Global Localization based on Evolutionary Optimization Algorithms for Indoor and Underground Environments

by

Juan Carballeira López

A dissertation submitted by in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Electrical Engineering, Electronics and Automation

Universidad Carlos III de Madrid

Advisor:

Luis Enrique Moreno Lorente

Tutor:

Luis Enrique Moreno Lorente

June 2022

Con un pouco de unto vello que o ben soupen aforrar e ca fariñiña munda xa tiña para cear. Fixen un caldo de groria que me soupo que la mar, fixen un bolo do pote que era cousa de envidiar.

Agradecimientos

En primer lugar quiero agradecer a mi familia, a mis padres Luis e Isabel y mi hermano Pablo por ser un referente en todos los sentidos aunque eso lo haga un poco más dificil a veces. Sin ellos esto no habría sido posible aunque no les haya hecho demasiado participes del proceso. A mis abuelos, que recuerdo con tanto cariño, a Alejo, Maribel, Carlos y Alexo, a Carla y a Alicia, a la que ya le explicaré esto cuando sea mayor.

En segundo lugar a mi tutor Luis Moreno por haberme dado la oportunidad de trabajar en esto, por haberme ayudado tanto, por sus grandes ideas y consejos y haber tenido tanta paciencia. Junto con él al resto del grupo de investigación: Santiago, Dolores, Dorin, Janeth, David Álvarez, David Serrano y al resto del personal del departamento y de la UC3M, ha sido un gusto ir a trabajar estos años. Incluyo también a Julio por ser mi primer tutorando y a Aitor por ir a escalar conmigo. Quiero resaltar entre todos ellos a mi compañero de fatigas Pavel, empezamos de la mano y de la mano apretando hasta el final.

También al resto del departamento y compañeros del master: Raúl, Quique, David, Edwin, Jorge (un correo y voy), Lisbeth, Luis, Carlos, Juanmi, Elisabeth, Olaya, Marcos....y alguno más que baja a comer conmigo y del que me olvidaré, pero no creo que se lo tome mal.

A la gente de Madrid, que si se HA convertido en mi hogar es principalmente gracias a la gente que me acompaña. A Laura, especialmente, por haberme soportado y ayudado tanto en los días de escritura, por la comida congelada, por enseñarme a escalar y por esas series interminables de capítulos de semanas. Pero sobretodo por ser una fuente de inspiración, admiración y por aportarme tantas cosas desde que nos conocimos, dolores de cabeza incluídos, pero principalmente muchas cosas buenas que ya te diré en persona. A Inés, Jon, Adriana, María (e incluso Raúl) por el día a día que compartimos, los buenos momentos juntos y las conversaciones trascendentales en los malos, es una gran suerte haberos encontrado por el camino. A la gente del único Rocódromo donde no escala nadie: Guillem, Silvia (y Gabi), Sergio y Juan especialmente, mi primer amigo "madrileño". A mi grupo de rock que duró 4 días. A tantos y tantos compañeros de equipo de fútbol que han pasado por mi vida y en los que encontré tantos amigos, en Ureca, la Cancha, el Javito y el club anteriormente conocido como Celta de Vino.

Á xente de Vigo, amigos de toda la vida que, como dice Fer con toda su sabiduría, ya es mucho decir, y tiene toda la razón. Espero no olvidar a nadie: a César, Du, Duni, Da, Susana, Isa, Rosa, Lara, Lucas, Peru, Bona, Siño, Andrea, Gabi, Nuñez, Jas, Perico, Migui y especialmente a Ainhoa por ser mi compañera de vida tantos años y haber empezado esta aventura conmigo. Aunque no os vea demasiado saber que estais ahí es muy importante para mí. También a la gente de la UVigo con la que aprendí tantas cosas, principalmete pinpón, especial mención para Kike y Davo, porque los veo más, no es personal.

Y por último pero no por ello menos importante...a Led Zeppelin, Pink Floyd, RATM, al Celta de Vigo y a mi perro Vladimir Gudelj Carballeira Vaamonde (Vlado).

Published and submitted content

During the development of this thesis the following research works were published:

A. Journals

- Using the Jensen-Shannon, Density Power, and Itakura-Saito Divergences to Implement an Evolutionary-Based Global Localization Filter for Mobile Robots. **Author**. July 2017IEEE Access PP(99):1-1 DOI: 10.1109/ACCESS.2017.2724199. This work is fully included in this thesis in chapters 3 and 4. The material from this source included in this thesis is not singled out with typographic means and references.
- Wildfire Spreading Simulator Using Fast Marching Algorithm.

  SNE Simulation Notes Europe 31(3):159-167 **Author**. DOI: 10.11128/sne.31.tn.10577

B.Conferences

- DENDT: 3D-NDT scan matching with Differential Evolution. July 2017Med Conference 2017. This work is partially included in this thesis in chapter 3. Whenever material from this source is included in this thesis, it is singled out with typographic means and an explicit reference

  Co-author. DOI: 10.1109/MED.2017.7984203
- Wildfire spreading simulator using Fast Marching algorithm

  Author. July 2019 10th EUROSIM Congress

Abstract

A fully autonomous robot is defined by its capability to sense, understand and move within the environment to perform a specific task. These qualities are included within the concept of navigation. However, among them, a basic transcendent one is localization, the capacity of the system to know its position regarding its surroundings. Therefore, the localization issue could be defined as searching the robot's coordinates and rotation angles within a known environment. In this thesis, the particular case of Global Localization is addressed, when no information about the initial position is known, and the robot relies only on its sensors. This work aims to develop several tools that allow the system to locate in the two most usual geometric map representations: occupancy maps and Point Clouds. The former divides the dimensional space into equally-sized cells coded with a binary value distinguishing between free and occupied space. Point Clouds define obstacles and environment features as a sparse set of points in the space, commonly measured through a laser sensor.

In this work, various algorithms are presented to search for that position through laser measurements only, in contrast with more usual methods that combine external information with motion information of the robot, odometry. Therefore, the system is capable of finding its own position in indoor environments, with no necessity of external positioning and without the influence of the uncertainty that motion sensors typically induce. Our solution is addressed by implementing various stochastic optimization algorithms or Meta-heuristics, specifically those bio-inspired or commonly known as Evolutionary Algorithms. Inspired by natural phenomena, these algorithms are based on the evolution of a series of particles or population members towards a solution through the optimization of a cost or fitness function that defines the problem.

The implemented algorithms are Differential Evolution, Particle Swarm Optimization, and Invasive Weed Optimization, which try to mimic the behavior of evolution through mutation, the movement of swarms or flocks of animals, and the colonizing behavior of invasive species of plants respectively. The different implementations address the necessity to parameterize these algorithms for a wide search space as a complete three-dimensional map, with exploratory behavior and the convergence conditions that terminate the search. The process is a recursive optimum estimation search, so the solution is unknown. These implementations address the optimum localization search procedure by comparing the laser measurements from the real position with the one obtained from each candidate particle in the known map. The cost function evaluates this similarity between real and estimated measurements and, therefore, is the function that defines the problem to optimize.

The common approach in localization or mapping using laser sensors is to establish the mean square error or the absolute error between laser measurements as an optimization function. In this work, a different perspective is introduced by benefiting from statistical distance or divergences, utilized to describe the similarity between probability distributions. By modeling the laser sensor as a probability distribution over the measured distance, the algorithm can benefit from the asymmetries provided by these divergences to favor or penalize different situations. Hence, *how* the laser scans differ and not only *how much* can be evaluated. The results obtained in different maps, simulated and real, prove that the Global Localization issue is successfully solved through these methods, both in position and orientation. The implementation of divergence-based weighted cost functions provides great robustness and accuracy to the localization filters and optimal response before different sources and noise levels from sensor measurements, the environment, or the presence of obstacles that are not registered in the map.

Resumen

Lo que define a un robot completamente autónomo es su capacidad para percibir el entorno, comprenderlo y poder desplazarse en él para realizar las tareas encomendadas. Estas cualidades se engloban dentro del concepto de la navegación, pero entre todas ellas la más básica y de la que dependen en buena parte el resto es la localización, la capacidad del sistema de conocer su posición respecto al entorno que lo rodea. De esta forma el problema de la localización se podría definir como la búsqueda de las coordenadas de posición y los ángulos de orientación de un robot móvil dentro de un entorno conocido. En esta tesis se aborda el caso particular de la localización global, cuando no existe información inicial alguna y el sistema depende únicamente de sus sensores. El objetivo de este trabajo es el desarrollo de varias herramientas que permitan que el sistema encuentre la localización en la que se encuentra respecto a los dos tipos de mapa más comúnmente utilizados para representar el entorno: los mapas de ocupación y las nubes de puntos. Los primeros subdividen el espacio en celdas de igual tamaño cuyo valor se define de forma binaria entre espacio libre y ocupado. Las nubes de puntos definen los obstáculos como una serie dispersa de puntos en el espacio comúnmente medidos a través de un láser.

En este trabajo se presentan varios algoritmos para la búsqueda de esa posición utilizando únicamente las medidas de este sensor láser, en contraste con los métodos más habituales que combinan información externa con información propia del movimiento del robot, la odometría. De esta forma el sistema es capaz de encontrar su posición en entornos interiores sin depender de posicionamiento externo y sin verse influenciado por la deriva típica que inducen los sensores de movimiento. La solución se afronta mediante la implementación de varios tipos de algoritmos estocásticos de optimización o Meta-heuristicas, en concreto entre los denominados bio-inspirados o comúnmente conocidos como Algoritmos Evolutivos. Estos algoritmos, inspirados en varios fenómenos de la naturaleza, se basan en la evolución de una serie de partículas o población hacia una solución en base a la optimización de una función de coste que define el problema.

Los algoritmos implementados en este trabajo son *Differential Evolution*, *Particle Swarm Optimization* e *Invasive Weed Optimization*, que tratan de imitar el comportamiento de la evolución por mutación, el movimiento de enjambres o bandas de animales y la colonización por parte de especies invasivas de plantas respectivamente. Las distintas implementaciones abordan la necesidad de parametrizar estos algoritmos para un espacio de búsqueda muy amplio como es un mapa completo, con la necesidad de que su comportamiento sea muy exploratorio, así como las condiciones de convergencia que definen el fin de la búsqueda ya que al ser un proceso recursivo de estimación la solución no es conocida. Estos algoritmos plantean la forma de buscar la localización óptima del robot mediante la comparación de las medidas del láser en la posicion real con lo esperado en la posición de cada una de esas partículas teniendo en cuenta el mapa conocido. La función de coste evalúa esa semejanza entre las medidas reales y estimadas y por tanto, es la función que define el problema.

Las funciones típicamente utilizadas tanto en mapeado como localización mediante el uso de sensores láser de distancia son el error cuadrático medio o el error absoluto entre distancia estimada y real. En este trabajo se presenta una perspectiva diferente, aprovechando las distancias estadísticas o divergencias, utilizadas para establecer la semejanza entre distribuciones probabilísticas. Modelando el sensor como una distribución de probabilidad entorno a la medida aportada por el láser, se puede aprovechar la asimetría de esas divergencias para favorecer o penalizar distintas situaciones. De esta forma se evalúa *cómo* difieren las medias y no solo *cuanto.* Los resultados obtenidos en distintos mapas tanto simulados como reales demuestran que el problema de la localización se resuelve con éxito mediante estos métodos tanto respecto al error de estimación de la posición como de la orientación del robot. El uso de las divergencias y su implementación en una función de coste ponderada proporciona gran robustez y precisión al filtro de localización y gran respuesta ante diferentes fuentes y niveles de ruido, tanto de la propia medida del sensor, del ambiente y de obstáculos no modelados en el mapa del entorno.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robotics, automation and "intelligent" systems have changed our way of life in the past century, from the way manufacturers and warehouses operate to, more recently, meddling in our daily life, affecting the way we commute every day, the way we communicate and our daily tasks at home. Robots and autonomous systems have evolved from aggressive industrial machinery aimed to deliver us from factory repetitive tasks into more flexible, mobile and autonomous devices, able to perform a task outside an specific workspace, react to unexpected situations, interact with all kind of objects and living beings. Hand in hand with this physical evolution, its computational ability or intelligence has evolved from simple active systems programmed to perform a task regardless of the environment, through reactive ones, able to sense, detect and interact and into intelligent systems capable of extracting conclusions from the environment, learn and decide.

Mobile robots are those that are able to move from one place to another autonomously, that is, without any physical help from external parties. Unlike the majority of industrial robots, typically manipulators, which are limited to an specific workspace, mobile robots have the specific quality of moving freely within the environment to achieve a desired goal. This mobility is precisely what makes them suitable for a large amount of applications, both in structured and non structured environments. Ground mobile robots are distinguished in Wheeled Mobile Robots and Legged Mobile Robots, but mobile robots also include Unmanned Aerial Vehicles (UAV's) or Autonomous Underwater Vehicles. They have become an essential tool in industrial an commercial environments. Mobile robots are easily found transporting materials in hospitals for several years. Warehouses take advantage of autonomous mobile robots to efficiently transport goods between loading zones and shelves for storage to complete customer orders. Military and security robotic systems have been historically on the lead of robotic research. And increasingly conquering domestic environments as consumer products, from household tasks, assistance for the

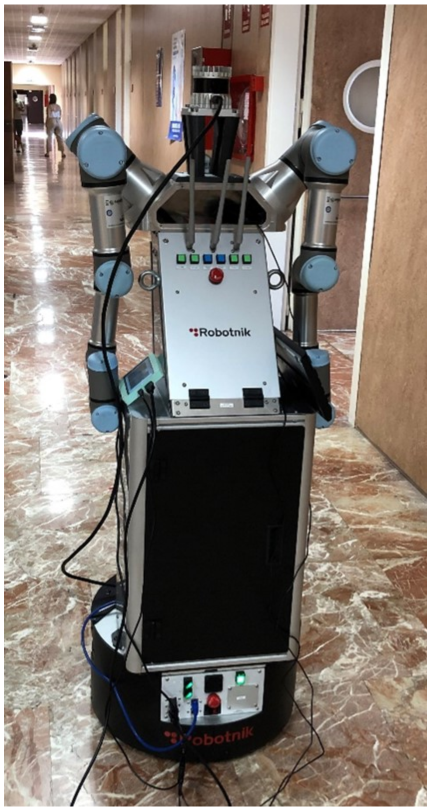

Figure 1.1: ADAM: Mobile manipulator platform developed in the Robotics Lab.

most dependent and as pure entertainment, good examples are autonomous vacuum cleaners or lawnmowers.

Therefore, a fully modern autonomous robot should be able to perform one or more specific or complex tasks obtaining information from the environment, moving throughout it without human assistance and avoiding circumstances that may prevent it from succeeding or that may risk its integrity. A further step would consist in learning and adopting new capabilities or new methods to perform the task. When fully autonomous, meaning no necessity of external assistance nor guidance, and working in a non-structured environment, the attention focuses in a fundamental issue: navigation. Navigation comprises everything a robot needs to get from point A to point B, interacting with the environment, as efficiently as possible. This problem breaks down into several sub-problems:

- It needs to know where it is (Localization).
- It needs to detect and avoid obstacles (Collision Avoidance).

- It has to be able to explore new terrain memorizing its surroundings (Mapping).
- It has to be able to plan a route from point A to point B (Trajectory Planning).

As its has been widely said and discussed there is not a specific or at least a unique definition of what a robot is, there are many kinds of robots, many kinds of autonomous robots, with different levels of autonomy, and the level required depends on the specific task and field. In addition, many other tasks or skills that define the performance of a robot rely on a proper self-location. This work will focus on the basis of navigation: localization, the capability of the system to know its current position and location within the environment.

Global Localization (GL) can be defined as the search of a robot's pose (position and orientation) in a two-dimensional (2D) or three-dimensional (3D) environment when the initial location is unknown. In this thesis, a set of optimization techniques based on Evolutionary Algorithms (EAs) will be developed as a continuation of the work carried out in the Robotics Lab research group of the Systems Engineering and Automation Department of the Carlos III University of Madrid (UC3M) in this field. These algorithms, which try to emulate nature in its way of selection for survival, will be presented and implemented as a feasible solution to this issue for different cases in 2D and 3D environments, both simulated and real. These maps represent the worst-case scenario of indoor or underground situations with no external information available, and the robot has to rely uniquely on its sensory system. The core of this selection is usually managed through an evaluation function. Different alternatives for these cost, fitness or selection functions will be presented as a more flexible approach than the commonly used Euclidean distance when comparing possible locations of the mobile robot based on range sensor information.

Real laser data used in this thesis has been acquired through the mobile platform ADAM(Figure 1.1), developed by the Robotics Lab group. The sensing capability of this platform includes RBG vision, time-of-flight Infrared data (IR), 2D and 3D laser telemetry, with a special prototype High-Density Light Detection and Ranging (HD-Lidar) designed specifically for mapping and localization. Differential Evolution algorithm (DE)[7] has been applied to a localization filter for simulated and semi-simulated environments, 2D and 3D occupancy maps of different indoor buildings. For 3D Point Cloud-based maps using real data acquired trough ADAM's LiDAR, Particle Swarm Optimization (PSO)[8] and Invasive Weed Optimization (IWO) [9] have also been developed and will be compared with the mentioned DE. The resulting GL-filters have been tested in different maps and different common situations including perception noise and the presence of obstacles. The overall performance and accuracy results prove to be comparable with other reserach group investigations, even outstanding in some situations.

The main objectives of this thesis are:

- Review the state of the art in localization for mobile robots and related topics as mapping and scan-matching.
- Review the state of the art on Evolutionary Optimization and its applications.
- Develop a Global Localization method based on Evolutionary Algorithms.
- Consider different approaches to map representation using different cost functions.
- Test and validate the proposed solutions in 2D and 3D environments.

This document is structured in five chapters. In this first chapter, an introduction to the Global Localization problem addressed in this thesis, the environment considered, and the data utilized together with the main objectives of this work were presented. A review of the existing solutions for mobile robot localization in known and unknown maps is provided in Chapter 2 under the title of State of the art, including different approaches to localization and the existing techniques regarding the utilization of laser beam sensor for geometric environment perception. A review of the existing stochastic optimization methods, with a deeper theoretical explanation of the implemented ones, is also included. Chapter 3 describes the different evolutionary techniques and their application to a multi-environment GL-filter, the considerations assumed for each map representation, and the adjustment of required parameters and data processing stages. A novel approach to optimization function selection for the Global Localization task based on probabilistic considerations of the laser measurements is presented, together with the specific cost functions implemented. The experimental results and their discussion are presented in Chapter 4, while the most remarkable conclusions and future developments are included in Chapter 5.

Chapter 2

State Of The Art

Once the set of tasks a mobile robot is determined to perform is defined, a challenging shared problem to solve for all of them is navigation through the environment or workspace. No matter what the task is or how thoroughly the characteristics of the environment are represented, to perform successful navigation, the robot accomplishes different phases: perception, localization, cognition, and motion control. In the perception phase, information about the surroundings is acquired through the robot's sensors to extract meaningful data to interpret the environment. If not previously provided, this interpretation could be used to perform a model commonly represented as a map. This map could include, in a simple categorization, geometrical, topological, or semantic information. The next step is the localization process, where the robot searches for its position within that map. Once the environment is represented and the position and orientation of the robot are known, trajectory planning and motion control stages will lead that robot towards its destination.

The main contribution of this document is to solve the Global Localization problem for 2D and 3D environments using evolutionary techniques on different map models. Although the recent tendency in autonomous robots is to combine the previously detached mapping and localization problems into Simultaneous Localization And Mapping (SLAM), a Global Localization module is still needed. The ability to accurately estimate the robot's position and orientation on a known map without any information about the initial location still solves the kidnapped robot situation, for instance, where even a mobile robot able to perform SLAM could lose previous information or could be abducted to an unidentified place. This chapter presents the current state of the art in the topics that affect our approach to GL: localization itself, different techniques of mapping focusing on metric representations through the use of Point Clouds (PC), SLAM, and Scan Matching techniques to search the spatial relation between PCs, and finally Evolutionary Algorithms and the different optimization functions that manage them.

2.1 Localization on Mobile Robotics

The main characteristic that differentiates a mobile robot from its predecessors, robotic arms and manipulators, is its ability to navigate within the environment, increasing its flexibility to adapt to a major number of tasks. As stated, navigation can not be performed without the localization module. The result of this localization provides position and orientation (x, y, θ) or $(x, y, z, \alpha, \beta, \gamma)$ depending if considering a 2D or 3D environment and the number of Degrees-Of-Freedom (DOF). Regarding the available information about the robot's initial location, the localization issue could be divided into position tracking, global localization, and kidnapped robot problem.

- Re-localization or tracking systems: Position tracking assumes that the initial position of the robot is known, at least with some uncertainty. The basic principle of these systems is to maintain a reliable estimation of the robot's pose for each lapse of time during its navigation. The vast majority of current localization algorithms present a solution to this problem mainly because of its simplicity and low computational cost in comparison to the GL situation. Tracking relies on proprioceptive (odometry mainly) and exteroceptive sensor information to reduce the uncertainty of the *estimate* or $belief$ for proper localization. A well-known and widely used example of this kind of algorithms is the Kalman and Extended Kalman Filters (EKF) [10], [11], [1], which acquire good results and are very efficient computationally speaking. The main drawback is that a fine initialization is necessary. In addition, statistical modeling of the properties of the system is to be determined accurately.

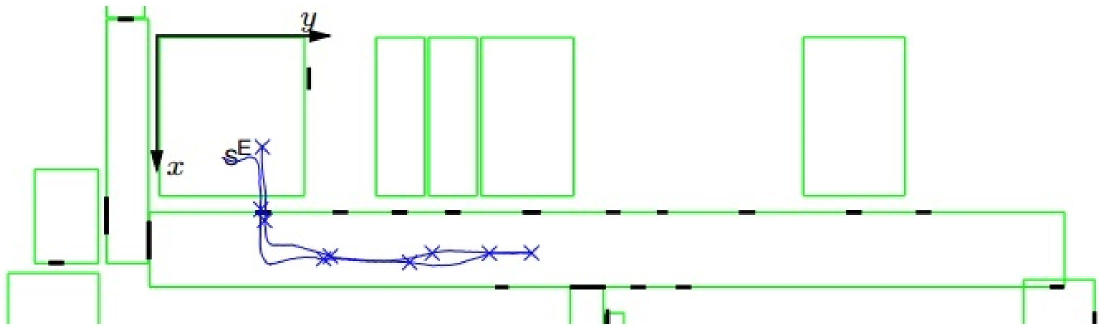

Figure 2.1: Tracking experiment using EKF [1].

- GL systems: On the other side, dealing with the absence of initial information opens up another category. This kind of methods do not assume any information on either position or orientation of the robot. Therefore, the whole environment must be considered as a search space, in contrast with tracking, where only a local area is necessary. This turns it into a more difficult task as the mathematical models as well as the information integration are more complex.

- Kidnapped Robot situation: A third category that could be considered a deviation from the previous ones deserves attention. In the kidnapped robot situation, a properly localized robot should be able to deal with the uncertainty of being abducted to an undefined position or losing pose tracking. Recovery from this situation is necessary and encompasses pose monitoring to be aware of being kidnapped and re-localization, which can be seen as a Global Localization problem.

We will focus on GL systems, as this work proposes different approaches to solve this issue. In a first division, GL can be classified into two different categories depending on how the information is provided: positioning and self-localization systems.

- Positioning systems rely on information exchange with the exterior, which may include both receiving position information from external sources (beacons) or even sending signals towards external sensors and receiving from them the location of the robot. In general, they use different methods to triangulate an estimation of the robot's location. Among the different positioning systems, the most widely known is the GPS. A positioning system that provides location and time information anywhere on the surface of the earth where there is no obstruction for electromagnetic waves toward four or more GPS satellites. In our case of study, indoor or underground spaces, it is far more difficult to establish a reliable positioning system. This is due to the difficulty of generating good coverage because of the necessity to place a high amount of emitters and receivers. Current indoor positioning systems using radio frequencies, vision, laser beams, or ultrasounds obtain accurate results but with the big disadvantage of incredibly depending on the external disposition of the elements, and even though they do not require a detailed map, which is an advantage, they surely depend on knowing the beacons' position, which makes them an almost inflexible system.

- Self-localization systems, on the other hand, are those that obtain the information directly from onboard sensors. Typical examples of these systems are LiDAR or computer vision-based for 2D and 3D scanning localization modules. Referring to the use of LiDAR, the idea is to acquire a set of distances from a laser sweep across the environment. This type of localization module requires a more detailed and complex predefined map than positioning systems. Self-localization systems have been studied in this project as our work is focused on indoor environments for which they are a more interesting solution from a conceptual point of view as well as more reliable and flexible. A particularization of self-positioning localization systems that do not require previous knowledge about the map is SLAM, first proposed in 1991 [12], which has become the main subject of study in the last decades. A review of SLAM is presented in section 2.1.2.

Different approaches, conceptual and mathematical, to all these different cases of GL problems have been considered. The next sections present two different categories for this approach: Probabilistic methods and Autonomous Map Building.

2.1.1 Probabilistic Localization Approaches

Probabilistic map-based global localization approaches identify and compute the probability of the robot to be located at each point of the environment representation or on a subset of candidate points. The exteroceptive and proprioceptive information noise affects the sensor data introducing uncertainty. Therefore, inevitably, localization can not be measured but only estimated. Among these methods, three categories could be distinguished:

- Bayesian-based methods: This type of method integrate motion and external sensor information into a *a posteriori* probability density function in a step of prediction and perception update. In a second step, the robot's pose is estimated according to specific criteria such as maximum density point or average value. Once it has converged, the probability distribution is focused on a small area surrounding the estimate. The key to these methods is to create an accurate model for the density function to cover the most feasible areas. In its basis, Bayesian approximation considers a continuous function for the probability density. Modifications of this concept have approximated and discretized this function to reduce complexity. These modifications have been widely studied and applied in the field of localization, common examples among them may be Grid-based probabilistic filters [13], [2], [14], Monte-Carlo (MC) localization methods [15]. MC reduces the search to a subset of particles to represent the approximate belief of the robot state, considering a small number of possible locations. In [16], Dellaert *et al.* applied a sampled-based representation of the density while Klaess *et al.* [17] considered 3D grid maps to apply MC with probabilistic observation models. Kummerle *et al.* [18] worked on a 6DOF localization of the robot for outdoor application using a MC particle filter by matching laser scans to a 3D map of the environment, a solution with the same approach as the presented in this work.

