restricting the domain to the graph that represents the environment and proving the relation between matrix associated to the exploration and the one associated to the underlying graph we are able to calculate the optimal exploration path.

Unlike most of the literature in Active SLAM we propose that the heuristic for graph exploration consist in traversing every edge once. We prove that this type of graphs resulted in great performance with respect to the optimal path, with results over the 97% of the optimal in some quality measures.

The Active SLAM algorithm *TIGRE* integrates the proposed graph extraction algorithm with our version of the incremental exploration algorithm that traverses every edge once. Our algorithm is based on a modification of the classic Tarry's algorithm for maze-searching that achieves the lower bound in approximation for

an incremental algorithm. We test our incremental system one typical exploration scenario and show that it achieves similar performance than off-line methods and also demonstrate that even the optimal off-line methods that visits every node have worst performance than ours.

Contents

Chapter 1

Introduction

Robotic arms have demonstrated their value boosting the productivity of many manufacturing industries. Unlike their predecessor the mobile robots can move freely thus increasing its potential applications but also increasing its complexity.

Mobile robots are currently present in application ranging from military drones and autonomous vehicles to automated vacuum cleaner. For all of these platforms we need to know its position in the space in order to control it. This insight reflects why the positioning systems are the cornerstone in the robotics community.

In some robotics applications the *Localization* is done with respect to a manually built map consisting in a set of landmarks or beacons with known position, the Global Positioning System (GPS) can be considered a beacon with known location. However in environments like indoors, caves, underwater or even in space the use of GPS is denied, furthermore in some applications like autonomous driving the GPS precision is not enough to drive safely.

Although the use of beacons is a possibility in industrial environments one of the less invasive way of localization based on natural landmarks is preferred. It have been proven very effective because many of the autonomous cars use this approach. The main constraint of this approach consist in the previously built map, by definition if the previously built map is not accurate, or nonexistent, this approach will fail.

The *Simultaneous Localization and Mapping (SLAM)* approach solves the constraint of localization using maps by simultaneously building the map in which the robot is moving. This approach have been called the *holy grail* of mobile robotics [Dissanayake et al., 2001] and nowadays there are many algorithms working with sensors as diverse as wheel odometry, ranging laser, ultrasound, monocular cameras, 3D-cameras, omnidirectional cameras and different types of robotic platforms including unmanned air vehicles (UAV), autonomous underwater vehicles (AUV), autonomous cars and even robotic spacecraft. State of the art SLAM algorithms provide accurate information on the metric representation of the environment. SLAM provides an appealing alternative to user-built maps, showing that robot operation is possible in the absence of an ad-hoc localization infrastructure.

In order to perform a task the mobile robot often requires a map, fundamental tasks like path planning or more complex like semantic segmentation of the environment are all based on a map. In fact, for many mobile robots the task consists in building the map, one example consist in exploring an environment ensuring full coverage and reporting to the human, another is the structural inspection where the 3D reconstruction is required.

In this thesis we treat the problem of building the map autonomously. Most practical applications of mapping consist in a human operator moving a robot through the environment, the robot gathers information passively and builds the map according to this information. Stopping criteria and quality of the reconstruction are usually subjective to the human operator.

The task of building a map autonomously based on the SLAM algorithm is called *Active SLAM*. In this thesis we first follow the classic active SLAM approach of choosing among a finite set of trajectories the one with the maximum value in terms of our utility function, nonetheless we move into a different approach based on graph theory.

Considering the link between the SLAM problem and the underlying graph we are able to mathematically define optimal exploration and use on-line graph exploration technique with convergence guarantees and close to optimal performance.

1.1 Contributions

In this thesis we focus on the problem of Active SLAM. First we follow the common approach, we generate a path planning algorithm using utility functions based on the uncertainty, however we differ from previous works by acknowledging its probabilistic nature by calculating its expected value. We demonstrate that this criteria reliably avoid zones void of features, additionally when the input graph is augmented with possible traversals it can leverage the possible savings in distance with the safety of that path.

In order to exploit the graph properties of the environment for the Active SLAM algorithm we transform the environment into a topological graph composed of approximate convex regions. The contributions derives from using contours to define the image and an incremental algorithm to deal with incremental exploration

One important contribution results from developing a different approach for the active SLAM task. Analyzing the topological graph structure of the environment we are able to determine the optimal exploration trajectory, using anindoor environment dataset we observe that a path that traverses every edge once in the topological graph achieves up to 97% of the result of the optimal trajectory in one of the typical criteria used, therefore we propose it as an exploration policy.

The policy of traversing every edge in the topological graph can be implemented using classical algorithms for on-line graph exploration, hence we can prescind from the utility functions and still being able to claim optimality in the exploration. Consequently we develop the *TIGRE* algorithm that consist in the incremental segmentation combined with exploration policy previously described and demonstrate its behavior in the common exploration scenario.

1.2 Thesis Outline

The remainder of the thesis is organized as follows:

- Chapter 2 *(Fundamentals)*, We review some of the basic concepts needed to develop this thesis. We focus in the graph SLAM technique and the Optimality Criteria to evaluate the uncertainty.

- Chapter 3 *(Expected Uncertainty based Path Planning)*, We consider the random nature of the uncertainty in path planning and calculate its expected value to evaluate the trajectories.

- Chapter 4 *(Graph Extraction)*, In preparation for graph based exploration we designed an algorithm to transform the partially explored map to a graph representation.

- Chapter 5 *(Optimal Information Path)* Based on a graph representation we analyze the optimal solution for common indoors structures, based on these observations we generate an exploration policy.

- Chapter 6 *(Graph based Active SLAM)* Based on the topological representation of a partially explored map and the conclusions derived from the previous chapter we proposed a new exploration policy and evaluate in typical exploration scenario.

- Finally in chapter 7 we present the conclusions and possible research directions departing from this thesis.

1.3 Publications

The following publications have been derived from this thesis

Conferences

- L. Fermin-Leon, J. Neira, and J.A. Castellanos. "Path planning in graph SLAM using Expected uncertainty". In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on. IEEE.2016.

  Ranking $\rightarrow$ CORE2017: A, h5-index : 50 in Google Scholar (Ranked 5th in Robotics)

- L. Fermin-Leon, J. Neira, and J. A. Castellanos. "Incremental contour-based topological segmentation for robot exploration". Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017.

  Ranking $\rightarrow$ CORE2017: B, h5-index : 71 in Google Scholar (Ranked 1st in Robotics)

- L. Fermin-Leon, J. Neira, and J. A. Castellanos. "TIGRE: Topological graph based robotic exploration". Mobile Robots (ECMR), 2017 European Conference on. IEEE, 2017.

  Ranking $\rightarrow$ h5-index : 12 in Google Scholar

Journal Publications

- L. Fermin-Leon, J. Neira, and J.A. Castellanos. "Exploration Efficiency in Active SLAM". Robotics and Autonomous Systems. (In Preparation).

Open Source Software

- TIGRE: Topological Graph based Robotic Exploration (`https://github.com/lfermin77/TIGRE`)
- Implementation of the Incremental Contour-Based Topological Segmentation in structured or unstructured environments (`https://github.com/lfermin77/Incremental_DuDe_ROS`)
- Wrapper for Dual Decomposition and ROS (`https://github.com/lfermin77/dude_ros`)
- Algorithm to evaluate the Expected Uncertainty in Path Planning (`https://github.com/lfermin77/Expected_Uncertainty`

Chapter 2

Fundamentals

We will discuss most of the theoretical basis we will need for the rest of the thesis

2.1 Simultaneous Localization and Mapping

Robot localization can be seen as a problem of coordinate transformation. Maps are described in a global coordinate system, which is independent of a robot pose [Thrun et al., 2005]. Localizing a robot implies finding the correspondence between these coordinate systems. In the scenario when some information about the global coordinate system is given, whether in form of localization of landmarks or GPS signal, and the robot localization can be done reliably with respect to these landmarks no further refinement needs to be done [Cadena et al., 2016].

In applications where the use of GPS is denied like indoors, underground or underwater environments, and no landmark is localized in a prior map the localization has to be done while building the map, this task is known as *Simultaneous Localization and Mapping*. Because we are building the map while localizing the robot we define the global coordinate system in our discretion and define every measure with respect to it.

According to [Thrun et al., 2005] there are two main forms of the SLAM problem, the *online SLAM* and the *full SLAM* problem. In the online version the task consists in estimating the momentary pose and the map. The term online is coined because it only involves the variables that persist at time t, i.e. the last pose and the map. Typical algorithms used in this problem are the *Extended Kalman*

Filter (EKF) where the variables are the poses and the Covariance matrix and the *Extended Information Filter (EIF)* where the variables are the information vector and the Information matrix (the inverse of the covariance matrix).

In contrast to the previous one the full SLAM problem consist in estimating the entire path of the robot along with the map, this subtle difference has implications in the type of algorithms that are typically used. One of the most commonly used for this task is the *Graph SLAM* where the landmarks and the path of the robot are modeled as nodes of the graph and the relation among them as its edges. In this thesis we will be based almost exclusively in this type of algorithm so we will review it in detail.

2.1.1 Graph SLAM

In graph-based SLAM, the poses of the robot are modeled by nodes in a graph and labeled with their position in the environment. Spatial constraints between poses, that result from observations or from odometry measurements, are encoded in the edges between the nodes.

The spatial constraints represented by the edges is considered to be a probability distribution over the relative transformation between pair of poses. These transformations come from either odometry measurements or the result of aligning observations of different robot poses. Once the graph is built the problem consist in finding the set of robot poses that best satisfies the constraints.

In graph SLAM the problem is divided in two tasks:

1. *Graph Construction,* Commonly known as “Front-End", consists in constructing the graph from the raw measurements. It deals with the problem of data association that consists in deciding whether two robot observations are related or not and include the corresponding edge in consequence

2. *Graph Optimization,* Also known as “Back-End". In this task the problem consists in determining the most likely configuration of robot poses given the edges of the graph. Because it deals with an abstract representation of the data the nature of the sensor is irrelevant.

In [Grisetti et al., 2010] they show that in graph-based SLAM the problem is simplified by abstracting the raw measurements into *“virtual measurements"*, the edges in a graph represent these measurements.

Considering the front-end provides correct information, the graph built $G = (V, E)$ consists in a set of nodes (or vertices, or points) and edges (or lines, or arcs). The set of nodes $V = \{v_1, v_2, \cdots v_n\}$ are associated to the number n of previous positions in the robot. Using the function $x(v_i) = \mathcal{N}(\mathbf{x}_i, \mathbf{P}_i)$ we relate the node number with its probability distribution. We will use the variable i for numbering nodes.

The set of edges $E = \{e_1, e_2, \cdots e_m\}$ are associated with the m number of virtual measurements. Every edge is composed of a pair of nodes $e_j = \{v_i, v_k\}$ where v_i and v_k are the nodes it connects. Let $z(e_j) = \mathcal{N}(\mathbf{z}_j, \mathbf{\Sigma}_j)$ the mean and the covariance of the virtual measurement. We will use the variable j for numbering edges.

Let's define $\mathbf{d}_j = d(\mathbf{x}_i, \mathbf{x}_k, \mathbf{z}_j)$ as the resulting vector of the function that computes the difference between the expected observation, in terms of position of the nodes, and the real observation gathered by the robot as $f(\mathbf{x}_i, \mathbf{x}_k, \mathbf{z}_j)$, we can write the log-likelihood l_j of the measurement $\mathbf{z}_j$ as

$$l_j \propto \mathbf{d}_j^T \mathbf{\Sigma}_j^{-1} \mathbf{d}_j \tag{2.1}$$

The maximum likelihood approach consist in finding the set of robot poses that minimizes the constraints associated to all observations. Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_n)$ be the vector of poses of robot we can write compactly the negative log-likelihood $F(\mathbf{x})$ as

$$F(\mathbf{x}) = \sum_{j=1}^{m} \underbrace{\mathbf{d}_j(\mathbf{x})^T \mathbf{\Sigma}_j^{-1} \mathbf{d}_j(\mathbf{x})}_{F_j(\mathbf{x})} \tag{2.2}$$

$$= \sum_{j=1}^{m} F_j(\mathbf{x}) \tag{2.3}$$

where we write $\mathbf{d}_j(\mathbf{x})$ to emphasize that it is a function of the vector of configurations $\mathbf{x}$. $F_j(\mathbf{x})$ represent the contribution of every edge e_j to $F(\mathbf{x})$. The optimal vector of configuration is then

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} F(\mathbf{x}) \tag{2.4}$$

Linearization

Following the demonstration of [Grisetti et al., 2010] we show the derivation of the algorithm.

Considering the function $\mathbf{d}_j(\mathbf{x})$ is in general non linear the common approach to solve the optimization in equation 2.4 consist in linearizing the system. Considering a good initial guess $\breve{\mathbf{x}}$ the term $\mathbf{d}_j$ can be written as

$$\begin{aligned} \mathbf{d}_j(\breve{\mathbf{x}}+\Delta\mathbf{x}) &\simeq \mathbf{d}_j(\breve{\mathbf{x}})+\mathbf{J}_j\Delta\mathbf{x} && (2.5)\\ &\simeq \breve{\mathbf{d}}_j+\mathbf{J}_j\Delta\mathbf{x} && (2.6) \end{aligned}$$

$$\text{where } \mathbf{J}_j = \left.\frac{\partial \mathbf{d}_j}{\partial \mathbf{x}}\right|_{\breve{\mathbf{x}}} \text{ and } \breve{\mathbf{d}}_j = \mathbf{d}_j(\breve{\mathbf{x}}) \quad (2.7)$$

Let's write the linearization of the maximum likelihood equation 2.3 for the contributions of every edge j.

$$\begin{aligned} &F_j(\breve{\mathbf{x}}+\Delta\mathbf{x}) && (2.8)\\ &= \breve{\mathbf{d}}_j(\breve{\mathbf{x}}+\Delta\mathbf{x})^T\mathbf{\Sigma}_j^{-1}\breve{\mathbf{d}}_j(\breve{\mathbf{x}}+\Delta\mathbf{x}) && (2.9)\\ &\simeq (\breve{\mathbf{d}}_j+\mathbf{J}_j\Delta\mathbf{x})^T\mathbf{\Sigma}_j^{-1}(\breve{\mathbf{d}}_j+\mathbf{J}_j\Delta\mathbf{x}) && (2.10)\\ &\simeq \breve{\mathbf{d}}_j^{\,T}\mathbf{\Sigma}_j^{-1}\breve{\mathbf{d}}_j + 2\underbrace{\breve{\mathbf{d}}_j\mathbf{\Sigma}_j^{-1}\mathbf{J}_j}_{\mathbf{b}_j}\Delta\mathbf{x} + \Delta\mathbf{x}^T\underbrace{\mathbf{J}_j^T\mathbf{\Sigma}_j^{-1}\mathbf{J}_j}_{\mathbf{Y}_j}\Delta\mathbf{x} && (2.11)\\ &\simeq c_j + 2\mathbf{b}_j\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{Y}_j\Delta\mathbf{x} && (2.12) \end{aligned}$$

Using this approximation we can write

$$
\begin{aligned}
F(\breve{\mathbf{x}}+\Delta\mathbf{x}) &= \sum_{j=1}^{m} F_j(\breve{\mathbf{x}}+\Delta\mathbf{x}) && (2.13)\\
&\simeq \sum_{j=1}^{m} (c_j + 2\mathbf{b}_j\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{Y}_j\Delta\mathbf{x}) && (2.14)\\
&\simeq \underbrace{\sum_{j=1}^{m} c_j}_{c} + \underbrace{\sum_{j=1}^{m} 2\mathbf{b}_j}_{\mathbf{b}}\Delta\mathbf{x} + \Delta\mathbf{x}^T \underbrace{\sum_{j=1}^{m} \mathbf{Y}_j}_{\mathbf{Y}}\Delta\mathbf{x} && (2.15)\\
&\simeq c + 2\mathbf{b}\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{Y}\Delta\mathbf{x} && (2.16)
\end{aligned}
$$

The matrix $\mathbf{Y}$ is the *Information Matrix* of the system. This expression is minimized solving the system

$$\mathbf{Y}\Delta\mathbf{x} = -\mathbf{b} \tag{2.17}$$

The solution of linearized system consist in

$$\mathbf{x}^* = \breve{\mathbf{x}} + \Delta\mathbf{x} \tag{2.18}$$

In the Gauss-Newton algorithm this result represent the new initial guess for the next iteration where the Jacobian matrix $\mathbf{J}_j$ is evaluated at the new linearization point and new $\mathbf{x}^*$ is calculated. This iterative procedure is repeated until convergence.

Characteristics of the Linearized System

The Jacobian matrix $\mathbf{J}_j$ only depends on the position of nodes that e_j connects, as a consequence it is a block matrix with zeros every where else, represented by $(\cdot)$, except by the elements associated with the nodes v_i and v_k.

$$\mathbf{J}_j = \begin{pmatrix} \cdot & \mathbf{J}_{j|i} & \cdot & \mathbf{J}_{j|k} & \cdot \end{pmatrix} \tag{2.19}$$

where $\mathbf{J}_{j|i}$ and $\mathbf{J}_{j|k}$ are the partial derivatives of the difference vector $\mathbf{d}_j$ with respect to the poses of the nodes v_i and v_k.

$$\mathbf{J}_{j|i} = \left.\frac{\partial \mathbf{d}_j}{\partial \mathbf{x}_i}\right|_{\breve{\mathbf{x}}} \tag{2.20}$$

$$\mathbf{J}_{j|k} = \left.\frac{\partial \mathbf{d}_j}{\partial \mathbf{x}_k}\right|_{\breve{\mathbf{x}}} \tag{2.21}$$

Consequently the information matrix associated to the edge e_j has a very interesting structure, it is a square block matrix with zero blocks everywhere except by the positions associated with the connecting nodes

$$\mathbf{Y}_j = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \mathbf{J}_{j|i}^T \mathbf{\Sigma}_j^{-1} \mathbf{J}_{j|i} & \cdot & \mathbf{J}_{j|i}^T \mathbf{\Sigma}_j^{-1} \mathbf{J}_{j|k} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \mathbf{J}_{j|k}^T \mathbf{\Sigma}_j^{-1} \mathbf{J}_{j|i} & \cdot & \mathbf{J}_{j|k}^T \mathbf{\Sigma}_j^{-1} \mathbf{J}_{j|k} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \tag{2.22}$$

and the error vector is

$$\mathbf{b}_j = \begin{pmatrix} \cdot \\ \mathbf{J}_{j|i}^T \mathbf{\Sigma}_j^{-1} \breve{\mathbf{d}}_j \\ \cdot \\ \mathbf{J}_{j|k}^T \mathbf{\Sigma}_j^{-1} \breve{\mathbf{d}}_j \\ \cdot \end{pmatrix} \tag{2.23}$$

The difference vector $\mathbf{d_j}$ depends only on the relative position of the connected nodes v_i and v_k. Consequently the error $F(\mathbf{x})$ associated to a set of poses is invariant under a rigid transformation of all the poses. This fact results in the equation 2.17 being under determined [Grisetti et al., 2010] [Thrun and Montemerlo, 2006].

The common solution consists in *“anchoring"* one of the nodes. This is done by choosing one node (usually the first) and setting its pose to $x(v_1) = \mathbf{0}^T$ and setting its information arbitrarily high (or equivalently setting the covariance matrix to $\mathbf{0}$). The same effect is achieved by suppressing the block row and block column associated to the chosen node [Kummerle et al., 2011].

2.2 Exploration Taxonomy

The exploration task as the problem of gathering information of the environment have been addressed with different approaches, we can classify the exploration according to the structure used to represent the space in *Geometry Based* and *Graph Based* [Dessmark and Pelc, 2004].

2.2.1 Geometry Based

In the geometry based exploration we can identify two scenarios: environment represented as a set of *polygons*, or represented as a *raster cells*.

Polygon based

In the first scenario a mobile robot has to explore an unknown environment modeled by a simple polygon, this type of exploration is related to the art gallery problem and has been studied in the computer science community. The solution comes from segmenting the polygon into convex regions. When the robot enters to a convex region every point in it becomes visible, consequently, once every convex region have been visited the whole polygon have been visited. Unfortunately, finding the minimum set of convex regions is demonstrated that is NP-hard [Lien and Amato, 2004].

Grid Based

The second scenario is the most common in the robotics community, in this case the environment is represented by an occupancy grid. Every cell in the grid represent the probability of being occupied. Typical examples are the one associated with the Active SLAM literature.

Most of the algorithms are related to the frontier-based exploration [Yamauchi, 1997], the frontiers are defined as border regions between the explored and unexplored cells. Given that the environment is enclosed this algorithm is guaranteed to converge to full exploration. Considering the robot selects the action with the highest utility in every instant this constitute a greedy exploration approach, consequently no optimality guarantees are given.

One important drawback of this algorithms is that the number of cells in the grid depends on the level of quantization used, low resolution implies larger cells that represent poorly the environment, high resolution implies large amount of data.

2.2.2 Graph Based

Using the abstraction level of the graphs to represent the environment we obtain some interesting properties. The graphs are used to represent a wide variety of problems, therefore, any result coming from this analysis can be extrapolated to other areas besides robotic exploration and vice-versa.

In the context of on-line graph exploration [Megow et al., 2012] the graph is represented as $G(V,E)$ with E the set of edges, and V the set of vertices. The common value to optimize is the cost of traversing an edge, usually the distance, and the behavior of the algorithm can always be compared to the optimal. The sequence of nodes to visit is known as *path*, the optimal path can always be calculated offline and then compared with performance of the online version. In this context we call an algorithm *c-competitive* if for a positive number c, it computes for any instance a path of total length at most c times the optimal offline path.

According to the assumptions we can differentiate three scenarios:

No node is recognizable

In this case the nodes are anonymous and recognizable only by a marker placed on them, the interest for this area is the minimum information the robot needs to accomplish the exploration. Classical results can be found in [Fraigniaud et al., 2005] and more recently in [Disser et al., 2016], some applications of this methods to real robots can be found in [Wang et al., 2014].

Node is recognizable, but edges not

Once the robot visits a node it learns the node, however the robot can only learn the end vertex of the edge and the cost of traversing it once the robot traverses it and recognizes the node in the other end. In this scenario the common goal is the *full graph exploration* and it consists in visiting every edge, and thus every vertex. The offline version of this problem is the polynomially solvable *Chinesse Postman*

Problem (CPP). For the online solution the lower bound is known to be reached using *Depht First Search* (DFS), with the worst case being traversing twice the distance of the optimal, in the case the graph have an eulerian path [Megow et al., 2012].

Node and Edges recognizable

Whenever a robot visits a node it learns the node and where every incident edge is heading, for example if the vertices correspond to cities, the searcher is able to see the road signs of routes to other cities including distance information and the name of the city it is headed. In this scenario the goal usually consists in exploring all the nodes. The offline solution to this task consist in one of the most studied optimization problem the *Traveling Salesman Problem* (TSP), which is known to be NP-Hard [Megow et al., 2012]. In [Miyazaki et al., 2009] they demonstrate that for the special case that all edges have equal weight, the standard Depth First Search (DFS) is 2-competitive. It yields a total tour not larger than twice the size of a minimum spanning tree (MST), a lower bound on the optimal tour. In the case of planar graphs the algorithm *ShortCut* proposed in [Kalyanasundaram and Pruhs, 1994] is 16-competitive, this represented the state of the art for almost two decades until the generalization proposed in [Megow et al., 2012] to graph with bounded genus.

2.3 Active SLAM

So far we have discussed the SLAM when no action in the robot is controlled, in that sense the robot perceives the world passively. Including in the problem the commands that affect the measurement we receive from the environment defines the *Active SLAM*.

In [Cadena et al., 2016] they survey the literature on active SLAM and claim that the popular framework for active SLAM consists in selecting the best future action among a finite set of alternatives. They consider this framework can expressed in three steps:

1. The robot identifies possible locations to visit in its current estimate of the map. These locations usually come from the frontier points described in

[Yamauchi, 1997].

2. The robot computes the utility of visiting each vantage point and selects the action with the highest utility.

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

The utility of the action is calculated using the predicted covariance matrix for the possible locations to visit and the characteristics of the associated estimated map. Most of the approaches calculate the utility as a linear combination of these metrics. The cost associated to the covariance matrix have been calculated using A-optimal [Sim and Roy, 2005] and D-optimal criteria [Carrillo et al., 2012b].

2.3.1 Optimality Criteria in Active SLAM

In order to associate a scalar number to the quality of the covariance matrix some optimality criteria are described in [Pukelsheim, 2006]. These criteria are defined for the information matrix $\mathbf{Y}$.

From the perspective of the theory of optimal design of experiments [Pukelsheim, 2006], different optimality criteria (e.g. A-Opt, D-Opt, E-opt) could be computed as different instances of the family of *information functions* of the information matrix $\mathbf{Y}$ (i.e. the inverse of the covariance matrix). Mathematically any optimality criteria can be expressed as $||\cdot|| : \mathbf{Y} \to \mathbb{R}$.

For an information matrix $\mathbf{Y}$ with size $u \times u$, the *information function* $||\mathbf{Y}||_p$ is defined in [Pukelsheim, 2006] as:

$$||\mathbf{Y}||_p = \begin{cases} \sqrt[p]{\frac{1}{u}\text{trace}(\mathbf{Y}^p)} & if \quad p \neq 0, p \leq 1 \\ \\ \sqrt[u]{det(\mathbf{Y})} & if \qquad p = 0 \end{cases} \tag{2.24}$$

with p an additional parameter that can be related to the optimality criteria used. Using the properties of the trace and the determinant of a matrix we can write equation 2.24 in terms of the eigenvalues λ_i of the matrix $\mathbf{Y}$

$$\|\mathbf{Y}\|_p = \begin{cases} \left(\frac{1}{u}\sum_{i=1}^{u}\lambda_i^p\right)^{\frac{1}{p}} & if \quad p \neq 0, p \leq 1 \\ \\ \prod_{i=1}^{u}\lambda_i^{\frac{1}{u}} & if \qquad p = 0 \end{cases} \tag{2.25}$$

The different optimality criteria are described in terms of different values for p:

1. E-Optimal: The lowest eigenvalue λ of the information matrix is the limiting behavior as p decreases, $\lim_{p \to -\infty} \|\mathbf{Y}\|_p = \lambda_{min}$
2. A-Optimal: The trace of the covariance matrix in the case $p = -1$
3. D-Optimal: The determinant of the information matrix is associated to the value of $p = 0$
4. T-Optimal: The trace of the information matrix is associated to the value of $p = 1$

Anchor Node effect

As described in [Thrun and Montemerlo, 2006] [Grisetti et al., 2010] the information matrix associated to the SLAM problem is rank deficient, every solution to the system is valid up to a generic transformation. In order to find a single solution the common strategy used by state of the art back-ends like g^{2}o [Kummerle et al., 2011] consist in anchoring one node, in terms of the information matrix its effect consist in eliminating the row and the column associated to the anchored node.

Transforming the information matrix with reference to one anchored node to be referenced to another node can be done simply using the Jacobians associated with the transformation

$$\boldsymbol{Y}_k = \mathbf{J}_{ki}\boldsymbol{Y}_i\mathbf{J}_{ki}^T \tag{2.26}$$

In linear algebra, two matrices $\mathbf{A}$ and $\mathbf{B}$ are called similar if $\mathbf{B} = \mathbf{P}^{-1}\mathbf{AP}$, as a consequence they share the set of eigenvalues. Because $\mathbf{J}_{ki}^T \neq \mathbf{J}_{ki}^{-1}$ the matrices $\boldsymbol{Y}_i$

and $\boldsymbol{Y}_k$ are not similar, as a consequence the set of eigenvalues is modified and in general every optimality criteria is different for any of the transformed matrices. The only exception consist in the D-Optimal which is the only optimality criteria invariant to transformations because the determinant is not modified.

$$\begin{aligned}
\det(\boldsymbol{Y}_k) &= \det(\mathbf{J}_{ki}\boldsymbol{Y}_i\mathbf{J}_{ki}^T) && (2.27)\\
&= \det(\mathbf{J}_{ki}) * \det(\boldsymbol{Y}_i) * \det(\mathbf{J}_{ki}^T) && (2.28)\\
&= 1 * \det(\boldsymbol{Y}_i) * 1 && (2.29)\\
&= \det(\boldsymbol{Y}_i) && (2.30)
\end{aligned}$$

Ideally we want the optimality criteria to be independent of the choice of the anchoring node, accordingly we propose to calculate the optimality criteria in the information matrix in terms of the nonzero eigenvalues before anchoring any node.

Chapter 3

Expected Uncertainty based Path Planning

3.1 Path Planning, from continuum to graphs

The basic path planning problem is to find collision-free paths for a moving object (robot) among stationary, completely known obstacles. This basic definition has been generalized in terms of finding the optimal path in terms of certain parameters, like distance, power consumption among others, these parameters are commonly summarized in cost functions to be optimized.

One of the early approaches came from the work on artificial potential fields [Khatib, 1986], where the robot and the obstacles are modeled as electrically charged particles. The result is a path defined in the continuum of the space of configurations of the robot.

This planning strategy was later improved by graph based algorithms like Probabilistic Road Map (*PRM*) [Kavraki et al., 1996] and then Rapidly-exploring Random Trees (*RRT*) [Kuffner and LaValle, 2000] that remains valid nowadays. In these algorithms the maps is built representing the space as a graph where the set of nodes are samples of the open space and the set of edges represent the possibility of traversing between any pair of nodes.

In *PRM* the nodes are sampled from the configuration space and then edges are added if there is a possible path connecting them. In *RRT* the graph is built by adding new nodes in random directions but always close to the graph currently being built, unlike the *PRM* the graph is built from the starting point outwards. In grid based algorithms the graph is given in terms of the connectivity to the neighbourhood in the grid.

Once the problem is transformed into a graph we can use graph based search algorithms. The graph traversal refers to the process of visiting each vertex in a graph, in graph search once the goal is reached it stops traversing the graph.

The classic algorithms for graph search are the *Depth-first search (DFS)* and *Breadth-first search (BFS)* [Diestel, 2000] which differ in the order the nodes are visited. In robotics community one of the most known is the Dijkstra's algorithm [Dijkstra, 1959], which can be seen as a generalization of the *BFS* for weighted graphs. The A^* algorithm [Hart et al., 1968] is a variation of the Dijkstra's algorithm where a heuristic term is used to improve performance.

3.2 Planning under Uncertainty

The most general formulation of the problem of path planning under uncertainty is choosing the optimal action in partially observable stochastic domain. This problem is known as the Partially Observable Markov Decision Process (POMDP) [Kaelbling et al., 1998]. Kaelbling et al. use techniques from operations research to solve the problem in these terms. In [Prentice and Roy, 2009] Prentice claims that this approach has limited success in addressing large real-world problem. They propose the *Belief Road Map (BRM)*, a variation of the Probabilistic Road Map (*PRM*) to include the uncertainty when searching in the $belief space$ (term quoted from operations research).

In *PRM*, a graph is constructed in which the nodes are a set of poses sampled in the free space and the edges are added between any pair of nodes as long as there exist a sequence of controls that allows the robot to move between them.

The *BRM* additionally simulates these controls in order to obtain the parameters to update the covariance when searching for the path with smallest final uncertainty using a *BFS* algorithm. For the experimental setup, the localization is achieved by placing RF beacons in a known map. The final position, and its uncertainty, are modeled using the combination of beacons and odometry signals. In this setup there's no place recognition and the expected uncertainty can be calculated more efficiently than [Kaelbling et al., 1998] from the expected received signal from the beacons.

Agha-mohammadi *et al* [Agha-mohammadi et al., 2014] claim that they break the curse of history in this POMDP (the optimal action depends on the traversed nodes, actions and observations previous to the current node) by including a feed back controller, which stabilizes the belief, thus solving the problem with standard Dynamic Programing (DP), like Dijkstra's [Dijkstra, 1959] algorithm. Without using local controllers Indelman *et al* [Indelman et al., 2015] limits the effects of the history using a model predictive control (MPC) scheme: at each time step the robot plans a suitable motion strategy over a time horizon. This algorithm is tested in different time horizon lags resulting in prohibitive large planning time when this horizon reaches 20 look-ahead steps (9000 secs).

In [Valencia et al., 2011] the problem is posed as finding the sequence of nodes to traverse in the graph coming from a Graph SLAM system. In this work the authors consider that the probability of getting lost (i.e. no loop closure) is directly related to the increments in uncertainty, under the assumption that better paths are the ones that keep the robot well localized throughout the whole trajectory. They propose a path planning algorithm that finds a sequence of nodes that minimizes the sum of the positive increments in uncertainty. This work uses a loop closure probability heuristic for path planning, including the following simplifications:

1. The loop closure probability heuristic does not consider the environment.

2. The decision of using only positive increments biases the solution into nodes with big loop closures even if the path is not reliable.

3. It only considers the uncertainty with the following node in the list, thus the cost function is associated to the probability of relocalizing with every node in the path, which can be substantially smaller than the probability of missing some nodes and still being able to relocalize in the end.

Like [Valencia et al., 2011], in [Carrillo et al., 2012a] they propose to find the path that keeps the robot well localized during the whole trajectory but using a different metric, the Sum of the D-Optimal criteria for the nodes in the path. They argue that D-Optimal gives the most accurate representation of the uncertainty enclosed in the covariance matrix and thus it should result in less uncertain paths. The problem is that the least uncertain path does not necessarily coincide with the most reliable path, because the probability of getting lost is related to the increments in uncertainty [Valencia et al., 2011] not to the absolute uncertainty of the nodes.

In [Kim and Eustice, 2014] an Active Visual SLAM is constructed with iSAM [Kaess et al., 2008] as back-end. A path is proposed that revisits some nodes and estimate the probabilities of tracking the features. The probability is estimated with a precalculated table that relates the saliency of the feature to the probability of tracking. The precalculated table of probability allows an accurate calculation of the expected uncertainty effectively considering the environment and improves its performance, getting a good trade off between uncertainty and path length, thus encouraging its use. However the probability estimated is restricted to its specific domain and the analysis does not consider the probability of missing some nodes and still being able to localize in the end.

In this thesis the expected value of the uncertainty is proposed, therefore the probability of the transition is needed. This probability is calculated using not only the distance dependence like [Valencia et al., 2011] when increasing the connectivity, but also the node dependence. This dependency is estimated through Monte Carlo simulations that can be extended to any matching algorithm.

Due to the non monotonicity of the evolution of the expected value, algorithms like Dijkstra's cannot be used so the search has to be done in the complete space. The number of trajectories in the complete space is exponential, however when the search is done in the reduced graph like [Carrillo et al., 2012a], the number of possible paths to evaluate becomes tractable.

3.3 Expected Uncertainty

In probabilistic SLAM, the uncertainty of a path can be determined with the covariance matrix associated to each pose, always calculated with respect to a base reference frame. The first node of the graph, or the starting point of the trajectory are usually used as base reference (although any node can be actually used).

Intuitively, the expected value of the covariance matrix of a destination point that we propose here is calculated by weighting the covariance resulting from the relocalization of the robot at every point of the trajectory, with the probability of this relocalization being successful. The more structure or texture in a given area, the higher this probability. If at a given point there is insufficient structure or texture, the covariance of the localization will be given mostly by odometry, so its corresponding covariance is weighted with the probability of relocalization failing.

Whenever a new node is reached, two hypotheses are produced (successful and failed relocalization) effectively doubling the number of hypotheses for every new node in the path. The main contribution of this chapter consists in considering the effects of the relocalization event in the covariance matrix to reduce the number of alternative hypotheses from 2^n to n.

3.3.1 Derivation

In graph SLAM the map is represented using a graph $G = (V, E)$ formed by a set of nodes $V = \{v_1, v_2 \cdots v_n\}$ and a set of edges $E = \{e_1, e_2 \cdots e_m\}$. Once the graph is optimized with respect to a base reference the nodes v_i contains the information of the previous poses of the robot, it includes the position $\mathbf{x}_i$, the associated covariance $\mathbf{\Sigma}_i$ and the readings $\mathbf{z}_i$ in that pose.

In the graph based settings the trajectory is defined as $T = \{n_0, n_1, \cdots n_m\}$ where the elements n_t contains the node v_i in the graph that needs to be visited in sequence. When the robot traverses from the node in n_{t-1} to the node in n_t there are two possible outcomes,

1. Succesful Relocalization (r), when the robot is able to relocalize, its covariance is the one of the node in n_t in the optimized graph.
2. Failed Relocalization ($\bar{r}$), in this case the new covariance of the robot pose comes from the propagation of its previous one with the odometry covariance $\mathbf{O}$ of traversing from n_{t-1} to n_t.

Because we have two possible outcomes we handle two hypothetical covariance matrices for every transition, as a consequence for a path with "m" nodes we obtain 2^{m-1} hypothetical covariance matrices.

Let $H = \{h_0, h_1, \cdots\}$ be the set of the hypothesis, where every term $h_k = (\mathbf{C}_k, prob\mathbf{C}_k, id_k)$ contains one of the hypothetical covariance $\mathbf{C}_k$ and "$prob\mathbf{C}_k$" is the probability of getting it, the term "id_k" is the sequence of relocalization events that defines unequivocally the hypothesis, for example $id_k = r, \bar{r}$ is the hypothesis of getting a successful localization with the node in n_1 and failed relocalization with the node in n_2, correspondingly we can write $\mathbf{C}_{r,\bar{r}}$ as the covariance associated with this hypothesis.

The exponential growth of the number of hypothesis can be greatly reduced if we consider that every successful relocalization produces the same covariance. In figure 3.1 we observe the way the redundant hypotheses that produce the same covariance can be merged to produce a number of hypotheses equal to the number of nodes in the trajectory.

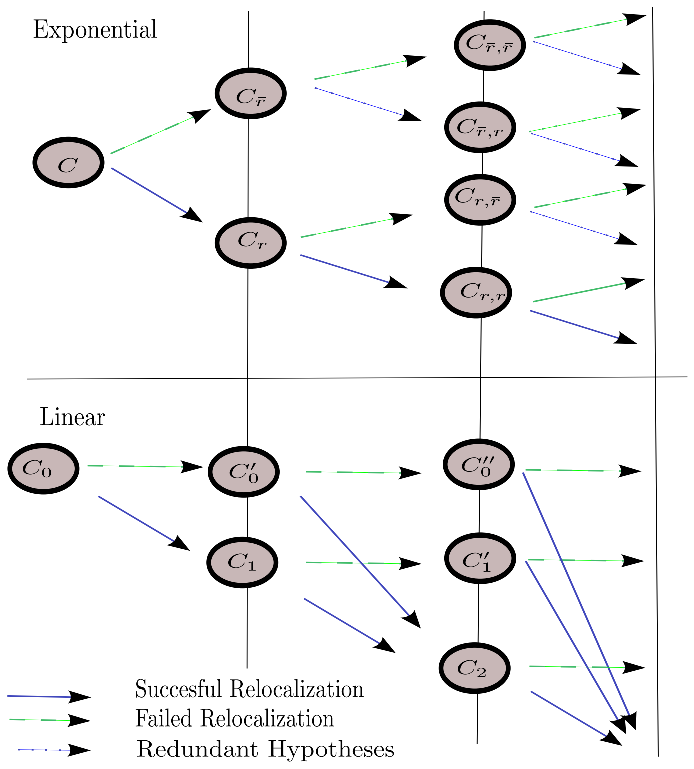


Figure 3.1: Relocalization Process: theoretically whenever a new node is reached, two hypotheses of covariance ($\mathbf{C}$) are produced (the successful $\mathbf{C}_{\dots,r}$ and failed relocalization $\mathbf{C}_{\dots,\bar{r}}$) aparently resulting in exponential growth in number of hypotheses. However successful relocalization at each step result in the same hypothesis ($\mathbf{C}_{\bar{r},r} = \mathbf{C}_{r,r} = \mathbf{C}_2$, in the next step $\mathbf{C}_{\bar{r},\bar{r},r} = \mathbf{C}_{\bar{r},r,r} = \mathbf{C}_{r,\bar{r},r} = \mathbf{C}_{r,r,r} = \mathbf{C}_3$), therefore the actual number of alternative hypotheses is linear

Considering the number of alternative hypotheses is the same that the number of nodes in trajectory we index the set of hypothesis H with the index of the last n_t with successful localization, for example, hypothesis h_2 is the hypothesis of having a successful relocalization with the node in n_2, and failing every relocalization afterwards, consequently $\mathbf{C}_2$ is the covariance associated to this hypothesis.

Let $\mathbf{C}^{(t)}$ be the covariance matrix associated to the robot pose when visiting the node n_t, this matrix is a random variable and can be characterized using its expected value. The expected uncertainty is calculated weighting every hypothetical covariance with its corresponding probability. Let's define, $\mathbf{C}_k^{(t)}$ the robot pose covariance when visiting the node n_t according to the hypothesis h_k and $prob\mathbf{C}_k^{(t)}$ its probability, with these definitions we can write

$$E[\mathbf{C}^{(t)}] = \sum_{h_k \in H} \mathbf{C}_k^{(t)} \cdot prob\mathbf{C}_k^{(t)} \tag{3.1}$$

We need to calculate the covariance of every hypothetical realization and its corresponding probability.

The covariance $\boldsymbol{C}_k^{(t)}$ is the result of the composition of the covariance from the last node with successful relocalization with the odometry of traversing every further node, consequently $t \geq k$. Let $\mathbf{\Sigma}_t$ be the pose covariance of the node in n_t in the optimized graph and $\mathbf{O}$ the covariance of the odometry we can write

$$\mathbf{C}_k^{(t)} = \begin{cases} \mathbf{\Sigma}_t & \text{for } t = k \\ f(\mathbf{\Sigma}_t, \mathbf{O}) & \text{for } t = k+1 \end{cases} \tag{3.2}$$

where the function f is the uncertainty propagation function. In this work we use the uncertainty propagation technique resulting from the linearization of the odometry equation [Kelly, 2004], in our case this equation is

$$\mathbf{C}_k^{(t)} = \mathbf{J}_1 \mathbf{C}_k^{(t-1)} \mathbf{J}_1^T + \mathbf{J}_2 \mathbf{O} \mathbf{J}_2^T \tag{3.3}$$

where $\mathbf{J}_1$ and $\mathbf{J}_2$ are the Jacobians of the transformation from the node in $\mathbf{n}_{t-1}$ to the node in $\mathbf{n}_t$. For the generic case $t > k+1$ we have the recursive equation

$$\mathbf{C}_k^{(t)} = f(\mathbf{C}_k^{(t-1)}, \mathbf{O}) \tag{3.4}$$

The probability associated to $\mathbf{C}_k^{(t)}$ implies considering every connection to the node n_t and every posterior unsuccessful localization afterwards. Using the conditional probability and defining p_t (and its complement $\overline{p_t}$) as the probability of getting a successful localization with the node in n_t, the equation can be written as:

$$prob(\mathbf{C}_k^{(t)}) = prob(\mathbf{\Sigma}_k) \prod_{\tau=k+1}^{t} \overline{p_\tau} \qquad \text{for } t > k \tag{3.5}$$

The estimation of p_t depends on the set of successful or failed localizations of every hypothesis and it will be described in the next section. The term $prob(\mathbf{\Sigma}_t)$ can be calculated using the fact that in every n_t, the probabilities add up to 1. Thus, the probability of the relocalization in node n_t can be calculated recursively with:

$$\sum_{k=0}^{t} prob(\boldsymbol{C}_k^{(t)}) = 1 \tag{3.6}$$

$$\sum_{k=0}^{t-1} prob(\boldsymbol{C}_k^{(t)}) + prob(\mathbf{C}_t^{(t)}) = 1 \tag{3.7}$$

$$prob(\mathbf{\Sigma}_t) = 1 - \sum_{k=0}^{t-1} prob(\boldsymbol{C}_k^{(t)}) \tag{3.8}$$

where $\mathbf{C}_t^{(t)} = \mathbf{\Sigma}_t$ comes from the equation 3.2.

3.3.2 Algorithm

We can now bundle up all of these equation in the algorithm 1. Summing up, we start with two inputs

1. Optimized SLAM Graph $G = (V, E)$, with set of nodes $V = \{v_1, v_2, \cdots v_n\}$ containing the information of the nodes of the graph where every node contains the pose and covariance matrix of every pose $v_i = (\mathbf{x}_i, \mathbf{\Sigma}_i)$.

2. Trajectory $T = \{n_0, \cdots n_m\}$, The elements of n_t this set contains the nodes v_i that should be visited in the graph G.

We begin in the node in n_0 and with covariance $\mathbf{\Sigma}_0$, from there on we define every alternative set of events in the variable hypothesis every time we visit a new node

- Hypothesis $H^{(t)} = \{h_0^{(t)}, h_1^{(t)} \cdots\}$, the elements $h_k^{(t)} = (\mathbf{C}_k^{(t)}, prob\mathbf{C}_k^{(t)})$ of this set contains every possible covariance and its corresponding probability when visiting the node in n_t. The number of hypothesis is $|H^{(t)}| = t+1$.

Every time the node in n_t is visited from the node in n_{t-1} we update the set of hypothesis using the recursive equations

$$\mathbf{C}_k^{(t)} = f(\mathbf{C}_k^{(t-1)}, \mathbf{O}) \tag{3.9}$$

$$prob(\mathbf{C}_k^{(t)}) = prob(\mathbf{C}_k^{(t-1)})\bar{p}_t \tag{3.10}$$

where "f" is the function that propagates the uncertainty of the hypothesis and p_t its probability of relocalization with the node in n_t. Visiting a new node implies generating a new hypothesis h_t and the components are

$$\mathbf{C}_t^{(t)} = \mathbf{\Sigma}_t \tag{3.11}$$

$$prob(\mathbf{C}_t^{(t)}) = 1 - \sum_{k=0}^{t-1} prob(\boldsymbol{C}_k^{(t)}) \tag{3.12}$$

This process is repeated every time we visit a new node till we reach the final node. The whole algorithm is represented in the figure 3.2.

The final step consist in calculating the expected uncertainty once the final node in n_m is visited.

$$E[\mathbf{C}] = \sum_{k=0}^{m} \mathbf{C}_k^{(m)} \cdot prob\mathbf{C}_k^{(m)} \tag{3.13}$$

In this algorithm we update the whole set of hypotheses in every step. Because the number of hypotheses only grows linearly with the number of steps we can

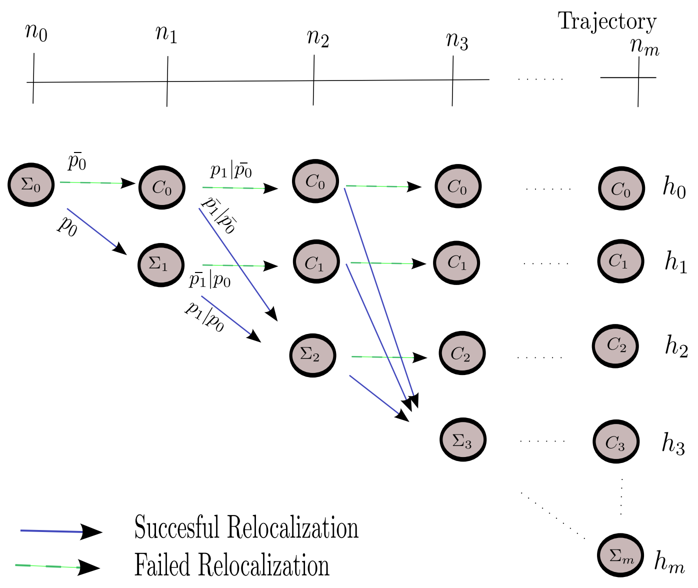


Figure 3.2: Evaluation of Hypotheses: Whenever there is a successful relocalization, the new uncertainty is the Graph's $\boldsymbol{\Sigma}_i$, otherwise the previous uncertainty is propagated from the last relocalization $\boldsymbol{C}_i$

conclude that the computational complexity of the algorithm is $\mathcal{O}(m^2)$ with m the size of the evaluated trajectory, this represents a major improvement from the $\mathcal{O}(m \cdot 2^m)$ of the naive implementation.

3.4 Relocalization Probability

The relocalization probability is estimated considering two different events: the localization event and the place recognition event. The localization is related to the uncertainty in the relative position between the robot and the node being evaluated. Place recognition deals with the ability to determine the correct position (relative to the node) based only on the sensor readings.

Considering these two events independent, the relocalization probability p_t to do a correct detection from the robot position $\mathbf{x}_t$ of the node in n_t is calculated using:

Algorithm 1: Expected Uncertainty

```
Input     : Trajectory T=$\{n_0, n_1, \cdots n_m\}$
            $n_i = (\boldsymbol{x}_i, \boldsymbol{\Sigma}_i, \boldsymbol{z}_i)$
Output    : Node_Expected_Uncertainty
Variables: H=$\{h_0, h_1 \cdots\}$
            $h_k = (prob_k, \boldsymbol{C}_k)$
Initialization;
$\mathbf{O}$ = Odometry_Covariance($n_0, n_1$)
$p$ = Relocalization_Probability $(\mathbf{O}, n_1)$
$h_0 = (prob_0, \mathbf{C}_0) = (1 - p, \mathbf{O})$
$h_1 = (prob_1, \mathbf{C}_1) = (p, \boldsymbol{\Sigma}_1)$
$H = \{h_0, h_1\}$
for $i \in \{2 \cdots m\}$ do
    $\neg prob_i = 0$
    for every hypothesis $h \in H$ do
        $\mathbf{O}$ = Odometry_Covariance $(n_{i-1}, n_i)$
        $\mathbf{C}$ = update_covariance $(\mathbf{C}_h, \mathbf{O}, n_{i-1}, n_i)$
        $p$ = Relocalization_Probability $(\mathbf{C}_h, n_i, T)$
        $h = (prob_h \cdot (1 - p)\ , \mathbf{C})$
        $\neg prob_i = \neg prob_i + prob_h$
    $prob_i = (1 - \neg prob_i)$
    /* add new Hypothesis                                          */
    $h_i = (prob_i, \boldsymbol{\Sigma}_i)$
    $H = H \cup \{h_i\}$
$\boldsymbol{C}_{Expected} = \sum_{h \in H} p_h \cdot \boldsymbol{C}_h$
Return $(\boldsymbol{C}_{Expected})$
```

$$p_t = p_{recognition} * p_{localization} \tag{3.14}$$

where $p_{recognition}$ is the place recognition probability and $p_{localization}$ is the localization probability.