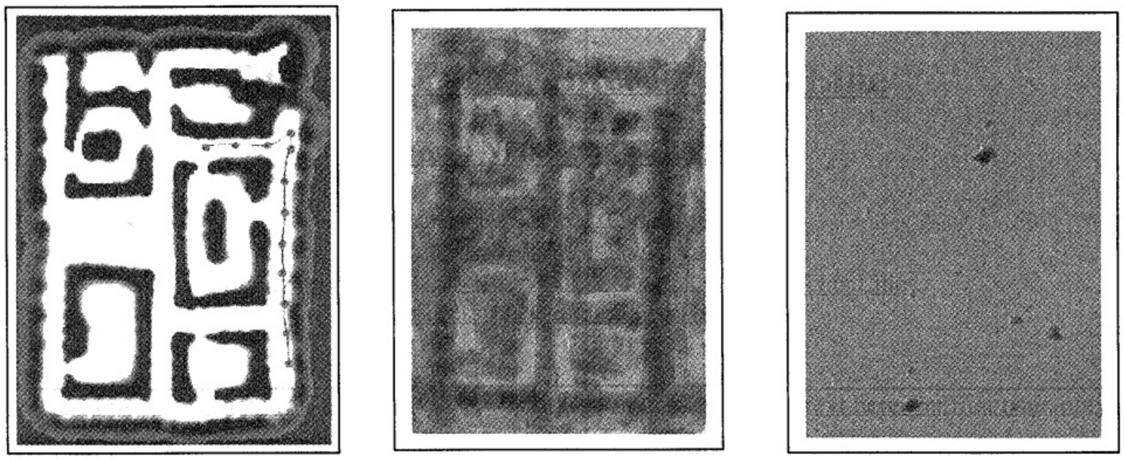
Figure 2.2: Evolution of the maximum probability position density with the amount of sensor readings over an occupancy grid map [2].

- Optimization-based methods: in this kind of algorithms, all the computational efforts are aimed at optimizing the value of a fitness or cost function. This function is designed to integrate all the system's information. Typically these methods are population-based, and after each cycle or iteration, the estimated pose of the robot results from the element with the optimum cost function value. Two different ideas result in two different strategies. The well-known Kalman filters derive the cost function to compute the solution. They present good computational performance, but they cannot deal with multi hypothesis distributions. Therefore, they have often been applied to tracking problems (re-localization), where only a single hypothesis is needed when computing the robot's pose. The second idea is to perform a stochastic search over the space of the optimum estimate. Evolutionary Algorithms or Metaheuristics such as DE or Particle Swarm Optimization (PSO) filters rely on this idea. Genetic Algorithm (GA) and Ant Colony Optimization (ACO) are a few examples of these kinds of algorithms. A specific review on these methods can be found in [19] and [20]. Both DE and PSO methods are compared with MC in [21]. Lisowski *et al.* [22] have implemented an hybrid version that mixes DE and MC. A different evolutionary technique called Harmony Search algorithm [23] is the basis of the GL filter designed by Mirkhania *et al.* [24]. As cited before, the DE method was applied in our previous work [25], [26], [27]. Ronghua and Birong [28] have proposed a mixture between MC and a genetic algorithm optimizer. Chien et al. [29] have applied PSO to implement a modified version of MC that avoids premature convergence. Other nature-inspired algorithms are used as in

[3], where Neto *et al.* proposed the Leader-Based Bat Algorithm to estimate the location through a smaller number of microbats in pursuit of the best place to start the colony. Comparison with standard Bat Algorithms, Particle Filter, and PSO is provided. An example of the performance of this method for the kidnapped robot problem is shown in Figure 2.3. The bat population (in blue) converges into a new position after an abduction.

Figure 2.3: LLBA algorithm behavior after robot kidnapping problem [3].

- Hybrid methods also called multi-hypotheses Kalman filters [30], [31], [32], [33] are another kind of algorithms were the set of solutions considers decision trees and geometric constraints. These methods imply Bayesian rules, are constituted by normal (Gaussian) probability distributions, but the generation, selection, or elimination of solutions are not only based on probability criteria. Each probability distribution is driven by a Kalman Filter, and to manage the problem, decision trees and geometric constraints are mixed together with probabilistic attributes to solve the problem. In [34], laser range measurements are fused with odometry to calculate a covariance for pose estimation through an MC method. The iterative Closest Point (ICP) matching algorithm uses that outcome as a starting point. A feature-based EKF is presented in [35] that extracts geometric features from the environment to perform the correction stage, while in [36] a combination between particle and Kalman filters is presented.

Different methods have been recently applied mainly for 3D environments and full

6DOF. Localization matching extracted features from data and model [37], multi-sensor fusion over an EKF-based solution for uneven indoor terrains [38], the use of laser range finder plus artificial reflector [39] and with the advance of technology in cameras and vision the increase of computer-vision techniques [40, 41], that turn localization into a scene recognition problem matching a different kind of features like Scale-Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF) [42]. Finally, as previously stated, the apparition of SLAM, which merges Localization and Mapping, has become the commonly selected approach to solve localization, mainly regarding 3D environments. The next section (2.1.2) reviews the state of the art in this particularization.

2.1.2 SLAM - Mapping

These previously discussed (Section 2.1.1) localization strategies all rely on the previous knowledge of the environment map. However, an increase in flexibility, the main attribute of mobile robotics, comes hand in hand with adaptation to large and dynamically changing environments. Global localization becomes a more difficult task in these environments, and previously conceived maps would intuitively lead us to expect higher inaccuracies in the estimation. These situations have been addressed in the past three decades through autonomous map building since the apparition of the SLAM concept [12]. The initial assumption for the SLAM technique is that there is no initial information at all, neither positioning nor environment representation. Therefore, the strategy of the robot is to explore its surroundings, acquiring sensory data, both proprioceptive and exteroceptive, recursively integrating this information with previous perceptions in a consistent manner to construct a map simultaneously with the navigation process. As stated, global localization estimates the current position and orientation of the robot within a map. For this map to be constructed, the robot's pose in each step must be known to maintain spatial consistency when incorporating new perceptions. Therefore, incremental mapping and localization are undetachable concepts.

The first concepts to be taken into account for robotic mapping are the desired or necessary complexity of the representation and the information sensing capacity available or required, depending on the autonomous system's purposes. An initial classification distinguishes between metric or geometrical, topological, and semantic maps. In almost every case, even for a topological or semantic representation, metric information is required for feasible navigation. Topological and semantic data represent a higher conceptual level, mainly focused on cognition and trajectory planning stages, and will be discarded in this work. Metric measurements are acquired through *idiothetic* and *allothetic*, proprioceptive and exteroceptive, self-proposition, and external purpose sensors. Those are typically encoders for odometry and 2D and 3D laser range sensors, LiDAR, InfraRed (IR), or Time-of-Flight (ToF) sensors for obstacle detection and environment measures.

Mathematically speaking, the SLAM problem has been divided into two different probabilistic approaches according to Siciliano *et al.* [43].

The first one, denominated $full$SLAM, addresses the problem by estimating the posterior of the complete robot's path together with the resulting map and its denoted by:

$$p(X_t, m|Z_t, U_t), \tag{2.1}$$

where X_t is the full path until instant t, m is the resulting map, $Z_t = \{z_1, ..., z_t\}$ contains the sensor perceptions for each pose in X_t and U_t is the propioceptive motion information for the path.

The second approach, *online*SLAM consists on the estimation of each current location of the robot independently:

$$p(x_t, m|Z_t, U_t), \tag{2.2}$$

where the difference is denoted by the term x_t which represents each individual location of the robot in contrast with the whole path estimation in *full*SLAM. Extracted from equations 2.1 and 2.2, the system must know two models to complete the process, a mathematical model that links the information from motion sensors u_t to locations x_t and another that converts exteroceptive perception z_t into these locations and the map m.

The use of probabilistic techniques has been widely studied for mapping [44, 45, 46] and initially led to satisfactory results when solving the problem posed. Typical examples are occupancy grid maps, where the map is divided into grids where every cell or voxel has an associated probability that distinguishes between free space, obstacle, or undetermined. Relative positioning between two different poses, determining its relative translation and rotation, is the key to map building and was typically approached with dead reckoning algorithms such as Kalman Filter, Markov, Montecarlo [47], and Extended Kalman Filter (EKF) [48]. The main drawback of these methods is that they typically rely on internal sensors such as encoders or Inertial Measurement Units (IMUs) and are consequently affected by their uncertainties. When the robot is moving, the position calculated by the internal sensors inevitably drifts, introducing inaccuracies and deviations. As a result, apart from dead reckoning from inertial sensors, external information obtained from the surrounding environment is usually needed for robot positioning, such as from vision sensors, range sensors, or GPS. With the development of visual sensors, visual-SLAM (vSLAM) has emerged as a successful approach. The use of vSLAM has been widely discussed as it simplifies sensor configuration, with less possible technical difficulties and simpler hardware mounted on the robot. As described in [49], vSLAM could be categorized in: featured based [50], direct [51, 52] and RGBD based [53] depending on whether it relies on features or distinctive points, the whole image or they combine image and depth information from RGBD cameras. Focusing on RGBD-SLAM, a technology that has been widely studied in recent years due to the apparition of low-cost sensors, the same direct [54], or feature-based [55] approaches have been developed with the particularity of the inclusion of depth information that could also be considered in isolation [56], with the same concept as laser-based SLAM.

There are different classifications that could be made on SLAM as it is a wide field of study, with multiple technologies involved and different types of maps and sensors implied in the process. Since this work presents techniques applied for both 2D and 3D geometrically described environments, an initial distinction could be made depending on the map dimensions and the DOF that define the robot's pose within that map. A pose in a two-dimensional map would be defined by a three-parameter vector $\overrightarrow{p} = (x, y, \theta)$ where θ represents the rotation over de z axis. In three-dimensional maps, the pose vector would be parametrized as $\overrightarrow{p} = (x, y, z, \alpha, \beta, \gamma)$ including the roll, pitch and yaw angles (α, β, γ). However, concessions are often made for 3D mapping depending on the DOF considered for the localization method and the availability of sensor information. Authors have constructed volumetric maps using a combination of horizontal a vertical 2D laser range finders [57, 58]. Nevertheless, it still remains a 3DOF localization method as the sensor used for that purpose still works in (x, y, θ) space.

Figure 2.4: 2D projection of 3D EKF-Slam resulting map presented in [4].

Focusing on the metric representation of the environment, SLAM could be classified into the next categories based on the strategy selected.

- EKF-based approaches: This approach was historically the first formulation for SLAM solutions. It consists of a probabilistic approach using a non-linear version of the Kalman Filter, the most widely used variant of Bayes filters. This type of filter integrates all the available information and computes all measurements (typically odometry and range information) to estimate the actual value of the robot's location. FastSlam is a well-known reference in this category [59]. Again like in pure GL, the uncertainty of sensor information could lead to inaccuracies. Initially developed for planar mapping using 2D laser range sensors, still is the most utilized SLAM strategy for 2D [60, 61, 62] and 3D SLAM solutions [4]. An example of the results can be observed in Figure 2.4. Cheein and Tobeiro propose in [63] a hybrid scheme between MC and EKF, which creates an uncertainty map to direct the robot to uncertainty regions while the MC scheme extracts these points and their estimated probability. In [61], Valiente *et al.* propose a comparison between Stochastic Gradient Descent and EKF. The latter presents better accuracy for low Gaussian noise environments. Limitations of these methods rely on assumptions of linear propagation means and covariances when normally sensor and motion models possess a non-linear nature.

Figure 2.5: (a) Laser measurements plotted according to raw odometry data (b) Occupancy map generated by the proposed SLAM algorithm [5].

- UKF-based approaches: this *evolution* of EKF was developed to cope with its limitations. Different authors apply this technique to SLAM [64, 65, 66]. The difference is the utilization of the unscented transform to linearize motion and measurement models. In [65], Schymura *et al.* develop a potential field strategy for the exploration of acoustic SLAM, proven to be more accurate and faster than previous methods. Fernandez *et al.* propose a strategy that combines odometry with panoramic images, applying MC localization. A UKF based approach with cartographer control reference using distinctive points in the surroundings was introduced in [67]

- Evolutionary-based approaches: Many evolutionary algorithms have been implemented to SLAM and mapping solutions. PSO heuristic methods are widely used for post-localization optimization. An improved version of FastSLAM was presented in [68] using PSO as a resampling method. In [69], an implementation of $(\nu + \lambda)ES$ and $(\nu + 1)ES$ evolutionary algorithms is considered for robot formation self-localization and multi-resolution mapping. Begum *et al.* propose a novel combination of Fuzzy Logic and Genetic Algorithm (GA) in [5]. Fuzzy logic substitutes normal distribution to model odometry uncertainty while GA iterates over a sample-based position uncertainty. The improvement for 2D mapping compared to odometry-based mapping can be seen in Figure 2.5

Most of these techniques have been mainly applied to planar mapping in 2D environment representation. When speaking of 3D representations for 6DOF localization, the environment information requirements increases, and so does the necessity to compensate for odometry inaccuracies and prediction through the environment information. Advances in vision, LiDAR end telemetric sensors hand in hand with computational capacity allow to switch from dead reckoning approaches to a more environment-based strategy. A well-known example of this technique was the apparition of the Iterative Closest Point (ICP) [70] applied in [71] to a mapping method based on the alignment of 3D scans combined with a heuristic for Loop Detection (when a point in the map is revisited and the position uncertainty can be updated). One of the more important concepts of laser-based SLAM is how to compare two scans reducing odometry errors between the two positions. This technique, known as Scan Matching, Scan Alignment, or Point Registration, estimates the rigid transformation between each pair of scans and thus the relative position of the robot between those scans. As each scan is received, it can be placed in reference to the previous one following this transformation and so recursively building a three-dimensional map. This strategy has become the main approach to 2D and 3D metric mapping, both for motion-based pose estimation error minimization and in isolation as a relative pose estimation method.

Algorithms developed in this thesis are applied to 2D environments and extended into two types of representation maps for 3D. Firstly, an occupancy grid map constructed through elevation of simulated and real 2D maps that we could consider semi-simulated. Finally, for the LiDAR 3D Point Cloud map obtained through the platform ADAM, mentioned in Chapter 1 and obtained through evolutionary-based

SLAM techniques [72]. Therefore, localization in 2D environments is given as a 3DOF vector, while for 3D is a full 6DOF localization method.

2.1.3 Scan Matching

This work presents an alternative approach to GL on previously known geometric maps applying evolutionary techniques. The importance of geometric representation of the space is crucial to understanding the solutions proposed for 2D and 3D environments. A review of the different localization and mapping techniques has been presented in sections 2.1.1 and 2.1.2 respectively. In this section, different approaches to Point Registration or Scan Matching are outlined. Scan Matching could be considered the milestone for this work as it encompasses both localization and mapping purposes in one concept. From now on, mapping references are always referred to as geometric Pont Cloud-based representations, although in some cases, image or RGB information is also present. One of the primary purposes of building a good consistent map is to be able to transform the scans or point cloud into another type of data like an occupancy grid to perform path planning and navigation. Scan registration optimizes the spatial error between perceptions to maximize the accuracy of the map. There are two wide categories on which the different point registration algorithms are spread depending on the approach: deterministic and probabilistic. To main algorithms deserve special attention as they have been the foundation of many implementations and variations through the years: Iterative Closest Point (ICP) [70] and Normal Distribution Transform (NDT)[73].

Figure 2.6: The Scan Matching Problem.

- Deterministic approaches:

  This category encompasses the ICP algorithm and its modifications. ICP is the predominant solution in this category and in point registration in general. It has been the seed and one of the most used and modified algorithms. Its functioning is quite straightforward, intuitive, and yet effective, the main reason for its popularity. The objective is to estimate the spatial transformation between a pair of scans or representative set of n-dimensional points implying rotation and translation (R, t) that matches both clouds and therefore obtaining the relative location of the sensor in those two instants (Figure 2.6). This is achieved by minimizing the least-squares of the distance of all the points that belong to different observations defined by scans A and B, which yields the minimum possible distances between them. Equation 2.3 depicts the L2-norm fitness function upon which the classic ICP algorithm is based, where the fitness value E is calculated by the summation of all Euclidean distances between each point a in A and each resulting point from applying rotation R and translation t to each point contained in the B set. N_a and N_b are the total number of considered points in both data sets. The ICP method has become the most common scan matching solution. This first approach paved the way for many other researchers that tried to improve its performance and implement different versions of the algorithm. ICP workflow consists of a recursive iteration, as each selected point in the source finds the closest point in the target. Then the transformation matrix is estimated and applied to the rest of the source selected points. The algorithm computes the root squared cost function and repeats the process until convergence conditions, typically when $E(R, t)$ falls under a threshold value. The key step and what implies the highest computational cost is the closest point selection step.

$$E(R,t) = \sum_{i=1}^{Na} \sum_{j=1}^{Nb} ||a_i - (Rb_j + t)||^2, \tag{2.3}$$

Many modifications and updates have been developed since its apparition, different distance considerations to minimize a cost function as Point To Line [74] or Point To Plane [75] which can increase the convergence speed.

Although a highly consistent method, one of the main drawbacks of ICP is its elevated computational cost. This seems obvious considering that laser sensor's technology and resolution have advanced enormously in the past years, increasing the scan sizes and inducing an exponential rise of computational times mainly during the nearest neighbor match stage computed for each point data of the source cloud. Therefore, reducing the number of representative points could optimize the solution. This reduction of data is mainly achieved by point cloud simplification or *downsampling* methods that could be random, uniform, or feature-based. Another approach was presented in [76], where, assuming that laser data is assorted in the same way when using the same sensor, modification of the coordinate system type into polar coordinates simplifies the nearest neighbor search by considering bearing in the polar coordinate space only. With the use of data structures like k-d tree [77] and octree, the closest point search is accelerated, increasing the ICP computational efficiency.

Other implementations of ICP try to avoid local minima convergence. Rough pre-alignments are considered previous to ICP refinement using feature descriptors or correspondences. Distinctive points are described by a set of parameters that define a characteristic point of the scan, considering the shape around that point, normally through normal vector computation. An initial alignment is fed to the ICP by matching features with similar descriptor values. Examples of the use of these features are Fast Point Feature Histogram [78] and SHOT [79]. A popular example of the pre-alignment strategy based on descriptors is RANdom SAmpling Consistency (RANSAC) [80].

The IMPR method, proposed by Lu and Milios [81], limits the maximum transformation considered (rotation and translation). Correspondences are found within a range of distance and rotation of the target, not considering the whole space. A variation combines ICP and IMRP. The translation is computed by the ICP method, and the rotation is estimated by the IMRP method.

Figure 2.7: Examples of 3D Mapping using solely LiDAR information and ICP-Scan Matching. Left: Indoor Environment. Right: Outdoor Environment.

- Probabilistic approaches:
  these methods are based on the use of Maximum Likelihood Estimator (MLE) [82]. The main representation of these approaches is the Normal Distribution Transform (NDT) method, which states that a scan can be subdivided into piece-wise continuous and differentiable probability density and uses this information to match successive scans using Newton's algorithm. Peter Biber introduced in [73] this new approach to solve the scan matching problem, which is different from the classical ICP. In this method, the matching is not done by finding the minimum distance between two scans. In the NDT algorithm,

the scans are represented by probability functions, and the alignment is done by getting the vector pose that optimizes a score that is computed according to the probability distributions. The values of the probabilities vary depending on the transform applied to the moving scan. The optimum score will be the minimum when both scans are aligned. By doing this, the NDT does not represent an occupancy grid but a probability grid. The first characteristic is that the point cloud space is divided into voxels of regular size. All the contained points in each voxel are taken into account to calculate the mean and the covariance matrix. The last step is to model each mean/covariance matrix pair through normal distribution. Finally, the PC is divided into N equally sized voxels defined by a normal distribution. NDT algorithm was expanded to 3D environments in [83] and compared to ICP, concluding that it could be considered faster. The probability of convergence was higher from a wider range of initial pose estimates, but so was the resulting rotation error.

Many other researchers have developed several variations of the NDT algorithm. For example, Takubo et al. [84] have designed an NDT version for high-resolution maps. Ulas and Temeltas have made a multi-layer NDT to extract features [85] and Einhorn and Gross [86] have implemented a 2D/3D SLAM method based on NDT maps. Most of these previous works have used the NDT method in the classical optimization way (like ICP), using an optimization method that relies on the gradient or Newton's algorithm. In our previous work [72] a Differential Evolution driven DE-NDT scan matching method was developed.

2.2 Evolutionary Algorithms

As it can be extracted from sections 2.1.1 and 2.1.2, the GL problem, for a previously known environment or through SLAM techniques, is an estimation process that involves uncertainties and probabilistic models. In this work, we consider the situation of localizing a mobile robot inside an indoor or underground environment, with no information from external sources (positioning) and no knowledge at all about its initial position, just each local scan obtained by a laser range finder. The larger the environment and the amount of data obtained (laser points), the bigger the search space for a solution. Obtaining a mathematical function or model to solve the problem implies simplifications that alter the outcome. In addition, the concept of localization itself requires adaptative approaches that should be flexible enough to adapt to different environments or information sources, implying that the more adaptative, general, or high level the strategy selected to solve this problem is, the easier to be expanded to a different situation or information sources. Therefore, considering all these qualities, a Stochastic Optimization approach is depicted as a feasible choice for this matter. In section 2.1.1 stochastic probability-based approaches have been presented, which were the base for the GL study.

Stochastic optimization methods use randomness to generate a non-deterministic solution or to emulate a non-deterministic behavior, implying rather than using a non-deterministic process that does not converge into a single solution from the same initial conditions to a deterministic problem (GL searches for a single solution even being an estimation) or introducing some kind of randomness in the process to imitate non-deterministic processes, with not one single solution. This randomness implied in the process does not mean that the optimization behavior is random. Optimization algorithms sample the space in a rational manner [87]. Classic stochastic approaches to approximate a probability density function are Markov and MC methods (2.1.1).

Another interesting concept of stochastic optimization is the inductive learning perspective. Inductive learning is defined as the capability of an individual to improve its characteristics through the passing of time when contrasted to an objective or reference. In optimization, this is represented in algorithms that are able to evolve a set of candidates or possible solutions in each iteration when contrasted to a function that evaluates each candidate's performance. This evaluation function receives many names in the literature: objective function, cost function, fitness function, or health function. Learning from the example establishes the assumption that a set of candidates as a representative sample of the whole domain could be a single solution or a population that iteratively, through a process of generation, test and selection, improves its knowledge about the environment (cost function). This perspective is acquired from k-armed bandit theory from the field of Game Theory [88], for enhancing and gaining information from the environment simultaneously with decision making (where to evolve towards). At the end of the 1980s and the 1990s, a sub-field of Artificial Intelligence (AI), Computational Intelligence, often referred to as $scruffy$ or $soft$ AI, started to focus on strategy-based algorithms. This field aimed to approach the constitution of complex behavior in a simpler or more intuitive manner, in a rather descriptive way based on the procedure more than the reason for success. This led to opening up to a wider spectrum of inspiration sources, and the result was the introduction of Natural and Biologically Inspired Computing, Fuzzy Systems, or Artificial Neural Networks. These techniques can be encompassed under the category of Metaheuristics. Those are strategy-based techniques that trade-off precision or quality in favor of effectiveness, simplicity, adaptability, and flexibility. Metaheuristics define a high-level strategy for a black-box problem type, where there is a poor mathematical definition of the problem. Basically, they are sets of procedures of the general appliance to optimization problems inspired by different fields of study, animals, plants, nature, biology, neurology, physic or cultural events, and many others. Metaheuristics are guides on a search for a near-optimal solution for non-deterministic unspecified problems. Although the proper nomenclature for this type of algorithm should be Metaheuristics, it should be noted that typically Neural Networks (NN) are detached from the rest of the approaches due to the great amount of investigation surrounding them in the last decade. The rest of the algorithms, even though not all of them involve genetic inspiration, are commonly denominated Evolutionary Algorithms.

Boosted by the advances of computational power in this century, these stochastic iterative processes have been possible to implement with the inherent computational waste. The sometimes inefficient random nature may revisit previously explored areas of the domain and may require a large number of samples to obtain an optimal solution. In contrast, the adaptability and non-dependence of prior knowledge of the problem can lead to innovative solutions in many implementations.

This section does not intend to present all algorithms, as the list is large and the common characteristics of some of them are difficult to extract but to overview different strategies that could be encompassed in a small classification. Depending on the field of inspiration, Metaheuristics excluding NN could be classified in:

- Nature Inspired:

  Nature-inspired or bio-inspired algorithms try to extract optimization techniques from different sources in nature. These sources are animal, plant, or genetically related. They have become the most commonly applied Metaheuristics for their adaptability and their intuitive understanding. The two most relevant categories for autonomous robotic applications are:

- Evolutionary Algorithms: There is an instant relation between stochastic optimization and evolution theory, adaptation, survival of the strongest, and Darwin's Theory. Evolutionary Algorithms try to mimic a simplified version of genetic material propagation mechanisms, mutation, and its adaptability to changes in the environment. The first expression of these algorithms was Genetic Algorithm (GA) [89], and modifications have created a wide range of other EAs. In this Metaheuristic, population-based, the parameters of the solution are codified as genes of each population member. The most qualified elements, in terms of the fitness function, contribute at a higher level to the genetic material of the next generation or offspring. DE [7] is a modified version of GA for a more exploratory approach in terms of global search. Differing from GA, all members of the population have random possibilities of contributing to the offspring. A extended review of DE is presented in section 2.2.1.