3.4.1 Localization Probability

The purpose of the navigation system is to make the position of the robot $\mathbf{x_k}$ be the position of the destination node $\mathbf{x_i}$. This relative position $\mathbf{x_d} = \ominus\mathbf{x_k} \oplus \mathbf{x_i}$ can be estimated as a new random variable with a Gaussian Distribution [Valencia et al., 2011] [Ila et al., 2010] with mean $\boldsymbol{\mu_d}$ and covariance $\mathbf{C_d}$.

$$\mathbf{x_d} \simeq \mathcal{N}(\boldsymbol{\mu_d}, \mathbf{C_d}) \tag{3.15}$$

$$\boldsymbol{\mu_d} = \ominus\boldsymbol{\mu_k} \oplus \boldsymbol{\mu_i} \tag{3.16}$$

$$\boldsymbol{C_d} \simeq \boldsymbol{J_d} \begin{pmatrix} \boldsymbol{\Sigma}_{ii} & \boldsymbol{\Sigma}_{ik} \\ \boldsymbol{\Sigma}_{ki} & \boldsymbol{\Sigma}_{kk} \end{pmatrix} \mathbf{J_d}^T \tag{3.17}$$

where $\oplus$ and $\ominus$ are the composition and inversion operators of transformations, $\mathbf{J_d}$ is the Jacobian of the transformation and $\boldsymbol{\Sigma}_{ii}$, $\boldsymbol{\Sigma}_{ik}$, $\boldsymbol{\Sigma}_{ki}$ and $\boldsymbol{\Sigma}_{kk}$ are the marginal covariances. The probability of the robot being within a neighborhood $\boldsymbol{\Omega}$ of the node $\boldsymbol{x}_i$, can be calculated integrating the Gaussian distribution.

$$p_{localization} = p(\boldsymbol{x}_d \in \boldsymbol{\Omega}) = \int_{\boldsymbol{\Omega}} \mathcal{N}(\boldsymbol{\mu_d}, \boldsymbol{C_d}) d\boldsymbol{x_d} \tag{3.18}$$

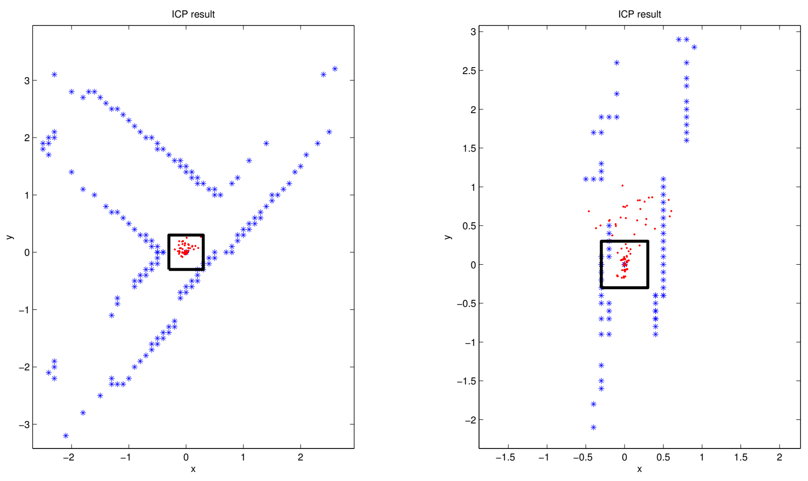
In [Valencia et al., 2011] this expression is simplified using the marginals. The experiments suggest that this approximation is consistent because $\boldsymbol{C_d}$ is close to diagonal. This enables to approximate equation 3.18 with the product of independent variables.

3.4.2 Place Recognition Probability

In order to estimate the probability of relocatization or place recognition at every step of a trajectory, a Monte Carlo algorithm is performed. Location node is randomly sampled with additive Gaussian noise. From the sampled position, a matching algorithm (ICP in our case) is used to compare the laser measurements with the map and try to recover the true position. If the error of the resulting location with the node's estimated location is under a threshold, relocalization or place recognition is considered a success, otherwise a failure. The corresponding probability of relocalization is then simply the proportion of successful tries.

Random noise is also added to simulate dynamic environments (Fig 3.3). Parts of information are removed from the laser scan in order to simulate occlusions.

This algorithm was tested with the laser scan readings from the Rawseeds Data Set [Ceriani et al., 2009]. In Fig. 3.3 it can be seen the results in two situations, a corridor and a corner.



(a) Narrow Distribution (b) Spread Distribution

Figure 3.3: ICP results in XY plane: The Monte Carlo simulations of the ICP produces a distribution of the error (Red), the samples inside the box (Black) are considered success, the rest failures.

3.5 Path Planning in Explored Environments

Based on the expected uncertainty of a path we can evaluate every trajectory. In the case of Explored Environments the graph representing the map is considered known and every possible path is represented in it. In this case we consider the safest path given between two given nodes considering all possible paths.

We perform an exhaustive search for every possible path and evaluate its expected uncertainty. In order to keep the tractability of the algorithm we perform the search in a reduced graph similar to [Carrillo et al., 2012a] where the redundant information is removed.

The graph reduction is done considering that nodes with only 2 edges represent consecutive nodes, these nodes do not produce any new paths, therefore they are not included in the reduced graph (their edges are inherited by its neighbours). Additionally in graph SLAM, any edge beyond the second one represent loop closures. Some algorithms recognize loop closures in consecutive nodes produce a set of consecutive nodes with more than two edges. In terms of possible paths these consecutive nodes produce cycles and multiple paths that are essentially the same. To avoid these multiple alias of the same path, whenever a node with 3 or more edges is found it is checked against its neighbours. If it is associated with a loop closure previously represented in the reduced map, the additional edge is not included.

The reduced graph is formed almost exclusively by nodes in the crossroads, thus producing a limited number of paths. The complete search is implemented with a recursive search for all the simple paths in the reduced graph.

3.5.1 Experiments

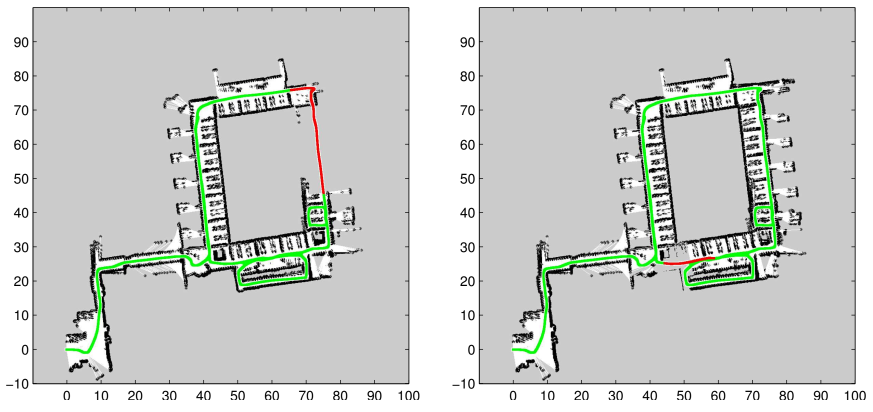
In order to test the algorithm in a realistic environment, we use the Rawseeds Dataset [Ceriani et al., 2009]. It belongs to the University of Milano-Bicocca and it contains conventional office-like features where a robot traverse it, the robot is equipped with Ultrasound sensor, Inertial Measurement Unit (*IMU*), several vision components and Laser Range Finder (*LRF*). In our experiments we consider exclusively the scans coming from the *LRF*.

The algorithm includes the parameters to calculated the probability of localization and probability of recognition. The localization probability is considered using the probability that the position of the robot be within a window of 60cm in x- and y-axis and 60° in angle of the node to be localized. Place recognition probability was estimated sampling from the same window using the node's 360° laser scan.

Unreliable Path Avoidance

The experiment was done to compare the avoidance of unreliable paths. For this purpose, all scans from a certain area of the environment are ignored for all purposes, as if the region was outdoors or completely void of features within the sensor range. Thus the links connecting nodes in this area are based only on wheel odometry.

In order to test the effect on the position of the unreliable zone two different regions void of features are simulated (Fig 3.4). The effect of the magnitude of the uncertainty is simulated multiplying the covariance matrix associated to the links in this zone by 10 and by 100. Namely the scenarios are:



(a) Scenario A and B (b) Scenario C and D

Figure 3.4: Fully Explored Scenarios: Information is removed from the regions inside the blue square to simulate regions void of features are within the sensor range, to simulate wheel odometry the covariance matrix associated to the links in this zone is multiplied by 10,(A and C) and by 100 (B and D)

1. Scenario A: The North East wing in the main building is removed and the covariance matrix of the corresponding links are multiplied by 10

2. Scenario B: The North East wing in the main building is removed and the covariance matrix of the corresponding links are multiplied by 100
3. Scenario C: The South West wing in the main building is removed and the covariance matrix of the corresponding links are multiplied by 10
4. Scenario D: The South West wing in the main building is removed and the covariance matrix of the corresponding links are multiplied by 100

There were a total of 879 sets of start and goal points represented among different scenarios. All possible paths between them are obtained and evaluated according the following criteria:

1. Expected Uncertainty (Our Proposal)

   The criteria to be evaluated consisting in the expected value of the final covariance

   $$Cost = E[\mathbf{C}] = \sum_{h_k \in H} \mathbf{C}_k \cdot prob(\mathbf{C}_k) \tag{3.19}$$

   we drop indices in equation 3.13 knowing the sum is in the final node over all of the whole set of hypothesis H. In order to assign an scalar to this expected matrix we use its determinant $det(E[C])$.

2. Mechanical Work

   The mechanical work criteria the cost function implemented was presented in [Valencia et al., 2013]

   $$Cost = \sum_{nodes} \Delta U_k^+ \tag{3.20}$$

   $$\Delta U_k^+ = \begin{cases} \Delta U_k & if \quad \Delta U_k \geq 0 \\ 0 & otherwise \end{cases} \tag{3.21}$$

   $$\Delta U_k = U_k - U_{k-1} \tag{3.22}$$

$$U_k = \frac{1}{|\mathbf{O}^{-1} + \boldsymbol{\Sigma}_{jj}^{-1}|} \tag{3.23}$$

where $\mathbf{O}$ is related to the odometric uncertainty between nodes and $\boldsymbol{\Sigma}_{jj}$ is the marginal covariance of the nodes.

3. The sum of the D-Optimal

   This criteria was proposed in [Carrillo et al., 2012a] is implemented with:

$$Cost = \sum_{nodes} det(\Sigma_{jj}) \tag{3.24}$$

4. Distance

   The shortest path between the given nodes.

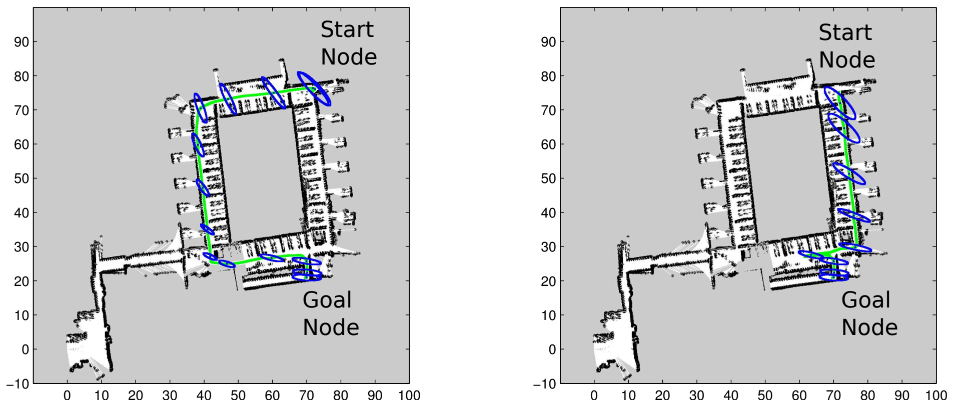
The success is measured considering the the avoidance of the unreliable zone. For every criteria the path that minimize its cost is chosen, the path that minimizes the cost but traverse the unreliable zone is considered an error. The Error Rate in the table 3.1 is the fraction of the total number of paths where minimizing a criteria implies traversing the unreliable zone.

Scenario	Expected Uncertainty	Work	Sum of D-Optimal	Distance
A	0	0	0	15.1%
B	0	0	0	15.1%
C	0	12.1%	47.1%	28.5%
D	0	0	44.1%	28.5%

Table 3.1: Error Rate of the four path planning algorithms analyzed in this work, for the four different scenarios considered.

In the first two scenarios, (Table 3.1) the size and the position allowed every criteria to avoid the unreliable zone. This simulation allowed to establish a baseline where every algorithm worked.

In scenarios C and D, the probability of getting the loop closure with the simulated magnitude of the uncertainty was negligible, however the error rate for



(a) Path with Minimal Work (b) Path with Minimum in the rest of Criteria

Figure 3.5: Error Example: The minimal work criteria tend to choose paths that return to low uncertainty nodes and then go to the goal even if this means going trough the zone void of features (a). The expected uncertainty increases wildly when traversing it, so the other path is chosen (b)

methods was bigger. In the case of the “Sum of D-Optimal" criteria the position effect is clear.

Based on the observations we can described some characteristics of the criteria for choosing the safest path. The distance is used as a baseline because sometimes the algorithms choose longest path even when the shortest avoid the unreliable zone.

For the “Sum of D-Optimal” the effect of the origin node is prominent, the nodes located at the farthest distance of the origin have associated the biggest covariance matrices, consequently the paths that minimize this criteria tend to avoid these regions even when it traverses the unreliable zone or a longer path.

In The “Mechanical Work” criteria the cost of traversing from one node to another with lower uncertainty is essentially “0” (equation 3.21) as a consequence the paths that minimize this cost are the one that traverse to low uncertainty nodes and then go to the goal, even when this means traversing the unreliable zone or a longer path. This situation occurred in the experiment described in figure 3.5.

The “Expected Uncertainty” of the paths that traverse the unreliable zone are very high. After accumulating uncertainty during the traversal of the zone void of features, the probability of getting a loop closure is very low, consequently the probability of getting a large uncertainty is also large and thus the expected uncertainty. The paths that avoids the unreliable zone always minimize this cost.

3.6 Path Planning in Partially Explored Environments

In partially explored environments the graph is considered incomplete, consequently potential paths are missing from the graph. In [Valencia et al., 2011] the authors consider the need for increasing the connectivity of the graph from the SLAM algorithm to find the optimal path. They proposed connecting the nodes whose probability of being closer than a predefined distance is above a threshold. The solution we propose is based on the Generalized Voronoi Graph (*GVG*), this technique solves the problem of connectivity between nodes close by. Additionally, in partially explored environments it can identify potential connections between nodes before the zone is fully explored as long as the sensors perceive free space in front of the vehicle connecting them.

The (*GVG* have been used in [Thrun, 1998] in the construction of topological maps, it is based on the generalized Voronoi Diagram of a shape. Points that lie at the same maximum distance of the contour of the shape form a diagram that condenses the information of the shape. The resulting diagram is transformed into a graph where nodes are the branch points and end points of the diagram and the edges are the paths that connect them.

Once we obtain the *GVG* we fuse it with the graph result from the graph SLAM algorithm. The edges from the Voronoi graph are mapped as a sequence of nodes in the SLAM based graph by choosing the closest node within a neighbourhood, if none is found no correspondence is made. The resulting graph includes the edges coming from the *GVG* and the covariance associated to these edges is the one associated to the odometry between them.

In this scenario a minimal increase in uncertainty can be compensated by a major reduction in distance, pondering this alternative we propose a cost function that weights the final expected uncertainty with the traversed distance. We use the product as cost function because the weight in the percentage increment is the same for both, i.e. a minimal increase in uncertainty is compensated by a major reduction in distance. The easiest way of doing this is using the product as cost function

$$Cost(T_i) = Distance(T_i) * Expected\_Uncertainty(T_i) \tag{3.25}$$

where T_i is the trajectory been evaluated. This parameter-free cost function avoids the necessity of tuning constants, thus increasing its robustness.

3.6.1 Experiment

In this experiment we test the feasibility of traversing unexplored areas than can result in shorter but still reliable paths. Similar to experiment in fully explored environments we use the rawseed dataset [Ceriani et al., 2009] with the same parameters for the probability estimation.

To simulate partial exploration, only part of the provided graph map is used. Because laser sensors can see forward several meters, a potential path can be obtained using the Voronoi Graph of the Occupancy Grid. Paths distances were extracted from the Voronoi Diagram in pixels and then transformed into meters.

The implementation of the GVG is based on the Matlab ® implementation of the thinning algorithm described in [Lam et al., 1992] applied to the image representing the occupancy grid.

In this scenario two alternative trajectories are evaluated, as it can be seen in fig. 3.6. The first one along the available graph and the second considering the unexplored area.

As a consequence of connection with the Voronoi graph a new path connecting the last position of the robot with the goal position appears, after setting the covariance matrix associated to the odometry of connecting these node the calculation are represented in table 3.2.

	Distance	Expected Uncertainty	Cost
Graph SLAM	31.7m	2.97	94.15
Traversing Unexplored	15.0m	3.11	46.65

Table 3.2: Path Comparison

The expected uncertainty of the traversing path reveals that with a small increase in uncertainty of 4,7%, the distance is greatly reduced, in a 52 %, these values are clearly represented in the cost function comparing the 94.15 versus 46.65.

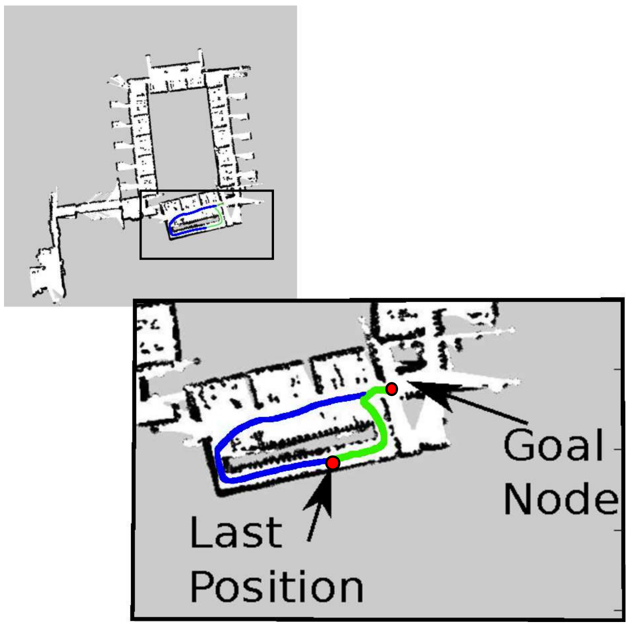


Figure 3.6: Paths in a partially explored map, traversing the unexplored (GREEN or light gray) there is a great reduction in distance with a negligible increase in the expected uncertainty compared to regular path traversing (BLUE or dark gray)

With this experiment we demonstrate that the topological decomposition of the environment increases the capabilities of our algorithm, it solves the problem of lack of connectivity reported in [Valencia et al., 2011] and it also determines when the risk of traversing unexplored regions is compensated with the savings in the distance traversed.

3.7 Conclusions

The path planning algorithm proposed evaluates alternative paths by estimating the final expected uncertainty. As can be seen in the experiments, this metric robustly avoids unreliable paths by quantifying not only the amount of uncertainty between in the links composing it, but also the probability of getting relocalized within the neighbourhood of each node.

Results presented in this work suggest that our criteria consistently avoids unreliable paths, unlike alternative metrics evaluated in this work.

This algorithm is generalized to partially explored environment expanding the graph with the edges coming from the Generalized Voronoi Graph and the definition of a cost function effectively including shorter paths as long as it contains only loop closures with low relative uncertainty.

This work was publish in the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016 [Fermin-Leon et al., 2016].

Chapter 4

Graph Extraction from Grid based Maps

4.1 From Maps to Graphs

When the problem of autonomous exploration is remapped as a graph problem most of the graph theory can be used to solve it efficiently, we have already seen this of the new approach in previous chapters with the graph SLAM formulation and the graph based path planning. The graph we have been using so far consist in the set of positions to be optimized as nodes and the measurements as the edges, however we observe that the augmented version of this graph that includes traversable edges not present in the original graph improves path planning algorithm.

The quality of the solution depends on how accurate the problem is represented by the graph. Because we are interested in mapping it is very important that the graph represents the map accurately. Given a map we seek to divide it into meaningful pieces connected among others in pairs. These regions will become the nodes in the graph and the edges will represent their connections to the neighbouring regions.

The maps divided this way are known as “topological maps", whose name is derived from *topology*, the branch of mathematics that studies the properties of objects invariant to deformations, in this case the connection between regions is the property preserved.

The problem of producing topological maps based on regular maps is known as “Topological segmentation". In the general case the number and characteristics of the regions are usually not well defined, the same space can have different topological representations. Usually the number of regions and its characteristics is defined by the application. In the topological maps of rooms, also known as “Room Segmentation" [Bormann et al., 2016], the regions can be narrowed to rooms and corridors and thus properly segmented.

In preparation for autonomous exploration we first seek for a segmentation in convex regions, this property guarantees that once the robot traverse it every element in it becomes visible. Secondly we need the segmentation to be incremental and stable, because we only have access to a partially explored map during exploration and we need this incremental segmentation to be similar to the final one.

4.2 Related Work

One interesting approach is proposed in [Bhattacharya et al., 2012] because it represents an important advance into mathematical guarantees, they represent the trajectories from the given source point to the destiny point in topological classes, the number of classes is defined by the number of obstacles and the ways to navigate around it. However this criteria fails if there are not trajectories or even in indoors spaces with several rooms but not obstacles. A similar approach to the graph extraction is presented in [Urcola et al., 2015] where the set of possible paths is clustered into topological classes, these classes are then represented with a discrete set of nodes and edges that constitutes the topological graph.

The most popular approach consist in beginning with the map represented as an occupancy grid and transforming it into an image. In this representation the image processing techniques are used to extract the topological graph.

In particular, according to the survey presented in [Bormann et al., 2016], the most popular image based approach to segment floor plans is the one based on the Generalized Voronoi Graph. Additionally, when they compare it with methods based on features, distance transform or mathematical morphology in their dataset, the Voronoi segmentation is the best approximation of the ground truth segmentation in most of the cases.

4.2.1 Voronoi-based Segmentation

The Generalized Voronoi Diagram can be defined as the centers of the maximal disks inscribed in the free cells image. These points form a set of intersecting lines. The Generalized Voronoi Graph (GVG) is built defining the intersecting points as nodes and the lines connecting them as edges.

The original formulation for segmentation based on Generalized Voronoi Graph comes from [Thrun, 1998]. In this work Thrun defined the critical points as the points on the Voronoi diagram that minimize clearance locally, i.e. doors. The regions are defined by the nodes in the GVG and extends until the critical points are reached.

Several heuristics can be introduced to improve GVG based segmentation. In [Bormann et al., 2016] Bormann *et al.* prune the extracted GVG collapsing the leave edges into the node points of their origins. Next, they define a set of critical point candidates, and then some heuristics in point selection, as well as merging, are applied. They classify the traditional approaches according to the criteria each follows: Morphological, Distance Transform, GVG and Features. An implementation of each approach is tested in 20 images, corresponding to indoor environments in different configurations (furnished and without furniture) and compared with human label segmentation. In this dataset, the Voronoi-based approach scores the highest quality in terms of precision and recall. However it exhibits one of the highest processing times compared with the other approaches.

Incremental versions of GVG improves its performance, increasing the possibility of being used on line in exploration tasks [Van Zwynsvoorde et al., 2000] [Choi et al., 2009] [Liu et al., 2011]. The most recent algorithm is presented in [Liu et al., 2015], they incrementally construct a Voronoi graph-based topological segmentation for semi-structured environments (unlike room segmentation). They report a high dependence of processing time with respect to cell size: the change of the cell size from 0.1m to 0.25m reduces the maximum processing time from 50 to 2.5 seconds[1].

[1]Processing time extracted from the figure 13 in [Liu et al., 2015]

4.2.2 Contour-Based Segmentation

Contour-based Segmentation is related to “Part Segmentation", a type of segmentation commonly associated to 3D meshes [Shamir, 2008]. Its goal is the segmentation of the object represented by the mesh into meaningful parts. When applied to 2D, the image is represented as a set of closed contours and processed to generate sub-contours. In the contour representation lies the most significant difference between this approach and the current approaches used for topological segmentation.

Unlike the pixel based segmentation, contour-based complexity doesn't depend on the image size (a rectangular shape can be represented by only 4 contour points), but rather in the characteristic of the contour.

One common criteria for part segmentation is convexity, finding the exact convex decomposition of the contour. According to [Lien and Amato, 2006], this decomposition can be costly to construct and can result in representations with an unmanageable number of components. Furthermore, if the polygon has holes the problem is NP-hard. The alternative they propose is the Approximate Convex Decomposition (ACD). In this way they can produce a hierarchical representation of convex decompositions of different levels of approximation. They claim the complexity of this decomposition to be $\mathcal{O}(nr)$ with n the number of vertexes and r the number of notchs (the most significant non-convex feature). This decomposition provides an insight of the characteristics of the shape.

4.3 Dual- Space Decomposition

Other recent works in part segmentation point to decompose two-dimensional shapes into functional and visually meaningful parts include the Dual-Space Decomposition [Guilin et al., 2014]. In this work the authors demonstrate that the resulting decomposition is statistically similar to the one produced by a human in a dataset composed by complex shapes from the MPEG7 database with added noise and the contour of some images, such as trees, human crowd, and bikes.

In Dual Space Decomposition (DuDe) the input is a polygon P, possibly with holes in it. The polygon and the holes are represented as a sequence of 2D points whose ends are assumed to be connected (same as a closed contour).The algorithm decomposes the polygon P based on the decomposition of the complement $\overline{P}$

(dual-space).