- Swarm Intelligence Algorithms: When considering intelligent behaviors in nature, it seems intuitive to appeal to swarm intelligence. Many animals benefit from rejecting individual conduct as they would not survive on their own. When observing nature, it is impressive to analyze some groups of animals' collective search for food, defense against predators, or route selection, and the parallelism with population-based algorithms seems natural. Many collective animals' behavior has been imitated, from ant colonies, swarms of bees, schools of fishes, or bacteria. The aim is to replicate the adaptability and scalability speed that collective intelligence implies. The most common particularizations of these algorithms are Bee Algorithm [90] and Particle Swarm Optimization (PSO)[8]. Bee Algorithm emulates food foraging of honey bee colonies. It is a population-based algorithm that establishes local searches through *exploratory bees*, communication possibilities to attract other bees of each exploratory bee depend on the cost function. In [91], BeA is used to distribute swarm robot members into different tasks depending on the necessities. Shortest trajectory planning using BeA is presented in [92]. A extensive explanation of PSO is given in 2.2.2.

- Physical Algorithms:

Physical algorithms encompass all those inspired by physical phenomena. It is a broad category that includes music-inspired algorithms like Harmony Search

[93], metallurgy inspired in Simulated Annealing [94], or society-based regarding knowledge and cultural habits in Cultural Algorithm [95]. In this author's opinion, the most remarkable algorithm for its peculiarity is Harmony Search, based on the search for harmony in jazz music improvisation. Each parameter of candidate solutions is represented by a musician in search of a good harmony in the band. Variations in the tone of each instrument in each iteration may find a better harmony that is stored in the memory of the group. Current work in our investigation group is working on applying HS to SLAM techniques. Another interesting physically inspired algorithm is Simulated Annealing (SA), inspired by metallurgic processes where metal is heated and slowly cooled to favor crystal formation. SA is a single element algorithm, randomly selected inside the search space, that manages the selection of a neighboring element through probability. Each element's temperature is calculated, but there is a probability of selecting a worse candidate even with the worst cost function value, escaping from local minimums.

- Probabilistic Algorithms:

  Probabilistic Algorithms generally alternate iterations between generating a set of possible solutions using a probabilistic model and converting the existing population into a probability model. Examples of this category are Cross-Entropy Method [96] and Bayesian Optimization Algorithm. Cross-Entropy assumes a probability distribution for the search space sampling using that distribution. Samples are determined individually by calculating the same type of distribution from a group of better candidates. With each iteration, the distribution is refined.

- Stochastic Algorithms:

  Although all optimization algorithms in this classification are considered stochastic, Stochastic Algorithms are those that lack an external source of inspiration rather than the stochastic concept itself and the strategy of the local search procedure. Examples of these algorithms are Random Search [97], Tabu Search[98] or Stochastic Hill Climbing [99]. Random Search is a direct search strategy based on uniform distribution sampling of the whole parameter searching space. Its heuristic is simple, for a limit of iterations, generate a random uniform distribution over the search space and, evaluate all samples, select the more efficient one in terms of cost until the iteration limit is reached. Its main applications in robotics are the initialization for other optimization methods and as a standard comparison of the performance of other algorithms.

As stated, we have selected three different nature-inspired algorithms to develop a GL filter that is presented in the next sections. Those are DE, PSO, and Invasive Weed Optimization (IWO), a rather recent bio-inspired stochastical optimization algorithm based on invasive habits of weed expansion. Selected algorithms are population-based, with an explorative nature which makes them suitable for large search spaces like GL in a full 3D environment. An explorative nature is needed regarding the search space considered in largely known maps and the number of local minimums encountered in possible symmetric locations.

2.2.1 Differential Evolution

The solution proposed is based on the DE method described by Storn and Price in [7] for global optimization problems over continuous spaces. It has been developed in previous work [25] and [27]. A theoretical description will be given in this section. Differential Evolution is included in the nature-inspired Evolutionary Algorithms. It is a population-based stochastic optimization method that presents an artificial representation of the mutation effects in evolution. Although it could be stated that the search space parameters are coded as *chromosomes*, it is not considered a genetic algorithm as its metaphoric definition is distant from accurate evolution theory explanations. DE considers a fixed set of possible solutions and performs a competition procedure for every single candidate in terms of the cost function value. Each competition confronts each element of the population and a mutated combination of three randomly selected elements. The possible next-generation member is a genetic (parameter) mixture of the original candidate and that mutated element. For each pair, the fittest in terms of evaluation function value reaches the next generation.

The algorithm starts with a randomly generated population of N_p candidates, defined by d parameters each, to cover the entire map. The size of this population remains constant during the whole process. Each d-dimensional vector $x_i^k = (x_{i,1}^k, ..., x_{i,d}^k)$ is a possible solution to the optimization problem in the k_{th} generation, where $i \in [0, Np]$, and works as starting point for Np parallel direct searches. After initialization, a new population is generated by adding the weighted difference between two candidates to a third vector. This operation is called *mutation*. A mutated vector's parameters are then mixed with each of the original vector's parameters, and a randomly selected mutated vector is assigned to each candidate. To complete each iteration, the two populations, old and new, are compared through a cost function in order to select the Np better members in each performance.

The first step after initialization in a DE algorithm is to generate new parameter

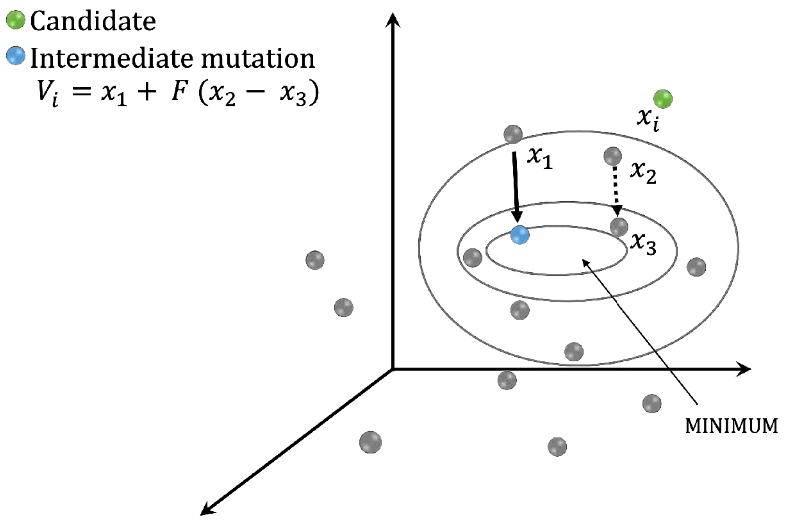

Figure 2.8: Example of a three-dimensional parameter mutated vector generation.

vectors by adding a weighted difference between two random members of the population to a third one and so, creating a mutated equivalent for each member of the old population. This way an intermediate population is created such as:

$$v_{i,k} = x_{r0,k} + F * (x_{r1,k} - x_{r2,k}),$$

with random different indexes *r1,r2* and *r3* $\in [1 - N_p]$ indicating the three randomly selected candidates in generation k and parameter $F > 0$, which controls the amplification of the difference or *mutation*. A three-dimensional example of this new member generation is shown in Figure 2.8.

In order to increase the diversity of the perturbed vectors, the crossover concept is introduced. Denoted by u_i^k, new intermediate population members are created from the combination of parameters from each original population member x_i^k of the old generation with other parameters from the previously mentioned mutated candidates v_i^k. To this point, the trial vector $u_i^k = (u_{i,1}^k, u_{i,2}^k, ..., u_{i,d}^k)$ is created with its parameters being:

$$u_{i,j}^k = \begin{cases} v_{i,j}^k, & \text{if } p_{i,j}^k \leq CR \\ x_{i,j}^k, & \text{otherwise} \end{cases}$$

where $p_{i,j}^k$ is a randomly chosen number among the interval [0,1], typically over a normal distribution, for each parameter j of each candidate i. d is usually defined as the number of chromosomes, the number of parameters that define the search space, and hence each candidate vector. The crossover stage is controlled by the

Figure 2.9: Illustration of the crossover process for D = 6 parameters.

CR variable, which marks the probability of assuming parameters from a mutated member of the population over the existing ones. An illustration of the crossover process for a 6-dimensional vector is given in Figure 2.9.

In order to decide whether this new intermediate population member should replace or not its competitor in the next generation of candidates, each one of the trial vectors u_i^k is compared with the target (original) member of the population x_i^k through a cost function. If vector u_i^k delivers a smaller cost function value than x_i^k then it will replace it as a member in the next generation, otherwise the old candidate x_i^k is retained.

There are many applications in several optimization fields of the DE metaheuristic. In the field of robotics, several examples have already been mentioned as our previous work on DE-based GL, [25, 26, 27] or DE-based NDT scan matching [72]. In [100], Joshi *et al.* present an optimization of the minimal representation approach for multi-sensor-based robotic manipulation path planning. A DE modified scheme which employs a statistical representation of the population rather than the actual estimates is presented in [101] for computational cost reduction in a Cartesian robot motion control. A multi-robot path planning optimization using a combination of DE optimization and Q-learning was developed in [102]. In [21], a comparative of population necessities, convergence rate, and speed between DE, PSO, and MC methods is performed for a 2DOF robot in a 2D map representation. Conclusions yield that metaheuristic approaches require fewer particles and less processing time to converge into a localization solution. Unluckily no accurate information is given.

2.2.2 Particle Swarm Optimization

A second solution is proposed based on another nature-inspired stochastic optimization algorithm, in this case, in the field of swarm intelligence. Our election, motivated by the simplicity of the concept, was PSO, formulated in 1995 by Eberhart and Kennedy in [8]. A theoretical description will be given in this section. It is a population-based Meta-heuristic that presents an artificial representation of the swarm behavior of schools of fishes or flocks of birds in the search for an optimum path during navigation or flight. PSO, in contrast with Bee Algorithm or Ant Colony, does not imitate a particular collective, hence the use of the world *particle*. Immediate parallelism can be established between the position evolution of each member of a swarm during movement towards a goal and the search space of a set of particles to find the optimum solution to a GL problem. The search space is sampled with a set of particles using a uniform distribution. Through all iterations, PSO maintains a particle memory and a global memory, where each candidate keeps track of its particular best position and the swarm's collective best position, respectively. On each iteration, every member is awarded a velocity, consisting of a weighted sum of current velocity and distances to personal and collective best.

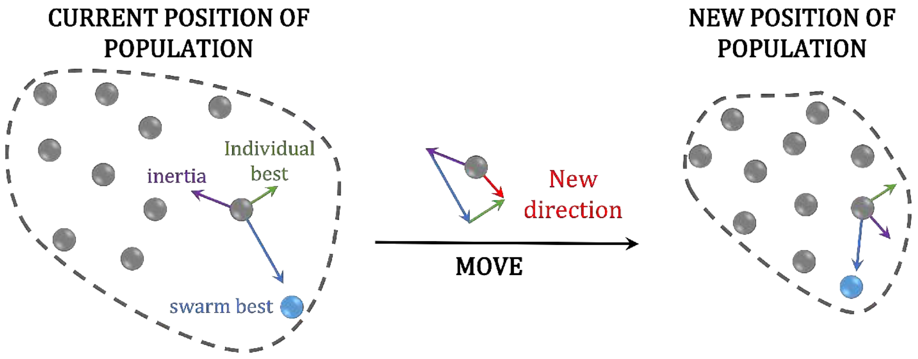

Figure 2.10: Illustrative example of swarm movement.

The algorithm starts with a randomly generated population of N_p particles, defined by d parameters each, to cover the entire domain. This is typically done through a uniform distribution. The size of this population remains constant during the whole process. Each d-dimensional position of particle $p_i^k = (x_{i,1}^k, ..., x_{i,d}^k)$ is a possible solution to the optimization problem in the k_{th} movement, where $i \in [1, Np]$, and works as starting point for Np parallel direct searches. After initialization, the position of the whole swarm is redistributed, recalculating each particle's velocity in terms of distance and direction within the search space coordinates by a weighted sum of vectors (Figure 2.10):

$$v_i(k+1) = [wv_i(k)] + [c_1 r_{d1}(p_i^{best} - p_i(k))] + [c_2 * r_{d2} * (p_{gbest} - p_i(k)];$$

where $v_i(k+1)$ represents the new velocity assigned to the i^{th} particle, w is the inertia weight indicating the influence of the current velocity $v_i(k)$, c_1 and c_2 are the bias weights attributed to the influence of the differences between i^{th} particle current position with its historical best p_i^{best} and with the swarm's global best p_{gbest} respectively. Coefficients r_{d1}, r_{d2} and r_{d3} are uniformly selected random variables $\in [0,1]$.

A maximum velocity for each dimension of the search space is set so that:

$$v_{i,j}(k+1) = \begin{cases} v_{i,j}(k+1), & \text{if } v_{i,j}(k+1) \leq vmax_j \\ vmax_j, & \text{otherwise} \end{cases}$$

where j represents each coordinate of the vector $\in [0,d]$ where d is the dimension of the domain. The new position of the i^{th} particle is therefore obtained from:

$$p_i(k+1) = p_i(k) + v_i(k+1),$$

In contrast with evolutionary algorithms, there is no selection mechanism or competition. The particles all remain the same, and they are simply modified by updating their position. Each iteration terminates with the update of individual and collective memories. After the set of particles' positions is modified, it is evaluated through a cost or fitness function defined depending on the optimization problem. If the new position $p_i(k+1)$ provides a better cost value than its personal best or than the global best, any of those are replaced. The new position becomes that singular particle's own reference or the entire swarm's reference.

$$p_i^{best} = \begin{cases} p_i(k+1), & \text{if } f_c(p_i(k+1)) \leq f_c(p_i^{best}) \\ p_i^{best}, & \text{otherwise} \end{cases}$$

$$p_{gbest} = \begin{cases} p_i(k+1), & \text{if } f_c(p_i(k+1)) \leq f_c(p_{gbest}) \\ p_{gbest}, & \text{otherwise} \end{cases}$$

PSO has been widely implemented in a wide variety of fields, including robotic applications. As a paradigm in swarm-intelligence-based algorithms, its utility in multi-robot scenarios is obvious [103, 104, 105, 106]. In [107] a GL filter using PSO for scan-matching based localization over grid maps was implemented, and results show localization errors lower than 10cms. Havangi *et al.* implemented a PSO-based sampling method for a FastSLAM improved version using unscented transform [108]. Another intuitive application is path planning. An example can be found in [109], where Roberge *et al.* present a comparison between implementations of GA and PSO for fixed-wing UAV's optimal trajectory planning in complex 3D environments by implementing multi-objective cost functions. The same approach is employed in [110] for combined path planning and obstacle avoidance applying the Probabilistic Road Method in a multi-objective cost function.

2.2.3 Invasive Weed Optimization

A third iterative bio-inspired metaheuristic has been considered for implementing a GL filter. In this case, a rather recent approach in comparison with the early mentioned DE and PSO (2.2.1,2.2.2), which mimics the colonial behavior and the survival adaptability of invasive weed plants. Invasive Weed Optimization was introduced in 2006 by Mehrabian *et al.* in [9], as an attempt to imitate the facility of weed plants to colonize unexpected fields, with adaptability to changes in the environment, endurance in the field, reproductive speed and rapid occupation and displacement of other plant species. These are interesting qualities for our case of study, where a large search domain and multiple and unpredictable local minimums are always to be expected, and good coverage of the variable domain appears like an appropriate solution. Like previously presented algorithms, IWO samples the domain through an initial randomly distributed set of plants. Each plant is awarded a distinctive reproduction capability depending on its suitability to endure in the environment. Depending on this reproductive ability, a number of seeds are spread within a distance of each plant's position in the domain. In each generation, the new set of scattered seeds is evaluated and competes with original plants so that only the fittest ones are selected once a maximum number of elements in the colony is reached.

The algorithm starts with a uniform distribution of an initial random population of N_{ini} seeds over the d-dimensional search space, where d indicates the number of parameters that determines a possible solution for the optimization problem. The size of this population, unlike previously analyzed heuristics, evolves in each generation k until a specific maximum N_{max} is reached. Each d-dimensional position of plant $p_i^k = (x_{i,1}^k, ..., x_{i,d}^k)$ is a possible solution to the optimization problem in the k_{th} generation of the plant colony, where $i \in [1, N_k]$, and works as starting point for N_k parallel direct searches. Each seed grows into a plant and is awarded for spreading a number of seeds depending on its position within the colony's fitness minimum and maximum, as shown in Figure 2.11. This fitness is evaluated through the cost function that defines the optimization problem.

Extracted from this concept, the number of seeds awarded to plant p_i^k is:

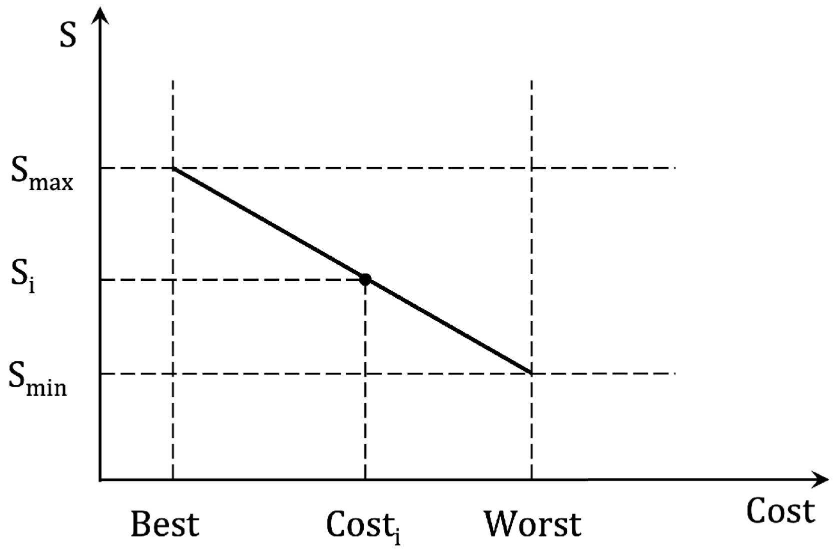

Figure 2.11: Seed retribution for each plant. S:No of seeds. Cost: fitness value.

$$S_i^k = S_{min} + (S_{max} - S_{min}) \frac{(c_{p_i^k} - c_{wrst})}{(c_{best} - c_{wrst})},$$

S_{max} and S_{min} are two constants representing the reproduction capability chosen for the more and less suitable plant in each iteration in terms of the objective function. This cost is represented by c_{best} and c_{worst} respectively and $c_{p_i^k}$ is the fitness function result for each plant i in the colony on generation k. For each plant, each S_i^k generated seed will be spread randomly within a distance from its parent plant. The new position is a random number within the normal distribution around the original position of the parent plant so that:

$$p_{i,j}^k = rand(f(x)),$$

$$f(x) = \frac{1}{\sigma_{it}\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-p_i^k}{\sigma_{it}})^2},$$

where $j \in [1, S_i^k]$ represents an index for every scattered seed from plant p_i^k, and $f(x)$ is the normal distribution over the original position p_i^k with standard deviation σ_{it}. The standard deviation for this normal distribution is defined by a variable standard deviation, σ_{it}:

$$\sigma_{it} = \frac{(it_{max} - k)^n}{(it_{max})^n} (\sigma_{initial} - \sigma_{final}) + \sigma_{final}, \quad (2.4)$$

As it can be extracted from equation 2.4, for each generation k the standard deviation σ_{it} decreases exponentially from an initial value $\sigma_{initial}$ to a final value σ_{final}. The exponent n controls the decreasing speed of the possible distance of dispersion of seeds on each iteration from initial and final values. Therefore, the capacity of each plant in the colony to spread its awarded seeds far from its own position decreases as the colony evolves to an optimum position in the search space.

In the final stage of the algorithm, in order to decide which elements endure in the next generation of the colony, plants and spread seeds are evaluated in a collective manner, not individually, as explained in section 2.2.1 for DE. The best elements succeed and remain part of the colony while the worst are eliminated. This occurs once the maximum possible plants in the colony N_{max} is reached. Until then, all seeds are accepted.

IWO is a relatively novel algorithm in contrast with previously presented DE and PSO or other more *classic* bio-inspired methods such as GA or Ant Colony; therefore, less literature can be found in robotic field applications. Nevertheless, there is some interesting work developed as [111], where a minimum energy suitable path for robotic arm trajectory planning using IWO was presented. In [112, 113] IWO is applied to solve navigation issues. Panda *et al.* [113] used a combination of PSO and IWO to optimize each path of a multi-robot system applied in dynamically changing environments, optimizing the time and distance of each robot mixed with collision avoidance. Another interesting approach is the optimization of Q values on an LQR controller to control the Robogymnast presented in [114].

2.2.4 Fitness Functions

As depicted in section 2.2, stochastic optimization algorithms or Metaheuristics are iterative strategy-based processes to estimate a solution for any problem that can be presented as the search for a combination of certain parameters that optimizes certain cost or fitness function. Therefore, Metaheuristics are not task-oriented algorithms but applicable methods to a broad amount of problems, hence the use of the prefix *meta*.

When applying any Metaheuristic, like bio-inspired optimization or Evolutionary Algorithms, to a given task, the efficacy of the method highly depends on the function that defines the problem or the target function to optimize. This function receives many names in literature, fitness function, cost function, health function, or target function.

Global Localization is a physical task, meaning that the goal is to obtain an accurate position of the robot in a 2D or 3D space, and typically it is achieved through acquiring metric information about the environment by measuring distances from the robot to different obstacles within the sensor range. Most common sensors are laser range finders or ToF sensors like IR depth cameras, so the goal is to find a good estimation that minimizes the error between prediction and actual observations of the sensor. Information from laser measurements comes in a discrete form, as a set of (x_i, y_i) or (x_i, y_i, z_i) points, depending on the dimensions of the environment, hence, GL could be reduced to a Scan Matching problem as introduced in section 2.1.3. Point registration techniques use distance metrics as cost functions, as their objective is to minimize the distance between observed and predicted points. Therefore the applied cost functions are distance-error minimization related. Among all of them, Root Mean Squared Error (RMSE) or L2 loss is the most common:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n}(p_i - \hat{p_i})^2}{n}},$$

where n is the amount of point pairs considered, p_i is the real observation point and $\hat{p_i}$ is the estimated observation. Other options involving distance error are, the Mean Absolute Error or L1 loss:

$$MAE = \frac{\sum_{i=0}^{n}|p_i - \hat{p_i}|}{n},$$

computed as the sum of absolute errors between actual and estimated points or a simple Mean Error (ME).

Iterative Closest Point, the reference on scan matching processes, uses the RMSE between transformed source points and target cloud in order to minimize the matching error, the expression for the ICP cost function can be observed in Equation 2.3. Feature-based ICP variations rely on walls, columns, or other singular features of the known map to distinguish among the different possible locations, matching is done for a reduced set of points, but the cost function remains the same. The absolute error was chosen in our previous work for a comparison with RMSE [115]. Other authors have also proposed different approaches. Donoso *et al.* minimize the distance between two sets of points through the Hausdorff distance [116]. Fox *et al.* [2] not only consider the perceived information but also minimize the "future expected uncertainty, which is measured by the entropy of the future belief distributions". The Mahalanobis distance is adopted by the future-based approach made by Arras *et al.* [30].

Chapter 3

Evolutionary Techniques for Global Localization Filters

As stated in previous chapters, a proper localization module is key to successful performance in a baste amount of tasks a mobile robot has to perform. The localization task can be defined as the knowledge or estimation of all the necessary parameters that define the robot's pose inside an environment. Two very different situations could be considered: Re-localization or tracking, where the system knows at least approximately its initial location, and global localization or kidnapped robot situation when there is no knowledge of the initial pose, this information was lost, or the robot was abducted to an unspecified place. Hence, global localization or at least recovery from a kidnap situation is necessary for an autonomous robot. This second kind of situation will be studied in the next chapters, and a set of implementations of feasible methods for solving this problem are presented.

Our approach is based on the assumption that there is previous knowledge of the geometric map that defines the environment where the autonomous mobile robot performs its tasks. Depending on this map, localization becomes a two-dimensional problem, where the pose of the robot is typically defined by position (x, y) and rotation over the z axis θ, or a simplified 3D where it is assumed that rotation only takes place within the horizontal plane and therefore the robot works with 4DOF, (x, y, z, θ). In previous work of our research group, an evolutionary filter solution was presented for a 2D map [25] which solved the GL problem satisfactorily and was then extended to 3D as an improvement [27]. This evolutionary filter, based on DE algorithm, has been the foundation of this work. Extracted from that work, the evolutionary filter structure, noise modeling, and a comparison between L1 and L2 norms as fitness functions for the selection mechanism served as an inspiration to continue the development into a fully 6DOF filter, with probabilistic modeling of the laser sensor measure and the implementation of different non-symmetric statistical divergences as cost functions to manage the evolutionary filter. In this work, we will address both 2D and 3D situations, but in the latter considering that the robot could vary any of the 6DOF that fully define a three-dimensional environment. Therefore its pose would be defined by $(x, y, z, \alpha, \beta, \gamma)$ encompassing position and orientation (roll, pitch, yaw) in a 3D space. But more important than the classification of 2D and 3D in this work is the difference between sparse and continuous maps, or Point Cloud-based and grid-based maps as it could also be presented. As it can be seen in Figure3.1 two different concepts of environment models will be utilized for indoor situations. A sparse Point Cloud environment model reconstructed through evolutionary mapping techniques presented in [72] and a continuous one, where a structured simulated laser cloud could be generated from any given position. The first is formed by a set of spatial points representing laser beam measures. This map is built through SLAM techniques concatenating local PCs in a consistent spatial manner to form a global PC map. Hence, the localization issue could be addressed as a global/local PC pair scan matching search, local regarding the current PC obtained from the robot in the actual position. On a second model, the 2D or 3D space is represented as an occupancy grid map. Therefore, the whole environment is represented as occlusion or free space in contrast with sparse PC representation. On occupancy maps, laser measures can be simulated over the model.

Figure 3.1: Different environment maps studied. Left: Real data sparse PC Structured Map. Top Right: 2D Occupancy Grid. Bottom Right: Simulated 3D Occupancy Grid.

A bio-inspired optimization solution is presented for both two-dimensional and three-dimensional environments and different conceptual map representations. The location of the robot within a map is represented as a set of possible position and orientation combinations spread over the environment space. Iterative stochastic optimization evolves this set of candidate locations throughout the space by weighting each estimate by a fitness function. For 2D and 3D occupancy grid maps, this stochastic engine is an implementation of the Differential Evolution method. On sparse Point Cloud 3D maps, three bio-inspired or Evolutionary Meta-heuristics are implemented, DE, Particle Swarm Optimization, and Invasive Weed Optimization. A new approach in contrast with commonly used Euclidean distance cost functions is presented, applying a set of probabilistic divergences using a different concept. The laser sensor information is modeled as a probability distribution over the measured distance. Statistical distances or divergences are now considered to measure the similarity between laser measurements, penalizing or favoring different situations for a more robust localization outcome.