The following definitions are useful to understand the algorithm.

4.3.1 Definitions

Convex Hull

The smallest convex set which includes the polygon, CH.

Pockets

The regions (p_i) inside the convex hull but not inside the contour. Basically the regions the polygon P needs to add to become CH.

$$\bigcup p_i = \overline{P} \tag{4.1}$$

$$\overline{P} \bigcup P = CH \tag{4.2}$$

Bridge

The segment of the convex hull that limits each *pocket*.

Convexity

A polygon is considered convex if for any pair of points inside the polygon, every point on the straight line segment that joins them is also inside the polygon. As mentioned above the smallest convex set which includes a polygon P is called the convex hull. Based on these definitions, several methods to measure the convexity of a polygon [Zunic and Rosin, 2004] based on either the probability of finding a straight line segment inside the polygon or in the similarity to its convex hull.

According to [Lien and Amato, 2006], these global measurements of convexity are not useful to identify where and how the polygon should be decomposed. They instead use the measure of concavity (complementary to convexity)

$$concavity(P) = \max_{x \in \partial P}\{dist(x, CH)\} \tag{4.3}$$

with P the polygon, ∂P its contour and $dist(x,CH)$ is the shortest distance (or path) from the point $x \in \partial P$ to the convex hull CH.

4.3.2 Decomposition

The Dual-Space decomposition works iteratively. Firstly the convex hull of the polygon is found, this defines the *pockets* and its corresponding bridges. The contour of these *pockets* is iteratively decomposed: every *pocket* is treated as a polygon with a new convex hull and new *pockets* that will be decomposed once again till no more pockets are found.

Because of the hierarchical relation of the DuDe decomposition (every polygon is the *pocket* of a parent polygon), it is possible to measure the distance from every point to its closest bridge, and next the distance from this bridge to the bridge in the parent polygon, and so on till the convex hull is reached. This path defines the distance to the convex hull and thus the concavity.

The set of points with concavity bigger than the concavity threshold are the pivot points of the cut. Finally the set of cuts, using these points, that maximizes the convexity (minimize the concavity) in the final decomposition, is chosen.

4.3.3 Implementation

In order to produce a topological segmentation of the 2D grid based map we transform the occupancy grid into a set of polygons, then we use the function `DuDe_Segment`[2] from [Guilin et al., 2014] to segment them. This function decomposes a polygon P represented by a set of n boundaries b_0, b_1, b_2, b_{n-1}, where b_0 is the external boundary and $b_{k>0}$ are boundaries of the holes, until a maximum allowed concavity is reached (Algorithm 2).

The first step consists in transforming the occupancy grid into a binary image, this is a common step for every topological map segmentation based on occupancy grids. This consists in applying thresholds to the image to obtain the obstacles and the free space. In the case a real robot is used the obstacles and free space are modified according to the robot's characteristics. We transform the binary image into a polygon representation using the *OpenCV* ® function `findContours`.

[2]Public code of Dual Space Decomposition can be found in `http://masc.cs.gmu.edu/wiki/Dude2D`

Using the retrieval tree mode in this function, we can find all the external contours (Parents) and the set of contours inside them (CHILDREN) in the binary image. Consequently, every set formed by a parent contour and the set of its child contours represents a polygon.

Looping through every external contour and its children, we obtain every decomposition and build the *Decomposed* set aggregating every decomposition.

Algorithm 2: Dual Space Decomposition of Binary_Image (`DuDe_Binary`)

```
Input      : Binary_Image, Max_Concavity
Output     : Decomposed = $\{c_0, c_1, \cdots\}$
             Contour $c_i = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots\}$
Variables: Contour_Sets :
                 Parents = $\{P_0, P_1, \cdots P_k\}$
                 Child$_i$ = {}
                 Dude_Contours = {}
             CHILDREN = $\{\text{Child}_0, \text{Child}_1, \cdots \text{Child}_k\}$
(Parents, CHILDREN) = findContours$_{tree}$(Binary_Image)
for every contour $P_i \in$ Parents do
    Dude_Contours =
        DuDe_Segment({$P_i$, Child$_i$}, Max_Concavity)
    Decomposed = Decomposed $\cup$ {Dude_Contours}
return Decomposed
```

4.4 Incremental Decomposition

During exploration, whenever a new a scan is processed, some new information is aggregated to the map. This information usually only modifies the contours connected to the new scan, leaving the rest unchanged (except during loop closings). This observation permits the implementation of the incremental version of the decomposition.

In the SLAM context a loop closure can potentially modify the whole map, in this scenario the whole map is decomposed. In this worst case scenario the processing time is the same as the batch implementation, in the regular exploration scenario (no loop closure) the expected processing time is lower.

The incremental decomposition is done using the algorithm 3, we now explain it step by step. The first step consists in obtaining the difference of the binary image between time step $k-1$ and k, the received occupancy grid is transformed into a binary image, it is then compared to the previous map image to produce the difference image.

The second step consists in processing the difference image to update the decomposition. The difference image is transformed into the set of contours "*Difference*" using `findContours`, with simple contour approximation and retrieval mode set as external (no hierarchy needed). Defining that two regions are in "contact" when they share points in their bounding contour, the set of contours are classified in:

1. *Unchanged*, Contours in previous decomposition with no contact with the "*Difference*" set.
2. *Modifiable*, It consists in expanding the set "*Difference*" with the contours in the previous decomposition in contact with it.

The set "*Modifiable*" is transformed into a reduced binary image and processed using the Dual-Space Decomposition of binary images `DuDe_Binary`. Finally the result of this decomposition is aggregated to the set of unchanged contours to produce the new decomposition. The new decomposition is stored for the next iteration (Algorithm 3).

4.5 Experimental Results

The first experiment evaluates the performance of the algorithm in the structured environment scenario (room segmentation). We evaluate our approach in both of its versions, batch (`DuDe_Binary`) and incremental, compared to the traditional approaches in the room segmentation dataset of [Bormann et al., 2016] in terms of time, precision and recall.

Algorithm 3: Incremental Decomposition

```
Input     : Difference_Image, Max_Concavity
Output    : Decomposed = $\{c_0, c_1, \cdots\}$
Variables: $Contour\_Sets$ :
                Previous = $\{p_0, p_1, \cdots\}$
                Difference = $\{d_0, d_1, \cdots\}$
                Modifiable = $\{\}$
                Unchanged = $\{\}$
                Modified_Contours = $\{\}$
DIFF = findContours$_{Ext}$(Difference_Image)
for every $p_i \in Previous$ do
    for every $d_j \in DIFF$ do
        if contours_connected($p_i, d_j$) then
            Modifiable = Modifiable $\cup$ $\{p_i, d_j\}$
        else
            Unchanged = Unchanged $\cup$ $\{p_i\}$
Expanded_Image = draw_contours(Modifiable)
Modified_Contours = DuDe_Binary(Expanded_Image, Max_Concavity)
Decomposed = Unchanged $\cup$ Modified_Contours
return Decomposed
```

The second experiment evaluates the performance in poorly structured environments. We use the three exploration sequences described in [Liu et al., 2015], namely a Parking Lot, a Tunnel and a Building. The lack of ground truth implies the quality of the decomposition cannot be determined using a precision and recall, thus we evaluate the algorithm in terms of the average processing time between the batch and the incremental version and show visual results of both segmentation methods.

The algorithms run in a Laptop Intel® Core™i7-2620M CPU @ 2.70GHz×4, under the robot operating system framework (*ROS*). Our *ROS* package is based on the function *IncrementalDecomposition* [3] (Algorithm 3) that takes the occupancy grid *ROS* message and the previous decomposition, processes it and then decomposes it with the public implementation of [Guilin et al., 2014].

The optimal value for the parameter "*Max_Concavity*" was found to be 2.5 *meters* for every scenario in both experiments, independent of the size, resolution and type of environment.

4.5.1 Environment with High Structure (Room Segmentation)

We use the maps proposed in [Bormann et al., 2016], consisting in 20 scenarios. Every scenario contains images of the rooms with and without furniture and the human segmentation of the image. We choose the evaluation in terms of time, precision and recall because the time can be compared (the characteristics of our computer are roughly the same) and precision and recall allows the comparison with a human labeled image.

In this context the precision and recall is evaluated between the human labeled regions R_{human} and the segmented R_{seg}, for every correspondence between regions the true positive tp represent the pixels in both regions, the false positives fp are the pixels in the segmented region but not in the human labeled and the false negatives fn are the pixels in the human labeled region but not in the segmented. Consequently $tp + fp$ is the region R_{human} and $tp + fn$ is R_{seg}. This can be summarized in the following equations.

[3]Public Implementation can be found in `https://github.com/lfermin77/Incremental_DuDe_ROS`

$$Precision = \frac{tp}{tp+fp} = \frac{R_{human} \cap R_{seg}}{R_{human}} \tag{4.4}$$

$$Recall = \frac{tp}{tp+fn} = \frac{R_{human} \cap R_{seg}}{R_{seg}} \tag{4.5}$$

In order to evaluate the incremental version of the algorithm we simulated the exploration with a virtual robot with a radius of laser range of 5 meters and perfect localization. In this situation the map is constructed incrementally and then processed with our algorithm. The precision and recall are calculated in the final map and the time reported is the average processing time per simulated scan.

Because the maps presented have different sizes, number of rooms and complexity, we present the results for every map using histograms for precision and recall (Fig 4.1) and time (Fig 4.2), .

The best result in precision and recall reported in this dataset correspond to the Voronoi-based segmentation [4], in Table 4.1 the average values are compared. The contour-based approach has a performance similar to the best evaluated in this dataset, in particular our incremental version does not sacrifice any quality in pursue of increased velocity.

		Voronoi	Batch	Incremental
No Furniture	recall	95.0 ± 2.3	86.3 ± 9.7	87.5 ± 10.7
	precision	94.8 ± 5.0	94.1 ± 3.1	93.8 ± 3.1
Furnished	recall	86.6 ± 5.2	85.8 ± 10.3	87.5 ± 10.7
	precision	94.5 ± 5.1	94.0 ± 3.2	93.8 ± 3.1

Table 4.1: Averaged Precision and Recall

The diversity on the size of the images in the dataset implies the time needed to process them is not consistent (the time reported for the Voronoi-based segmentation is $12.0 \pm 14.2s$). To evaluate the processing time we use the histograms in

[4]The recall in the Not Furniture scenario is 98.1% ± 2.4% for the mathematical morphology based, however, in the rest of scenarios the Voronoi-based clearly overcomes it.

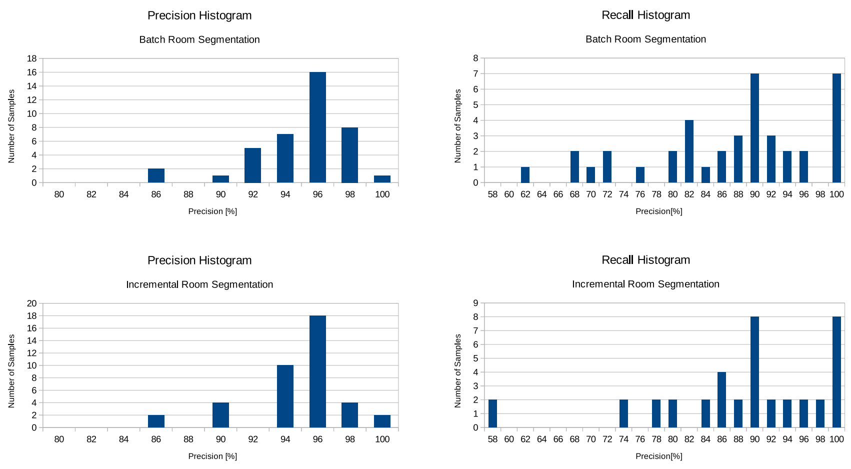
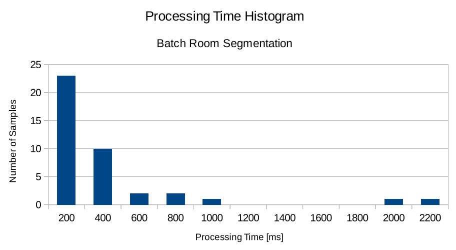


Figure 4.1: Precision and Recall Histograms, In spite the diversity of the maps the precision is very localized, in both cases 95% of the maps scores precisions above 94%. The effect of the heterogeneity of the maps in terms of complexity and size affects importantly the recall, we score recalls above 84% in the batch mode and 86% in the incremental mode for the 85% of the maps



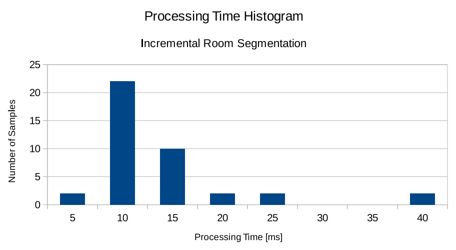


Figure 4.2: Processing Times, Using contour based segmentation 95% of the maps in the dataset are processed under 1s, with our incremental version the processing time is below 25 ms for the 95% of the maps

fig.4.2. The batch version of the contour-based segmentation is one order of magnitude faster than the Voronoi-based. Our incremental approach further reduces these processing time to tens of miliseconds.

Summing up, the proposed Contour-Based Segmentation algorithm obtains similar quality to the state of the art approaches, one order of magnitude faster, without any application of oriented heuristics. Our incremental algorithm further reduces these times, without sacrificing quality.

Qualitatively, the decomposition is similar to the human labeled, with the difference that corridors are usually over segmented (Fig. 4.3), this is because this algorithm is based on the convexity of the regions, one “*L*" shaped corridor is always segmented (as a minimum) in two regions.

4.5.2 Environment with Low Structure

The lack of ground truth for segmentation in poorly structured scenarios implies the evaluation of the quality can only be made using parameters being affected by it. The number of regions can indicate if a region is over segmented (value too high) or under segmented (value too low).

As expected, the average time for the incremental version is lower than the batch version (Table 4.2). On the other hand, the histogram of the processing time shows an interesting behavior (fig 4.4): in the case of the incremental version, the distribution is almost completely localized independent on the map resolution, but the batch version grows monotonically spreading through all *X* axis.

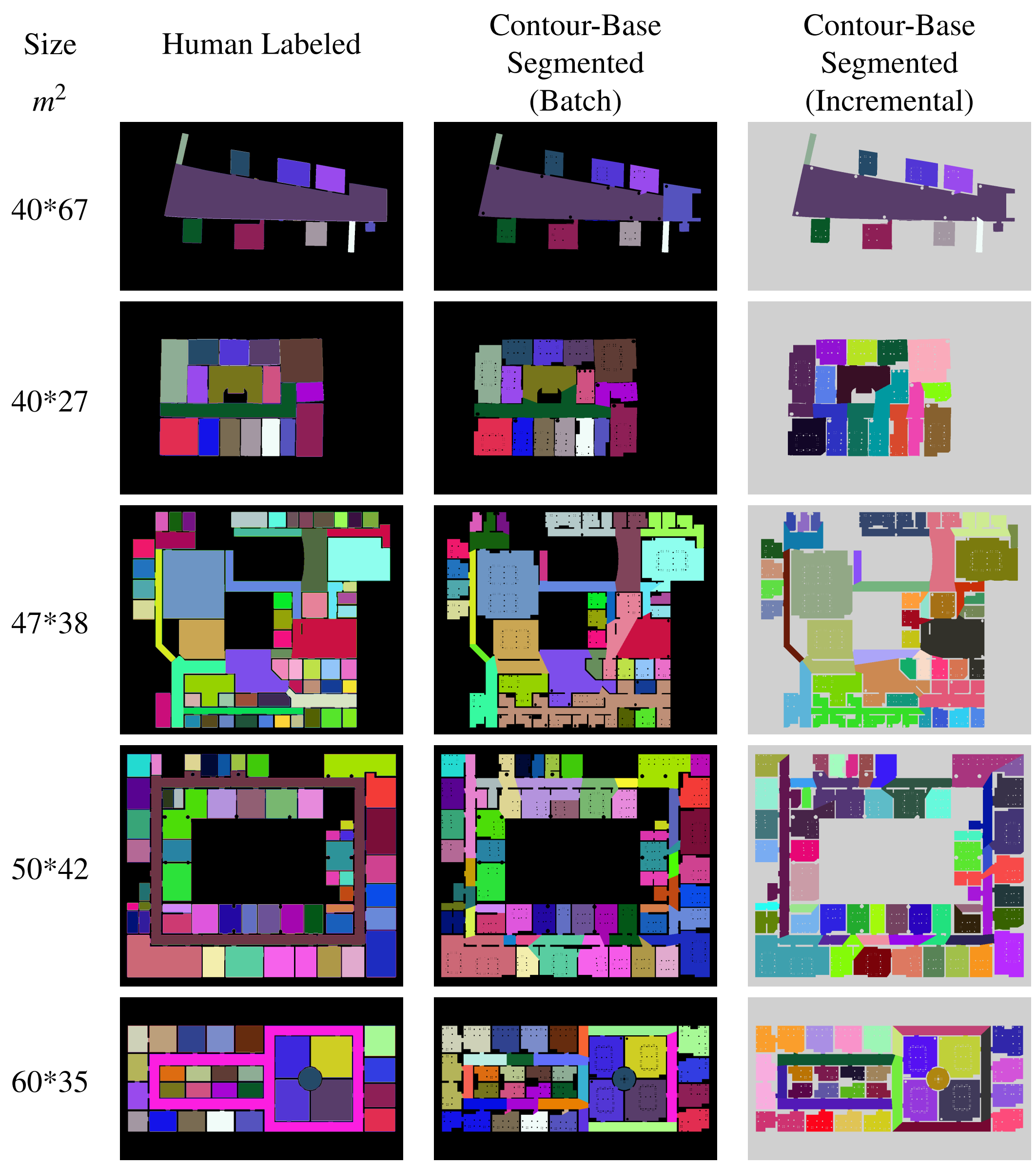

Size
m^2
Human Labeled
Contour-Base Segmented (Batch)
Contour-Base Segmented (Incremental)
40*67
40*27
47*38
50*42
60*35

Figure 4.3: Some results on the datasets in [Bormann et al., 2016], Although the corridors are over segmented, the rooms are usually well differentiated

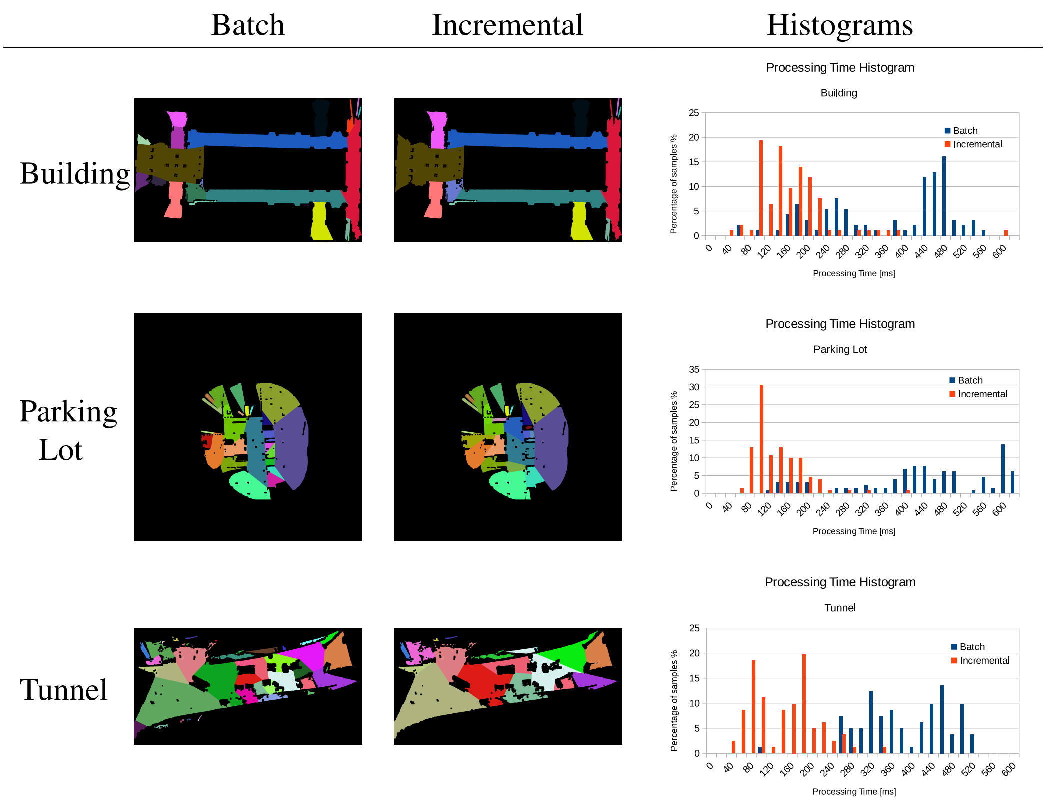


Figure 4.4: Exploration sequences in [Liu et al., 2015] are processed with the incremental and the batch version of the algorithm. Both methods produce visually appealing segmentations. The number of regions in the incremental version is slightly lower than batch counterpart. The histograms show the processing time of our incremental algorithm is under 200ms for the 95% of the frames processed, on the other hand in the batch version the time to process increases with every frame producing and spread distribution

Scenario	Resolution	Processing Time [ms]		Number of Regions	
	[m/pixel]	Batch	Incremental	Batch	Incremental
Parking Lot	0.050	427 ± 156	104 ± 51	16	12
Building	0.025	331 ± 131	137 ± 76	13	11
Tunnel	0.025	376 ± 85	135 ± 63	21	20

Table 4.2: Segmentation in low structure environments

Figure 4.4 shows that when the number of regions are similar (table 4.2), both decompositions are similar. Additionally it can be observed that the number of regions produced by the incremental algorithm is consistently smaller in this dataset. This seems to indicate a tendency to under segment environments with low structure, compared to the batch version.

Figure 4.5 shows the process of the incremental segmentation, where after successive maps update the over segmented regions are merged.

4.6 Conclusions

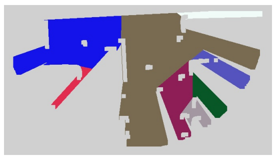
We introduce a new contour-based approach for topological segmentation. This segmentation has the following advantages over traditional approaches:

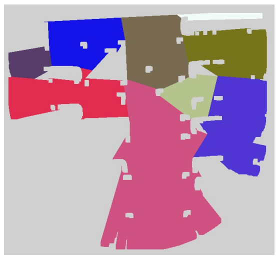
1. Velocity: batch times are comparable to the simplest segmentation cases, with quality comparable to the most complicated cases; an incremental version allows on-line use.

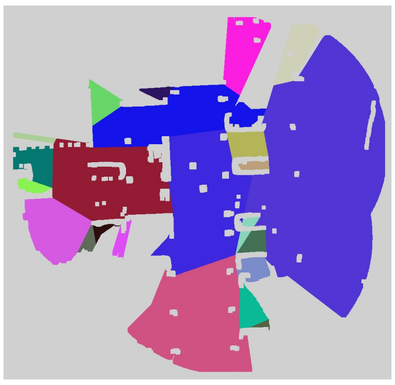
2. Grid Size Independence: contour based decomposition depends only on the contour characteristics, not in the image size.

3. Quality of segmentation: the resulting segmentation scores high in precision and recall in the comparison with human labeling.

4. Flexibility: it works in environments ranging from highly to poorly structured, with comparable results to human labeling.


(a)


(b)


(c)

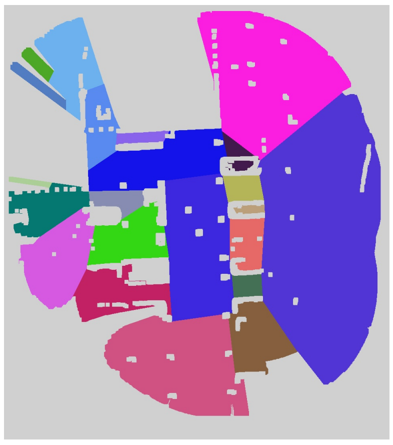

(d)

Figure 4.5: Time Sequence of the Incremental Topological Segmentation: Regions over segmented in the first scan (top) are eventually merged in successive map updates. The regions (and their colors) are updated only in the contours connected to the new information

5. Tuning of a single parameter: the only parameter in the algorithm is the maximum concavity ($Max\_Concavity$) allowed, it is measured in meters. Empirically, the same value ($2.5m$) is use for highly and poorly structured environments, this hints this parameter could be use as a constant for map segmentation.

This work was presented in the IEEE International Conference on Robotics and Automation 2017 [Fermin-Leon et al., 2017a].

Chapter 5

Optimal Information Path

In order to define an optimal exploration we need to define some measurement of performance and then we can variate the variables we can control in order to optimize that measurement. In active SLAM we focus in the exploration for reconstruction and we know we have a limiting behavior, the longer the path the better the map. Although many other exploration variables can be associated to the exploration, the optimization of the length of the path and the quality of the reconstruction constitute the quintessential exploration problem.

The length of the path can be measured directly in terms of meters traveled by the robot and it is also related other terms like exploration time and power consumption. The quality of the map can be a more abstract idea, because the SLAM problem after linearization can be seen as a multivariate Gaussian distribution the covariance matrix, and its inverse the information matrix, is a good indicator of the quality of the reconstruction i.e. the better the information matrix the better the reconstruction. Considering we have the function that maps the information matrix to a real number, the problem of *Optimal Exploration* can be properly defined.