3.1 The Localization Principle

The classic localization hypothesis depicts a robot keeping track of its movement through a known environment using odometry from its onboard sensor (encoder). The inherent uncertainty of this odometry causes confusion about the actual position of the robot in each instant, growing exponentially the more the movement continues. Therefore, the system must periodically confirm its current position within the environment or map, avoiding the uncertainty of increasing unbounded. Typical exteroceptive sensor systems may include vision, laser range finders, or ultrasonic sensors. The sensor information is then combined with the odometry to correct both position and uncertainty. In conclusion, an accurate position can not be measured directly. The localization system using that information will estimate the best candidate. These beliefs about the robot's location are typically represented as probability density functions [6]. An intuitive example of the correction of the perception phase is shown in Figure3.2. The prediction obtained by the odometry is corrected through external info (laser beam), and therefore uncertainty shrinks on the probability density function (dotted to continuous line in Figure3.2 (b)).

Therefore classic localization approach is divided into prediction update and correction update, and the available information on any instant of time t is:

$$Y_t = \{z_{0:t}, u_{0:t}\} = \{z_0, u_0, z_1, u_1, ..., z_{t-1}, u_{t-1}\}.$$

where $z_{0:t}$ and $u_{0:t}$ contain the external and odometry information respectively and so the probability density function for any instant could be $p(x_t|Y_t, m)$ where m represents the known map of the environment. It is assumed that in this stochastic process, future states do not depend on the past state but only on the present state, and so the recursive determination of future probabilistic density functions could be computed from a Bayesian point of view, assuming that the observation z_t is conditionally independent from previous measurements.

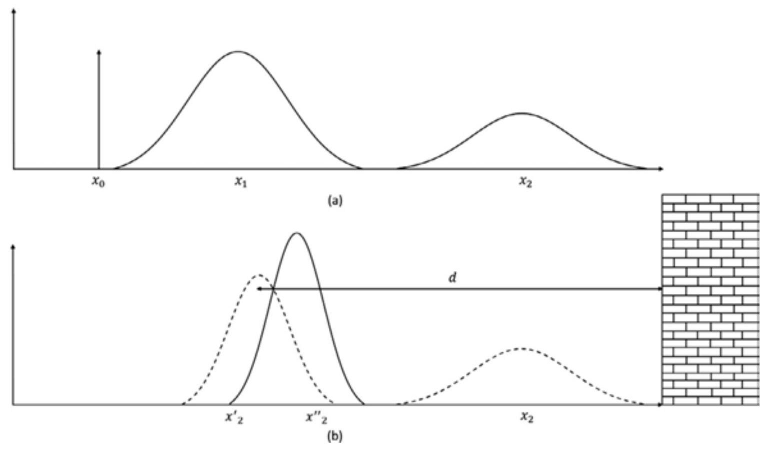

Figure 3.2: Localization motion prediction and perception update (a) Prediction (b) Perception [6].

$$p(x_t|Y_t) = \int_{\mathbb{R}_n} p(z_t|x_t)p(x_t|Y_{t-1})dx_t. \tag{3.1}$$

$$p(x_{t+1}|Y_t) = \int_{\mathbb{R}_n} p(x_{t+1}|x_t, u_t)p(x_t|Y_t)dx_t. \tag{3.2}$$

Equations 3.1 and 3.2 represent the measurement update and prediction respectively providing the solution to the bayesian recursive estimation problem. The multidimensional integrals of these equations can not be solved, have no analytical solution when the probability models are neither linear nor Gaussian. In equation 3.1 $p(x_{t+1}|x_t, u_t)$ represents the probabilistic motion model or a probabilistic generalization of the kinematics, and its computed from the state space model of the robot: $x_{t+1} = f(x_t; u_t) + v_t$. It represents the probability density of the following possible states x_{t+1} based on current state x_t and the input u_t. The term v_t takes into account the motion noise often modeled as a Gaussian distribution.

The second term $p(z_t|x_t)$ is considered the probabilistic observation model. It represents the posterior density over the possible sensor measurements, based on the uncertainty about the external information captured by exteroceptive sensors expressed as $zt = h(xt) + \epsilon_t$ where the term added ϵ_t expresses the noise sensor, typically represented as a normal distribution as well. With the assumption that the environment map is known, it is possible to compute the distance measurements z_t that would be observed from the estimated position x_t.

Based on the representation of the probability density function, different types of localization filters have been applied. As shown in equations 3.1 and 3.2, the Bayesian recursive filter entails the evaluation of integrals that are even feasible, as they are composed of known functions, require a high computational cost, and are based on the next iteration estimates that involve an associated error.

There are many variations that approximate de Bayesian approach by discretizing the continuous density function $p(x_t|Y_t)$ over a limited space.

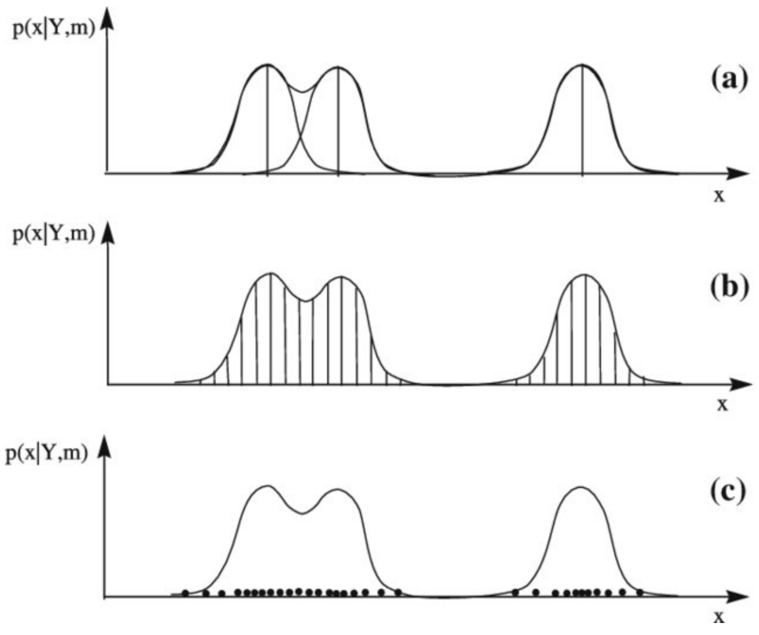

Figure 3.3: Probability density approximations: (a) mixture of Gaussians, (b) Piecewise approximation, (c) Monte Carlo approximation.

As it is represented in figure 3.3, those are Mixture of Gaussians, Piecewise approximation, and Monte Carlo approximation. Mixture of Gaussians approximates the density distribution into a sum of weighted Gaussian distributions replacing integrals by a finite sum, many examples of its utilization in localization can be found in the literature [1], [32], [31]. Piecewise approximation divides the space into cells. Each one of the cells in that grid is assigned an associated probability, and again the integral is approximated to a finite sum. Piecewise is successfully used for localization in [13], [2]. Finally, the Monte Carlo approximation is a particle-based method. The probability density function is sampled accordingly to the probability distributions, and again the integral is approximated into a sum of weighted samples [117], [15], [16].

Although Bayesian-based methods explain the localization problem and are used to estimate a solution for it, they lack simplicity for non-linear or non-Gaussian situations. Each estimate parameter in $\mathbb{R}^n$ has an associated value on the density probability function. The density function is calculated in that space for all the conditions given by the robot and the sensors for a given time t, but this must be managed by a weighting function to determine the unique estimation $\hat{x_t}$. Hence, localization is an optimization problem that can be solved by calculating the Maximum A Posteriori estimator (MAP)

$$\hat{x}_t^{MAP} = argmax_x p(x_t|Y_t) = \prod_{i=1}^{t} p_e(z_i|x_i) \prod_{i=1}^{t} p_v(x_i|x_{i-1}, u_{i-1}) p(x_0) \tag{3.3}$$

where $p_e(z_i|x_i)$ and $p_v(x_i|x_{i-1}, u_{i-1})$ represent the probability density function for the observation and motion respectively including noise e of perception z_i and noise v for odometry u_{i-1}. Equation 3.3 can be expressed in a recursive manner as:

$$\begin{aligned} f_0(x_t) = \sum_{i=1}^{t} logp(z_i|x_i) + \sum_{i=1}^{t} logp_v(x_i|x_{i-1}, u_{i-1}) + logp(x_0) \\ = logp_e(z_t|x_t) + logp_v(x_t|x_{t-1}, u_{t-1}) + f_0(t-1), \end{aligned} \tag{3.4}$$

and assuming that there is an optimal solution at time $t-1$ the location estimation is solved by calculating:

$$\hat{x}_t = max_x(logp_e(z_t|x_t) + logp_v(x_t|x_{t-1}, u_{t-1})),$$

A recursive estimation to optimize equation 3.3 improves Bayesian methods in terms of computational costs and implementation facility apart from being less dependent on statistical hypothesis. In this work, we are considering laser range finder sensors both from 2D and 3D measuring of the environment, and there is no motion information or odometry. Equation 3.3 can be simplified into:

$$\hat{x}_t = max_x(logp_e(z_t|x_t),$$

where the estimation depends entirely on the probability of observing z_t from x_t. Assuming that laser sensor measurement's error can be modeled as a Gaussian probability distribution of zero mean and variance σ, the combined probability of all sensor beams forming a laser sensor could be denoted by:

$$p(z_t|\hat{x}_t) = \prod_{i=1}^{N_s} p(z_{t,i}|\hat{x}_t) =) = \prod_{i=1}^{N_s} \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-1/2 \frac{(z_{t,i} - \hat{z_{t,i}})^2}{\sigma^2}}, \tag{3.5}$$

where N_s es the number of beams employed by the sensor, $z_{t,i}$ is the actual measurement of beam i in instant t and $\hat{z_{t,i}}$ is the same beam measure expected if the robot was in position $x_{t,i}$. Introducing this expression in equation 3.4 and eliminating constant terms we obtain:

$$f_0(x_t) = -\sum_{i=1}^{N_s} \frac{(z_{t,i} - \hat{z_{t,i}})^2}{2\sigma^2}$$

and therefore the resulting cost function is the minimization of $f_0^{'}(x_t) = -f_0(x_t)$, the L2-norm:

$$f_{L2}(x_i^t) = \sum_{i=1}^{N_s} \frac{(z_{t,i} - \hat{z_{t,i}})^2}{2\sigma^2} \tag{3.6}$$

where $z_t = (z_{t,1}, ..., z_{t,N_s})$ is the observation or distance vector provided by the 3D laser scanner in t, $\hat{z}_{t,1} = (\hat{z}_{t,1}, ..., \hat{z}_{t,N_s})$ are the expected observations for that measurements if the robot was situated in $x_{t,i}$ and σ^2 is the observation error variance. N_s the number of laser beams in each scan.

A suitable cost function for Global Localization based on observation estimation only, assuming Gaussian noise distributions for the laser beam sensor, is the RMSE or L2-norm between actually observed measures and expected measures if the position of the robot was our considered estimate. One of the main contributions of this work is to implement a different approach to this idea, improving the performance in some situations where the L2-norm does not cope due to the constrained situation of expecting normally distributed perturbations on the laser, not considering other types of noise or unmodeled obstacles.

3.2 Environment Models

Once the localization problem is formulated and a feasible solution is presented, it is necessary to comment on the different conceptual representations considered of the environment. As stated in our proposed solution, localization is approached as the estimation of position and orientation from a motion-less point of view (in opposition to tracking) and assuming that a map of the environment is known. With this premise, the objective is to find an estimated position through the search space that minimizes a specific cost function. This is performed by comparing the actual perception of the robot from its real position against the estimated perception from each candidate. In this work, 2D and 3D map representations are considered. Two different representation concepts involve a different approach in each case: approximated representations (simulated or semi-simulated) and real data maps. Both are presented in this section.

3.2.1 Simulated Environments

Simulated and semi-simulated maps in this thesis are based on occupancy grid maps, both in 2D and 3D. It has been assumed that the map is represented by an occupancy grid map, which is one of the most typical approaches in GL. In this type of map, the 2D environment is discretized using cells with a fixed size. A value between zero and one is computed for each cell to represent the probability of being occupied. If each cell of the map is denoted by m_{ij}, a 2D full map can be defined by the following set:

$$m = \{m_{ij} : \quad 1 \leq i \leq n, \quad 1 \leq j \leq o\}, \tag{3.7}$$

where i and j are positive integers that take the map limits into account (n and o). The occupation probability of each cell is $p(m_{ij})$. This notation lets us define the map estimation problem as the computation of the individual probabilities of each cell. Equation 3.7 is easily extended to 3D representation denoting m_{ijk} and $p(m_{ijk})$ as cell or voxel and its probability respectively, where k represents the third dimension.

Since the map is known and subdivided into cells with a determined occupation probability, this representation is helpful for laser modeling purposes. Information about the environment is given by laser sensor measurements, both planar for 2D and multi-plane full rotation model for 3D.

Figure 3.4: Semi-simulated model example and laser scan from estimated pose.

In this type of map, the sensor is a laser scanner that measures the cells crossed by each laser beam. These measurements are simulated both from actual and estimated positions, meaning that no real data is introduced. The laser sensor perception is simulated by computing the distance covered by each laser beam to the first occupied cell in that direction. Hence, each obstacle position is measured in cells. Our algorithms have been tested over blueprint representations and real-data-based occupancy grids for 2D environments. For 3D maps, a projection of these two-dimensional maps over the z axis has been carried out. Maps considered may represent real or testing environments, but no real sensor data is used over occupancy maps. We will refer to them as simulated or semi-simulated situations. An example of a 3D laser sweep simulation over a 3D projected 2D real map is shown in Figure 3.4 to illustrate this concept. The advantage that occupancy maps bring, in contrast to Point Cloud-based representations, is their continuity. When considering an estimated position on the map, the expected perception from that pose can be simulated when the laser angular resolution is known, searching in a computationally feasible manner for the first obstacle for each laser beam.

Our sensor model for 2D simulated environments is formed by a vector of 61 readings separated 3° covering 180° Field Of View (FoV). For 3D LiDAR simulations, the sensor model covers a 360° rotation over the horizontal plane with a 5° resolution, FoV over pitch rotation encompasses 30° with a resolution of 5°. The 3D sensor model covers 7 planes of 72 measures each.

3.2.2 Real Environments

In contrast with simulated environments, where obstacles or occupied spaces are represented in a continuous way through occupancy maps, sparse real-data-based environments are also considered in this thesis. This information is obtained using a 3D LiDAR sensor. Different dispositions and angular resolutions available in the market induce a higher or lower detail level obtained from the environment but also higher or lower implicit computational cost. First implementations of lidar sensing were typically approached by a perpendicular disposition of two 2D sensors. In this work, a sensor with a 360° FoV in the horizontal plane and 90° on the vertical sensor has been utilized, with an angular resolution of 0.7°.

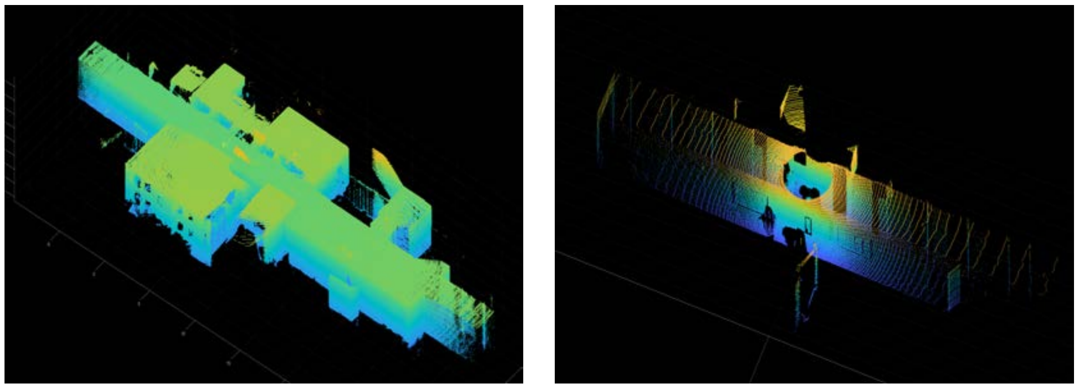

Figure 3.5: Point Cloud map example and laser scan from estimated pose.

The resulting measurements from this sensor come in the form of a Point Cloud, a structured set of three-dimensional points representing the traveled distance of each laser beam from the sensor to the nearest obstacle in that direction. This concept does not differ from the one presented in section 3.2.1 for simulated occupancy maps. The PC-based environment map is constructed through SLAM and Scan-Matching techniques introduced in Chapter 2, matching each reading from the LiDAR sensor to the previous one, obtaining the rigid transformation that optimizes the difference in terms of distance of significant points. As described in section 2.1.3, there are different methods to register point-cloud pairs. This registration could be local, matching two individual scans, or global, where each scan is registered with the accumulated global model. The advantage of local Scan Matching is that it can be corrected with loop detection mechanisms when, in a global case, if an incorrect match is introduced, it can not be corrected. Also, regarding the technique, Scan Matching is divided into probabilistic and deterministic methods, being the latter, through the ICP method, its maximal representation. Depending on the points considered, Scan Matching can be divided into point-based, feature-based, or mixed.

An example of PC maps utilized in this work is shown in Figure 3.5. This map was constructed by Evolutionary Optimization driven point-based local Scan Matching. Each scan has been matched individually with its predecessor, without the use of distinctive points or features. An iterative method based on the DE algorithm seeks the best-estimated transformation (translation+rotation) between scans to minimize the L2-norm, as considered in the ICP method [77]. In Figure 3.5 the resulting map and a single scan can be observed.

The Global Localization approach presented in this work for this type of known map could be considered as a global Scan Matching between the PC map and a single scan. This single scan represents the perception from the robot's current position, to be determined. Three different bio-inspired solutions, presented in section 2.2, are implemented for this task. Before describing its implementation, some pre-processing steps have to be presented in order to reduce computational costs.

Downsampling

A single scan coming from a high-resolution state-of-the-art LiDAR sensor is constituted by more than 260000 points. Evaluating this amount of comparisons through a cost function can be computationally highly expensive. In addition, the entire map is possibly formed by 500 or more single PC. This amount of data involved is interesting for modeling purposes. High-resolution sensors capture more details of the geometry of the environment, but in return, processing time increases, and using the original size of each scan is hardly inefficient computationally speaking. To deal with this issue, the amount of points involved has to be reduced. This process is commonly known as downsampling. There are three different concepts applied in downsampling methods:

- VoxelGrid Downsampling: The 3D space that contains the cloud is divided into equal-sized voxels or cubes. Each cube is reduced to the centroid value of points contained in its interior. This method returns a more uniformly distributed point cloud, reducing redundant points into one single point representing each voxel. The geometric characteristics of the cloud are maintained with a custom resolution.

- Non-Uniform VoxelGrid Downsampling: The concept applied is similar to VoxelGrid, in the sense that a set of points is reduced to its centroid value. In this case, the size of the grid is variable, depending on a maximum amount of points instead of a fixed size. Each group of user-specified n points is converted into one, determined by its centroid. This method is less computationally efficient but, unlike VoxelGrid, maintains the density distribution over the cloud.

- Random downsampling: The number of points is reduced in a percentage. Random points are selected within the cloud. This is yet simple in concept but more effective in point registration tasks. Although the spatial characteristics of the cloud are possibly affected by shape and feature recognition, selected points are real in contrast with previous methods.

As our task is to obtain an accurate point cloud registration between the actual local scan and global cloud map, Random Downsampling is the suitable choice, as it allows data reduction, thus maintaining real measures. An example the downsizing step is shown in Figure 3.6. The top figure shows the complete result, the colored cloud represents the global map including all points, the red dots indicate the local scan after a successful localization matching. The bottom cloud shows the same result using the downsized clouds.

3.3 Evolutionary Localization Filters

Proposed global localization solutions are based on bio-inspired and evolutionary Metaheuristics. These algorithms are probabilistic but do not involve derivatives or probability density functions in order to estimate an optimal solution. Three different methods have been implemented and are detailed in this section, but there are common ground characteristics among the three of them. Algorithm 1 describes an overview of the common scheme for the different methods. All three algorithms, Differential Evolution, Particle Swarm Optimization, and Invasive Weed Optimization, are population-based. A set of elements, candidates, or particles is allocated and evaluated through a fitness function to consider their qualification (lines $3-6$ in

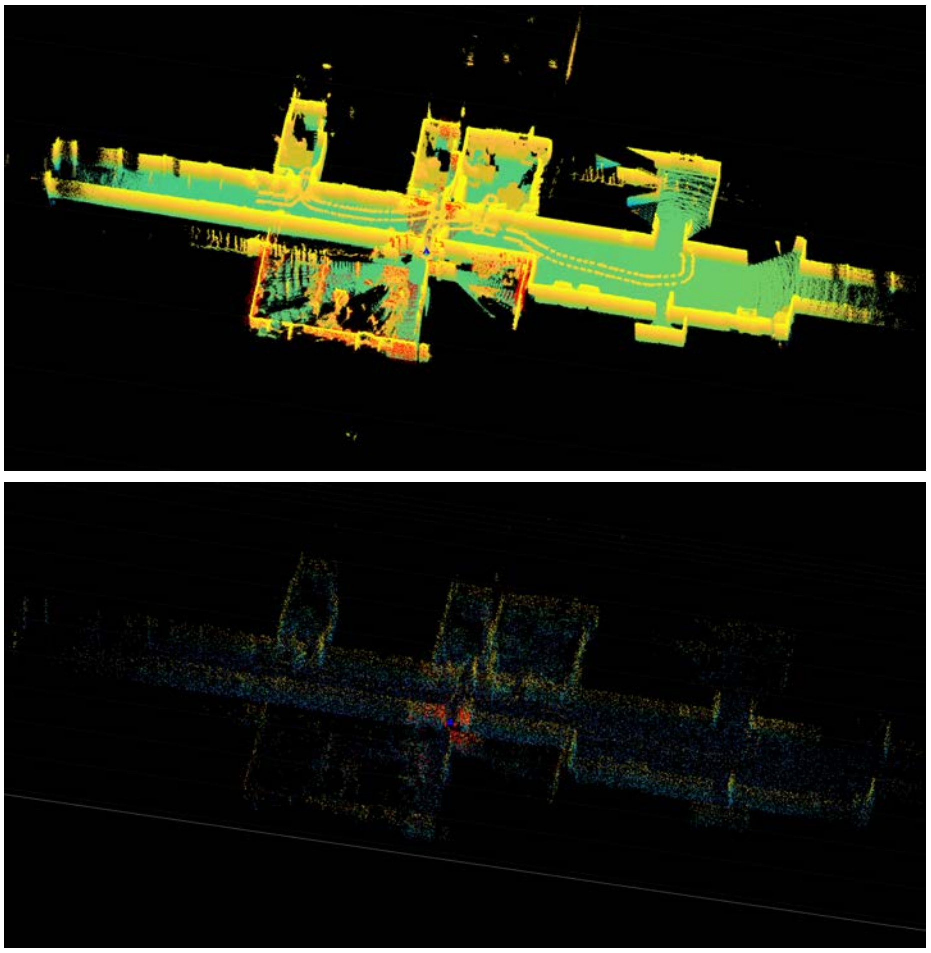

Figure 3.6: Example of localization success. Pose: [x,y,z]=[0.72,-0.04,-0.03]. Top: Original cloud. Bottom: Downsized cloud.

Alg. 1). This function evaluates the similarity between real and estimated perception data. All methods are population-based, where every member represents a possible pose of the robot. We will address 2D and 3D solutions, but explanations in this section will address a 6DOF consideration over a 3D domain. Therefore, each candidate is represented as a 6-parameter vector $\hat{p}_i = (x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i)$ where i is an index $\in [1, N_p]$ indicating each candidate on population N_p. Vector $\hat{z}_i = (z_1, ..., z_{N_s})$ is defined as the perception vector from the estimation, where N_s is the number of laser beams involved. This perception information is not known but only estimated. Just real data from the actual location of the robot is an input to the system. This induces the main difference between simulated and real environments and is reflected inside the $PERCEPTION\_EST$ step in lines 4 and 10 of Algorithm 1. Depending on the type of map, vector $\hat{p}_i$ is obtained in two different ways:

Algorithm 1 Metaheuristic Global Localization.

```
 1: function $Ev\_6DOF\_GL(global\_map, real\_scan)$
 2:     $INITIALIZATION \leftarrow (params, pop)$
 3:     for $i = 1 : Npop$ do
 4:         $est\_dists(i) \leftarrow PERCEPTION\_EST(map\_type)$
 5:         $cost(i) \leftarrow EVALUATION(fitness\_function, est\_dists(i))$
 6:     end for
 7:     while (CONVERGENCE CONDITIONS) do
 8:         for $i = 1 : Npop$ do
 9:             $new\_pop \leftarrow EVOLVE(alg\_params)$
10:             $est\_dists(i) \leftarrow PERCEPTION\_EST(map\_type)$
11:             $cost(i) \leftarrow EVALUATION(fitness\_function, est\_dists(i))$
12:         end for
13:         $SELECTION$
14:         $[solution, error] \leftarrow min(cost)$
15:         $convergence\_checking(...)$
16:     end while
17: end function                                  ▷ return solution, error
```

- Laser measurements from each candidate in occupancy grid maps can be simulated from each pose as explained in section 3.4, calculating the number of cells crossed by each laser beam until reaching the first obstacle. LiDAR model simulation from real and estimates from 1 to N_p are compared through a cost function.