Using a graph based approach we are able to model mathematically the exploration problem. We are able to pose this model as convex optimization and we demonstrate the solution is unique, and that the maximum optimal is asymptotically reachable. Based on the results on the floor plans in dataset we are able to propose a heuristic that achieves performance similar to the optimal behavior.

We propose the Eulerian Path as a policy for exploration and demonstrate that the obtained paths achieves similar performance to the mathematical optimal. The

Eulerian Path can be found incrementally, consequently our exploration policy proposal based on an incremental map is similar to the solution of the full problem of the optimal exploration when the full map is known.

5.1 Related Work

In [Khosoussi et al., 2014] they observed the relation between the optimality criteria used to evaluate the active SLAM algorithms and the connectivity indices of the associated graph being built during the Graph-based SLAM. Based on this link we can now use the latest work in connectivity optimization to produce an efficient exploration.

Some important connectivity indices of the graph comes from the analysis of the eigenvalues of one of the matrix associated to the graph, the *Laplacian Matrix*. In one of the seminal works on connectivity optimization [Boyd, 2006] Boyd considers the problem of optimization of several graph problems related to the Laplacian eigenvalues like algebraic connectivity and minimum total effective resistance, these problems are linked to the maximization of the E-Optimal and A-Optimal criteria. The optimization of the determinant of a generic matrix has been addressed in [Vandenberghe et al., 1998] and it is related to the D-Optimal maximization. Although most of these problems has not closed form solution Boyd demonstrated that these kind of problems are convex, which means that they can be solved efficiently using convex optimization techniques like semi-definite programming.

The work on eigenvalues optimization has seen its application in the robotic community in the multirobot scenario. In consensus based robot network the algebraic connectivity is related to the convergence rate. In [Zavlanos et al., 2011] they optimize the configuration of the robot in order to maximize this value.

Inspired by the work [Vandenberghe et al., 1998] we propose a generic formulation of exploration in terms of any of the commonly used optimality criteria and demonstrate the optimal is unique.

5.2 From Information Matrix to Laplacian Matrix

In [Khosoussi et al., 2014] they reveal the relation between the optimality criteria and the indices of the underlying graph representation of the SLAM problem. Considering that all measurements have the same noise covariance matrix $\mathbf{\Sigma}$, for linear functions they show that information matrix can be extracted only combining it with the matrix associated to the structure of the graph called *Graph Laplacian*. We will expand this work and define which conditions must be fulfilled in order to be applied in 2D and 3D SLAM scenarios.

5.2.1 Graph Laplacian

Let the graph G be defined as $G = (V,E)$, the set of nodes $V = V(G) = \{v_1, v_2, \cdots v_n\}$ the set of numbered nodes of size $n = |V|$, the set numbered edges $E = E(G) = \{e_1, e_2, \cdots e_m\}$ of size $m = |E|$. For each edge $e_j = \{v_i, v_k\}$ we define v_i and v_k the nodes it connects. For clarity we will use "i" when counting nodes and us "j" when counting edges.

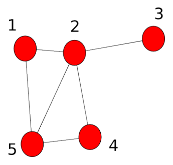
The Laplacian Matrix of a graph $\mathbf{L}$ of size $n \times n$ can be interpreted as a particular case of the discrete Laplace operator. It can be expressed in terms of other matrices associated to the graph, those are:

Adjacency Matrix (A)

Square matrix of size $n \times n$ where the columns and rows represent the nodes of the graph. The term $A_{i,j}$ is set to 1 if nodes v_i and v_j are connected, and 0 otherwise (Figure 5.1). It is only defined for simple graphs, i.e. graphs without multiple edges or self loops.

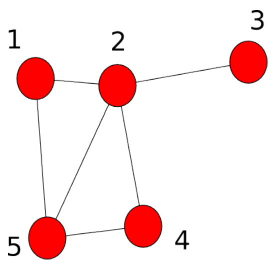
Degree Matrix (D)

Diagonal matrix of size $n \times n$ where the terms $\mathbf{D}_{i,i}$ are the degree of the nodes, i.e. the number edges connected to the node v_i.



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Figure 5.1: Adjacency Matrix



$$\mathbf{D} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

Figure 5.2: Degree Matrix

Vertex-Edge Incidence Matrix (Q)

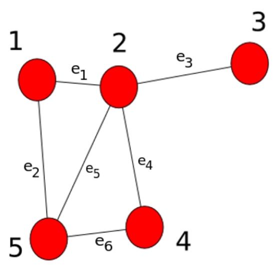
Rectangular Matrix of size $n \times m$ that relates the vertex with edges. The columns of the matrix is formed for the vectors $\mathbf{q}_j$ associated to the edge e_j in the form $\mathbf{Q} = (\mathbf{q}_1|\mathbf{q}_2|\cdots|\mathbf{q}_m)$ where the terms in the vector are $[\mathbf{q}_j]_i = -[\mathbf{q}_j]_k = 1$ if the edge e_j connects the nodes v_i and v_k and zero elsewhere.

The Laplacian matrix can be calculated as

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{5.1}$$

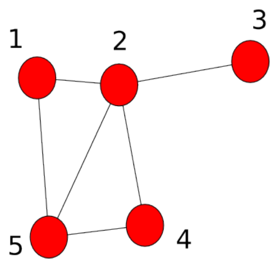
Therefore, in the Laplacian matrix the diagonal entries are given by the degree of the nodes and the off-diagonal entries are set to -1 when the nodes are connected.

We can also define the Laplacian matrix in terms of the vertex-edge incidence matrix



$$\mathbf{Q} = \left(\mathbf{q}_1 \mid \mathbf{q}_2 \mid \mathbf{q}_3 \mid \mathbf{q}_4 \mid \mathbf{q}_5 \mid \mathbf{q}_6 \right) = \left(\begin{array}{r|r|r|r|r|r} 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 0 & -1 & -1 \end{array}\right)$$

Figure 5.3: Incidence Matrix



$$\mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 2 & -1 \\ -1 & -1 & 0 & -1 & 3 \end{pmatrix}$$

Figure 5.4: Laplacian Matrix

$$\mathbf{L} = \mathbf{Q}\mathbf{Q}^T \tag{5.2}$$

Using equation 5.2 we can observe the effect in the Laplacian of every single edge.

$$\mathbf{L} = \mathbf{Q}\mathbf{Q}^T \tag{5.3}$$

$$= (\mathbf{q}_1|\mathbf{q}_2|\cdots|\mathbf{q}_m)\begin{pmatrix} \mathbf{q}_1^T \\ \hline \mathbf{q}_2^T \\ \hline \vdots \\ \hline \mathbf{q}_m^T \end{pmatrix} \tag{5.4}$$

$$= \mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_2\mathbf{q}_2^T + \cdots + \mathbf{q}_m\mathbf{q}_m^T \tag{5.5}$$

Defining a new matrix $\mathbf{E}_j = \mathbf{q}_j\mathbf{q}_j^T$ associated to the edge e_j we can write the Laplacian matrix as a sum of the effect of each individual edge.

$$\mathbf{L} = \sum_{j=1}^{m} \mathbf{E}_j \tag{5.6}$$

The matrix $\mathbf{E}_j$ is associated to the edge $e_j = \{v_i, v_k\}$. This matrix has the property that the terms of diagonal associated to the nodes $\left(\mathbf{E}_j\right)_{i,i} = \left(\mathbf{E}_j\right)_{k,k} = 1$, the off-diagonal terms $\left(\mathbf{E}_j\right)_{k,i} = \left(\mathbf{E}_j\right)_{i,k} = -1$ and 0 otherwise.

$$\mathbf{E}_j = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \mathbf{1} & \cdot & \mathbf{-1} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \mathbf{-1} & \cdot & \mathbf{1} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \tag{5.7}$$

Weighted Laplacian

One generalization of the Laplacian matrix consist in considering a weighted graph. The case of the labeled graph when the edges are labeled by real numbers is known as weighted graph. It can be represented adding a new component to the elements of the edge, i.e. $e_j = \{v_i, v_k, \alpha_j\}$.

We can generalize the equation 5.6 for the Weighted Laplacian as

$$\mathbf{L_w} = \sum_{j=1}^{m} \alpha_j \mathbf{E}_j \tag{5.8}$$

where the term α_j is a real positive number. The case with no weight can be considered using $\alpha_j = 1$.

5.2.2 Relation between Laplacian Matrix and Information Matrix

We will demonstrate the deep connection between the Laplacian matrix and the Information matrix, and discuss the case that all measurements have the same noise covariance matrix $\mathbf{\Sigma}$. We need to use the Kronecker product ($\otimes$) for matrices **A** and **B**.

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} [\mathbf{A}]_{1,1}\mathbf{B} & \cdots & [\mathbf{A}]_{1,n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ [\mathbf{A}]_{n,1}\mathbf{B} & \cdots & [\mathbf{A}]_{n,n}\mathbf{B} \end{pmatrix} \tag{5.9}$$

Using the Kronecker product we can write the following

Theorem 5.2.1. *The information matrix in Graph SLAM can be approximated for small error in terms of the edges as*

$$\mathbf{Y} = \sum_{j=1}^{m} \mathbf{Y}_j \simeq \sum_{j=1}^{m} \mathbf{E}_j \otimes \mathbf{\Sigma}_j^{-1} \tag{5.10}$$

In this equation $\mathbf{E}_j$ is an $n \times n$ matrix where “n" is the number of nodes and $\mathbf{\Sigma}_j$ is the $b \times b$ matrix where “b" is the size of the configuration vector to be estimated, then the Kronecker product $\mathbf{E}_j \otimes \mathbf{\Sigma}_j^{-1}$ is a $nb \times nb$ block matrix.

Proof. In the general problem of SLAM we define the configuration vector $\mathbf{x}_i$ that contains the position $\mathbf{t}_i$ and the orientation $\boldsymbol{\theta}_i$ of the vertex v_i. We will use the operators $\oplus$ for composition of transformation and $\ominus$ for the inverse transformation as defined in [Smith et al., 1990].

$$\mathbf{x}_{ik} = \ominus \mathbf{x}_i \oplus \mathbf{x}_k \tag{5.11}$$

In the SLAM context the information is received in terms of relative transformation, similar to the previous cases this transformation is considered a random variable

$$\mathbf{x}_{ik} = \bar{\mathbf{x}}_{ik} \oplus \mathbf{d}_{ik} \tag{5.12}$$

with d_{ik} a random variable $\mathcal{N}(0, \mathbf{\Sigma}_j)$. Setting the transformations with respect to an arbitrary world position "w" we can write

$$\mathbf{x}_{wi} = \hat{\mathbf{x}}_{wi} \oplus \mathbf{d}_i \tag{5.13}$$

$$\mathbf{x}_{wk} = \hat{\mathbf{x}}_{wk} \oplus \mathbf{d}_k \tag{5.14}$$

using the composition properties we write

$$\begin{aligned} \mathbf{x}_{ik} &= \ominus \mathbf{x}_{wi} \oplus \mathbf{x}_{wk} && (5.15)\\ &= \ominus \mathbf{d}_i \ominus \hat{\mathbf{x}}_{wi} \oplus \hat{\mathbf{x}}_{wk} \oplus \mathbf{d}_k && (5.16)\\ &= \ominus \mathbf{d}_i \oplus \hat{\mathbf{x}}_{ik} \oplus \mathbf{d}_k && (5.17)\\ &\simeq \hat{\mathbf{x}}_{ik} \ominus \mathbf{J}_{ki}\mathbf{d}_i \oplus \mathbf{d}_k && (5.18)\\ &\simeq \hat{\mathbf{x}}_{ik} \oplus \mathbf{d}_{ik} && (5.19) \end{aligned}$$

Assuming small errors

$$\mathbf{d}_{ik} \simeq -\mathbf{J}_{ki}\mathbf{d}_i + \mathbf{d}_k \tag{5.20}$$

Setting this equation in the common reference frame in w

$$\begin{aligned} \mathbf{d}_{ik|w} &\simeq \mathbf{J}_{wk}\mathbf{d}_{ik} && (5.21)\\ &\simeq -\mathbf{J}_{wk}\mathbf{J}_{ki}\mathbf{d}_i + \mathbf{J}_{wk}\mathbf{d}_k && (5.22)\\ &\simeq -\mathbf{J}_{wi}\mathbf{d}_i + \mathbf{J}_{wk}\mathbf{d}_k && (5.23)\\ &\simeq -\mathbf{d}_{i|w} + \mathbf{d}_{k|w} && (5.24)\\ &\simeq \begin{pmatrix} \mathbf{I} & -\mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{d}_{i|w} \\ \mathbf{d}_{k|w} \end{pmatrix} && (5.25) \end{aligned}$$

Considering the full vector $\mathbf{d} = (\cdot\ \mathbf{d}_{i|w}\ \cdot\ \mathbf{d}_{k|w}\ \cdot)$ the equation can be written

$$\mathbf{d}_{ik|w} = \begin{pmatrix} \cdot & -\mathbf{I} & \cdot & \mathbf{I} & \cdot \end{pmatrix} \mathbf{d} \tag{5.26}$$

When solving the graph SLAM problem based on the maximum likelihood approach we find the set of robot poses that minimizes the constraints associated to the observations (equation 2.3). The function to be minimized is

$$F = \sum_{j=1}^{m} F_j = \sum_{j=1}^{m} \mathbf{d}_j^T \boldsymbol{\Sigma}_j^{-1} \mathbf{d}_j \tag{5.27}$$

where $\mathbf{d}_j$ is the random variable associated to the transformation between the end nodes of the edge e_j, represented in the world reference w. Considering the edge e_j connects the nodes n_i and n_k we can write $\mathbf{d}_j = \mathbf{d}_{ik|w}$. Consequently we can express the function F_j as

$$F_j = \mathbf{d}_j^T \boldsymbol{\Sigma}_j^{-1} \mathbf{d}_j \tag{5.28}$$

This can now be written as

$$F_j = \mathbf{d}^T \begin{pmatrix} \cdot \\ -\mathbf{I} \\ \cdot \\ \mathbf{I} \\ \cdot \end{pmatrix} \boldsymbol{\Sigma}_j^{-1} \begin{pmatrix} \cdot & -\mathbf{I} & \cdot & \mathbf{I} & \cdot \end{pmatrix} \mathbf{d} \tag{5.29}$$

$$= \mathbf{d}^T \underbrace{\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \boldsymbol{\Sigma}_j^{-1} & \cdot & -\boldsymbol{\Sigma}_j^{-1} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & -\boldsymbol{\Sigma}_j^{-1} & \cdot & \boldsymbol{\Sigma}_j^{-1} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}}_{\mathbf{Y}_j} \mathbf{d} \tag{5.30}$$

Using Kronecker product we can write $\mathbf{Y}_j$ as

$$\mathbf{Y}_j \simeq \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \mathbf{\Sigma}_j^{-1} & \cdot & -\mathbf{\Sigma}_j^{-1} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & -\mathbf{\Sigma}_j^{-1} & \cdot & \mathbf{\Sigma}_j^{-1} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \tag{5.31}$$

$$\simeq \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & -1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & -1 & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \otimes \mathbf{\Sigma}_j^{-1} \tag{5.32}$$

$$\simeq \mathbf{E}_j \otimes \mathbf{\Sigma}_j^{-1} \tag{5.33}$$

where $\otimes$ is the Kronecker product and $\mathbf{E}_j$ is the matrix associated to the edge e_j (equation 5.6). Finally we can write the complete information matrix as:

$$\mathbf{Y} = \sum_{j=1}^{m} \mathbf{Y}_j \tag{5.34}$$

$$\mathbf{Y} \simeq \sum_{j=1}^{m} \mathbf{E}_j \otimes \mathbf{\Sigma}_j^{-1} \tag{5.35}$$

■

5.2.3 Every Measurement with same error distribution

In [Khosoussi et al., 2014] they claim that many SLAM front-ends and many popular datasets are consistent with the assumption that every measurement can be modelled as a Gaussian distribution with the same covariance matrix $\mathbf{\Sigma}$ in that case we can use the corollary of the theorem 5.2.1

Corollary 5.2.2. *In the case that that every measurement have the same noise covariance matrix $\mathbf{\Sigma}$ the equation 5.35 becomes*

$$\mathbf{Y} = \mathbf{L} \otimes \mathbf{\Sigma}^{-1} \tag{5.36}$$

Proof. This can be easily demonstrated using Kronecker product properties and setting $\mathbf{\Sigma}_j = \mathbf{\Sigma}$

$$\mathbf{Y} = \sum_{j=1}^{m} \mathbf{E}_j \otimes \mathbf{\Sigma}^{-1} \tag{5.37}$$

$$= \left(\sum_{j=1}^{m} \mathbf{E}_j \right) \otimes \mathbf{\Sigma}^{-1} \tag{5.38}$$

$$= \mathbf{L} \otimes \mathbf{\Sigma}^{-1} \tag{5.39}$$

■

One direct consequence of this corollary is that the unordered set of λ_t eigenvalues of the information matrix can be calculated directly.

Let $\mu_1, \cdots \mu_n$ be the ordered set of eigenvalues of the laplacian matrix $\mathbf{L}$ with size $n \times n$ and $\nu_1, \cdots, \nu_b$ be those of the noise covariance matrix $\mathbf{\Sigma}^{-1}$ with size $b \times b$ (listed according to multiplicity). Using the properties of the spectrum of the Kronecker product we know that the eigenvalues of the information matrix $\mathbf{Y} = \mathbf{L} \otimes \mathbf{\Sigma}^{-1}$ are:

$$\lambda_{i+(j-1)n} = \mu_i \cdot \nu_j \tag{5.40}$$

The connectivity indices of the graph are calculated using the eigenvalues of the Laplacian matrix removing the zero eigenvalue, namely

$$\begin{array}{lrl} \text{Number of Spanning trees} & \mathtt{t}(G) = & \prod_{i=2}^{N} \mu_i \\ \text{Kirchoff Index} & Kf(G) = & \sum_{i=2}^{N} \mu_i^{-1} \\ \text{Algebraic Connectivity} & \alpha(G) = & \mu_2 \end{array} \tag{5.41}$$

We can express the connectivity indices as instances of the information functions in terms of the eigenvalues after removing the zero eigenvalue. Using the same criteria for the information function we remove the b number of zero eigenvalues. For $p \neq 0$ and $p \leq 1$ we have

$$\|\mathbf{Y}\|_p = \left(\frac{1}{(n-1)b} \sum_{t=b}^{bn} \lambda_t^p \right)^{\frac{1}{p}} \tag{5.42}$$

$$= \left(\frac{1}{(n-1)b} \sum_{i=2}^{n} \sum_{j=1}^{b} (\mu_i \cdot \nu_j)^p \right)^{\frac{1}{p}} \tag{5.43}$$

$$= \left(\frac{1}{(n-1)b} \left(\sum_{i=2}^{n} \mu_i^p \right) \left(\sum_{j=1}^{b} \nu_j^p \right) \right)^{\frac{1}{p}} \tag{5.44}$$

$$= \left(\frac{\sum_{i=2}^{n} \mu_i^p}{(n-1)} \right)^{1/p} \left(\frac{\sum_{j=1}^{b} \nu_j^p}{b} \right)^{1/p} \tag{5.45}$$

$$= \|\mathbf{L}\|_p \|\mathbf{\Sigma}^{-1}\|_p \tag{5.46}$$

In the case that $p = 0$

$$\|\mathbf{Y}\|_p = \prod_{i=b}^{b(n-1)} \lambda_i^{\frac{1}{b(n-1)}} \tag{5.47}$$

$$= \prod_{i=2}^{n} \prod_{j=1}^{b} (\mu_i \cdot \nu_j)^{\frac{1}{b(n-1)}} \tag{5.48}$$

$$= \left(\prod_{i=2}^{n} \mu_i^{\frac{1}{n-1}} \right) \left(\prod_{j=1}^{b} \nu_j^{\frac{1}{b}} \right) \tag{5.49}$$

$$= \|\mathbf{L}\|_p \|\mathbf{\Sigma}^{-1}\|_p \tag{5.50}$$

As a consequence any information function applied to the un-anchored information function can be found applying it to the Laplacian Matrix **L** and the information matrix **Σ** separately.

$$\|\mathbf{Y}\|_p = \|\mathbf{L}\|_p \|\mathbf{\Sigma}^{-1}\|_p \tag{5.51}$$

Furthermore, under these conditions we demonstrate the exact connection between the optimality criteria and the connectivity index of the underlying graph.

5.2.4 Every Measurement with Bounded Measurement Noise

In the case the uncertainty in the edge e_j is bounded by

$$\mathbf{\Sigma}_j^{-1} \preceq \alpha_j \mathbf{\Sigma}^{-1} \tag{5.52}$$

where $\alpha_j \geq 1$ is an scalar, $\mathbf{\Sigma}^{-1}$ is a constant information matrix and the symbol "$\preceq$" implies the difference is $\alpha_j \mathbf{\Sigma}^{-1} - \mathbf{\Sigma}_j^{-1}$ is positive definite. The equation 5.35 now becomes

$$\begin{aligned}\mathbf{Y} \simeq \sum_{j=1}^{m} \mathbf{E}_j \otimes \mathbf{\Sigma}_j^{-1} &\preceq \sum_{j=1}^{m} \mathbf{E}_j \otimes \alpha_j \mathbf{\Sigma}^{-1} \\ &= \sum_{j=1}^{m} \alpha_j \mathbf{E}_j \otimes \mathbf{\Sigma}^{-1} = \mathbf{L_w} \otimes \mathbf{\Sigma}^{-1}\end{aligned} \tag{5.53}$$

with $\mathbf{L_w}$ the Laplacian defined for the weighted graphs. Accordingly we obtain a similar relation between the optimality criteria applied to the information matrix of the system $\mathbf{Y}_j$ and the one applied to the Laplacian matrix $\mathbf{L_w}$ of the weighted graph

$$||\mathbf{Y}||_p \leq ||\mathbf{L_w}||_p ||\mathbf{\Sigma}^{-1}||_p \tag{5.54}$$

In general every positive definite matrix, and consequently every uncertainty matrix, is trivially bounded by

$$\mathbf{\Sigma}_j^{-1} \preceq \nu_b \mathbf{I} \tag{5.55}$$

where ν_b is the largest eigenvalue of $\mathbf{\Sigma}_j^{-1}$, therefore the equation 5.54 is always valid. The equality is produced when the bound in equation 5.52 is sharp, i.e. $\mathbf{\Sigma}_j^{-1} = \alpha_j \mathbf{\Sigma}^{-1}$.

5.3 Optimal Path

The quintessential task in autonomous exploration consist in finding the shortest path that produces the best map, because the map is not known beforehand this path has to be found incrementally. In the scenario of Graph SLAM the length of the path is related to the number of traversed edges and the quality of the map is related to the optimality criteria applied to the information matrix.

In order to evaluate an exploration policy we compare its performance with the optimal. This optimal path depends on the current map to be explored and needs to be calculated off-line. The rest of this section deals with the problem of the calculus of this optimal path for the case of different optimality criteria.

5.3.1 Off-line Optimal Path

Considering the case that the covariance associated to every measurement is the same (or bounded by a real positive number) we observe that the result of applying the information function to the information matrix is proportional to the one applied to the Laplacian matrix (equation 5.51)

$$||\mathbf{Y}||_P \propto ||\mathbf{L}||_p \tag{5.56}$$

Instead of maximizing optimality criteria on the information matrix, we maximize the connectivity index of the underlying graph. The problem of the optimal path can be stated as:

Problem 5.3.1. *Given the graph $G(V,E)$ and a maximum number of traversals "d", find the set of traversals that maximizes a connectivity index of the graph.*

Different connectivity indices of the graph can be represented using a variation of the information function. Lets define the eigenvalues of **L** in the ordered vector $\mu = [\mu_1 \mu_2 \cdots \mu_n]$ with $0 = \mu_1 \leq \mu_2 \cdots \leq \mu_n$.

Instead of removing the row and column to obtain the reduced Laplacian $\mathbf{L_r}$ like [Khosoussi et al., 2014], we simply remove the zero eigenvalue and calculate the information functions on the remaining. The logic behind removing one row and column comes from setting one node of the graph as a reference for every other node, i.e. the measurement covariance respect to the world reference is the zero [Thrun and Montemerlo, 2006] [Grisetti et al., 2010]. Removing the zero eigenvalue without referencing any other node permits to extract the generic properties of the quality of the map and thus compare one-to-one with the connectivity indices of the underlying graph.

$$\|\mathbf{L}\|_p = \begin{cases} \left(\frac{1}{n-1} \sum_{i=2}^{n} \mu_i^p \right)^{\frac{1}{p}} & if \quad p \neq 0 \\ \prod_{i=2}^{n-1} \mu_i^{\frac{1}{n-1}} & if \quad p = 0 \end{cases} \tag{5.57}$$

This way we can express several connectivity criteria used in the spectral graph theory with different values of $p \leq 1$

1. Average Degree of Nodes ($p = 1$)
2. Number of Spanning Trees ($p = 0$)
3. Kirchoff Index ($p = -1$)
4. Algebraic Connectivity ($\lim_{p \to -\infty} \|L\|_p = \mu_2$)

Now we can define the characteristics of the solution. Lets define the vector of traversals $\mathbf{x} \in \mathbb{Z}^{+^m}$, the term $\mathbf{x}_j$ represents the number of times the edge e_j has been traversed. Using equation 5.6 we can represent the Laplacian $\mathbf{L}$ as a sum of the effect of every edge $\mathbf{E}_j$, fusing these definitions we can calculate the Laplacian matrix of every set of edges and the number of traversals simply using the following equations

$$\mathbf{L}(\mathbf{x}) = \sum_{j=1}^{m} \mathbf{E}_j \mathbf{x}_j \tag{5.58}$$

where $\mathbf{L}(\mathbf{x})$ represents the Laplacian as a function of the vector $\mathbf{x}$. Finally we can pose the problem of the Optimal exploration as an optimization problem to find the optimal vector of traversals $\mathbf{x}^*{}_{d,p}$ such as

$$\begin{aligned} \text{maximize} \qquad & \|\mathbf{L}(\mathbf{x})\|_p = \left\| \sum_{j=1}^{m} \mathbf{E}_j \mathbf{x}_j \right\|_p \\ \text{subject to} \qquad & \sum_{j=1}^{m} \mathbf{x}_j \leq d \end{aligned} \tag{5.59}$$

The problem posed in this terms can be seen as 0-1 integer programming problem, it has been solved exactly in [Vandenberghe et al., 1998] in the case where the function to optimize is the determinant (related to the D-Optimal) and propose that other functions can be solved, in a similar approach [Sagnol et al., 2015] second-order cone programming to optimize the solution. In [Wan et al., 2008] they optimize the algebraic connectivity. Although some solution can be found in specific domains it is known that the general case is NP-complete.