- On the other side, for real PC maps, this is not applicable, as it is a sparse model with a much higher amount of points involved. Laser perception modeling from each candidate, unlike in occupancy grid maps, is computationally non-efficient. Local scans, even after downsampled, are formed by thousands of points. Calculating the first obstacle (in this case represented by a point of the global map) in each direction of each point of the real scan is computationally very expensive, even more, due to the sparsity of the global map. There is no certainty that a projection of the vector pointing at each point of the local scan from a candidate location of the robot will coincide with a point of the global cloud. A recursive search over that line, within a distance threshold from each step of that beam direction, would be necessary but unmanageable in terms of computational time for thousands of points and N_p candidates. Another approach is needed. This concept is illustrated in Figure 3.7. For this type of map, the perception from the estimation will consist of applying a

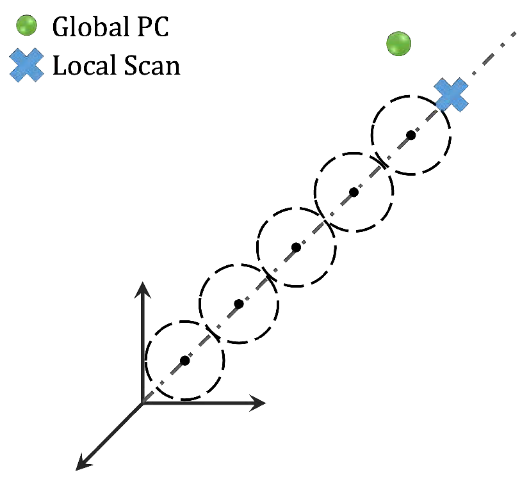

Figure 3.7: Recursive search over scan point direction.

transformation to the real observed point cloud to place it in reference to each candidate's position and, therefore, evaluating its matching with the global map cloud. Hence, a local/global iterative Scan-Matching method where each estimated transformation applied to the actual laser scan represents a candidate localization of the robot in the three-dimensional space. Once each candidate vector $\hat{p}_i = (x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i)$ is spread over the 3D space and the real local scan from the true position is transformed into those coordinates, coincidence with the global map is evaluated by the cost function.

Once the initial population is distributed on the 3D space and evaluated, the members of that population are modified or evolve into new candidates and evaluated (Lines 9 and 12 of Alg. 2.2) These steps occur in a different manner depending on the Metaheuristic and are developed in the next sections. $EVALUATION$ step is common for all bio-inspired methods. It is only defined by the selected fitness function and the method to obtain a comparison of scans depending on the map type. As commented previously in this section, perception from each estimated candidate in occupancy grid maps is obtained through laser scan modeling, while in PC maps, we consider a point registration process between the true scan transformed into the candidate's pose and the global map. In both environments, we consider a point-to-point evaluation to quantify the similarity between scans. In occupancy maps, that evaluation is intuitive as modeled laser scans from true and estimated positions contain the same amount of points and are sorted, meaning that each position of the vector containing each measure distance represents the same beam from the laser sensor (angle in the horizontal and vertical resolution of the LiDAR sensor). There is a spatial correlation between laser beams from real and candidate positions; hence they can be evaluated in a sorted way.

When considering sparse PC-based environment representations, the evaluation occurs by comparing the true scan with the global map when that scan is translated and rotated into each candidate's location. Imagine this concept as searching where a piece of the puzzle best fits by recursively placing it in different possibilities while mounting the whole puzzle. A point-to-point evaluation to quantify the similarity has to be approached in a different manner. For each point contained in the estimated scan, a counterpart in the global map is allocated. This pair of source and target points are obtained by implementing the Nearest Neighbor Search (NN-search). $kNN-$search algorithms are recursive search algorithms that find the k nearest neighbors regarding a specific distance. In our case, Euclidean distance is considered. The most common implementation of nn-search takes advantage of a structured tree-based division of data. For our task, a NN search powered by a 3d-tree is utilized. A 3d-tree structure is a 3D space partitioning data structure that recursively bisects the longest dimension of the 3D search space considered, forming boxes that are related to each other by a tree structure representing the *parent* and *child* for each bisection. Given the coordinates of a 3D point, this tree structure accelerates the search by descending through the tree structure, discarding a big part of the volume in each decision.

In our implementation, once the real PC scan is transformed into each candidate's location, the global cloud is reduced to the volume of the candidate scan to reduce the computational cost, and for every point of the local cloud, the nearest neighbor is considered in the global map. Each pair of points is compared in the cost function.

Once the population candidates have been modified and evaluated, the *SELECTION* steps (line 13) decides the members that remain in the next iteration.

3.3.1 DE-Based Global Localization

The localization algorithm described in this section, DE, is part of the formerly mentioned stochastic optimization-based methods. More specifically, it is included among the EA's, which somehow try to imitate the evolution model of the species proposed by Charles Darwin, following a series of nature-related concepts such as mutation, selection, and reproduction.

This section presents an implementation of DE for a GL-filter for 6DOF over a 3D known map. As commented, DE is a population-based algorithm. The population set contains candidates to be the correct location of the robot. These candidates are weighted by a cost value that compares sensor information from the true pose to sensor estimates from the candidate using the known map. The engine of the filter is the DE method that was first developed by Storn and Price [[7] and has been applied to solve optimization problems in multiple fields. The algorithm has been presented in 2.2.1

and its implementation is detailed in this section. The method described is presented for 3D environments for 6DOF global localization over a known map. Although experiments have been carried out over 2D environments with 3DOF, explanations can be easily reduced to that case. Algorithm 2 presents a pseudocode scheme of the implemented filter.

The objective is to estimate a 6 coordinate vector $\hat{p_i} = (x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i)$ that correspond to the robot's pose in a known map in position (x_i, y_i, z_i) and orientation over Euler angles $(\alpha_i, \beta_i, \gamma_i)$. Before the iterative procedure, an initialization process encompasses various steps (line 2-8 in Alg. 2). First, a pre-processing step is required to input the sensor scan from the real location (line 2 in Alg. 2). This step differs when considering occupancy maps or PC maps. In occupancy maps, the real pose is introduced by the user, and sensor measurements are simulated in the known map. This concept was explained in section 3.2.1 and an example of the resulting laser simulation is depicted in Figure 3.4. In this implementation, our sensor model is formed by a vector of 61 readings in the horizontal plane (x, y) separated 3° covering 180° Field Of View (FoV) for 2D simulated environments. For 3D LiDAR simulations, the sensor model covers a 360° rotation over the horizontal plane with a 5° resolution, FoV over pitch rotation encompasses 30° with a resolution of 5°, covering 7 planes of 72 measures each. This configuration can be modified to increase the detail level of the map information or reduced for more computational efficiency. Our resolution selection is a compromise between speed and performance on the maps tested. In PC-based environments, a real laser scan is directly obtained by the LiDAR and introduced into the algorithm through a previous downsampling method to reduce the number of points. The downsampling method was set to a random selection of 10% points of the cloud. Again, lower downsampling would increase the detail of the environment sensed but, in contrast, elevate computational times. Through any of these two methods, depending on the type of map, the *real_scan* input of the system is introduced.

DE is a population based bio-inspired stochastic algorithm so the method starts by selecting a configuration of N_p candidate locations through the environment (line 3 in Alg. 2). This population contains candidates to be the correct location of the robot $\hat{p_i} = (x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i)$. Depending on the optimization task, this population can be distributed based on *a priori* knowledge, in this case no previous location or estimation is considered so these candidates are randomly spread over the map volume following a uniform distribution.

On the next step of initialization, the different parameters that define the algorithm are selected. The scale factor F defines the mutation capability of the population, CR is the crossover rate, the capability of the population to acquire ADN information (parameters of the location) from mutated candidates. Both parameters influence the exploratory nature of the algorithm. After these steps, the algorithm

Algorithm 2 DE-based 3D Global Localization.

```
1: function $DE\_6DOF\_GL(global\_map, real\_position)$
2:     $real\_scan \leftarrow preprocessing(real\_position)$          ▷ perception from real location
3:     $pop \leftarrow randompop(N_P)$
4:     $parameters \leftarrow (F, CR, NP, max\_iter)$
5:     for $i = 1 : N_P$ do
6:         $est\_scan(x_i) \leftarrow PERCEPTION\_EST(global\_map)$
7:         $cost(x_i) \leftarrow fitness\_function(real\_scan, est\_scan)$          ▷ cost initial pop
8:     end for
9:     while (not CONVERGENCE) do
10:        for $i = 1 : N_P$ do
11:            $MUTATION(F)$          ▷ $v_i = x_1 + F(x_2 - x_3)$
12:            $CROSSOVER(CR)$          ▷ return $u_i$,genetic combination of $x_i$ and $v_i$
13:            $est\_scan(u_i) \leftarrow PERCEPTION\_EST(global\_map)$
14:            $cost(u_i) \leftarrow fitness\_function(real\_scan, est\_scan)$
15:                    ▷ cost function value calculation for next generation
16:            $SELECTION$ with $THRESHOLDING$
17:        end for
18:        $DISCARDING$          ▷ removing worse candidates
19:        $[solution, error] \leftarrow min(cost)$
20:        $convergence\_checking(...)$
21:    end while
22: end function          ▷ return solution, error
```

loop commences, and the candidate's laser perception is obtained and evaluated by a fitness function. Again, different approaches are considered depending on the map type. For occupancy grids, the laser from each member of the population is simulated, considering its position and orientation on the map, using the same method previously described. For real maps, each candidate involves a transformation of the *real_scan*. After this transformation, each corresponding point on the global map is computed through a NN-search as mentioned in section 3.2.2. Hence, for each candidate, a laser perception of the global map from that location is obtained. The sizes of the two vectors $z_i = (z_{i,1}, ..., z_{i,N_s})$ and $\hat{z}_t = (\hat{z}_{t,1}, ..., \hat{z}_{t,N_s})$ is equal to the amount of beams considered by modeling or PC downsampling and a point-to-point fitness function can be applied. As mentioned in section 3.1, on static consideration and assuming normally distributed noise, a suitable function is the L2-norm. Alternatives to this function are presented in section 3.4. Each candidate vector from the initial population x_i is awarded a cost value based on the fitness value that compares the *real_scan* and *est_scan*.

Once the parameters and initial population are initialized, the iterative process of the stochastic search commences in line 9. This loop continues until one convergence condition is satisfied. Convergence conditions applied are common for all implementations and are described later in this chapter. In each iteration and for the whole population set (lines $10-17$), the DE-filter creates the candidates to replace the current population members. Two evolutionary operators are applied to generate the new candidates. First, the mutation combines members of the population to form mutated vectors. After that, the diversity is increased by the crossover stage. New candidates are created by mixing (combining parameters) of current population members in each iteration and mutated ones.

The first step inside this loop is to generate new location vectors by adding a weighted difference between two random members of the population to a third one and so, creating a mutated equivalent for each member of the old population. This way an intermediate population is created such as:

$$mut_i^k = pop_a^k + F(pop_b^k - pop_c^k), \tag{3.8}$$

with random different indexes a,b and $c \in [1, N_p]$ indicating three selected population members in generation or iteration k and parameter $F > 0$, a scale factor that controls the intensity of the *mutation* (the evolution rate) by the amplification of differential variation. Notice that non of the elements pop_a, pop_b, pop_c coincide with element pop_i, the candidate vector considered in the loop. These three candidates are selected randomly among the rest of the population, ensuring that $a \neq b \neq c$ for population diversity. A mutated vector is generated by perturbing element pop_a with the scaled difference of $pop_b - pop_c$ generating a mutation of the robot's pose for each original member of the population. Mutation can be generated from a random candidate or the best candidate in each iteration. The first shows better general performance increasing exploratory qualities while mutation from best candidate implies faster convergence for local or previously biased searches as position-racking. Mutation Factor F must be a positive value, values of $F > 1$ have shown no better performance that optimum values between $0-1$ for same tasks. $F = 1$ would decrease population diversity as $pop_a^k + (pop_b^k - pop_c^k) = pop_b^k + (pop_a^k - pop_c^k)$. Our selected value is a variable F decreasing from 0.9 to 0.3 in two steps depending on the proximity to convergence conditions, explained in section 3.3.4.

In order to increase the diversity of the perturbed vectors, the crossover concept is introduced. Denoted by tr_i^k, new intermediate population members are created from the combination of parameters of the position and orientation from each original population member pop_i^k of the old generation with other parameters from the previously mentioned mutated candidates mut_i^k. To this point, the trial vector $tr_i^k = (tr_{i,1}^k, tr_{i,2}^k, ..., tr_{i,d}^k)$ is created with its parameters being:

$$tr_{i,j}^{k} = \begin{cases} mut_{i,j}^{k}, & \text{if } p_{i,j}^{k} \leq CR \\ pop_{i,j}^{k}, & \text{otherwise.} \end{cases} \tag{3.9}$$

where $p_{i,j}^{k}$ is a randomly chosen number among the interval [0,1], typically over a normal distribution, for each parameter j of each candidate i. The dimensions of both vectors is 6, recombination forms another possible location of $tr_i^k = (x_t, y_t, z_t, \alpha_t, \beta_t, \gamma_t)$. The crossover stage is controlled by the CR factor, which marks the probability of assuming parameters from a mutated member of the population over the existing ones. This variable is set to 0.85 in our implementation to encourage recombination for a higher diversity, which is important in big search spaces.

In order to decide whether this new intermediate population member should replace or not its competitor in the next generation of candidates, each one of the trial vectors tr_i^k is compared with the target (original) member of the population pop_i^k through a cost function. If vector tr_i^k delivers a smaller cost function value than pop_i^k then it will replace it as a member in the next generation, otherwise the old candidate pop_i^k is retained. Again like in line 6 of Algorithm 2, a new perception over the global map is generated for each trial vector.

The selection step evaluates the new trial and the current one by a cost function and selects the fittest as a population member for the next generation. A thresholding mechanism is added to the selection process to reduce the eagerness of the algorithm. This mechanism introduces a limit or threshold to choosing new solutions. When the fitness values are compared, the new candidate is only accepted if the difference between costs is larger than a pre-specified threshold. The objective is to avoid the optimization caused by the sensor noise. This threshold is set as a percentage of the cost function value of the older candidate. It was empirically set to 0.98 based on an expected sensor noise of 2% as described in our previous work [27]. The selection step is represented by:

$$pop_i^{k+1} = \begin{cases} tr_{i,j}^{k}, & \text{if } f_c(tr_i^k) < f_c(pop_i^k) * \tau \\ pop_i^{k}, & \text{otherwise.} \end{cases} \tag{3.10}$$

where $f_c()$ represents the cost value, and $\tau = 0.98$ the thresholding value selected.

The thresholding mechanism can cause degradation of the convergence properties because fewer solutions are accepted in each iteration. In order to increase the speed, the discarding step is implemented after a new generation is selected (line 18). At the end of each iteration, the worst population members (according to their costs) are substituted by solutions that are close to the best candidates. After each iteration, the worst 5% of the population members are substituted by one of the members of the best 20%. When the algorithm converges, the best population member is chosen to be the estimate of the robot's pose on the known map.

3.3.2 PSO-Based Global Localization

A second stochastic solution has been implemented to solve the mobile robot GL problem. PSO, like DE, is part of the formerly mentioned nature-inspired methods. In this case, a paradigm of swarm-intelligence-based Meta-heuristics was introduced in 1995 by Eberhart *et al.* [8]. Like the previous method, PSO is population-based. As explained in section 2.2.2, a set of possible solutions or particles evolve, changing its position on each iteration. Each movement of each particle depends on a velocity determined by a weighted sum of its previous velocity (inertia), its historical best position in terms of cost, and the historical best position of the whole swarm. Again, no information about the robot's position is known, only the global map and the local scan. In this thesis, the PSO method is implemented for real environments based on sparse PC information. Therefore, the initialization process is equivalent, with a uniform distribution of random particles (candidates) and the same downsampling method for the real scan from the actual position. The pseudocode for this implementation of the PSO-GL method is presented in Algorithm 3, initialization covers lines $2-10$.

The objective remains the same, to estimate a 6 coordinate vector, represented as particle $\hat{p_i} = (x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i)$ that corresponds to an optimum estimation of the robot's pose in a known map in position (x_i, y_i, z_i) and orientation over Euler angles $(\alpha_i, \beta_i, \gamma_i)$. In PC maps, as explained, this particle represents a transformation (translation and rotation) over 6DOF, which optimizes the matching between local scan (robot scan) and global scan (map). On the next step of initialization, the different parameters that define the algorithm are selected. The velocity weights: w for the inertia, c_1 and c_2, for the personal and global best direction, and finally, the maximum module of the velocity vector, $VelMax$ (line 3). The main difference between the PSO initialization step and DE is that PSO maintains a particle memory and a global memory, where each candidate keeps track of its particular best position and the swarm's collective best position, respectively. Hence, after spreading these particles and evaluating their fitness function, a global best and a particle best are updated (lines 8 and 10).

Once the initialization finishes, the swarm is spread over the 3D map, the parameters that manage the algorithm are selected, and the iterative optimization process commences. This occurs in the main loop from lines 11-25 of Algorithm 3 until any of the convergence conditions are satisfied. These conditions are explained later in this section.

In each iteration, the PSO-filter moves each candidate particle to a different location in the search space. The next position of each particle is a result of its previous coordinates plus a velocity vector. The calculation of this vector is the key to PSO performance and follows the next equation:

Algorithm 3 PSO-based 3D Global Localization.

```
function $PSO\_6DOF\_GL(global\_map, real\_position)$
    $real\_scan \leftarrow dowsampling$                          ▷ perception from real location
    $parameters \leftarrow (w, w_{damp}, c_1, c_2, VelMax, N_P, max\_iter)$
    $particles \leftarrow randompop(N_P)$
    for $i = 1 : N_P$ do
        $est\_scan(i) \leftarrow PERCEPTION\_EST(real\_scan, global\_map)$
        $cost(i) \leftarrow fitness\_function(real\_scan, est\_scan)$          ▷ cost initial pop
        $best(i) \leftarrow update$
    end for
    $glbest \leftarrow update$
    while (not CONVERGENCE) do
        for $i = 1 : N_P$ do
            $velocity(i)$ with $threshold \leftarrow CALCULATE$
                                    ▷ $v_i = wv_i + c_1(best(i) - p_i) + c_2(glbest - p_i)$
            $MOVE$                                               ▷ $p_i = p_i + v_i$
            $est\_scan(i) \leftarrow PERCEPTION\_EST(real\_scan, global\_map)$
            $cost(i) \leftarrow fitness\_function(real\_scan, est\_scan)$
                                    ▷ cost function value calculation for next generation
            $best(i) \leftarrow update$
        end for
        $REDUCE\ INERTIA(w_{damp})$
        $glbest \leftarrow update$
        $[solution, error] \leftarrow glbest$
        $convergence\_checking(...)$
    end while
end function                                                 ▷ return solution, error
```

$$v_i(k+1) = [wv_i(k)] + [c_1 r_{d1}(p_i^{best} - p_i(k))] + [c_2 * r_{d2} * (p_{gbest} - p_i(k)];$$

Three different terms influence the velocity that will displace particles in each iteration. A first-term denominated *inertia* term, $wv_i(k)$, accounts for the current velocity of the particle in that same iteration. Inertia weight w defines how influential this current velocity is. On the other terms, the *historical* terms, the position of each particle's historical best, and the swarm's collective historical best are considered. Two different weights control this influence, c_1 and c_2 which are multiplied by a random number $\in [0, 1]$ to introduce variety by randomness on the optimization search. A maximum velocity vector $VelMax$ establishes a threshold for each component of the resulting velocity so that:

$$v_{i,j}(k+1) = \begin{cases} v_{i,j}(k+1), & \text{if } v_{i,j}(k+1) \leq VelMax(j) \\ VelMax(j), & \text{otherwise} \end{cases}$$

Conceptually, the inertia weight w controls the trade-off between global and local exploration of the particles for a given maximum velocity $VelMax$. This maximum also enhances global search, but it is controlled by the inertia/historical ratio. The swarm may converge faster with a higher $VelMax$ but to a possible local minimum depending on the inertia weight. In our implementation, the maximum velocity is set to a percentage of each dimension of the global map to favor exploratory nature on the larger dimensions of the map due to its asymmetries. $VelMax$ is set to 10% of each coordinate's limits in the map, both for position and orientation. Studies like the one presented in [118] suggest, through experimental adjustment of the inertia weight with regard to the maximum velocity values, that for high maximum velocities, the inertia weight can be lower in comparison with *historical* weights, where statistically a higher global weight c_2 in comparison with particular best weight c_1 shows better convergence results. In our implementation, based on expectancy of global search requirements $VelMax$ was set to a quite high proportional value while $c_1 = 1.25$ and $c_1 = 1.75$ in comparison with the initial value of inertia $w = 1$. An interesting characteristic has been introduced, a variable inertia weight that decreases in each iteration through a dumping factor, $w_{k+1} = w_k * 0.99$. For each iteration, this weight decreases on a 1% of its previous value. This is intended to favor the displacement of the set of candidates towards the global best once the population is expected to converge to a solution. A value of 0.99 implies a low damping factor, but as several iterations are expected, this could prevent our implementation from converging rapidly to a local minimum. Inertia is corrected after each iteration (line 21) once all the present particles of the swarm have been moved to a new location and evaluated (line 17).

Unlike DE or other evolutionary and nature-inspired Metaheuristics, PSO lacks a selection stage. All particles remain on duty until the optimal solution has been estimated. The update phase modifies the behavior of the swarm for each movement. After each move, every single particle updates its historical best (line 19), and the global best is updated for the whole swarm, altering the velocity calculation for the next movement if these values have improved.

3.3.3 IWO-Based Global Localization

A third and last Metaheuristic is implemented for the Global Localization problem solution in PC-based maps. The same approach is considered an optimal solution for the location of the robot in a 3D sparse map representation, including the full 6DOF configuration that defines position and orientation in this known map. IWO, presented in 2006 by Mehrabian *et al.* [9] is another population-based bio-inspired stochastic optimization algorithm. This Metaheuristic, as presented in section 2.2.3 attempts to emulate the reproductive and endurance qualities of weed plants. Each plant in the colony is awarded with a reproductive capability, determined by the number of seeds spread. These seeds are scattered within a distance from the original plant until a maximum number of plants is reached, following a normal distribution of variable variance. The value of this dispersion rate evolves with each iteration, decreasing exponentially for each generation of the colony. The pseudocode for this IWO-based GL filter implementation is shown in Algorithm 4. The parameters that manage the algorithm are introduced in line 3. S_{max} and S_{min} represent the maximum and the minimum number of seeds awarded for the fittest and weakest plant of each generation, σ_{ini} and σ_{fin} are the first and last values of variable variance that manages the possible spreading distance and n or exp is the exponential value by which this variance decreases through each iteration.

Again common initialization routines are present in this implementation. Preprocessing step in line 2 consists of a 10% random downsampling. Lines 6 and 16 refer to the obtainment of a homologous point on the global map for each point in the $real\_scan$ when it is rotated and translated to each estimate's location.

The algorithm starts with a uniform sampling of the map, spreading a set of plants in a random manner. Each plant $\hat{p}_i = (x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i)$ represent a possible solution for the location of the robot. IWO considers a variable population in the colony. N_{Pini} members constitute the first generation. This population increases each iteration until a maximum is reached. After the initial population is scattered throughout the volume of the map, each candidate's local scan is evaluated in a point-to-point matching with its homologous on the global map. The population is sorted by this cost value, and the best and worst locations are registered (lines 9 and 10).

The algorithm starts to iterate until convergence conditions are fulfilled. The main loop covers lines 11-25. Each plant in the colony is awarded a number of seeds following the next equation:

$$S_i^k = S_{min} + (S_{max} - S_{min})\frac{(c_{p_i^k} - c_{wrst})}{(c_{best} - c_{wrst})}, \tag{3.11}$$

The worst candidate location of the robot, with cost value c_{wrst}, receives the minimum number of seeds, the best c_{best} receives the maximum and the intermediate population $c_{p_i^k}$ follow a proportional distribution between both values S_{max} and S_{min} depending on the relative position in the colony regarding fitness. The values of minimum and maximum seed affect the global or local search tendencies of the algorithm. The bottleneck in terms of computational cost in these implementations is the NN-search process. Therefore a compromise solution between a high number of seeds and efficiency has been applied. The selected values are $S_{max} = 6$ and $S_{min} = 2$. Some implementations propose no reproductivity for the worst candidates but as the basin

Algorithm 4 IWO-based 3D Global Localization.

```
1: function $IWO\_6DOF\_GL(global\_map, real\_position)$
2:     $real\_points \leftarrow preprocessing(real\_position)$    ▷ perception from real location
3:     $parameters \leftarrow (S_{max}, S_{min}, \sigma_{ini}, \sigma_{fin}, exp, N_{Pini}, N_{Pmax}, max\_iter)$
4:     $colony\_ini \leftarrow randompop(N_Pini)$
5:     for $i = 1 : N_P$ do
6:         $est\_points(i) \leftarrow PERCEPTION\_EST(map\_type)$
7:         $cost(i) \leftarrow fitness\_function(real\_points, est\_points)$    ▷ cost initial pop
8:     end for
9:     $glbest \leftarrow update$
10:    $glworst \leftarrow update$
11:    while (not CONVERGENCE) do
12:        for $i = 1 : N_P$ do
13:            $S(i) \leftarrow reproductive\_seeds(S_{max}, S_{min}, \sigma_{ini}, glbest, glworst)$
14:            $RANDOM\ SPREAD\ SEEDS \leftarrow S(i)$
15:                ▷ dispersion distribution $\sigma = f(\sigma_{ini}, \sigma_{fin}, exp)$
16:            $est\_points(i) \leftarrow PERCEPTION\_EST(global\_map)$
17:            $cost(i) \leftarrow fitness\_function(real\_points, est\_points)$
18:                ▷ cost function value calculation for next generation
19:        end for
20:        $SELECTION$    ▷ removing worse candidates until $N_{Pmax}$
21:        $glbest \leftarrow update$
22:        $glworst \leftarrow update$
23:        $[solution, error] \leftarrow glbest$
24:        $convergence\_checking(...)$
25:    end while
26: end function    ▷ return solution, error
```

of attraction of the optimum solution is small, meaning that the optimum solution could be proximal to a highly expensive candidate, a relatively high number of seeds is awarded to the worst estimates.