Some recent work have tackle the problem of adding an edge incrementally [Hassan-Moghaddam et al., 2017] using some heuristics, however not optimal solution has been found incrementally. Additionally these heuristics do not include the restriction that the set of edges should represent a feasible path (in consecutive edges the end node of first edge should be the start in the following).

Lets define the vector which produces the optimal index in equation 5.59 as $\mathbf{x}^*_{d,p}$, consequently we define the optimal value as the function evaluated in that vector $\mathscr{L}_{d,p} = \|\mathbf{L}(\mathbf{x}^*_{d,p})\|_p$.

The norm of the Laplacian have some useful properties, we name them here and the their proofs are in the appendix A.

Property 5.3.1. *$\mathscr{L}_{d,p}$ have an upper bound for $p \leq 1$.*

$$\mathscr{L}_{d,p} \leq \frac{2}{n-1} d \tag{5.60}$$

Property 5.3.2. *$\mathscr{L}_{d,p}$ is monotonically increasing with respect to the maximum traversals allowed d and the parameter of the information function p.*

In practical terms it means that monotony property is guarantee for every information function applied to the Laplacian Matrix. Additionally it implies that given a restriction of maximal traversals the maximum value is achieved traversing the maximum number of traversals

5.3.2 Information Rate

State of the art criteria to evaluate SLAM algorithms presented in the KITTY data set [Geiger et al., 2013] consist in dividing the accumulated error by the distance traversed. Considering the information functions are related to the error we similarly define the *Information Rate* as the quotient between the information function and the number of traversals.

$$E(\mathbf{x}, p) = \frac{\|\mathbf{L}(\mathbf{x})\|_p}{\sum_{j=1}^{m} \mathbf{x}_j} = \frac{\left\| \sum_{j=1}^{m} \mathbf{E}_j \mathbf{x}_j \right\|_p}{\sum_{j=1}^{m} \mathbf{x}_j} \tag{5.61}$$

This quantity relate two of the most important concepts in exploration: the quality of the map in terms of the accumulated information, and the length of the path needed to achieve this information. The task of exploration implies achieving the highest amount of information with the shortest path, it implies maximizing the information rate.

This function have several important properties, one of the most important is that it has a global maximum independent of the number of traversals, this maximum value constitute the *Optimal Graph Exploration*. In order to demonstrate that we need a few properties of $E(\mathbf{x}, p)$, these properties will be proven in appendix A.

Property 5.3.3. *The Information Rate is* Scale Invariant*:*

$$E(\alpha \mathbf{x}, p) = E(\mathbf{x}, p) \tag{5.62}$$

Property 5.3.4. *The Information Rate is* Bounded*:*

$$E(\mathbf{x}, p) \leq \frac{2}{N-1} \tag{5.63}$$

In order to find the optimal we can set the following integer maximization.

$$\begin{aligned}
&\text{maximize} && E(\mathbf{x},p) = \frac{\|\mathbf{L}(\mathbf{x})\|_p}{\sum_{j=1}^{m} \mathbf{x}_j} \\
&\text{subject to} && \sum_{j=1}^{m} \mathbf{x}_j \le d \\
& && \mathbf{x}_j \in \mathbb{Z}^{+^m}
\end{aligned} \tag{5.64}$$

The integer programming is known to be NP-hard, consequently no practical solution can be found. One approach to integer programming is called *relaxation*, we remove the constraints of **x** of being a vector of non negative integers and replace it with the relaxed version of being non negative real numbers in order to approximate the solution. In our case we can use the property of the scale invariant to calculate the solution when $\sum_{j=1}^{m} \mathbf{x}_j = 1$, because for every other value we only need to scale the solution, this way we can find the optimal value and its associated vector.

$$\begin{aligned}
&\text{maximize} && \|E(\mathbf{x},p)\| = \|\mathbf{L}(\mathbf{x})\|_p \\
&\text{subject to} && \sum_{j=1}^{m} \mathbf{x}_j = 1 \\
& && \mathbf{x}_j \in \mathbb{R}^{+^m}
\end{aligned} \tag{5.65}$$

Because the set of information functions is concave it is guaranteed that the function has a maximum value $\mathscr{E}_p$, achievable for $\mathbf{x}_p^* \in \mathbb{R}^m$ and it is unique.

$$\mathscr{E}_p \ge E(\mathbf{y},p) \tag{5.66}$$

The optimal value $\mathscr{E}_p$ represents the theoretical maximum, the performance of every exploration can be compared with this value to define the efficiency of the estimator.

Considering the fact that this solution comes from the *relaxation* in the original optimization problem the value of $\mathscr{E}_p$ represents an upper bound for the constrained version.

5.3.3 Optimal Path as Maximization with Linear Matrix Inequalities

The Laplacian matrix can be expressed as a weighted sum of matrices

$$\sum_{j=1}^{m} \mathbf{x}_j \mathbf{E}_j = \mathbf{L} \tag{5.67}$$

Finding the set of **x** that maximizes the criteria is efficiently posed as a convex optimization problem, although there is not a close form solution a convex problem is guarantee to converge to an optimal value. The convex optimization consist in minimizing a convex function where every constraint are defined between matrices using the positive semi definite comparison "$\succeq$" where $\mathbf{A} \succeq \mathbf{B}$ if and only if $\mathbf{A} - \mathbf{B} \succeq \mathbf{0}$.

$$\begin{array}{ll} \text{maximize} & F(\mathbf{x}) \\ \text{subject to} & G(\mathbf{x}) \succeq 0 \\ & \mathbf{A}\mathbf{x} = \mathbf{b} \end{array} \tag{5.68}$$

A subset of convex optimization problem with tractable solutions is the Semi Definite Programming (SDP)

$$\begin{array}{ll} \text{maximize} & \mathbf{c}^T \mathbf{x}) \\ \text{subject to} & \sum \mathbf{x}_j \mathbf{F}_j \preceq 0 \\ & \mathbf{A}\mathbf{x} = \mathbf{b} \end{array} \tag{5.69}$$

The A-Optimality criterion deals with the problem of determinant maximization which have been studied in [Vandenberghe et al., 1998]. The A-Optimality and E-Optimality criteria have been addressed in terms of semi definite programming [Boyd, 2006].

The relaxed version of the optimal Information Rate in equation 5.65 is the same problem of having a weighted graph and assigning the optimal set of weights that maximizes the criteria in the weighted Laplacian.

The specific shape for the optimization problems are

D-Optimal

The D-Optimal design has been discussed in [Vandenberghe et al., 1998] where the optimization is defined for real numbers like the relaxed version of the optimal problem

$$\begin{aligned} &\text{minimize} && -\log\det(\mathbf{L} + \left(\frac{1}{n}\right)\mathbf{1}\mathbf{1}^T) \\ &\text{subject to} && \sum \mathbf{x}_j \mathbf{E}_j = \mathbf{L} \succeq 0 \\ &&& \mathbf{1}^T\mathbf{x} = 1 \end{aligned} \tag{5.70}$$

where **1** is the vector formed by only 1’s.

A-Optimal

The problem of the A-Optimal design is linked to maximizing Kirchoff index of the graph. The Kirchoff index is also known as the total effective resistance assuming every edge is one resistor. The specific problem of finding the minimum total effective resistance is calculated in [Boyd, 2006] using Semi Definite Programming (SDP).

$$\begin{aligned} &\text{minimize} && \text{Trace}(\mathbf{Y}) \\ &\text{subject to} && \begin{pmatrix} \mathbf{L} + \left(\frac{1}{n}\right)\mathbf{1}\mathbf{1}^T & \mathbf{I} \\ \mathbf{I} & \mathbf{Y} \end{pmatrix} \succeq 0 \qquad (5.71) \\ &&& \sum \mathbf{x}_j \mathbf{E_j} = \mathbf{L} \succeq 0 \\ &&& \mathbf{Y} = \mathbf{Y}^T \succeq 0 \\ &&& \mathbf{1}^T\mathbf{x} = 1 \end{aligned} \tag{5.72}$$

The matrix **Y** is one slack matrix introduced to solve the problem.

E-Optimal

The E-Optimal criteria is related to the lowest non zero eigenvalue of the Laplacian Matrix. This eigenvalue is also related to the spectral partitioning of the graph, in the consensus multi-robot network this eigenvalue defines the velocity of convergence, in [Boyd, 2006] it is defined as the fastest mixing Markov process and define the problem as a SDP:

$$\begin{aligned} \text{minimize} \quad & \gamma \\ \text{subject to} \quad & \sum \mathbf{x}_j \mathbf{E_j} = \mathbf{L} \succeq 0 \\ & \gamma \mathbf{I} \preceq \mathbf{L} + \beta \mathbf{1}\mathbf{1}^T \qquad (5.73) \\ & \mathbf{1}^T \mathbf{x} = 1 \qquad (5.74) \end{aligned}$$

where γ, β and **x** are the variables to be optimized

5.4 Experimental Results

In this section we first verify the results of the properties of the information of the paths derived in previous section, secondly we analyze the pattern of the solution of the optimal path in typical indoors environments.

Finally based on the analysis we propose an exploration policy and compare its performance with respect to the optimal path in indoor environments.

5.4.1 Optimal Path Properties

The objective of this section consist in confirming some of the properties of the optimal path demonstrated in the previous section.

We perform an exhaustive search to find all possible graph H_t derived from the original graph $G = (V,E)$ such as they contain the same set of nodes V and the subset of (possible repeated) edges E. We can describe these graphs $H_t = (V,F_t)$ using the mathematical formalism

$$H_t = (V,F_t) \;\Big|\; F_t \subset E^d,\ |F_t| = d \qquad (5.75)$$

where "d" is the maximum number of traversals defined in the equation 5.59 and E^d is the notation for the multiset containing up to "d" repeated edges of the original graph $G = (V,E)$. $|F_t|$ is simply the number of elements in the set.

For a given maximum number of traversals d we evaluate the chosen criteria for every possible graph H_t and select the one that maximizes it. Considering that $\|\mathbf{Y}\|_p \propto \|\mathbf{L}\|_p$ (equation 5.56) we apply the information function to the Laplacian matrix of every graph H_t according to the values of p that represent the different optimality criteria, namely

- T-Optimal: $p = 1$
- D-Optimal: $p = 0$
- A-Optimal: $p = -1$
- E-Optimal: $p \to -\infty$

The first experiment consist in analyzing the complete graphs. The complete graphs are the graphs where every pair of distinct nodes is connected by an edge. The complete graph on n vertices is named K_n, in the figure 5.5 we observe K_4 composed of 4 nodes. This family of graphs is important for our analysis because all of the non zero eigenvalue of its laplacian matrix are equal, as a consequence every optimality criteria produces the same value. Finally, when the number of traversals d coincides with the total edges in K_n every information function reaches the same value, and consequently the optimal.

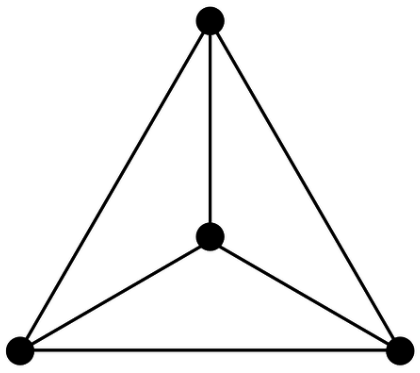


Figure 5.5: K_4 Graph, Every node is connected to the rest of the nodes. Analyzing this graph we can verify the properties of the information function

In the figure 5.6 coming from the analysis of the information of the K_4 graph we observe the properties enunciated earlier. The upper bound correspond to $p = 1$ associated to the T-Optimality criteria, every other function lies bellow it. The monotony with respect to the number of traversals d is evident no matter what optimality criteria is used. Finally the monotony with respect to the parameter p can be seen because we can write

$$\text{T-Optimal} \geq \text{D-Optimal} \geq \text{A-Optimal} \geq \text{E-Optimal} \tag{5.76}$$

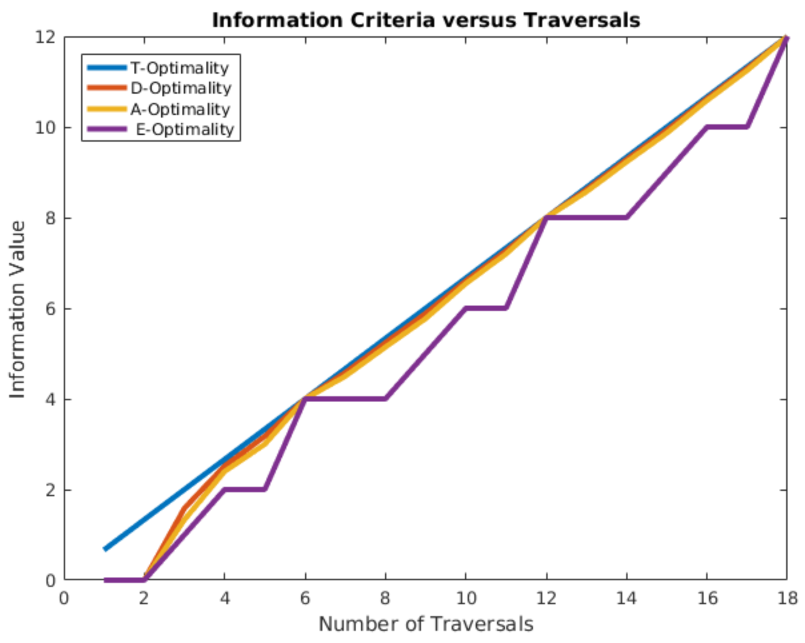


Figure 5.6: The upper bound on the information for every optimality criteria is the information of the T-Optimal. The functions are monotonously incresing with respect to both, the number of traversals and the parameter p, associated to the different criteria

One important feature deals with the fact that when the number of traversals is lower than the number of nodes the information according to every optimality criteria is zero, once there is a tree connecting every node the values grows.

In the figure 5.7 we observe the properties of the Information Rate of the graph. The upper bound property of the Information Rate is easily observed. The scale invariant property can be observed for the values of traversals of 6 and 12 and 18, in these cases the optimal information rate is reached when every edge is traversed the first and the second time.

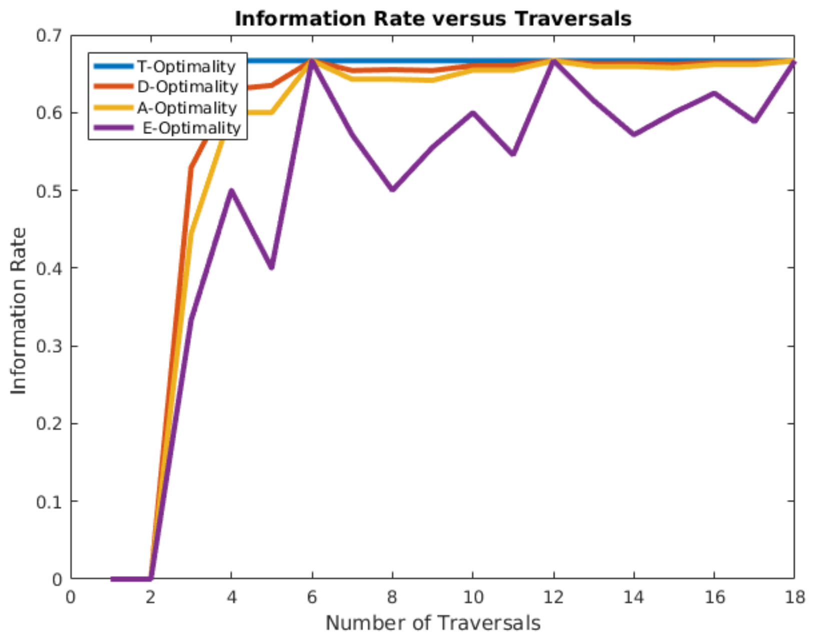


Figure 5.7: In the K_4 graph the optimal information rate is reached for every information function when the number of traversals is a multiple of the number of edges. The scale invariant is demonstrated for $d = 6, 12, 18$

In this first experiment we avoided the constraint that the set of edges must be traversable, i.e. the edges should be ordered in a sequence where the end node of every edge is the starting node of the following. Adding this restriction the graphs of the information and information rate are modified. In K_4 there is no plausible path that visits every edge only once, as a consequence in figure 5.8 the optimal is not reached when the value of traversals is $d = 6$, however in every graph there is always a path that visit every edge exactly twice producing the optimal information when $d = 12$. The information rate reaches exactly the same optimal that the unconstrained version confirming that the constraint of being a path do not modify the optimal value.

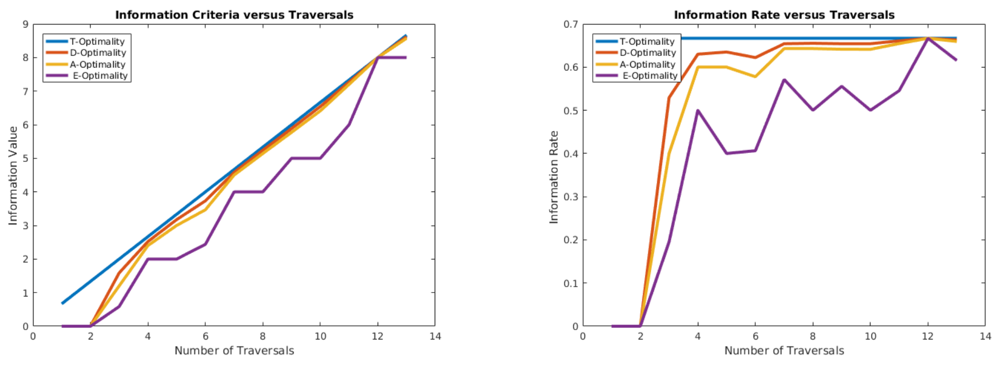


Figure 5.8: Restricting the solution to the set of traversable paths the evolution of the curve is modified, nevertheless the same optimal value is reached when $d = 12$, i.e. traversing every edge twice.

In the third experiment we show the results for the generic graph imposing the constraint of being a traversable path. In the figure 5.9 we observe on the left the topological graph and the evolution of its information when more edges are traversed. One important difference from this graph to the complete graph K_4 is that the optimal value is different according to optimality criteria used to calculate the information.

In figure 5.10 we can observe that the information according to every optimality criteria tends to one optimal value, this value is the one defined by equation 5.66 as $\mathscr{E}_p$ for the different values of p.

5.4.2 Optimal Path for Indoors Environment

In [Aydemir et al., 2012] they present a dataset where the maps of 130 indoors environments and its corresponding topological graphs are given. For every topological graph we solve the optimization problem presented in equation 5.65 to obtain the optimal set of weights $\mathbf{x}$ that maximizes the information according to the most common criteria, namely: E-Optimality, A-Optimality and D-Optimality.

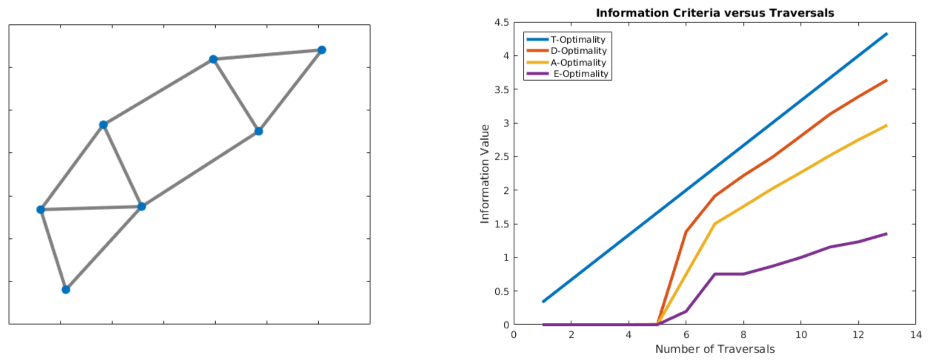


Figure 5.9: For a generic Graph the rate of the information growth is different according to the optimality criteria used, in general there are not intersection in these curves

In order to solve the optimization problem we use the function `fmincon` in MATLAB® using the default *interior-point* algorithm. This is not the most efficient algorithm to solve the optimization problem, however at present we are only analyzing the solution, we are not interested in efficiency.

According to [Aydemir et al., 2012] the structure of indoors environment is accurately represented using the *small world* networks model, one property of these networks is that small number of nodes concentrate many edges while many nodes have very low number of connections. In figure 5.11 one of the graphs from this dataset clearly represents this property. The thickness of the edges are associated to the optimal weight assigned by the optimization algorithm. We observe how the optimization according to the E-Optimality criteria assign more weight in the connection between the nodes with high degree, i.e. more connections. On the other hand, the optimization according to the D-Optimality criteria produces a balanced distributions on the weights.

Even when there is a big loop that can reduce the uncertainty in figure 5.12 the optimization according E-Optimal still produces bigger weights in the edges connecting the high degree node, in the A-Optimality these edges are still important but more balanced, and finally for the D-Optimality no big difference is made.

In figure 5.13 there are several cycles but according to the E-Optimal based optimization the most important is the tree connecting the nodes with higher degree,

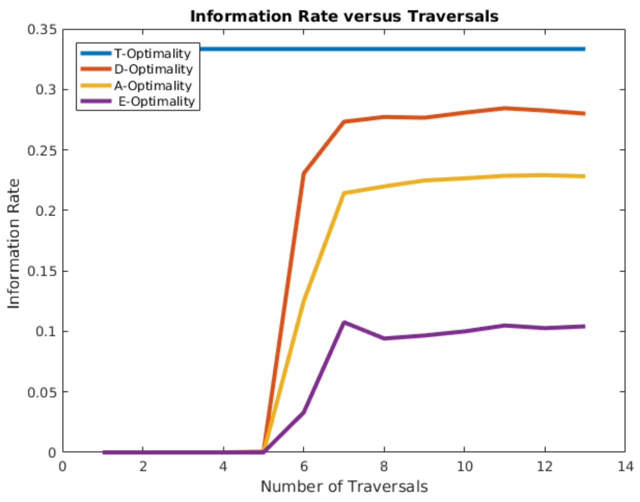


Figure 5.10: Analizing the information rate the trend to the optimal value is visible. This optimal value $\mathscr{E}_p$ is the result of the optimization in the equation 5.65

in the D-Optimal based optimization the bigger weights are in the nodes with low number of connections, the A-Optimal optimization emphasizes a different tree than the E-Optimal based and increases the weights associated to the cycle, but not as homogeneous as the D-Optimal.

5.4.3 Exploration Efficiency of Eulerian Path Policy

Many of the algorithms for active SLAM consist in choosing an interesting point and heading towards it [Cadena et al., 2016], most of them are based in the frontier exploration [Yamauchi, 1997], once every point have been explored the task is over. Considering the exploration task as visiting a set of points this common approach is similar the task of visiting every node in the graph-based exploration.

In SLAM once the data association is solved we are able to recognize a place we have been before. However even when we are able to recognize some exploration direction the destination is only known once we traverse onto it. According to the taxonomy on the graph based explorations presented in Chapter 2 the Active SLAM is in the scenario of nodes recognizable but edges not. The common task

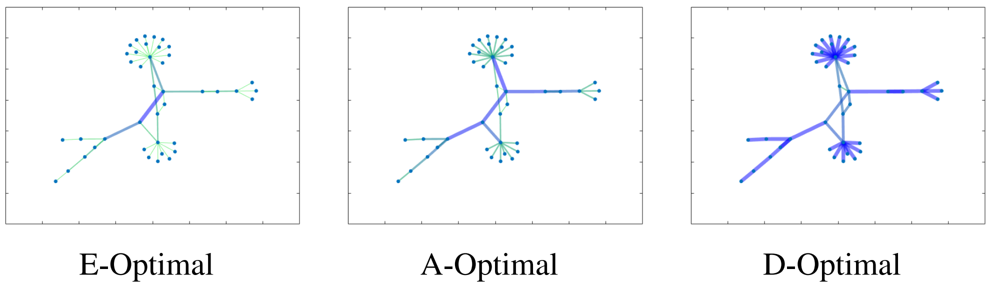


Figure 5.11: Indoors Graph with *small-world* topology, few nodes with many connections. The thickness of the edge is proportional its weight in the optimal graph. E-Optimal based optimization emphasizes the edges connecting the nodes with many connections while D-Optimal based optimization the weights are evenly distributed, the A-Optimal based can be seen as a transition between the others

in this scenario is the *full graph exploration* that consists in traversing every edge and consequently visiting every vertex.