For each candidate plant p_i in generation k, S_i^k seeds are scattered randomly within a distance from its parent plant, the new position is a random candidate location within the normal distribution around the parent value of each parameter so that:

$$p_{i,j}^k = rand(f(x)),$$

$$f(x) = \frac{1}{\sigma_{it}\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-p_i^k}{\sigma_{it}})^2},$$

where $j \in [1, S_i^k]$ represents an index for every scattered seed from plant p_i^k, and $f(x)$ is the normal distribution with 0 mean over the original position p_i^k with standard deviation σ_{it}. The standard deviation for this normal distribution is defined by a variable standard deviation, σ_{it}:

$$\sigma_{it} = \frac{(it_{max} - k)^n}{(it_{max})^n}(\sigma_{initial} - \sigma_{final}) + \sigma_{final}, \tag{3.12}$$

These parameters are less intuitive to adjust than previous optimization schemes. The standard deviation in each iteration controls how global the search over the 3D space may be. A high initial value of σ_{ini} combined with a low exponential n ensures wide global search on the first iterations. This may cause a non-convergence of the algorithm due to instability. These parameters were empirically tested and in our experience convergence ratio improves with a lower $\sigma_{ini} = 0.8$ combined with a proper sampling of the search space with the initial population. σ_{fin} and exponential n are set to 0.1 and 2, respectively, low values to more fine local search when the algorithm is expected to converge.

In the selection stage (line 20), when every new seed from every generation is evaluated by the fitness function, the selection mechanism is responsible for deciding which members of the colony remain. IWO proposes a collective selection. The fittest members among new seeds and previous plants endure. The maximum members of the colony are selected through parameter N_{Pmax}. Before this number of candidates is reached, no element is removed. The best and worst candidates are registered for the next iteration.

3.3.4 Convergence Conditions

All the presented implementations are based on an iterative process to stochastically optimize a fitness function that defines our problem. This iterative process continues until an optimum value of this function is obtained, but in the majority of situations, this value is unknown. The algorithm must rely on other sets of considerations. There are some situations where convergence can be easily noticed, for example, constrain-related optimization, where if all constraints are satisfied, the algorithm has converged. Multi-objective or multi-modal optimization convergence criteria are more difficult to define. GL is a multi-modal optimization problem as it can be described as a problem with multiple solutions, meaning that different locations may lead to very similar optimum values of the fitness function. In this type of problem, considering the cost value as a convergence criterion may be problematic, as it is not easy to quantify. Besides, several fitness functions will be presented in the next section, with different ranges of values. Even considering the L2-norm, where we could expect an optimum value of the cost function for the estimated solution close to 0, as low distances between map and scan are expected, the presence of noise and obstacles can lead to confusion if convergence criteria are based on cost function values.

Other options are population-based convergence conditions. Examples of these are differences between the best and worst candidates in each iteration. This can lead to premature convergence after initialization, with non-optimal but similar cost values in the population. Besides, information regarding the fitness value on optimal solutions is needed.

Another classic approach is iteration-based criteria or time limitations. A maximum number of iterations allowed is set or a maximum amount of time. These are suitable for unknown fitness function optimal values.

In these implementations, as the optimum fitness values can not be expected, and the use of several fitness functions is considered. A mix of iteration and population-based conditions has been implemented that do not ensure convergence but have experimentally proven to lead to good results. Three different convergence conditions can terminate the procedure:

- Number of iterations: A number of maximum iterations are reached. This criterion is totally empiric, and this maximum is set to an elevated number in order to let the algorithm converge through other methods.
- Invariance: A number of iterations where neither the best, the worst, nor the average cost function of the population varies.
- Total convergence: All population members possess the same cost. Experiments have shown that this criterion must be combined with a minimum of iterations and a suitable population number to avoid fast convergence.
- Normal convergence: For each iteration, the best, worst and average costs of the population are considered. If the worst/best ratio and the average/best ratio are below 105%, convergence is considered. This is suitable in large search spaces with large population numbers. Otherwise, it can lead to premature convergence.

3.4 Divergences: Implementation of Fitness Functions

The fitness function represents the evaluation mechanism for a stochastic optimization algorithm to differentiate between candidates and evolve to an optimum solution. Therefore, it represents a general concept that may vary in its implementation in as many ways as existing problems to solve. A representative fitness function is strictly related to the robustness of any Metaheuristic considered. In the task that concerns us, it is reduced to a suitable manner to distinguish between a set of distances provided by two laser scans. How accurately this difference is evaluated and how precisely the possible situations are represented in a mathematical function will have a great influence on how effective the GL procedure is. In robotics, this situation has been typically addressed through the L1-norm and the L2-norm because of their good behavior, mainly in terms of accuracy and speed. The main disadvantage when using these types of metrics is that they are symmetric, meaning they do not provide any information on how the sets differ but only provide quantity information.

As explained in section 3.1, a suitable cost function for global localization tasks is the RMSE or L2-norm. Equations 3.3 to 3.6 assume a recursive optimization method, where no information from the motion sensors is received. Only exteroceptive perception from a laser beam model is provided. If the sensor noise is modeled as a Gaussian distribution, the combined probability of all sensors (eq. 3.5) is introduced in the recursive version of the MAP optimization solution (eq. 3.3), induces the L2-loss equation (Eq.3.6). Although expecting a normally distributed sensor noise is assumable, the idea is restricted to a limited perspective of the variety of noise distributions and perturbations that may affect the laser observation. Besides, RMSE has proven to obtain non-satisfactory results when the normal distribution assumption diverges from reality.

A different approach is followed in this work, considering both sets of laser measurements as probability distributions. Therefore the distance or difference between probability distributions can be evaluated through a series of statistical metrics or divergences. In this work, four different probability-based cost functions have been implemented based on the following divergences: Kullback-Leibler (KL) [119], Jensen-Shannon (JS) [120], Density Power (DP) [121], and Itakura-Saito (IS) [122]. The objective is to test the performance of the GL module with this type of cost function in contrast with Euclidean-based metrics, checking the capabilities of the GL-filters (section 3.3) are maintained and evaluating the specific advantages of this approach. According to the state of the art, the most common strategy when developing a cost function for this type of filter is to choose an asymmetric metric. The KL divergence, for instance, is an asymmetric distance included in a wider class of metrics known as Csiszár- Morimoto $f-$divergences. The main advantage of asymmetry is that it allows to penalize or favor different situations depending on the error between real pose and estimated measured distance. A cost function with an asymmetric divergence can be designed to improve the performance in some unexpected situations. This feature has been exploited to increase the robustness of the localization method in the presence of occlusions. Utilizing divergence-based fitness functions requires that the data provided by the laser has to be modeled as a probability distribution. Laser measurements are no longer defined only by a specific distance but through a probability distribution that represents the whole range of the beam. Different types of distributions are merged into this beam model with different weights that are combined to model possible events. Cases with possible occlusions in the real perception are favored by the coefficient selection on the fitness function. Hence, the probability profiles and the divergence-based cost functions allow for improving the evolutionary-based localization performance when the robot is in an environment with significant occlusions. The asymmetry of the distance is exploited to increase robustness. In this section, the perception model that makes it possible to improve localization in environments with occlusions is described. Then, all the different divergence-based fitness functions are detailed.

3.4.1 Laser Beam Probabilistic Model

As stated previously, the different divergences applied are dependent on the probability distribution profiles. These profiles are selected based on the specific problem to solve and the chosen method of perception based on a laser range sensor. Each laser measurement is constituted by N_s returned distances, depending on the resolution required when modeling or the downsampling applied to the PC scan. Each component of the beam vector $(z_{t,i})$ can be modeled as a probability distribution depending on the distance obtained for each unidirectional beam. This model is often named *raytracing* or *raycasting* as it can be observed as the evolution of the surface explored by each laser beam over the environment to calculate the expected quantity. Therefore, in occupancy grid models, each cell's probability can be modified as the laser sensor reveals its path traveling free space cells until the collision.

Different authors have proposed a Bayesian model to approximate the behavior of the range sensor beam model for dynamic environments [123]. These probabilistic models provides us $p(z_t|x_t, m)$ for a LiDAR sensor, which denotes the probability of obtaining the perception z_t on environmental representation m if the actual position of the robot is x_t. Several considerations have to be evaluated, as the realistic approach must consider inherent measurement noise introduced by this type of sensor. In addition, perturbations between the actual perception of the sensor and the known map occur and must be considered, aside from inaccurate pose estimation. Hence, a probabilistic model must differ from an ideal sensor model where the observation can be expressed by the equation that defines the theoretical observation from a certain position.

As a quick review on sensor measurement modeling, researches as [124], [2] present a discrete grid mapping in contrast with geometric continuous approaches [125], [126], [127]. Other groups implement a non-normal probabilistic density sonar distance over a discrete grid map. In these approaches, the probability must be computed for all possible locations of the robot, a computationally inefficient task to accomplish in real-time implementations. A more time-efficient perspective is presented in [2] by considering only the distance to the first obstacle for each single laser beam. The model represents different types of scenarios. The laser beam detects modeled obstacles present in the map representation or an unmodeled obstacle. This approach was extended [127] by Thrun *et al.* into considering the presence of people in the surroundings of the robot and a threshold distance measurement representing the maximum range.

In the model presented in this work two different sources of distance measurements are considered: modeled or unmodeled obstacles. The laser may indicate the distance from the possible position in a particular direction to an obstacle present in th known map or an unpredicted presence. The evolutionary-based stochastic search engine evaluates the similarity of the estimate's laser vector and the actual perception vector, uncertain positions behind the obstacle detected must be considered in the model. In that way, the one-dimensional probability distribution for each pair of beams cover the same distance in case the first obstacle detected for each measure compared is at a different distance. The information for the space behind the obstacle is absent, and the model should represent this situation. With all these assumptions the probability distribution for a single beam can be described as:

$$p(z_{t,k}|x_t, m) = k_h p_{hit}(z_{t,k}|x_t, m) + k_o p_{occl}(z_{t,k}|x_t, m) + k_u p_{unkn}(z_{t,k}|x_t, m), \quad (3.13)$$

where p_{hit}, p_{occl}, and p_{unkn} are the probability densities that represent known or modeled obstacles, possible occlusions (unexpected obstacles), and uncertain places respectively, and may reflect a value between $\in [0, 1]$.

The coefficients k_h, k_o and k_u are constant weights considered to favor or penalize certain situations on the comparison.

A Gaussian distribution is considered to model the probability around the distance measure obtained from the sensor, integrating measurement noise, an assumable consideration as commented in previous section 3.1. this distribution can be expressed as:

$$p_{hit}(z_{t,k}|x_t, m) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z_{t,k}-z^*_{t,k})^2}{2\sigma^2}}, \quad (3.14)$$

where $z^*_{t,k}$, is the distance indicated by the laser sensor, σ^2 is the sensor noise variance.

Unexpected or unmodeled occlusions are modeled with a uniform distribution. These obstacles shorten the laser distance in regard to expected map occlusion. A clear example of this type of perturbation can include people in the view range of the laser or small mobile objects. For the posterior points, the unknown spaces after the laser measurement, a uniformly distributed model is considered, from the obstacle

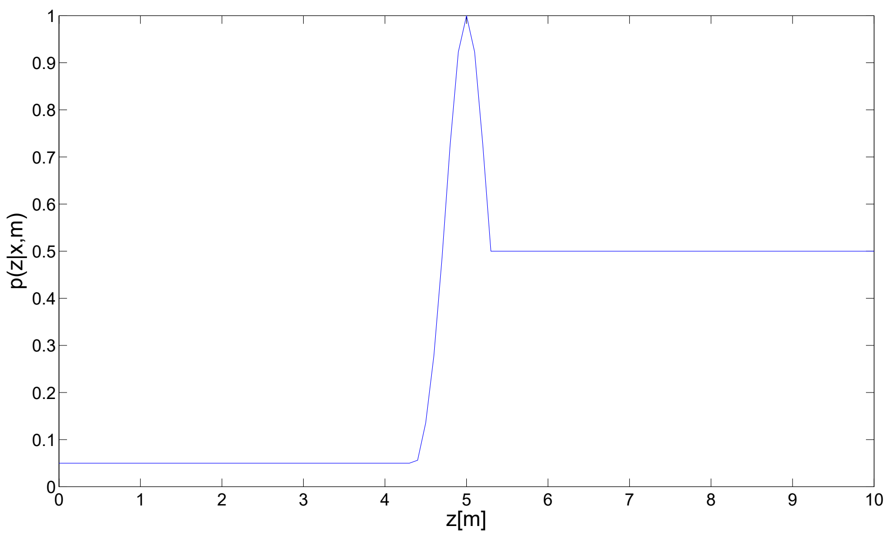

Figure 3.8: Probability distribution for a single laser beam.

to the maximum range of the sensor. Through the weight coefficients k_o and k_u, the influence of unexpected measurements and unknown spaces in the probability distribution is controlled, as the value of p_{occ} and p_{unkn} is one when an occlusion or unknown space is found. This concept and Equations 3.14 and 3.13 can be clarified by observing Figure 3.13 where the probability distribution that models a single laser beam is represented.

The distance reflected by the laser measurement is 5 m in this example, and p_{hit} is represented by a normal distribution. The mean of this Gaussian is equal to 5, while the variance depends on the expected sensor noise. The distance before the sensor measure is a uniform distribution with a very low value ($k_o = 0,05$). This is due to the fact that if the laser measurement is distributed around $5m$, the presence of an unmodeled obstacle is very unlikely on the ray casting principle. For the unknown spaces beyond the Gaussian distribution, where there is no information, the probability value chosen is $k_u = 0,5$, a value between occupied and free space.

The cost function compares the real measurement with the estimated one. The probability distribution for the estimated measurement is

$$p(\hat{z}_{t,k}|\hat{x}_t, m) = \hat{k}_h p_{hit}(\hat{z}_{t,k}|\hat{x}_t, m) + \hat{k}_o p_{occl}(\hat{z}_{t,k}|\hat{x}_t, m) + \hat{k}_u p_{unkn}(\hat{z}_{t,k}|\hat{x}_t, m), \qquad (3.15)$$

where all the same equivalent values are applied for the same possible situations.
These two probability distributions in Equations 3.13 and 3.15 are previously normalized to add up to one and then fed to the cost function for laser comparison. The

normalization factor is not included for simplicity in the equations. The different situations considered inside the cost function based on the values of real and estimated distributions are explained in the next section.

3.4.2 Robustness To Occlusions - The Coefficient Selection

Considering the optimum candidate location of the robot, an occlusion originates a difference when evaluating the actual and estimated scans. This error is induced by unmodeled obstacles, those not represented on the map. This can be observed in Figure 3.9. The real measurements (left) perceive that obstacle, but it is not considered in the laser scan from the optimum estimate (right) as it is not modeled in the known map. Therefore, what could be the optimum estimate, the accurate position of the robot, presents an occlusion error.

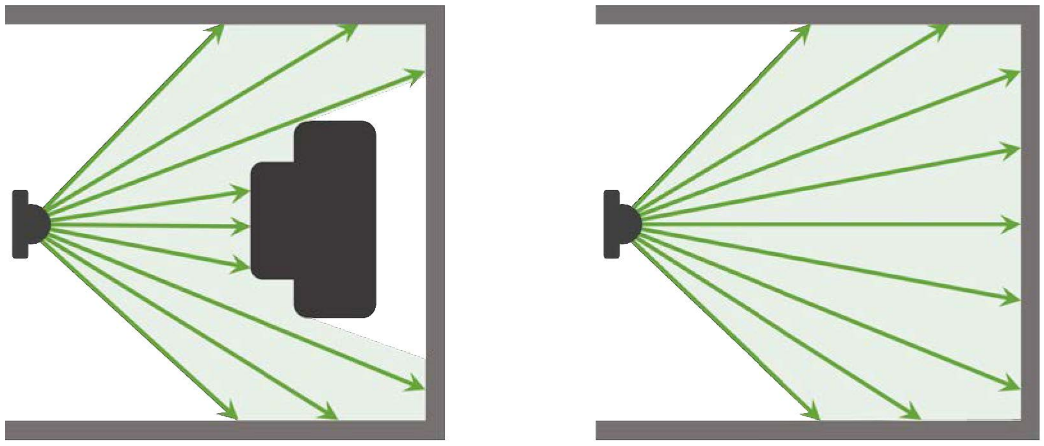

Figure 3.9: Laser scan example of occlusion. Left: possible occlusion, unmodeled obstacle. Right: laser scan without unmodeled obstacles. (Laser beams given by the estimate and not contained in the real measurements in dashed lines).

The divergence-based cost functions will compute the distance between two probability distributions (P_S and $P_{\hat{S}}$, real observation and estimated one) for each laser measurement of the scan. The probability distribution for each laser beam is defined according to the concepts explained in this Chapter. Equation 3.15 defines the probability distribution for each laser beam. The different weights present in this equation p_{hit},p_{occ} and p_{unkn}, are adjusted empirically to deal with possible occlusions. This is depicted in Table 3.1, where the different values for these coefficients in distinct situations are shown. This section presents different considerations and coefficient introduction to deal with the possibility. In summary, this implementation exploits the non-symmetry on the consideration of possible errors in the same laser beam.

Lower measurements by the real scan are considered as possible occlusions, while higher values are penalized due to their infeasibility.

Table 3.1: Values for the constants on Equations 3.8 and 3.15 depending on the relation between the real measurement $z_{t,k}$and the estimation $\hat{z}_{t,k}$.

	$z_{t,k} << \hat{z}_{t,k}$	$z_{t,k} \leq \hat{z}_{t,k}$	$z_{t,k} > \hat{z}_{t,k}$	$z_{t,k} >> \hat{z}_{t,k}$
k_o	0.1	0.1	0.1	0.95
$\hat{k}_o$	0.05	0.05	0.05	0.05
k_h	0.9	0.9	0.9	0.95
$\hat{k}_h$	0.95	0.95	0.95	0.05
k_u	0.15	0.5	0.9	0.95
$\hat{k}_u$	0.5	0.5	0.5	0.05

If we define $z_{t,k}$ as the laser measurement k from the actual position and $\hat{z}_{t,k}$ as the same measurement form the estimate location, the different possible situations are:

- $z_{t,k} << \hat{z}_{t,k}$: The measured real distance is considerably lower than the estimate. In this case, the cost function will not penalize this difference as a possible occlusion is considered: The probability profiles maintain a similar distribution to the one shown in Figure 3.8. The weight coefficient for this situation is represented in the first column of Table 3.8. The main modification to the reference probability distribution is the decrease of $k_u = 0.15$ not to penalize the unknown space behind the possible occlusion.

- $z_{t,k} \leq \hat{z}_{t,k}$: The real distance is equal to or slightly under the estimate. An occlusion is feasible, but the influence of sensor noise is also possible. By maintaining the unknown weight of the real probability, this difference is slightly penalized over the section behind possible obstacle distance. This factor has no impact when $z_{t,k} = z_{t,k}$.

- $z_{t,k} >> \hat{z}_{t,k}$: Considering the optimum estimation, this situation is impossible and therefore is strongly penalized. Coefficients selected for this situation are shown in the last column of table 3.8. All equivalent weights are set to a difference of $0.95 - 0.05$ to award the maximum cost. The estimate is, with a high probability, wrong and should be discarded by the GL-filter.

- $z_{t,k} > \hat{z}_{t,k}$. The estimated distance is slightly higher than the real one. Again in an ideal situation, this case will induce a wrong estimation, although it is not as heavily discarded as it could be neighboring the optimum. In addition, the possible presence of sensor noise may alter the error in this direction. A weight of $k_u = 0.9$ for the unknown space behind the real measure has proven to penalize fairly.

3.4.3 Divergence-Based Fitness Functions

Although there are multiple divergences that could be applied to implement the cost function of the GL strategy, four metrics have been chosen in this work. Each one will be explained in this section. The KL divergence is a non-symmetric statistical distance that measures how probability distribution P differs from Q or the *surprise* when referring to a situation with actual distribution P by using Q as a model. The JS divergence can be viewed as a symmetric version of the KL divergence. The DP divergence can be equivalent to a quadratic error version when using probability distributions. The IS distance is similar to the KL divergence when its formula is analyzed. These divergences belong to different families of metrics with interesting properties and connections between them. Different works have been published to establish formal relations between metrics in information theory [128], [129],[130]. Among them, Cichocki and Amari [128] have studied wide families of divergences named Alpha, Beta, and Gamma. The dissimilarity measures applied in this paper can be included in the first two categories. The researchers use power functions to generalize the KL divergence and to obtain different classes of divergences. They report that power functions allow to increase the robustness with respect to outliers and, therefore, the performance is better or more flexible (an example can be found in [131]). By using this approach, it is possible to define three families of divergences (Alpha, Beta, and Gamma) that can be viewed as generalizations of the KL divergence. All classes are linked, and it is possible to do transformations between them [132]. These families are derived from the well-known Csiszar–Morimoto (CM) $f-$divergence and the Bregman divergence. The CM $f-$divergences are obtained using the following equation:

$$d_{CM}(P||Q) = \sum_i q(i) f(\frac{p(i)}{q(i)}) \tag{3.16}$$

where f is the generator function, and p and q are the densities of two probability distributions P and Q. The Bregman divergences are given by:

$$d_{CM}(P||Q) = \sum_i [\Phi(p(i)) - \Phi(q(i)) - \frac{\delta\Phi}{\delta q(i))}(p(i) - q(i))], \tag{3.17}$$

where Φ is the generator function. The Alpha divergence [133] is a special case of CM $f-$divergence defined by the following formula:

$$d_A^{(\alpha)}(P||Q) = \frac{\sum_i [p^\alpha(i) q^{1-\alpha}(i) - \alpha p(i) + (\alpha - 1) q(i)]}{\alpha(\alpha - 1)}, \tag{3.18}$$

This metric depends on the variable parameter α. For example, when $\alpha \longrightarrow 1$, the generalized KL divergence is obtained $(\sum_i [p(i) ln(p(i)/q(i)) + p(i) - q(i)])$.

From the CM divergences, it is possible to establish some basic properties for the Alpha divergences: non-negativity ($d_A^{(\alpha)} \geq 0$ and $d_A^{(\alpha)} = 0$ if and only if $P = Q$), convexity with respect to both P and Q, continuity (continuous function of real variable α), duality ($d_A^{(\alpha)} = d_A^{(1-\alpha)}$), etc. The reader can consult [128] to find more properties and a more detailed explanation. The Beta divergence [121], [134] is obtained from the Bregman divergence:

$$d_B^{(\beta)}(P||Q) = \frac{\sum_i [p^\beta(i) + (\beta - 1) q^\beta(i) - \beta p(i) q^{\beta-1}]}{\beta(\beta - 1)}, \tag{3.19}$$

This divergence is dependent on β. The connection between the Bregman and the Beta divergences is strong, and the properties of the Bregman divergence are also valid for the Beta divergence: non-negativity ($d_B^{(\beta)} \geq 0$ and $d_B^{(\beta)} = 0$ if and only if $P = Q$), convexity with respect to P, linearity (a positive linear combination of Bregman divergences is also a Bregman divergence), invariance under affine transforms, etc. The Beta divergence has a single global minimum equal to zero for $P = Q$ and increases with the absolute value of the difference between p and q. The cost functions are detailed and connected to the families of divergences. Finally, an illustrative example of a single laser beam is shown.

Kullback-Leibler

The KL divergence, proposed by Kullback *et al.* in 1951 [119] can be defined as "a non-symmetric measure of the difference between two probability distributions P and Q. This concept describes the "a measure of the expected number of extra bits required to code samples from P when using a code based on Q, rather than using a code based on P" in information theory. In statistics, P is associated with the actual distribution (our real perception vector of distances), while Q is an approximation or a model of that probability distribution (our estimated candidate's perception). The definition in a continuous domain is:

$$d_{KL}(P||Q) = \int_{-\inf}^{+\inf} p(x) \ln \frac{p(x)}{q(x)} dx, \tag{3.20}$$

where p and q are the densities of P and Q. When applied to to discrete spaces like our approach:

$$d_{KL}(P||Q) = \sum_i p(i) \ln \frac{p(i)}{q(i)}. \tag{3.21}$$

The KL divergence exists for probability distributions in which the densities add up to one ($\sum_i p(i) = \sum_i q(i) = 1$). The individual components (each i) are included in the formula only if $q(i) > 0$ and $p(i) > 0$. The quantity $0 \ln 0$ is assumed to be zero. It has some interesting properties:

1. $d_{KL}(P||Q) \geqslant 0$.
2. $d_{KL}(P||Q) = 0 \Leftrightarrow P = Q$.
3. $d_{KL}(P||Q) \neq d_{KL}(Q||P)$.
4. If $P_1(x)$, $P_2(y)$, $Q_1(x)$ and $Q_2(y)$ are independent distributions, with joint distributions $P(x, y) = P_1(x)P_2(y)$ and $Q(x, y) = Q_1(x)Q_2(y)$, then $d_{KL}(P||Q) = d_{KL}(P_1||Q_1) + d_{KL}(P_2||Q_2)$.

Analyzing the third property, the KL divergence from P to Q does not coincide with the value calculated from Q to P. Hence; the KL divergence is normally referred to as a distance rather than a metric due to this is non-commutative property. This variable can also be interpreted as the average of the logarithmic difference between P and Q, where the average is weighted by P.

If $S(x, z) = \{m_{i,j}^{x,z}\}$ is the area (in cells) that is crossed by an observation z when the robot is located at x, the KL divergence for a given orientation (k) can be expressed by

$$d_{KL}^k(P_{S(x_t,z_{t,k})}||P_{\hat{S}(\hat{x}_t,\hat{z}_{t,k})}) = \sum_{i,j \in S_T} p_{S(x_t,z_{t,k})}(m_{ij}) \ln \frac{p_{S(x_t,z_{t,k})}(m_{ij})}{p_{\hat{S}(\hat{x}_t,\hat{z}_{t,k})}(m_{ij})}, \tag{3.22}$$

where S_T is the maximum between $S(x_t, z_{t,k})$ and $\hat{S}(\hat{x}_t, \hat{z}_{t,k})$. amount of accupancy cells traveled by the whole laser sweep $S(x_t, z_{t,k})$ differs from the surface in term of cells covered by a determined candidate $\hat{S}(\hat{x}_t, \hat{z}_{t,k})$. In our laser model, each free cell is added to the distance measured by each beam, until the first uncertain value is reached, just after the obstacle. The additive property of the KL divergence (property 4 on the list) is applicable because of the assumption that each cell's probabilty value is independent.

A simplification of Equation 3.22 could be denoted as:

$$d_{KL}^k(P_{Sk}||P_{\hat{S}k}) = \sum_{i,j \in S_T} p_{Sk}(m_{ij}) \ln \frac{p_{Sk}(m_{ij})}{p_{\hat{S}k}(m_{ij})}, \tag{3.23}$$

where $p_{Sk}(m_{ij})$ is the probability of the cell m_{ij} of being occupied with observation z covering the area S from location x. This formula compares observations and estimates with the same bearing. It obtains the KL divergence for a given orientation. The KL divergence equation for the complete scan formed by Ns observations can be expressed as:

$$d_{KL}(P_S||P_{\hat{S}}) = \sum_{k=1}^{N_s} d_{KL}^k(P_{Sk}||P_{\hat{S}k}). \tag{3.24}$$

As indicated in its properties, the KL divergence value is positive or equal to zero. The minimum value occurs when the discrete probability density from the actual and estimated positions are the same. A correction factor is introduced to distinguish between places where the same cost value is obtained but the number of occlusions is different:

$$KLD = d_{KL}(P_S||P_{\hat{S}})e^{\frac{N_{occ}}{N_s}}, \tag{3.25}$$

where N_{occ} represents the number of occlusions ($z_{t,k} << \hat{z}_{t,k}$) found. This correction factor has been chosen empirically. A typical problem of the KL divergence has to be considered. When $q(i)$ is very low for a particular i, the specific term $p(i) = q(i)$ can dominate the result. This issue has not been found in the experiments that have been performed in this work. Observing Figure 3.8 and Table 3.1, $P_{\hat{S}}$ is in the interval [0.05, 0.95]. In this way, the difference between larger and lower probabilities is not large enough to cause the cited problem in the current implementation.

Density Power

DP divergence was developed by Basu et al. [121]. Following the same ideas explained in the previous sections to develop the cost function of the GL filter, the next formula is applied to compute the DP divergence for a given orientation:

$$d_{DP}^k(P_{Sk}||P_{\hat{S}k}) = \sum_{i,j\in S_T} \left[p_{Sk}^{1+\alpha}(m_{ij}) - \left(1+\frac{1}{\alpha}\right) p_{\hat{S}k}(m_{ij}) p_{Sk}^{\alpha}(m_{ij}) + \frac{1}{\alpha} p_{\hat{S}k}^{1+\alpha}(m_{ij}) \right]. \tag{3.26}$$

This dissimilarity measure is a version of the Beta divergence. It has the same properties: a single global minimum for $P_{Sk} = P_{\hat{S}k}$, non-negativity, and increment dependent on the absolute value of the difference between densities. The authors in [12] believe that the most important motivation to study this metric from a practical point of view is to increase the robustness of the learning algorithms with respect to outliers.

It can be appreciated that this metric is dependent on a variable factor. It has been reported in [121] that when $\alpha = 1$, the DP divergence is equivalent to the L2-norm. The indefinition that appears when $\alpha = 0$ is solved by using the L'Hôpital's rule. The generalized KL divergence is obtained in this case.

Due to the versatility of the current method, this factor could be set to a fixed value or introduced as an additional parameter in the population set to be optimized. Both options have been tested. However, in the current version of the method, α will tend to one if Equation 3.26 is optimized. According to the formula, lower values are obtained for the cost function when $\alpha = 1$. A different approach to the optimization problem is needed in order to exploit α as an additional optimization factor. When the whole scan is considered, the next formula is computed:

$$d_{DP}(P_S||P_{\hat{S}}) = \sum_{k=1}^{N_s} d_{DP}^k(P_{Sk}||P_{\hat{S}k}). \tag{3.27}$$

Finally, the cost value when the correction factor that considers occlusions is introduced is computed by

$$DPD = d_{DP}(P_S||P_{\hat{S}})e^{\frac{N_{occ}}{N_s}}. \tag{3.28}$$

It has to be remarked that one advantage of this formula is that, since α is fixed to one, the cost value is equivalent to a quadratic error version but using the probability profiles approach. The DP divergence has been recently applied to point set registration in computer vision [135], multivariate analysis [136], or active learning [137].

Itakura-Saito

IS divergence was proposed in the sixties by Itakura and Saito [122]. As KL it is part of the non-symmetric Bregman divergences. It is applied to the current problem following the same ideas described for the other divergences. For a given orientation of the laser scan, it is expressed as

$$d_{IS}^k(P_{Sk}||P_{\hat{S}k}) = \sum_{i,j \in S_T} \left[\frac{p_{Sk}(m_{ij})}{p_{\hat{S}k}(m_{ij})} - \ln \frac{p_{Sk}(m_{ij})}{p_{\hat{S}k}(m_{ij})} - 1 \right]. \tag{3.29}$$

It is directly derived from Equation 3.19 when $\beta = 0$. Therefore, the properties of the Beta divergences are inherited. It can be observed that the Beta divergences connect the IS divergence ($\beta = 0$), the generalized KL divergence. ($\beta = 1$), and the Euclidean L2-norm ($\beta = 2$). The same can be concluded using different values of α because Equation 3.26 relies on a different version of the Beta divergence. Cichocki and Amari [128] explain that the choice of β is related to the statistical distribution of the data sets. For example, the optimal choice for the normal distribution is $\beta = 2$ and for the gamma distribution is $\beta = 0$. Although the problem studied here cannot be strictly linked to a known probability distribution, this analysis suggests that a further study about the statistical properties of the problem would help to choose the most suitable metric. When the whole scan is considered, it is equal to:

$$d_{IS}(P_S||P_{\hat{S}}) = \sum_{k=1}^{N_s} d_{IS}^k(P_{Sk}||P_{\hat{S}k}). \tag{3.30}$$

Finally, the cost value when the correction factor that considers occlusions is introduced is computed by

$$IS = d_{IS}(P_S||P_{\hat{S}})e^{\frac{N_{occ}}{N_s}}. \tag{3.31}$$

Jensen-Shannon

The JS divergence [120] is derived from the KL divergence. Given the KL divergence between two probability distributions (Equation 3.24), the JS divergence is

$$d_{JS}(P||Q) = \frac{1}{2}[d_{KL}(P||M) + d_{KL}(M||Q)], \tag{3.32}$$

where $M = (P + Q)/2$.

This metric can be classified as a symmetrized Alpha divergence. In general, there are two ways to symmetrize divergences. The first option is to use Equation 3.32. The JS divergence is a special case of Alpha divergence that is obtained when Equation 3.32 is applied to symmetrize the Alpha divergence (using d_A instead of d_{KL}) and the limit for $\alpha \longrightarrow 0$ is computed. The Jeffreys divergence [138] is obtained with the second symmetrization option:

$$d_{JF}(P||Q) = \frac{1}{2}[d_A(P||Q) + d_A(Q||P)], \tag{3.33}$$

In this case, the limit for $\alpha \longrightarrow 1$ has to be computed for Equation 3.33. Due to its relation with the JS divergence, some results using the Jeffreys divergence are included in the experiments. The next formula is applied to compute the JS divergence for the whole scan:

$$d_{JS}(P_{Sk}||P_{\hat{S}k}) = \frac{1}{2}[d_{KL}((P_{Sk}||M) + d_{KL}(P_{\hat{S}k}||M)], \tag{3.34}$$

where $M = (P_{Sk} + P_{\hat{S}k})/2$.

This divergence is also known as information radius (IRad) [139] or total divergence to the average [140]. It has important differences with respect to the KL divergence because it is symmetric, and it always results in a finite value. The square root of the JS divergence is usually referred to as JS distance [141].

This metric is based on Jensen's inequality [142] and the Shannon entropy. Jensen's inequality 'relates the value of a convex function of an integral to the integral of the convex function'. The application of its related theorems to information theory is used to define both the KL and the JS divergences. The Shannon entropy, named after the American mathematician Claude Shannon, is a well-known measurement used in information theory that is defined as $H = -\sum_i p(i)\log_b p(i)$.

The JS divergence has been applied to multiple fields, such as computer science, machine learning, medicine, history, etc. Some examples are given in papers about quantum information theory [143], machine learning [144], and medicine [145].

Finally, the cost value when the correction factor that considers occlusions is introduced is computed by

$$JSD = d_{JS}(P_S P_{\hat{S}})e^{\frac{N_{occ}}{N_s}}. \tag{3.35}$$

Illustrative example

An example where the divergences are computed for a single laser orientation can be observed in Figure 3.10. The first measurement (k=1) of the laser scan from the real pose and the estimate are represented in a grid map. The first step is to model the probability distributions of the laser beams. Different options have been defined. The values that have to be considered for computing the four divergences are shown in Table 3.2. This case is only an introductory example to explain the concepts behind the last sections.

For each cell crossed by each, the laser beam in its bearing direction in increasing order, different values are awarded to the free space (0.05), obstacles hit by the laser (0.95), and the uncertain cells right before obstacle detection (0.5) as indicated in Table 3.1. Represented in Figure 3.10, the first laser beam from the real position is constituted by traveling 11 free cells, and 13 probability values form the discrete probability distribution p_1 where cell 12 is denoted by the 0.95 as it represents the *hit* value and immediately posterior cell is the uncertainty probability (0.5). The same procedure is followed for the first beam of the estimate's perception (p_2), formed by 9 free space cells, an obstacle *hit*, and several uncertainty values after cell number 10 to square the number of cells compared. These probabilities are not yet normalized, as indicated by the KL divergence definition. After adding the terms of the fourth row, the KL divergence is equal to 0.3475. The average sum of the fifth and sixth rows is the JS divergence, which is 0.3897 in this case. The DP divergence is equal to 1.2150. The IS divergence is 3.6578. It can be observed that the parameter that is used to measure the difference between probability distributions is completely different

Figure 3.10: Measurement comparison from real pose (left) and estimate(right). Distance measured in cells on a occupancy grid map.

depending on the metric.

Table 3.2: Probability distributions for the example shown in Figure 3.10. Values: free space=0.05, obstacle=0.95, unknown space=0.5. $p_1(cell) = p_{sk}(m_{ij}), p_2(cell) = p_{\hat{s}k}(m_{ij}).p_M(cell) = [p_1(cell) + p_2(cell)]/2$. Left column: divergence for which the parameters of each row are needed.

	$cell$	1	2	...	9	10	11	12	13
	$p_1(cell)$	0.05	0.05	...	0.05	0.05	0.05	0.95	0.5
	$p_2(cell)$	0.05	0.05	...	0.05	0.95	0.5	0.5	0.5
	$p_M(cell)$	0.05	0.05	...	0.05	0.5	0.275	0.725	0.5
KL	$p_1(cell)ln\frac{p_1(cell)}{p_2(cell)}$	0	0	...	0	-0.14725	-0.1151	0.6098	0
JS	$p_1(cell)ln\frac{p_1(cell)}{p_M(cell)}$	0	0	...	0	-0.1151	-0.0852	0.2568	0
JS	$p_2(cell)ln\frac{p_2(cell)}{p_M(cell)}$	0	0	...	0	0.6098	0.2989	-0.1858	0
DP	$[p_1(cell) - p_2(cell)]^2$	0	0	...	0	0.8100	0.2025	0.2025	0
IS	$\frac{p_1(cell)}{p_2(cell)} - ln\frac{p_1(cell)}{p_2(cell)}$	0	0	...	0	1.9971	1.4026	0.2581	0

Chapter 4

Experimental Results and Discussion

This chapter is divided into three different sections regarding each type of environment representation. The experiments over 2D occupancy grid maps are presented in the first section while its extension for the 3D version can be found in section 4.2. Finally, section 4.3 shows the experiments for sparse PC-based GL. All three sections demonstrate the robustness of the different solutions for GL presented in this work regarding the accuracy, sensor noise response, and different considered perturbations, based on the uniform distribution of singular occlusions and different distribution of unmodeled obstacles. Furthermore, the different tests are presented for various Meta-heuristics in the case of PC-based maps and the different divergence-based fitness function implementations.

4.1 2D Simulated Environments

This section will present several experiments conducted with the DE-based GL algorithm over 2D occupancy grid maps. The objective of is to validate the performance of the method and the influence of the different divergences implemented as cost functions (KL, DP, IS, JS). Different variables can measure the performance of each method. Among them, the robustness to occlusions and the accuracy are of special interest. Other characteristics to be measured are the robustness before sensor noise levels, population requirements, and convergence properties.

Two-dimensional performance is tested in simulated environments obtained from the blueprints of the Department of Systems Engineering and Automation (DISA) of the Carlos III University of Madrid. Different sizes of the department's blueprints are taken into account depending on the qualities evaluated. A map of the entire building floor can be seen in Figure 4.1, and Figure 4.4 shows only the DISA. Real

Figure 4.1: GL in simulated map. All units in cells.

maps are also considered, occupancy map representations of 2D maps based on real laser data. Two examples are tested in this section: a small representation of the DISA department (Figure 4.2), and a real map of the Intel Laboratory provided by Dieter Fox, which has been widely used in localization research and thus will be very helpful as a performance comparison. As a reminder, the laser sensor is simulated over a 180° FoV, formed by 61 distance measurements. Configuration parameters of the DE-filter are: mutation coefficient $F = 0.8$ and crossover rate $CR = 0.75$. The sensor noise is modeled as a Gaussian distribution over each laser beam reading, a standard deviation of 2% is considered except when specified. DP-divergence, as explained in section 3.4 is controlled by a variable parameter, in these experiments $\alpha = 1$, as a probability equivalent of the L2-norm.

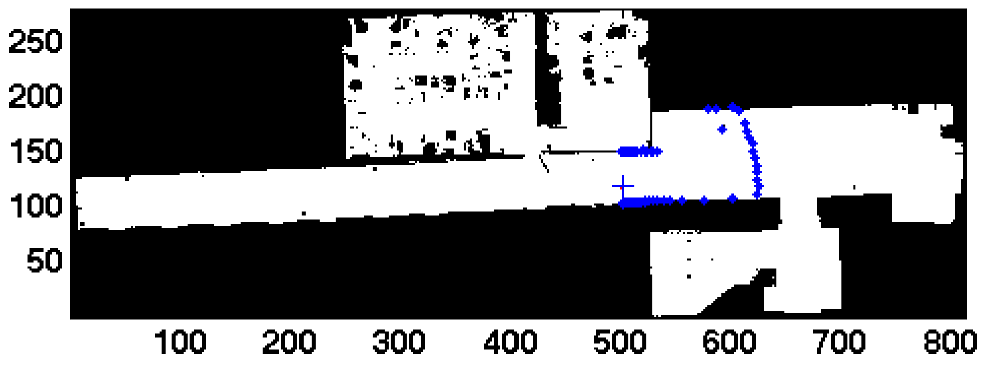

Figure 4.2: GL in a real map. All units in cells.

4.1.1 Performance and Accuracy

One of the main performance characteristics to be measured and compared is the accuracy and the proximity of the optimum estimation to the real location. A GL method's accuracy is a key value to optimum performance.

In the first experiment, the localization outcome when the robot is situated in diverse locations of the simulated map of Figure 4.1 are registered in Tables 4.1 and 4.2. The population requirements are only shown in the last table for simplicity. Each cell is a square of 12.1 cm side, and the total map has $960x300 = 288,000cells(4200m_2)$.

The robot's position and orientation are defined by a vector of three coordinates (x, y, θ). The first two parameters define the position, and the orientation is given by the third coordinate (zero when pointing right in the horizontal direction, increasing clockwise). The positioning error is expressed in cm and defined by the difference between the robot's true position and the estimate. The orientation error measures differences in degrees.

Table 4.1: Translation GL error when the robot is located in the simulated map of Figure 4.1. Errors in mean ± standard deviation (cm).

Pose	KL	DP	IS	JS
$(750, 50, 90)$	1.69 ± 0.97	1.72 ± 1.27	1.99 ± 1.13	2.46 ± 1.02
$(600, 130, 180)$	8.29 ± 7.01	11.52 ± 3.83	12.56 ± 7.9	10.66 ± 4.45
$(860, 60, 0)$	1.36 ± 0.48	2.19 ± 1.99	1.38 ± 1.00	1.61 ± 1.22
$(805, 245, 0)$	1.15 ± 0.57	2.36 ± 1.26	1.63 ± 1.10	3.11 ± 1.20
$(230, 30, 0)$	6.24 ± 1.81	6.11 ± 2.51	6.10 ± 0.99	7.08 ± 1.07
$(190, 33, 0)$	5.86 ± 1.30	9.40 ± 2.46	9.40 ± 1.71	9.08 ± 2.43
$(568, 84, 3)$	2.87 ± 0.91	5.69 ± 4.35	3.51 ± 3.18	4.76 ± 4.56

When speaking about qualitative performance, a distinction between localized or not localized is necessary. A distance threshold must be defined to evaluate if the estimated location matches the real pose (the distance between them is lower than a specified threshold) or if the estimation and the reality do not match. The success rate expresses the percentage of coincidences between reality and estimation over the total amount of trial tests performed. Since, during the experiments, the true location is known, it is possible to define a distance threshold, which is fixed to 50 cm.

Analyzing the results shown in Table 4.1, the position error when the robot is located in 7 distinct poses is in the interval $[1.15, 12.56]cm$ for all the different divergence analyzed. This range error is accurate enough to consider that the GL issue is solved for all fitness functions considered. We cannot establish a clear behavior pattern when comparing the different performances in terms of positioning error between the distinct cost functions. However, KL divergence seems to have one of the lowest, if not the lowest, positioning errors for all the different positions tested. The rest of the tested options do not yield conclusive results.

Table 4.2: Orientation error when the robot is located in the simulated map of Figure 4.1. Errors in mean $\pm$ standard deviation ($^\circ$).

Pose	KL	DP	IS	JS
$(750, 50, 90)$	0.04 ± 0.03	0.06 ± 0.05	0.05 ± 0.02	0.04 ± 0.03
$(600, 130, 180)$	0.16 ± 0.47	0.09 ± 0.06	0.09 ± 0.84	0.18 ± 0.18
$(860, 60, 0)$	0.00 ± 0.01	0.07 ± 0.10	0.03 ± 0.08	0.05 ± 0.09
$(805, 245, 0)$	0.02 ± 0.03	0.08 ± 0.06	0.00 ± 0.01	0.05 ± 0.06
$(230, 30, 0)$	0.15 ± 0.14	0.18 ± 0.19	0.13 ± 0.10	0.38 ± 0.19
$(190, 33, 0)$	0.20 ± 0.19	0.77 ± 0.29	0.75 ± 0.22	0.67 ± 0.34
$(568, 84, 3)$	0.34 ± 0.12	0.11 ± 0.07	0.19 ± 0.13	0.18 ± 0.11

Table 4.3: Percentage of localization success when the robot is located in the simulated map of Figure 4.1.

Pose	N_p	KL	DP	IS	JS
$(750, 50, 90)$	250	95	90	90	90
$(600, 130, 180)$	350	95	100	95	100
$(860, 60, 0)$	250	100	100	100	100
$(805, 245, 0)$	250	100	100	100	100
$(230, 30, 0)$	300	95	100	100	90
$(190, 33, 0)$	400	90	100	100	100
$(568, 84, 3)$	250	100	100	100	100

Referring to the observed orientation error (Table 4.2), its value is proximal to zero with all divergences and stays among the interval [0.00, 0.77]deg. Therefore, it can be concluded that all cost functions show optimum performance in orientation accuracy. Table 4.3 shows the percentage of success in finding a location for each one of the divergences tested in the different locations. As it can be seen, this factor is over 90% for every case studied.

To extend our analysis, accuracy experiments over real maps have been conducted. First, Figure 4.2 shows an example of GL over a real map registered by the Robotics Lab. In this map, each cell is a square of 5.6 cm side, and the whole map contains

Table 4.4: Translation GL error when the robot is located in the real map of Figure 4.2. Errors in mean $\pm$ standard deviation (cm).

Pose	KL	DP	IS	JS
$(500, 120, 0)$	5.88 ± 3.27	4.84 ± 1.66	3.07 ± 2.18	3.54 ± 1.20
$(650, 50, 0)$	4.16 ± 1.62	6.40 ± 0.88	5.01 ± 1.54	4.73 ± 1.67
$(750, 170, 0)$	2.06 ± 0.51	2.20 ± 0.48	2.26 ± 0.32	2.17 ± 0.63
$(560, 120, 90)$	3.62 ± 0.92	3.47 ± 0.78	4.59 ± 0.63	4.52 ± 1.07
$(785, 135, 0)$	2.43 ± 1.55	3.08 ± 2.17	2.67 ± 1.93	2.55 ± 1.89
$(390, 170, 0)$	0.45 ± 0.84	0.02 ± 0.01	0.38 ± 0.60	0.62 ± 0.85
$(100, 100, 0)$	1.58 ± 1.27	1.68 ± 0.77	0.71 ± 0.57	1.51 ± 1.17

Table 4.5: Orientation GL error when the robot is located in the real map of Figure 4.2. Errors in mean $\pm$ standard deviation ($^\circ$).

Pose	KL	DP	IS	JS
$(500, 120, 0)$	0.16 ± 0.08	0.14 ± 0.10	0.14 ± 0.09	0.07 ± 0.07
$(650, 50, 0)$	0.36 ± 0.23	0.77 ± 0.15	0.51 ± 0.41	0.47 ± 0.28
$(750, 170, 0)$	0.12 ± 0.12	0.19 ± 0.24	0.18 ± 0.12	0.15 ± 0.15
$(560, 120, 90)$	0.45 ± 0.05	0.42 ± 0.06	0.49 ± 0.05	0.42 ± 0.06
$(785, 135, 0)$	0.04 ± 0.13	0.07 ± 0.22	0.17 ± 0.30	0.15 ± 0.19
$(390, 170, 0)$	0.07 ± 0.10	0.00 ± 0.01	0.06 ± 0.05	0.07 ± 0.07
$(100, 100, 0)$	0.05 ± 0.04	0.01 ± 0.02	0.03 ± 0.04	0.11 ± 0.10

800x275=220,000 cells, equivalent to a 689.92 m^2 area representation. Table 4.4 refers to the positioning error when the robot is located in different locations on the real map. This error is in every case tested among the interval $[0.02, 5.88]cm$ and the orientation error shown in Table 4.5 is in the interval [0.00, 0.77]$^\circ$. The magnitude of these results is very similar in both simulated and real maps. We must notice that the resolution, in this case, is higher, with cells of 5.6 cm against the previous 12.1cm, which means that the same error in terms of cells results in almost half the error in centimeters. The percentage of localization success can be seen in Table 4.6 and again is over 90 % for every localization considered.

Finally, the accuracy has also been tested over the Intel Laboratory real map, which has been widely used in localization research and will be very helpful in comparing results with other GL methods proposed. This map is composed of 600x600=360,000 cells of 5 cm side each which represents an area of 900 m^2. The positioning error varies between 0.15 and 2.45 cm, while the orientation is between 0.01 and 0.21°. The percentage of success, in this case, is over 95% for each one of the four locations tested. Again, the accuracy results derived for the experiments on

Table 4.6: Percentage of localization success when the robot is located in the real map of Figure 4.2.

Pose	KL	DP	IS	JS
$(500, 120, 0)$	100	100	100	100
$(650, 50, 0)$	95	100	90	100
$(750, 170, 0)$	100	100	100	100
$(560, 120, 90)$	90	100	95	100
$(785, 135, 0)$	100	100	100	100
$(390, 170, 0)$	100	100	95	100
$(100, 100, 0)$	100	100	100	100

Figure 4.3: GL in Intel Lab real map. All units in cells.

real maps do not help us conclude if one of the metrics is more suitable for GL than the others. In addition, we may not even say that KL divergence tends to present lower error among them in real environments. The values are very similar both for position and orientation error, and the performance is accurate independently of the cost function chosen.

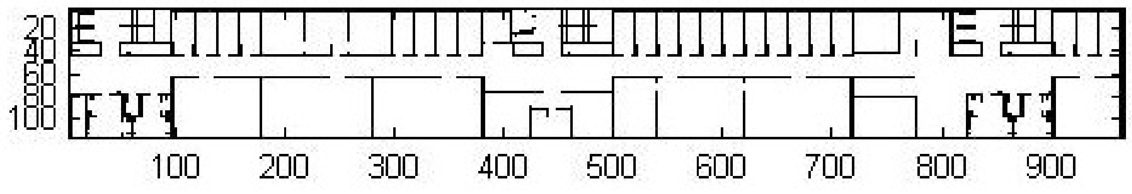

Figure 4.4: GL in a medium-size simulated map. All units in cells.

Other research groups have proposed different localization methods that can be

Table 4.7: Translation GL error when the robot is located in the real map of Figure 4.3. Errors in mean ± standard deviation (cm).

Robot's Pose	KL	DP	IS	JS
$(380, 125, 0)$	0.15 ± 0.19	0.68 ± 0.61	0.73 ± 0.58	1.04 ± 0.61
$(320, 120, 10)$	1.02 ± 1.46	0.83 ± 1.60	1.11 ± 1.51	0.93 ± 1.01
$(440, 258, 21)$	1.96 ± 4.12	2.05 ± 1.09	1.37 ± 2.02	2.45 ± 0.69
$(75, 520, 3)$	0.53 ± 0.35	2.01 ± 1.05	0.85 ± 1.04	0.72 ± 0.65
$(80, 50, 90)$	7.97 ± 7.36	13.38 ± 9.73	13.5 ± 9.15	12.10 ± 12.01
$(122, 120, 15)$	11.93 ± 9.77	7.22 ± 6.10	9.39 ± 7.67	10.20 ± 6.91
$(180, 60, 0)$	0.23 ± 0.15	0.52 ± 0.24	0.25 ± 0.17	0.30 ± 0.22

Table 4.8: Orientation GL error when the robot is located in the real map of Figure 4.3. Errors in mean ± standard deviation (°).