The *Eulerian Path* is the path that visits every edge once, this is the shortest possible path that produces the full graph exploration. We have seen in the previous experiment that for the complete graphs it maximizes every information criteria. While solving the famous *Seven Bridges of Königsberg* problem Euler proved that a graph needs that every node have an even degree in order to have an Eulerian Path, although not every graph fulfill this condition when allowing every edge be traversed twice this path is always traversable, furthermore using the scale invariant property of information rate we know this double traversal path will produce the same information rate that the virtual eulerian path.

Unlike the common approaches in the Active SLAM we propose traversing every edge as the policy for exploration. This policy recognizes the nature of the graph based exploration scenario and there are algorithms that can find this path incrementally.

In order to evaluate the performance of any algorithm we can compare it with the theoretical optimal, in this case we will compare the quality of the path applying the information function to the discovered graph and compare it with the optimal information for the graph.

Using the optimal information calculated for the the dataset presented in [Aydemir et al., 2012] calculated in the previous experiment we calculate the explora-

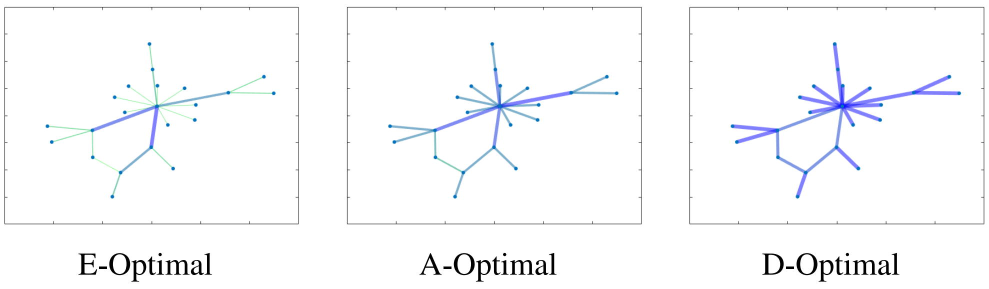


Figure 5.12: In the case of one loop the pattern of the optimal set of weights is repeated, in E-Optimal based the few edges connecting the high degree nodes are emphasized whereas in D-Optimal based the weighs are distributed

tion efficiency as the quotient of the information rate when every edge is traversed once(G_1) and the optimal information rate ($\mathscr{E}_p$).

$$Eff = \frac{\|\mathbf{E}(G_1)\|_p}{\mathscr{E}_p} \tag{5.77}$$

The average results in the table are presented in table 5.1, it can be seen that using the most common optimality criteria like D-Optimality and A-Optimality the efficiency of the proposed path is very high. In terms of the E-Optimality, which is related to minimize the maximum relative uncertainty, the variability in the efficiency is so high that prevent us for giving any guarantee.

Criteria	Average Efficiency
D - Optimality	$98.9 \pm 1.0\%$
A - Optimality	$89.3 \pm 8.6\%$
E - Optimality	$50 \pm 24\%$

Table 5.1: Exploration Efficiency

One final remark is that the A-Optimal is associated to the Kirchoff index of the graph, in the SLAM case it is related to the average of the uncertainty between every possible pair of nodes in the graph. Considering this we believe the A-Optimal criterion is the most appropriate for the active SLAM task.

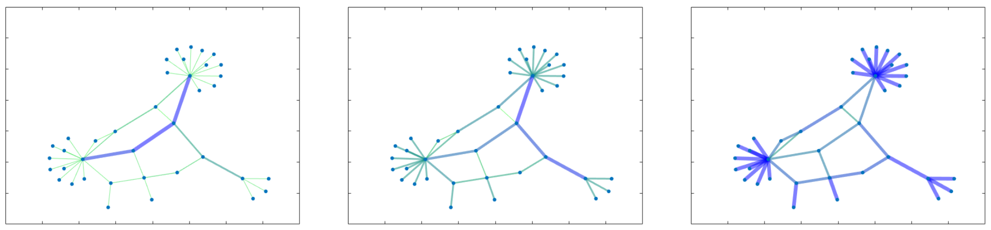


Figure 5.13: The third scenario with several cycles is very interesting because the weight pattern is very different according the optimality criteria to be optimized. In the E-Optimal based the edges inside the big loop are emphasized whereas in D-Optimal based the the edges inside the big loop are the less weighted. In the A-Optimal based the set of edges lightly emphasized is different from the edges of the E-Optimal based optimization

5.5 Conclusions

We demonstrated the relation between the optimality criteria of the Laplacian matrix and the un-anchored information matrix.

We demonstrate that optimal information rate has an optimal value which is independent of the size of the path.

We probe experimentally that the path that traverses every edge is very close to the optimal when evaluating E-Optimality or A-Optimality. The high variation in the case of E-Optimality prevents any efficiency claim. Considering the most commonly used criteria are the D-Optimal and the A-Optimal the proposed path is very efficient in common applications.

Chapter 6

Graph based Active SLAM

In the chapter 5 we proposed that traversing every edge once is the most appropriated exploration task in Active SLAM based on the characteristics of the problem, where the destination of the trajectory is only learned once the robot have traverse it. We also demonstrate that in topological graphs coming for real indoors maps the proposed path produces a very efficient algorithm when compared with the optimal exploration.

The next step consist in convert this optimal path into an exploration policy.

The first problem comes from building the graph from the current state of the map, this problem is solved in the incremental version of the topological segmentation described in chapter 4 where the graph is built incrementally emulating the on-line graph exploration.

The second problem comes from transforming this optimal path into a path achievable in on-line exploration. The path that traverse every edge once in a graph is known as *Eulerian Path*, however this path is not possible for all the graphs, the only mathematical guarantee is that a path traversing every edge twice is always achievable. In the case the graph can not have one eulerian path the *Chinese Postman Problem (CPP)* ask for the minimum length path that traverse every edge at least once but not more than twice.

The incremental algorithms to traverse unknown graphs comes from the maze-searching literature ([Lumelsky, 1991]). These classical algorithms offer mathematical guarantees of the length of the path. The Trèmaux algorithm, the basis of the *Depth First Search (DFS)*, produces a path that visits every edge exactly twice.

The Tarry's algorithm can be seen as a generalization of the Trèmaux and have the same path length, however its formulation permits introducing early termination criteria like the one proposed by Fraenkel's in [Fraenkel, 1970] which produces a path length between the minimum possible, defined by the solution of the *CPP*, and the original algorithm, that traverse every edge exactly twice. According to Fraenkel not further reduction of this interval of path lengths can be achieved by incremental algorithms and time has proven him right.

In this work we modify the maze-searching algorithm proposed by Tarry to include the Active SLAM constraints and insert it into an autonomous robotic exploration pipeline *TIGRE* that begins with a partially explored map, transform it into an partially explored graph and decides the next direction according to the Tarry's modified algorithm.

The resulting path is compared with the optimal path calculated off-line to visit every edge (solving *CPP*) or to visit every node (solving the traveling salesman problem *TSP*). The performance of the *TIGRE* is similar to the solution of the *CPP* which produces the lowest error in the reconstruction.

6.1 Tarry's Maze-Searching Algorithm

Early work on motion planning [Lumelsky, 1991] pointed out the relationships between maze-searching algorithms and robot motion planning with incomplete information driving the attention of the community towards classical algorithms (e.g. Trèmaux, Tarry and Fraenkel algorithms) that could be suitably adapted to motion planning problems as the exploration task considered in this work.

In particular, Tarry's algorithm [Gross and Yellen, 2006], an efficient algorithm to explore mazes, could be easily adopted within an exploration task because, from the knowledge of a partially known graph, it guarantees that every edge of the still unknown graph will be traversed exactly twice, once in every direction, with the exception of the edges incident to the start and target nodes, which will be traversed once each.

The assumptions of the Tarry's algorithm are:

1. *Node Recognition*, it can be recognized when a node has been previously visited.

2. *Edges Traversed Recognition*, upon arrival to a node, the edges that have been previously traversed outwards the node are known.

3. *Entrance Edge Recognition*, the entrance edge, i.e. the edge we traversed the first time we arrived to the node, is known for every visited node.

Under these assumptions, the algorithm can be easily described as: *Arriving at any node continue via any edge which has not yet been traversed outward, but choose the entrance edge only as a last resort*.

It is worth noting that, adding an extra condition as *Every time you arrive to a previously visited node by a new path, return by the path you came*, then the Trèmaux method is obtained, which is the basis of the Depth First Search (DFS) algorithm for graph traversal.

6.2 Topological Graph-based Robotic Exploration

In this section we describe the TIGRE algorithm, a topological graph-based robotic exploration algorithm rooted on the previously mentioned Tarry's maze-searching algorithm and that allows a robot to autonomously explore its navigation environment modeled as an undirected connected graph. Starting from an empty graph the algorithm evolves until a stopping criterium is satisfied.

At each time step an incremental contour-based topological segmentation algorithm [Fermin-Leon et al., 2017a] extracts the graph-based representation from the output of a graph-SLAM algorithm [Kummerle et al., 2011]. Every region in the topological segmented map is represented by one node in the graph and one additional node is used for the unexplored area. The graph's edges represent every pair of regions connected by a physically traversable path, including regions driving towards unexplored area. Physical traversability of all the edges of the topological graph increases the reliability of the obtained model for subsequent robotic tasks.

6.2.1 Tarry's Assumptions in Graph-SLAM

The graph SLAM algorithm for the map reconstruction satisfies, within performance bounds, the assumptions of the Tarry's algorithm described in the previous section:

1. *Node Recognition*, recognizing a previously visited node relates to the loop-closing capabilities of the algorithm. In our work we enforce loop-closing behaviour by navigating the vehicle in the vicinity of previously stored robot's poses within the navigation region.

2. *Edges Traversed Recognition*, the edges between nodes of the topological graph are defined by the traversed edges, and its direction vector, of the graph-SLAM algorithm

3. *Entrance Edge Recognition*, because the time history of the edge traversals is stored, the entrance edge is just the first, in chronological order, edge traversed into the region.

6.2.2 On-line Exploration Terminating Conditions

Fraenkel's constraints [Lumelsky, 1991] reduce the exploration path length introducing a counter associated with the number of nodes with unexplored edges, guarantying that every edge will be traversed at least once but never more than twice, once in every direction. In our case we keep track of the number of the remaining edges to be traversed, and once every edge is traversed the exploration is over. Consequently, our exploration path length lies within an analogous interval, i.e. between the minimum number traversals, achieved by the off-line CPP algorithm when the full graph is known, and twice the number of edges of the graph.

Additionally, we modify the Tarry's algorithm by a break-tie criteria when multiple edges could be chosen to be traversed at a given time step of the evolution of the algorithm. In the original Tarry's algorithm whenever a new node is reached and there are several edges that can be traversed, one edge is chosen randomly. We modify this criteria by choosing, in those situations, an edge leading to a previously visited node instead of an edge leading to the unexplored area, thus enforcing the loop-closing behaviour and increasing the precision of the underlying graph-SLAM solution.

6.2.3 The TIGRE Algorithm

Algorithm 4 describes the pseudo-code of the proposed topological graph-based exploration algorithm[1]. Initially, the algorithm builds on top of the ROS navigation stack for the functions `simulator` and `navigate`.

Then, the `Graph_SLAM` function is based on the implementation reported in [Lazaro et al., 2013] of g2o [Kummerle et al., 2011] as a back-end for the optimization, by using the odometry readings and the 2D laser scan to update the pose graph and the grid-map. The function `Topological_Segmentation` segments the input grid-map, associating a label to each segmented region by using the implementation reported in [Fermin-Leon et al., 2017a] of the contour-based segmentation algorithm [Guilin et al., 2014].

Next, the function `Build_Topological_Graph` builds an annotated topological graph. The information of the nodes includes their order in the sequence. Using this information the function `Extract_Incident_Edges` finds the valid edges connected to the current region (edges not traversed outwards) and classify them into either *Frontier_Edges* (leading to unexplored areas) or *Link_Edges* (leading to previously visited areas).

Finally, from those subsets of the incident edges the next goal location for the vehicle is selected. The algorithm favors the selection of goals leading to previously visited areas to enforce the loop-closing behaviour in the case of ambiguity.

6.3 Experimental Results

In this section we report simulation results, in the Player/Stage simulation environment [Gerkey et al., 2003], to illustrate the behaviour of the topological graph-based robotic exploration algorithm proposed in previous sections when an autonomous vehicle is navigating within the 20m $\times$ 20m *Cave* environment [Howard and Roy, 2003]. A C++ implementation of the TIGRE algorithm has been programmed on top of both simulation and navigation functions of the ROS package.

Figure 6.1 shows the online reconstruction of the navigation environment both from the geometrical perspective of the grid-map provided by the graph-SLAM

[1]The source-code of the algorithm is available at `https://github.com/lfermin77/TIGRE`

```
Algorithm 4: TIGRE Algorithm
  Input     : Map, Topological_Map, Pose_Graph
  Output    : Topo_Graph = (V, E)
  Variables: Link_Edges = {}
             Frontier_Edges = {}

  Exploration_Completed = FALSE
  Commands = 0
  while Exploration_Completed = FALSE do
      [Scan, Odometry] = simulator(Commands)
      [Pose_Graph, Map] = Graph_SLAM(Scan, Odometry)
      Regions_Set = Topological_Segmentation(Map)

      Topo_Graph = Build_Topological_Graph(Map, Regions_Set,
        Pose_Graph)
      [Frontier_Edges, Link_Edges] = Extract_Incident_Edges(Topo_Graph)
      if Link_Edges = {} & Frontier_Edges = {} then
          Exploration_Completed = TRUE
      else
          Remove_Entrance_Edge(Link_Edges)
          if Link_Edges = {} & Frontier_Edges = {} then
              Goal = Entrance_Edge(Topo_Graph)
          else
              if Link_Edges $\neq$ {} then
                  Goal = Extract_Goal(Link_Edges)
              else
                  Goal = Extract_Goal(Frontier_Edges)
      Commands = navigate(Goal)
```

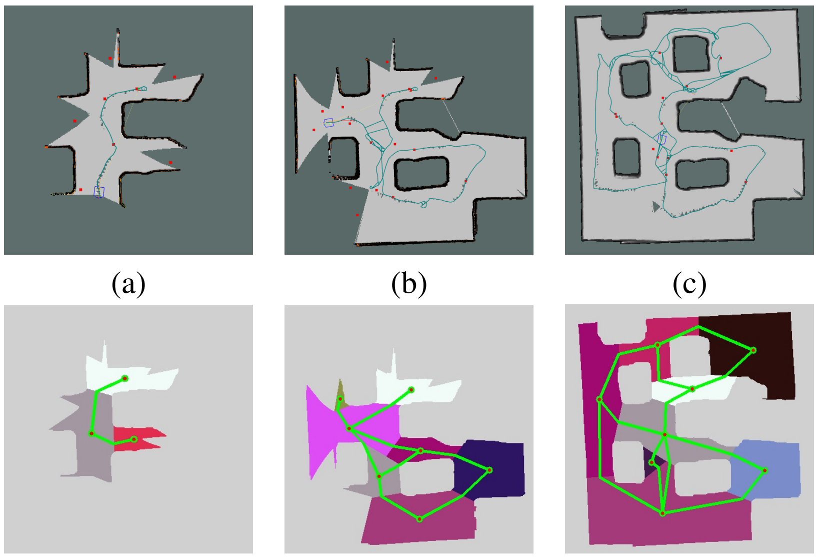
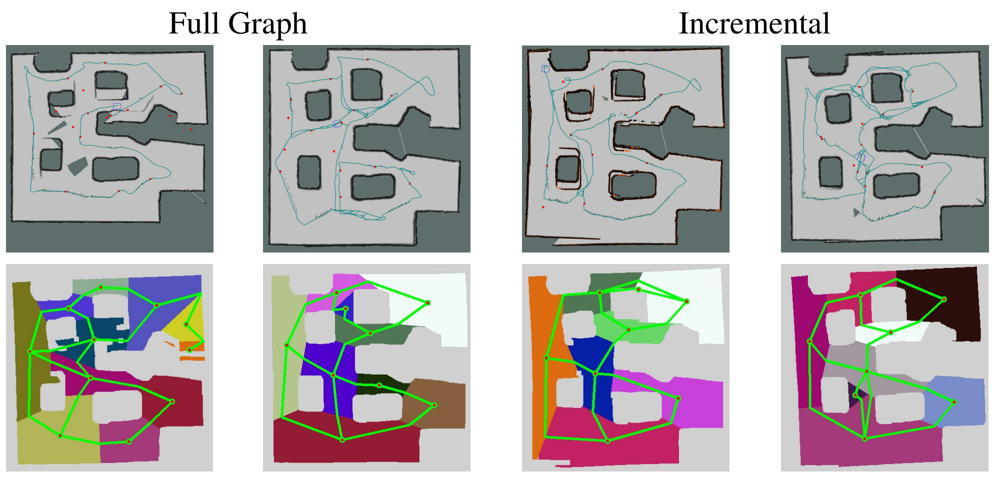

(a) (b) (c)
(d) (e) (f)

Figure 6.1: Snapshots of the exploration task: (a) and (d) represent early stages of exploration task and its decomposition with the current topological graph (green); in (b) the robot chooses to traverse the edge presented in (e) first, rather than exploring the nearby frontier. The exploration is considered completed in (c) and (f).

algorithm, and the online topological graph used for the robot exploration task derived for the incremental contour-based topological segmentation. The computation of segmented regions, and their representative nodes are sufficiently topologically stable for the execution of the TIGRE algorithm. In some cases, a region is over-segmented and 2-connected regions appear but the performance of the topological exploration algorithm is unaffected because when only one edge drives out of a node, no decision is required.

Different performance metrics, as the average of 10 replications of the experiment, are reported to compare the behaviour of the graph-SLAM algorithm when the topological graph-based model of the navigation is provided by: (i) an off-line full graph-based solution to the Travelling Salesman Problem (TSP); (ii) an off-line full graph-based solution to Chinese Postman Problem (CPP); (iii) an on-line greedy Frontier-based algorithm; and (iv) our on-line TIGRE algorithm.



(a) TSP $\bar{\varepsilon} \simeq 0.72 \pm 0.46m$ (b) CPP $\bar{\varepsilon} \simeq 0.21 \pm 0.07m$ (c) Frontier-based $\bar{\varepsilon} \simeq 0.61 \pm 0.37m$ (d) TIGRE $\bar{\varepsilon} \simeq 0.24 \pm 0.03m$

Figure 6.2: Comparison of the performance of different off-line, that use the full-graph, and on-line, that incrementally build the graph, algorithms. The average estimated error $\bar{\varepsilon}$ of the robot poses is shown together with its standard deviation computed from the ten replications of the experiment.

Figure 6.2 plots the final grid-maps obtained by each algorithm with the overlaid exploration trajectory. Also the final topological segmentations and the topological graphs are shown. Additionally, figure 6.3 plots the evolution of the rate robot pose error over distance for the complete exploration path length. From the computation of the mean error of the poses of the vehicle along its trajectory (recall that ground-truth is available in the simulation tool) we conclude that the best performance corresponds to the CPP-algorithm, the worst performance corresponds to the TSP-algorithm and that the TIGRE algorithm outperforms both the TSP and Frontier-based algorithms. The results agree with the intuition that both TSP and Frontier-based search for the shortest exploration path faster at the expense of reducing the number of loop-closing during the graph-SLAM execution. Both CPP and TIGRE, visiting all the edges of the graph, result in larger path lengths but in improved estimation errors.

Finally, figure 6.4 describes the evolution of the area coverage (percentage of

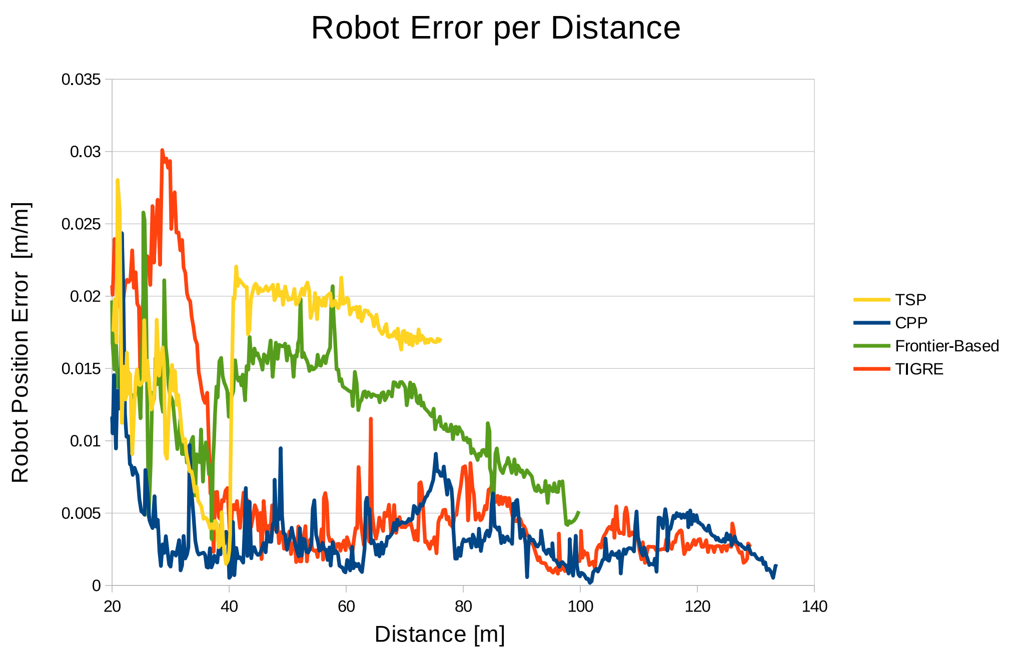


Figure 6.3: Evolution of the estimated error of the robot pose divided by the distance traveled versus the distance traveled. Relevant behaviour appears as the traveled distance increases.

the explored cells of the grid-map) versus the path length. Clearly the TSP algorithm results, by definition, in the shortest exploration path length due to its inherent feature of driving the vehicle always to unexplored terrain visiting all the nodes of the graph but only a subset of the edges. Similarly, the Frontier-based, with its greedy approach towards unexplored terrain reports short exploration path length in this case-study. In both cases, the number of edge re-traversals (and therefore loop closures) is very low, thus a similar performance, in terms of estimation error is obtained for both of them. On the contrary, the behavior of the TIGRE and the CPP bear similarities because they force the traversal of every edge of the graph, at least once, as mentioned in the previous sections, and therefore the distance travelled by the vehicle is larger, with frequent re-traversals of the edges of the graph (see the plateaus in the figure), and therefore achieving a better estimation error.

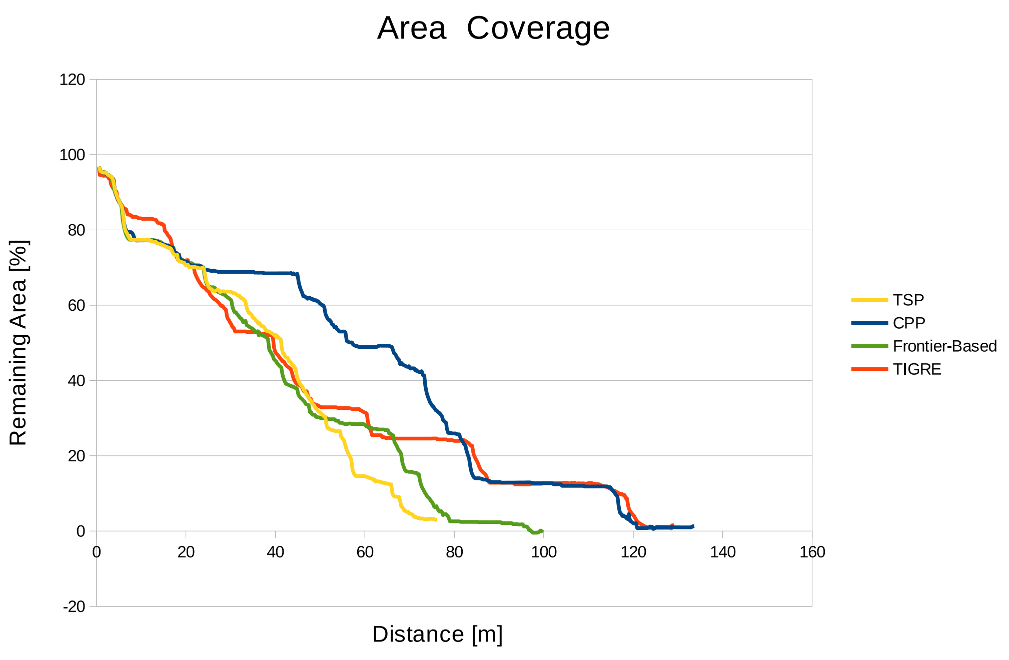


Figure 6.4: Map area coverage versus the distance traveled by the robot.

6.4 Conclusions

This paper focused on the problem of autonomous robotic exploration and map building supported by high level information provided by online topological segmentation that incrementally generates an undirected connected graph of the environment. The exploration problem is formulated as the traversal of all the edges of the online graph.

The online TIGRE algorithm is proposed that integrates Graph-SLAM features, contour-based topological segmentation, incremental graph construction and online decision-making by an adaptation of a constrained depth-first search based graph traversal algorithm. Simulation results suggested a close behaviour of the online TIGRE algorithm with off-line algorithms that require the knowledge of the full graph of the environment (CPP), and that it outperfoms, in terms of error estimation of the robot poses other online (Frontier-based) and off-line (TSP) algorithms.

Further work is aimed at thoroughly evaluating the TIGRE algorithm in more complex scenarios and its comparison against other exploration methods reported in the literature.

This work was presented in the European Conference on Mobile Robots 2017 [Fermin-Leon et al., 2017b].

Chapter 7

Conclusions and future work

7.1 Conclusions

The graph theory is a branch of mathematics with more than two centuries of work. Whenever a problem in any area can be stated as a graph problem every tool in the graph theory can potentially be used to solve it. In our case we use the tools from graph theory, spectral graph theory, on-line graph exploration,pr among others to design several stages of the Active SLAM problem.

Current algorithms in path planning under uncertainty are essentially a graph based planning. Unlike previous works in the area we include the probability of getting a good relocalization which resulted in recognizing the stochastic nature of the problem resulting and planning in terms of the expected uncertainty. One important finding is that the graph coming from the Graph-SLAM algorithm is *not* the best to pose the problem, in fact including edges of potential traversable paths coming from the Voronoi Diagram increases the number of possible routes where a shorter route can be traversed without risk of getting lost.

When the environment is segmented as a set of convex regions we can guarantee that every point in the region becomes observed once the robot enters it, with this formulation we can safely transform the robotic exploration problem to the on-line graph exploration problem. Most of the algorithms to segment the environment are based on grid based representation of the environment and commonly uses heuristics associated to indoors environment, these methods do not guarantee the convexity of the regions decomposed. Consequently we decided to choose a new method based on contours, not pixels, this way our algorithm produced approximated convex regions, additionally it was very robust to noise, very stable and obtained a great performance in public datasets. Changing the paradigm was superior than to fine tune one old method.

Efficiency is a measure of performance when optimal is known. In active SLAM there is not a predefined optimal, in that scenario the only way to evaluate an algorithm is with respect to other. We define the quintessential exploration task where the only variables are the traversed distance and the quality of the reconstruction, we also substitute the continuous domain of the environment to the topological graph that represents it, because every region is assumed to be convex visiting every edge implies full coverage. In this set we can finally analyze the problem of robotic exploration.

One important tool for the optimal path analysis consists in establishing the connection between the Information matrix of the graph SLAM and the Laplacian matrix of the underlying graph. Using this relation we can link the robotic exploration with the spectral graph theory being able to demonstrate the relation between the connectivity indices of the graph and the different optimality criteria applied to the Information matrix.

Finally we were able to calculate the optimal exploration path for any map decomposed into a topological graph.

Once the optimal exploration is defined we can evaluate our proposal in different scenarios. Using the indoors maps from dataset we were able to compare the performance of our proposal of traversing every edge with the optimal and it resulted in a great performance, achieving in average 98% of the optimal when measuring the D-Optimal and 90% when measuring the A-Optimal.

The last chapter consist in integrate the modules into the exploration pipeline. In the experiments we observed how frontier based algorithms behave similar to the one associated to the solution of the *TSP*, whereas our algorithm behaves similar to the off-line solution of the *CPP* which achieves the best performance in terms of the error in the estimation.

7.2 Future Work

The optimal on-line graph exploration is agnostic to the type of environment represented as a graph. The main limit in order to be used in a 3D exploration scenario consist in being able to segment the 3D environment into a set of convex regions. One efficient incremental segmentation of 3D environment will permit the optimal exploration y 3D scenario.

In the optimal exploration analysis we assumed that every time the robot gets to a previously visited region it is able to recognize it, however recognizing the stochastic nature of this event permits to make a more accurate prediction of the quality of the map. Considering every possible scenario, whether the robot is able to recognize or not previously visited regions, leads to multiple possible reconstructions. We already treated this problem using the expected uncertainty when planning a path, the natural extension consist in evaluating the expected uncertainty during exploration.

Appendix A

Proofs

A.1 Properties of the Information Function applied to the Laplacian Matrix

We will probe that $||\mathbf{L}(\mathbf{x}^*_{d,p})||_p$ is monotonically increasing in terms of both, "p" and "d". We also demonstrate that every information function has an upper bound linear in the number of traversals d.

Theorem A.1.1. *The set of information functions is an ordered set. Let* $\mathbf{Y}$ *be a positive definite matrix and* p, q *real numbers with* $p \leq q$ *then*

$$||\mathbf{Y}||_p \leq ||\mathbf{Y}||_q \tag{A.1}$$

This theorem is demonstrated in [Pukelsheim, 2006] as a consequence of the Jensen inequality.

Corollary A.1.2. *Let* $G(V, E)$ *the connected graph with Laplacian* $\mathbf{L}$*, the set of information functions have an upper bound for* $p \leq 1$.

$$||\mathbf{L}(\mathbf{x})||_p \leq \frac{2}{n-1}k \leq \frac{2}{n-1}d \tag{A.2}$$

Proof. This comes from the fact that in information functions if $p \leq 1$ then $||\mathbf{L}(\mathbf{x})||_p \leq ||\mathbf{L}(\mathbf{x})||_1$. Calculating $||\mathbf{L}(\mathbf{x})||_1$ we obtain the upper bound.

$$\begin{aligned}
||\mathbf{L}(\mathbf{x})||_1 &= \frac{1}{n-1}\sum_{i=2}^{n}\mu_i = \frac{1}{n-1}\sum_{i=1}^{n}\mu_i && \text{(A.3)}\\
&= \frac{1}{n-1}\cdot \operatorname{trace}(\mathbf{L}) && \text{(A.4)}\\
&= \frac{1}{n-1}\cdot \operatorname{trace}\left(\sum_{j=1}^{m}\mathbf{E}_j\mathbf{x}_j\right) && \text{(A.5)}\\
&= \frac{1}{n-1}\left(\sum_{j=1}^{m}\operatorname{trace}(\mathbf{E}_j\mathbf{x}_j)\right) && \text{(A.6)}\\
&= \frac{1}{n-1}\left(\sum_{j=1}^{m}\mathbf{x}_j\operatorname{trace}(\mathbf{E}_j)\right) && \text{(A.7)}
\end{aligned}$$

Using the definition of $\mathbf{E}_j$ we observe there are only two nonzero elements in the diagonal of these matrix, they are $[\mathbf{E}_j]_{ii} = [\mathbf{E}_j]_{kk} = 1$, as a consequence $\operatorname{trace}(\mathbf{E}_j) = 2$. Combining it with the previous equation we have

$$\begin{aligned}
||\mathbf{L}(\mathbf{x})||_1 &= \frac{1}{n-1}\left(\sum_{j=1}^{m}\mathbf{x}_j(2)\right) && \text{(A.8)}\\
&= \frac{2k}{n-1} && \text{(A.9)}\\
&\leq \frac{2}{n-1}d && \text{(A.10)}
\end{aligned}$$

■

We demonstrated the upper bound for every Laplacian only depends on the number of nodes n and the number of traversals k, independent of the topology of the graph.

Theorem A.1.3. *Let $G(V,E)$ the connected graph with Laplacian $\mathbf{L}$, the maximum information function is monotonically increasing with respect to the maximum traversals allowed d.*

Proof. This is a consequence of the concavity property of the information functions [Pukelsheim, 2006]. Lets consider the maximum information of the Laplacian $\|\mathbf{L}(\mathbf{x}^*_{k,p})\|_p$ among all possible traversals of length k according to the criteria associated to p. Now lets define $\mathbf{x_k}$ as a generic vector with $\sum x_i = k$. Using the definition we know that

$$\|\mathbf{L}(\mathbf{x}^*_{k,p})\|_p \geq \|\mathbf{L}(\mathbf{x_k})\|_p \tag{A.11}$$

One possible Laplacian matrix, not necessarily the optimal, comes from adding one generic edge to the optimal set of edges for k. Using the concavity of the information functions we can write

$$\begin{aligned} \|\mathbf{L}(\mathbf{x_{k+1}})\|_p &= \|\mathbf{L}(\mathbf{x}^*_{k,p}) + \mathbf{E}_j\|_p && \text{(A.12)} \\ &\geq \|\mathbf{L}(\mathbf{x}^*_{k,p})\|_p + \|\mathbf{E}_j\|_p && \text{(A.13)} \\ &\geq \|\mathbf{L}(\mathbf{x}^*_{k,p})\|_p && \text{(A.14)} \end{aligned}$$

Combining equations A.11 and A.14 we have

$$\begin{gathered} \|\mathbf{L}(\mathbf{x}^*_{k+1,p})\|_p \geq \|\mathbf{L}(\mathbf{x_{k+1}})\|_p \geq \|\mathbf{L}(\mathbf{x}^*_{k,p})\|_p \\ \|\mathbf{L}(\mathbf{x}^*_{k+1,p})\|_p \geq \|\mathbf{L}(\mathbf{x}^*_{k,p})\|_p \end{gathered} \tag{A.15}$$

■

We demonstrated that every information function is monotonously increasing with the number of traversals. In practical terms it means that monotony property is not a restriction for choosing the information function to apply to the Laplacian Matrix. The following corollary also deals with another practical issue

Corollary A.1.4. *The maximum value of* $\|\mathbf{L}(\mathbf{x})\|_p$ *is achieved traversing the maximum possible number of traversals*

$$\|\mathbf{L}(\mathbf{x}^*)\|_p = \mathscr{L}(d,p) \tag{A.16}$$

Proof. Considering the information function is monotonically increasing we know that if $k \leq d$ then $\mathscr{L}(k,p) \leq \mathscr{L}(d,p)$, consequently $\mathscr{L}(d,p)$ is the maximum value in the interval. ■

This implies that we only need to search for the vectors of traversals $\mathbf{x}$ with length d.

A.2 Properties of the Information Rate Index

We will probe that $E(\mathbf{x_k})$ is Scale Invariant and Bounded.

Property A.2.1. *Scale Invariance:* $E(\alpha\mathbf{x_k}, p) = E(\mathbf{x_k})$

Proof. Using the homogeneity of the information functions $||\mathbf{Y}(\alpha\mathbf{x_k})|| = \alpha||\mathbf{Y}(\mathbf{x_k})||$ we have

$$\begin{aligned} E(\alpha\mathbf{x_k})||_p &= \frac{||\mathbf{L}(\alpha\mathbf{x_k})||_p}{\alpha k} && \text{(A.17)} \\ &= \frac{\alpha||\mathbf{L}(\mathbf{x_k})||_p}{\alpha k} && \text{(A.18)} \\ &= \frac{||\mathbf{L}(\mathbf{x_k})||_p}{k} && \text{(A.19)} \\ &= ||\mathbf{E}(\alpha\mathbf{x_k})||_p && \text{(A.20)} \end{aligned}$$

■

Property A.2.2. *Bounded*

Proof. Trivial consequence of corollary A.1.2

$$\begin{aligned} E(\mathbf{x}, p) = \frac{||\mathbf{L}(\mathbf{x})||_p}{k} &\leq \frac{\frac{2k}{n-1}}{k} && \text{(A.21)} \\ &\leq \frac{2}{n-1} && \text{(A.22)} \end{aligned}$$

■

In order to find the optimal we can set the following integer maximization.

$$\begin{aligned} \text{maximize} \quad & E(\mathbf{x_k}, p) = \frac{||\mathbf{L}(\mathbf{x_k})||_p}{k} \\ \text{subject to} \quad & k \leq d \\ & (\mathbf{x_k})_i \in \mathbb{Z}^{+^m} && \text{(A.23)} \end{aligned}$$

One approach to integer programming is called *relaxation*, we remove the constraints of $\mathbf{x_k}$ of being a vector of integers and replace it with the vector $\mathbf{y_k}$ consisting on real numbers in order to approximate the solution. In our case we can use the property of the scale invariant to find the solution when $k = 1$, because for every value of $k \neq 1$ we only need to scale the solution, this way we can find the optimal value and its associated vector

$$\begin{array}{ll} \text{maximize} & \|E(\mathbf{y_k}, p) = \|\mathbf{L}(\mathbf{y_k})\|_p \\ \text{subject to} & k = 1 \\ & (\mathbf{y_k})_i \in \mathbb{R}^{+^m} \end{array} \tag{A.24}$$

Because the set of information functions is concave the it is guaranteed that the function has a maximum value $\mathscr{E}_p$, achievable for $\mathbf{y} \in \mathbb{R}^m$ and it is unique. Additionally it is achieved for $\mathbf{y}^*$,

$$\mathscr{E}_p \geq E(\mathbf{y_k}, p) \tag{A.25}$$

The optimal value $\mathscr{E}_p$ represents the theoretical maximum, the performance of every exploration can be compared with this value to define the efficiency of the estimator. Now we will demonstrate that the value $\mathscr{E}_p$ is the asymptotic value when restoring the integer constraint.

Property A.2.3. *The optimal value $\mathscr{E}_p$ represents the asymptotic value of equation 5.64*

$$\mathscr{E}_p = \lim_{k \to \infty} \left(\max \left(\frac{\|\mathbf{L}(\mathbf{x_k})\|_p}{k} \right) \right) \tag{A.26}$$

Proof. In order to demonstrate we need to recall the scale invariant property and some analysis of the integer numbers. Every real number y with a finite number of decimal digits r can be written as a weighted sum of integer numbers s_i with the property $0 \leq s_i \leq 9$.

$$y = \sum_{i=0}^{r} s_i * 10^{-i} \tag{A.27}$$

Multiplying by 10^r reveals that $10^r y$ is an integer number. Let's consider the vector $\mathbf{y}^*$ solution of the relaxed maximization problem. When the maximum number of decimal digits of any real number in the vector is bounded by r and using the scale invariant property we can write

$$\begin{aligned}
\mathscr{E}_p &= \max\left(E(\mathbf{y}^*), p)\right) && \text{(A.28)}\\
&= \max\left(E(10^r y^*, p)\right) && \text{(A.29)}\\
&= \max\left(\frac{\|\mathbf{L}(10^r \mathbf{y}^*)\|_p}{10^r}\right) && \text{(A.30)}
\end{aligned}$$

Because $10^r y^*$ is an integer vector we have the solution of the equation A.23 set in the integer domain. In order to finish the demonstration we take the limit $r \to \infty$ to consider the cases of real number with infinite number of decimal digits.

$$\begin{aligned}
\mathscr{E}_p &= \max\left(\lim_{r\to\infty}\left(E(10^r y^*, p)\right)\right) && \text{(A.31)}\\
&= \max\left(\lim_{r\to\infty}\left(\frac{\|\mathbf{L}(10^r \mathbf{y}^*)\|_p}{10^r}\right)\right) && \text{(A.32)}
\end{aligned}$$

Now let's define $k = 10^r$ and we can write

$$\mathscr{E}_p = \lim_{k\to\infty}\left(\max\left(\frac{\|\mathbf{L}(\mathbf{x_k})\|_p}{k}\right)\right) \qquad \text{(A.33)}$$

■

Bibliography

[Agha-mohammadi et al., 2014] Agha-mohammadi, A., Chakravorty, S., and Amato, N. (2014). Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, 33(2):268–304.

[Aydemir et al., 2012] Aydemir, A., Jensfelt, P., and Folkesson, J. (2012). What can we learn from 38,000 rooms? reasoning about unexplored space in indoor environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4675–4682. IEEE.

[Bhattacharya et al., 2012] Bhattacharya, S., Likhachev, M., and Kumar, V. (2012). Topological constraints in search-based robot path planning. *Autonomous Robots*, 33(3):273–290.

[Bormann et al., 2016] Bormann, R., Jordan, F., Li, W., Hampp, J., et al. (2016). Room segmentation: Survey, implementation, and analysis. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1019–1026. IEEE.

[Boyd, 2006] Boyd, S. (2006). Convex optimization of graph laplacian eigenvalues. In *Proceedings of the International Congress of Mathematicians*, volume 3, pages 1311–1319.

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

[Carrillo et al., 2012a] Carrillo, H., Latif, Y., Neira, J., and Castellanos, J. A. (2012a). Fast minimum uncertainty search on a graph map representation. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2504–2511. IEEE.

[Carrillo et al., 2012b] Carrillo, H., Reid, I., and Castellanos, J. (2012b). On the comparison of uncertainty criteria for active slam. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2080–2087.

[Ceriani et al., 2009] Ceriani, S., Fontana, G., Giusti, A., Marzorati, D., Matteucci, M., Migliore, D., Rizzi, D., Sorrenti, D. G., and Taddei, P. (2009). Rawseeds ground truth collection systems for indoor self-localization and mapping. *Autonomous Robots*, 27(4):353–371.

[Choi et al., 2009] Choi, J., Choi, M., and Chung, W. K. (2009). Incremental topological modeling using sonar gridmap in home environment. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3582–3587. IEEE.

[Dessmark and Pelc, 2004] Dessmark, A. and Pelc, A. (2004). Optimal graph exploration without good maps. *Theoretical Computer Science*, 326(1-3):343–362.

[Diestel, 2000] Diestel, R. (2000). *Graph theory*. Springer-Verlag Berlin and Heidelberg GmbH & amp.

[Dijkstra, 1959] Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

[Dissanayake et al., 2001] Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., and Csorba, M. (2001). A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241.

[Disser et al., 2016] Disser, Y., Hackfeld, J., and Klimm, M. (2016). Undirected graph exploration with θ (log log n) pebbles. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 25–39. Society for Industrial and Applied Mathematics.

[Fermin-Leon et al., 2017a] Fermin-Leon, L., Neira, J., and Castellanos, J. (2017a). Incremental contour-based topological segmentation for robot exploration. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2554–2561. IEEE.

[Fermin-Leon et al., 2017b] Fermin-Leon, L., Neira, J., and Castellanos, J. (2017b). Tigre: Topological graph based robotic exploration. In *Mobile Robots (ECMR), 2017 European Conference on*, pages 1–6. IEEE.

[Fermin-Leon et al., 2016] Fermin-Leon, L., Neira, J., and Castellanos, J. A. (2016). Path planning in graph slam using expected uncertainty. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4594–4601. IEEE.

[Fraenkel, 1970] Fraenkel, A. (1970). Economic traversal of labyrinths. *Mathematics Magazine*, 43(3):125–130.

[Fraigniaud et al., 2005] Fraigniaud, P., Ilcinkas, D., Peer, G., Pelc, A., and Peleg, D. (2005). Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2-3):331–344.

[Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

[Gerkey et al., 2003] Gerkey, B., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, volume 1, pages 317–323.

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

[Gross and Yellen, 2006] Gross, J. L. and Yellen, J. (2006). *Graph Theory and Its Applications*. Chapman and Hall/CRC.

[Guilin et al., 2014] Guilin, L., Zhonghua, X., and Jyh-Ming, L. (2014). Dual-space decomposition of 2d complex shapes. In *27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH. IEEE.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107.

[Hassan-Moghaddam et al., 2017] Hassan-Moghaddam, S., Wu, X., and Jovanović, M. R. (2017). Edge addition in directed consensus networks. In *American Control Conference (ACC), 2017*, pages 5592–5597. IEEE.

[Howard and Roy, 2003] Howard, A. and Roy, N. (2003). The robotics data set repository (radish).

[Ila et al., 2010] Ila, V., Porta, J., and Andrade-Cetto, J. (2010). Information-based compact pose slam. *Robotics, IEEE Transactions on*, 26(1):78–93.

[Indelman et al., 2015] Indelman, V., Carlone, L., and Dellaert, F. (2015). Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments. *The International Journal of Robotics Research*, 34(7):849–882.

[Kaelbling et al., 1998] Kaelbling, L., Littman, M., and Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134.

[Kaess et al., 2008] Kaess, M., Ranganathan, A., and Dellaert, F. (2008). isam: Incremental smoothing and mapping. *Robotics, IEEE Transactions on*, 24(6):1365–1378.

[Kalyanasundaram and Pruhs, 1994] Kalyanasundaram, B. and Pruhs, K. (1994). Constructing competitive tours from local information. *Theoretical Computer Science*, 130(1):125–138.

[Kavraki et al., 1996] Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580.

[Kelly, 2004] Kelly, A. (2004). Linearized error propagation in odometry. *The International Journal of Robotics Research*, 23(2):179–218.

[Khatib, 1986] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer.

[Khosoussi et al., 2014] Khosoussi, K., Huang, S., and Dissanayake, G. (2014). Novel insights into the impact of graph structure on slam. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2707–2714. IEEE.

[Kim and Eustice, 2014] Kim, A. and Eustice, R. M. (2014). Active visual slam for robotic area coverage: Theory and experiment. *The International Journal of Robotics Research*, page 0278364914547893.

[Kuffner and LaValle, 2000] Kuffner, J. J. and LaValle, S. M. (2000). Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE.

[Kummerle et al., 2011] Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g 2 o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE.

[Lam et al., 1992] Lam, L., Lee, S.-W., and Suen, C. Y. (1992). Thinning methodologies-a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 14(9):869–885.

[Lazaro et al., 2013] Lazaro, M. T., Paz, L. M., Pinies, P., Castellanos, J. A., and Grisetti, G. (2013). Multi-robot slam using condensed measurements. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1069–1076. IEEE.

[Lien and Amato, 2004] Lien, J.-M. and Amato, N. M. (2004). Approximate convex decomposition of polygons. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 17–26. ACM.

[Lien and Amato, 2006] Lien, J.-M. and Amato, N. M. (2006). Approximate convex decomposition of polygons. *Computational Geometry*, 35(1âĂŞ2):100 – 123. Special Issue on the 20th {ACM} Symposium on Computational Geometry 20th {ACM} Symposium on Computational Geometry.

[Liu et al., 2015] Liu, M., Colas, F., Oth, L., and Siegwart, R. (2015). Incremental topological segmentation for semi-structured environments using discretized gvg. *Autonomous Robots*, 38(2):143–160.

[Liu et al., 2011] Liu, M., Colas, F., and Siegwart, R. (2011). Regional topological segmentation based on mutual information graphs. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3269–3274. IEEE.

[Lumelsky, 1991] Lumelsky, V. J. (1991). A comparative study on the path length performance of maze-searching and robot motion planning algorithms. *Robotics and Automation, IEEE Transactions on*, 7(1):57–66.

[Megow et al., 2012] Megow, N., Mehlhorn, K., and Schweitzer, P. (2012). Online graph exploration: New results on old and new algorithms. *Theoretical Computer Science*, 463:62–72.

[Miyazaki et al., 2009] Miyazaki, S., Morimoto, N., and Okabe, Y. (2009). The online graph exploration problem on restricted graphs. *IEICE transactions on information and systems*, 92(9):1620–1627.

[Prentice and Roy, 2009] Prentice, S. and Roy, N. (2009). The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*.

[Pukelsheim, 2006] Pukelsheim, F. (2006). *Optimal design of experiments*. SIAM.

[Sagnol et al., 2015] Sagnol, G., Harman, R., et al. (2015). Computing exact *d*-optimal designs by mixed integer second-order cone programming. *The Annals of Statistics*, 43(5):2198–2224.

[Shamir, 2008] Shamir, A. (2008). A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556.

[Sim and Roy, 2005] Sim, R. and Roy, N. (2005). Global a-optimal robot exploration in slam. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 661–666. IEEE.

[Smith et al., 1990] Smith, R., Self, M., and Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer.

[Thrun, 1998] Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

[Thrun and Montemerlo, 2006] Thrun, S. and Montemerlo, M. (2006). The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429.

[Urcola et al., 2015] Urcola, P., Lazaro, M., Castellanos, J., and Montano, L. (2015). Generation of probabilistic graphs for path planning from stochastic maps. In *Mobile Robots (ECMR), 2015 European Conference on*, pages 1–7. IEEE.

[Valencia et al., 2011] Valencia, R., Andrade-Cetto, J., and Porta, J. (2011). Path planning in belief space with pose slam. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 78–83.

[Valencia et al., 2013] Valencia, R., Morta, M., Andrade-Cetto, J., and Porta, J. (2013). Planning reliable paths with pose slam. *Robotics, IEEE Transactions on*, 29(4):1050–1059.

[Van Zwynsvoorde et al., 2000] Van Zwynsvoorde, D., Siméon, T., and Alami, R. (2000). Incremental topological modeling using local voronoi-like graphs. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 2, pages 897–902. IEEE.

[Vandenberghe et al., 1998] Vandenberghe, L., Boyd, S., and Wu, S. (1998). Determinant maximization with linear matrix inequality constraints. *SIAM journal on matrix analysis and applications*, 19(2):499–533.

[Wan et al., 2008] Wan, Y., Roy, S., Wang, X., Saberi, A., Yang, T., Xue, M., and Malek, B. (2008). On the structure of graph edge designs that optimize the algebraic connectivity. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 805–810. IEEE.

[Wang et al., 2014] Wang, H., Jenkin, M., and Dymond, P. (2014). Deterministic topological visual slam. In *Proceedings of the Fifth Symposium on Information and Communication Technology*, pages 126–135. ACM.

[Yamauchi, 1997] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151. IEEE.

[Zavlanos et al., 2011] Zavlanos, M. M., Egerstedt, M. B., and Pappas, G. J. (2011). Graph-theoretic connectivity control of mobile robot networks. *Proceedings of the IEEE*, 99(9):1525–1540.

[Zunic and Rosin, 2004] Zunic, J. and Rosin, P. L. (2004). A new convexity measure for polygons. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(7):923–934.