Pose	KL	DP	IS	JS
$(380, 125, 0)$	0.01 ± 0.03	0.08 ± 0.08	0.02 ± 0.02	0.12 ± 0.08
$(320, 120, 10)$	0.18 ± 0.28	0.08 ± 0.05	0.16 ± 0.21	0.12 ± 0.13
$(440, 258, 21)$	0.11 ± 0.04	0.11 ± 0.09	0.21 ± 0.16	0.07 ± 0.05
$(75, 520, 3)$	0.02 ± 0.01	0.09 ± 0.04	0.06 ± 0.06	0.04 ± 0.02
$(80, 50, 90)$	0.98 ± 1.2	$0.45.38 \pm 0.73$	0.5 ± 0.15	0.810 ± 1.01
$(122, 120, 15)$	0.93 ± 0.77	0.22 ± 0.10	0.39 ± 0.67	0.20 ± 0.91
$(180, 60, 0)$	0.23 ± 0.15	0.52 ± 0.24	0.25 ± 0.17	0.30 ± 0.22

Table 4.9: Percentage of localization success when the robot is located in the real map of Figure 4.3.

Robot's Pose	Np	KL	DP	IS	JS
$(380, 125, 0)$	300	100	100	100	100
$(320, 120, 10)$	300	95	100	100	100
$(440, 258, 21)$	300	100	95	100	100
$(75, 520, 3)$	300	100	100	100	100
$(80, 50, 90)$	300	74	84	62	74
$(122, 120, 15)$	300	98	100	100	100
$(180, 60, 0)$	300	100	100	100	100

compared to our technique. The position error published in [116] by the research group of Donoso is $[8, 15]cm$. The average translation error of the technique proposed by Se *et al.* [146] is $7cm$. Their average angular error is one degree. Similar errors have been found when investigating the results of other research groups in the same type of environment. Since these algorithms rely on different concepts and assumptions, it is necessary to study the other techniques' configuration parameters and experiment conditions to present a more exhaustive comparison.

Another important aspect that must be taken into account when testing the method's accuracy is its reaction to different amounts of noise in the sensor. As it is known, every existing sensor's measurement comes with an inherent noise that deviates the measurement from the correct value.

Figure 4.5: Sensor noise influence. Left: position error vs. sensor noise. Right: success vs. sensor noise.

In this case, we will model the laser's noise as a Gaussian with deviation σ from the real value. This Gaussian noise was set to a typical value of 2% in the previous experiments, but in the next experiment, the performance reaction to increasing this level will be tested. A smaller part of the building map was considered in this experiment just to reduce the computational cost. Figure 4.4 shows the simulated partial map considered, and the results are shown in the chart of Figure 4.5. The left of the figure shows the positioning error when the laser noise increases from 0 up to 10%. The error grows almost linearly for every divergence, and acceptable error levels are obtained even with high noise. Again there is a slight advantage when using KL divergence over the rest, especially with higher noise levels. Interestingly, JS divergence error experiments a higher growth from a 7% noise rising for 4 to 14 cm. The right figure shows the percentage of success. It can be observed that the algorithm finds an acceptable solution in more than 95% of the cases even with a 10% noise level. In addition, it is not affected by noise, at least at these levels, applying the DP cost function.

4.1.2 Occlusions: Uniform Noise

This section explains the performance of the DE-based Gl filter and the different divergences before occlusions caused by the presence of people or small mobile obstacles. As described in section 3.4, this type of occlusion is modeled as a uniform distribution noise that shortens the distance measured by each laser beam. In this work, the behavior of the algorithm when using KL, DP, IS, and JS-based cost functions has been tested.

The objective is to study the algorithm's performance when the true location is in a distinguishable place of the map shown in Figure 4.4. A uniform distribution has selected to model the random measurements that represent the occlusion of the sensor measurement. The minimum value is the 25% of the sensor distance, and the maximum value is the 75% of the laser measurement. In this way, the contaminated laser reading can be expressed by

$$z_{k,c} = (1-\epsilon)N(z_k,\sigma) + \epsilon U(0.25z_k, 0.75z_k), \tag{4.1}$$

where the laser measurement ($z_{k,c}$) can be given by the real sensor distance ($(N(zk),\sigma)$) or the uniform distribution $U(0.25_{zk}, 0.75_{zk})$ depending on the contamination level, expressed as a value from 0 to 1 representing the percentage of contaminated measurements, $\epsilon \in [0,1]$. The mean of the normal distribution N is the laser distance (z_k), and the sensor noise is represented by the standard deviation ($\sigma = 2\%$ in this experiment). The range of the uniform distribution is $[0.25_{zk}, 0.75_{zk}]$. Occlusions considered in this experiment are not present in the known map. The considered pose is $(x,y,\theta) = (60,50,0)$. An example with a 50% of contamination is shown in Figure 4.6. It can be appreciated that the uniform noise causes significant changes in the laser scan.

Figure 4.6: Laser scan with 50% level of contamination by uniform noise. Units in cells.

The study of the influence of the contamination level is presented in Figure 4.7. The horizontal axis corresponds to the percentage of contaminated measurements (contamination level). The vertical axis is used for the position error (left). The percentage of success is in the right part.

Figure 4.7: Left: GL error vs. uniform noise (percentage of contaminated measurements) in the simulated map of Figure 4.4. Right: success vs. uniform noise.

The results are outstanding in all cases if the localization error is considered. Therefore, the probability profiles that are defined to model the different metrics cause a great improvement in the method behavior when uniform noise is included. The errors are lower than 5 cm even with a 50% of contamination. This means that the 50% of the laser readings are wrong measurements originated by the uniform noise. The same error is obtained with 5% of contamination when the quadratic error cost function is used. Besides, when using the quadratic error, it is not possible to obtain the solution when the contaminated noise is greater or equal to 28%. Analyzing the positioning error, the results for every cost function tested are very similar and robust, with a maximum of 10 cm error up to a 60% of contaminated or obstructed laser measurements. From this 60%, the behavior differs depending on the divergence tested, highlighting the more lineal behavior of the IS divergence, which maintains an almost linear growth from the beginning while the rest experiment a sudden increase of slope, being the DP the one with the worst results with high levels of noise. Worth noting that, despite a worse response with a higher amount of obstructions ($> 60\%$), KL and DP show a better behavior, and lower error, until that point, especially around a 40-60% noise.

Regarding the success rate, similar conclusions are obtained. The DE-GL method obtains optimum results independently of the divergence used with a 100% success up to a 60% level of occlusions. From this point, the percentage decreases. IS divergence is the metric with the best results reducing its success only to 90% with an 80% noise level. The results of the rest of the divergences are quite similar, decreasing with certain linearity from a 100% to around 60% success from a 60-65% to an 80% noise.

To reinforce this idea, an equivalent test has been performed on the Intel Real map of Figure 4.3. In this case the robot is located in $(x, y, \theta) = (380, 125, 0)$ and sensor noise σ is again 2% an the same uniform noise distribution has been introduced (Equation 4.1). The results show a great performance with less than 12 cm error up to an 80% occluded measures (higher resolution of the map must be taken into account) for every single divergence with optimum outcomes in terms of success rates until 65-70%. It is interesting to mention the remarkable difference in the influence between sensor noise (Gaussian) and occlusions (Uniform), with a much lower influence of the latter. We must remember that similar behaviors can be found with an 8% (Figure 4.5) sensor noise and with a 70% uniform noise.

Wang *et al.* [147] have published a particle-based localization filter for high-occluded and dynamic environments. They have tested their technique in a 0.1 m resolution map. Their lower errors are 5 cm and 1.1 degrees with 2 moving people and 10 cm and 1.5 degrees with 8 moving people. The authors have defined a variable called occlusion ratio, which is similar to the percentage of uniform noise presented here. Although they do not present exact values depending on the occlusion ratio, they obtain an error of several meters when the occlusion ratio is approximately 30%. Their experiments are focused on the tracking problem in a real scenario.

According to the results presented in this section, it can be concluded that the probability profiles that are defined to deal with occlusions produce a great improvement of the method capabilities in this type of situation.

4.1.3 Occlusions: Unmodeled Obstacles

In this section, the GL filter is examined when obstacles not modeled in the available floor plant are added. Measuring the performance before these type of occlusions is quite difficult, as it is not easy to determine which approach is more suitable due to the variety of possible dispositions and influential factors. Two different experiments have been implemented when the robot is in the map of Figure 4.4. First, a single remarkable obstacle is situated in front of the robot, and the percentage of success is computed when the robot is approaching the obstacle. Figure 4.8 registers the results. The scan on the right side of this figure corresponds to a 57 cell ($6.9m$) distance between the robot and the obstacle.

The occlusion caused by the unmodeled obstacle complicates the localization process. In the right side of Figure 4.8, it can be observed that the laser perception is drastically modified regarding the known map. The key conclusions are derived from the analysis of the percentage of success, where it can be seen that all the different divergences present optimum results for at least 36 cells (4.3m) distance, being KL the more robust with a 100% success when the obstacle is 31 cells from the robot. The minimum distance with optimum results when using the quadratic fitness function was 87 cells ($10.5m$). This distance is now in the interval [31,37] cells depending

Figure 4.8: Occlusions originated by a remarkable obstacle. Left: percentage of success vs. distance to obstacle. Right: Example os the scan from 57 cell distance.

on the divergence. Since all metrics result in a similar performance, the probability profile is also excellent for this situation. Although no significant differences are found between cost function implementations, the best results are obtained with the KL divergence. In the worst scenario, all the options achieve the true location of the robot at least in half of the tests when the obstacle is at 2.6m (22 cells). This means that with this GL approach if the robot is at this distance from an unpredicted obstacle that creates a major occlusion in its sensor, it will be able to locate itself even a 50% of the time.

Tsou and Wu [148] have published a localization method where feature matching is used to deal with dynamic obstacles. They have performed a similar experiment where a dynamic obstacle is added to the scanning area. Two different distances to the obstacle are tested: 434 cm with 2% dynamic information and 62 cm with 22% dynamic information. The average errors are 18.6 cm for the first case (2%) and 24.8 cm for the second one (22%). The success rate is approximately equal to 90% in both cases. Their algorithm is tested in a squared map (18.5m side) where possible locations are generated using different resolutions. The results with the highest resolution, which is 49 cm, are reported here.

In a second test, a singular pose when the robot is located in the map of Figure 4.4 was selected. Different small unmodeled objects are added. This process is shown in Figure 4.9 where the real laser scans are presented in the top row of the image and the estimation scan on the bottom row. The number of obstacles grows from left to right (5,15, and 45 for each column). Quantitative results are presented in Table 4.10. The number of occlusions $(0-45)$ is equal to the number of laser measurements affected by the unmodeled obstacles. The GL errors and percentages of success are shown in the table, comparing the different divergences implemented. The limit to consider successful cases has been increased to 70cm in this experiment.

Figure 4.9: Occlusions originated by multiple obstacles (increasing amount from left to right). Top Row: real laser scan from the true pose. Bottom Row: real laser from the estimated pose.

When 37 laser readings out of 61 are wrong measurements caused by occlusions, the correct location is obtained in all cases, and the worst error is 7.55 cm. The error that was reported with the quadratic cost function in the same circumstances (with occlusions) was in the interval [10.57, 11.76] cm. The errors of the current metrics are significantly lower in all cases. When the number of occluded measurements was greater or equal to 17, the quadratic-based method failed to reach the solution, and the success rate was 0%. After this analysis, it is clear that the probability profiles technique is the optimum option in situations with unmodeled obstacles. Comparing the results when using different metrics, the error with and without obstacles up to 37 occluded measurements is similar. The accuracy is maintained when there are occlusions.

Although the error with the DP divergence is slightly higher when there are 17 occluded readings, it still presents an acceptable value. No significant differences are observed between metrics. For higher levels of occlusion(43 and 45 measurements), the error is higher, and the success rate decreases. The IS and the JS divergences show better results ([12.22, 13.54] cm). If the success rate is examined, the worst performance is obtained with the DP divergence. The other three fitness functions present similar values.

Table 4.10: GL performance with multiple unmodeled obstacles. True location:(55, 55, 0). Number of occlusions in the left column. Type of divergence in the upper left corner. Errors in mean±standard deviation (cm). Sensor noise: 2%.

	KL		DP		IS		JS	
Occlusions	Error	Success	Error	Success	Error	Success	Error	Success
0	3.50 ± 1.54	100	4.32 ± 1.98	100	4.35 ± 1.41	100	4.19 ± 2.08	100
5	4.43 ± 1.42	100	2.61 ± 1.25	100	2.09 ± 1.59	100	3.55 ± 1.92	100
8	0.69 ± 0.34	100	2.25 ± 1.58	100	1.90 ± 1.36	100	1.89 ± 1.34	100
17	2.40 ± 2.85	100	6.33 ± 2.60	100	2.08 ± 1.12	100	2.66 ± 1.26	100
37	7.55 ± 2.92	100	3.68 ± 2.10	100	4.13 ± 4.13	100	2.45 ± 1.45	100
43	11.74 ± 3.38	90	36.94 ± 13.01	26	13.49 ± 2.40	92	12.41 ± 3.26	92
45	50.87 ± 1.22	18	60.75 ± 13.11	22	13.54 ± 2.81	18	12.22 ± 3.23	22

4.1.4 Tracking performance

The last experiment shows the robot's capabilities when it is executing a path. The typical scenario of a mobile robot involves the continuous reception of laser scans. The robot is navigating, and, at the same time, the information provided by sensors is received. The previous experiments have to be considered a more difficult problem to solve because a single laser scan is the only source of information. In addition, when a single scan is used, it is possible to find places where the laser readings are very similar. For example, this situation happens in environments with symmetric offices. In order to be successful in these cases, motion and more laser scans from different locations are required.

An advantage of the current approach is that some parameters can be relaxed after convergence if the robot's pose has been correctly estimated. The objective of this operation is to improve the computational cost. The population is limited to 20 candidates, and the number of iterations is 100 in this experiment after convergence. These parameters are not reduced in the first execution. The computational cost in these conditions is in the interval [1.21, 1.32] s. In tracking, the algorithm can be focused on the obtention of a faster response because the GL problem has already been solved. However, the population size and the number of iterations can be modified if the objective is to increase the accuracy.

The path that is followed is displayed in Figure 4.10. The histograms of the position and orientation errors while the robot is navigating are plotted in Figure 4.11. The mobile robot begins situated in location (100, 500, 0). The first part of the path covers the hallway that can be seen in the lower part of the image. Then, the aisle on the right side is traversed. The final location, with coordinates (402, 206, 185), and the laser scan that is received from this point are shown in the figure.

In Figure4.11, it can be appreciated that the filter obtains the correct location even after the first laser reading (all errors lower than 10 cm). This result must be

Figure 4.10: Robot's path and laser scan from the final location. Starting point: bottom left corner. Units in cells.

considered an advantage when compared to other techniques. For example, the MC-based approaches need more motion-perception cycles to return an accurate solution because the sensor information has to be integrated into the motion model. If the position error is checked, it remains lower than 10 cm for all divergences. This error depends on different factors. Since the sensor noise is dependent on the distance to the obstacles, the error is larger when the robot is situated in larger rooms. The variation of the laser measurements is more significant when the robot is turning. Therefore, the error could be worse in sharp turns. As previously commented, a simple adjustment that can be applied to improve the accuracy (if necessary) is to increase the population size and the maximum number of iterations. The orientation errors are lower than 0.7° for all divergences. If the cost functions are compared, satisfactory results are obtained with all of them, and all the errors are in the same interval. Most errors are lower than 6 cm and 0.5 degrees. No significant differences are found between them.

If other methods are analyzed, a similar experiment is presented by Zhang *et al.* [149]. The authors have implemented an algorithm called SAMCL. One of the objectives of their tests is to establish a comparison between their technique and MC. The idea is to measure the pose tracking error in an indoor corridor. The authors have reported that their errors improve those of the MC basic version in pose tracking and GL. If their method is compared to the filter presented in the current paper, the main difference can be found when the error of the first motion-perception cycle is

Figure 4.11: Histograms of the errors for the path displayed in Figure 4.10. Top: position error. Bottom: orientation error. Number of points of the path with an error in the interval defined by each bin in the vertical axis.

analyzed. First, their error is around 80 cm, and, after that, it decreases to 20 cm when more scans are received. As can be observed, the performance is worse in the first execution.

4.1.5 Population requirements and computational cost

The study of the population requirements of the GL method is relevant because this parameter has an important influence on the computational cost. The population size that is required has been examined for different locations in the simulated map (Figure 3). The experiment consists of measuring the minimum set that is needed to obtain the maximum percentage of success. Only cases with 100% of success are shown. The aim is to check if any divergence needs lower particles than the other ones. The position errors and the required number of particles are displayed in Table 4.11. The orientation error is omitted for simplicity.

Table 4.11: Optimum position error and population size when the robot is in the simulated map of Figure 4.1. Errors in mean ± standard deviation (cm).

	KL		DP		IS		JS	
Robot's Pose	Error	Pop	Error	Pop	Error	Pop	Error	Pop
$(50, 60, 0)0$	$4.64 \pm 0, 84$	50	4.48 ± 1.08	35	4.84 ± 0.62	55	4.62 ± 0.88	40
$(790, 40, 90)$	4.63 ± 0.85	200	3.45 ± 1.05	205	4.30 ± 1.75	195	3.42 ± 2.07	201
$(700, 80, 180)$	8.66 ± 10.82	140	9.65 ± 4.96	165	12.23 ± 2.60	145	10.50 ± 5.61	170

Although some differences can be appreciated, there is no evidence to choose one metric among the others. In addition, this parameter is highly dependent on different variables such as the location (larger rooms need fewer particles because it is easier to reach the minimum) or the perceptual ambiguities (symmetries). Therefore, more experiments are needed to present a more exhaustive study of the population requirements, which is challenging work to be accomplished in the future. This section is intended to introduce the study of this parameter.

The time complexity of the filter is $O(DE_GL) = N_{iter} * N_P * N_s$, where N_{iter} is the number of iterations. This parameter depends on the number of iterations, the population size, and the number of laser scan measurements. The computational complexity is dependent on the number of iterations when the algorithm is executed for a fixed number of particles and a laser scan with a constant resolution ($N_p * N_s$ is constant). The time complexity of other evolutionary-based methods is similar. The MC technique has to be viewed from a different perspective because it integrates the motion information to compute the robot's pose. Many motion-perception cycles are needed to converge. Anyway, the time complexity of the MC-based algorithms is highly dependent on the number of particles. The whole search space (in GL without motion) must be covered to assure the convergence to the true solution. Therefore, the population size that is required is larger. Several comparisons between DE and variants of MC are presented in [150] and [151]. In both papers, the MC-based method needs more particles to succeed.

An experiment has been conducted to study the parameters that influence the computational cost. The objective is to compare the times per iteration depending on the population size for different locations and maps. The number of iterations that are needed to converge is also presented. Table 9 illustrates the results. The success rate is 100% in all cases. The algorithm is implemented in MATLAB in a computer with a 2.7 GHz Intel Core i5 processor. Different conclusions can be drawn from the table. No significant differences are found when comparing metrics. The cost highly depends on the number of particles. For 50 particles, the time per iteration is in the interval [26, 27] ms. For 100 candidates, the time is [47, 66] ms. For 200 members, the cost is [88, 112] ms. The iterations to converge depend on the map. In general, larger maps require more iterations. The method needs [362.4, 405.5] iterations for the map of Figure 3 (288,000 cells). When the robot is in the map of Figure 6 (120,000 cells), it needs [188.9, 216.3] iterations. In Figure 4 (360,000 cells), [350.1, 438.3] iterations are required. All metrics show similar numbers. However, a more exhaustive study is necessary to compare the convergence properties.

Table 4.12: Times per iteration for different locations and maps. Results in mean±standard deviation.

Map/Pose	N_p		KL	DP	IS	JS
Figure 4.1	200	time (ms)	91 ± 3	88 ± 2	95 ± 4	98 ± 5
(860,60,0)		N_{iter}	382.6 ± 50.2	387.6 ± 42.9	362.4 ± 61.6	418.1 ± 40.6
Figure 4.1	100	time (ms)	51 ± 3	50 ± 4	53 ± 5	63 ± 9
(860,60,0)		N_{iter}	397.6 ± 61.0	376.1 ± 59.5	405.5 ± 74.6	378.9 ± 70.5
Figure 4.4	100	time (ms)	50 ± 4	48 ± 4	47 ± 3	50 ± 4
(60,60,0)		N_{iter}	189.9 ± 42.9	212.0 ± 57.1	200.3 ± 37.5	188.9 ± 20.1
Figure 4.4	50	time (ms)	26 ± 3	26 ± 3	27 ± 2	26 ± 2
(60,60,0)		N_{iter}	209.3 ± 42.5	216.3 ± 41.9	193.1 ± 32.05	192.0 ± 25.6
Figure 4.3	200	time (ms)	112 ± 6	111 ± 5	110 ± 2	111 ± 2
(350,120,0)		N_{iter}	371.9 ± 63.1	438.3 ± 45.7	424.5 ± 55.96	435.6 ± 40.37
Figure 4.3	100	time (ms)	61 ± 3	66 ± 5	65 ± 5	63 ± 3
(350,120,0)		N_{iter}	377.5 ± 74.5	388.3 ± 69.24	350.1 ± 59.0	433.5 ± 50.2

4.2 3D Simulated Environments

This section presents the experiments performed over 3D occupancy grid maps. The algorithm tested in this experiment is the DE-based GL filter presented in section 3.3.1. The performance regarding the described divergence-based fitness functions is analyzed. The different situations evaluated are an extension of those presented in section 4.1 for 2D maps. The different tests show the performance of the developed method in terms of localization accuracy, response to different amounts of sensor noise, and to several types of occlusions: uniformly distributed and unmodeled obstacles. In all these tests, the sensor considered is composed of a $7x72$ beam laser scan, covering a 360° view with 5° resolution over 7 vertical planes separated 5°. Two map examples are tested in this section, a 3D representation of the blueprint of the DISA department on Carlos III University of Madrid (4.1), which is represented in Figure 4.12. This could be considered a simulated map, as no real data has been utilized. The second representation is a 3D elevation of the before-mentioned Intel Laboratory map, built using real 2D laser data. This can be considered a semi-simulated environment. The extension on dimension z is a simulation. The configuration parameters for the DE-filter are: $F = 0.9$, and $CR = 0.75$. The variable factor of DP-divergence is set to $\alpha = 1$. Each laser beam is modeled as a Gaussian distribution with standard deviation $\sigma = 2\%$ except when a different value is specified.

Figure 4.12: 3D simulated map. All units in cells. Red dots indicate locations considered in Table 4.13.

4.2.1 Performance and Accuracy

In the first experiment, GL results over different positions of the map represented in Figure 4.12 are registered in Tables 4.13,4.14 and 4.15. Each cell is a cube of 12.1 cm side, and the total environment represented is formed by 960x300x25=7,200,000 cells covering a $12600m^3$ volume. The population required for each tested location is only shown in Table 4.13 for simplicity. The robot's location is represented by $(x, y, z, \alpha, \beta, \gamma)$ covering the possible 6DOF of a 3D space. The same threshold distance between optimum estimation and real position is considered a successful localization, 50cm.

Table 4.13 shows the percentage of successful localizations for each position for each metric utilized. These positions are represented as red dots in Figure 4.12. Divergence-based functions yield better solutions than L2-norm for almost every position considered, except position 4. Among the rest of the options, the results are quite similar. It could be mentioned that JS-divergence provides the worst result of all situations tested for position 4.

Regarding the positioning error in Table 4.14 a clear behavior pattern can not be extracted among divergences. Although KL divergence obtains the lowest error in 50% of the locations, the difference with the rest of the divergences is not remarkable. The positioning error is in the interval [0.90-9.58]cm for all metrics, and the GL problem is solved for all metrics. A comparison with the L2-norm shows a very different result. The positioning error of this metric is higher for all positions evaluated with errors in an interval between [3.04-12.93] cm, which is also accurate enough.

Table 4.13: Percentage of GL success when the robot is located in the simulated map of Figure 4.12.

	Robot's Pose	N_p	L2	KL	IS	JS	DP
1	(150,750,15,0,0,0)	300	100	100	100	100	100
2	(135,555,15,-0.4,0.5,270)	350	100	100	100	100	100
3	(220,250,8,4,-2,10)	450	91	100	100	100	100
4	(90,840,4,1,0,0)	450	82	42	60	31	36
5	(275,550,18,-1,2,200)	400	39	100	100	94	100
6	(50,400,5,0,0,0)	250	100	94	100	100	100
7	(130,600,10,0,0,180)	450	39	67	70	78	83
8	(60,860,15,-2,3,0)	300	100	100	100	100	100
9	(84,568,12,0,4,3)	300	100	100	100	100	100
10	(245,805,8,2,-1,0)	250	100	100	100	100	100

The orientation error can be observed in Table 4.15. This orientation represents the average value of the three Euler rotation angles (α, β, γ) for simplicity. None of the different angles show a distinctive tendency even though it can be highlighted that, in case of failure to localize the robot, the biggest error appears in the *yaw* angle γ (rotation around z axis). Referring to the observed orientation error, its value is proximal to zero for all metrics and stays among the interval [0.08, 0.99]deg for all positions except number 4. Therefore, it can be concluded that all cost functions show optimum performance in orientation accuracy. Position number 4, $(90, 840, 4, 1, 0, 0)$, was selected to test the performance in a small corner inside a non-singular room of the map, therefore it is a challenging position due to the many symmetries that this environment presents.

The accuracy of the GL method has been tested over the Intel Laboratory occupancy grid extension to 3D. This map is composed of 600x600x25 =9,000,000 cells covering a volume of 1125 m^3, the side of each cubic cell is 5cm in this case, so again, a positioning error of the same amount of cells compared to the previous map will result in a lower error in cms. Observing Tables 4.16 and 4.17 it can be determined that the performance is equivalent to our simulated environment. A difference between the L2-norm and the divergences is clearly noticeable regarding the positioning error. While divergence-based function provides an error in the range from 0.79 to 9.92 cm, less-squared difference's error raises up to 27cm. In terms of success rate, Table 4.16 shows equivalent results with its homologous. The achieved localizations

Table 4.14: Translation GL error when the robot is located in the real map of Figure 4.12. Errors in mean ± standard deviation (cm).

Robot's Pose	L2	KL	IS	JS	DP
1	3.20 ± 3.30	1.09 ± 0.61	1.66 ± 1.14	1.21 ± 0.67	1.27 ± 0.69
2	9.97 ± 3.03	1.71 ± 0.64	3.56 ± 2.67	1.96 ± 0.94	3.19 ± 1.62
3	9.62 ± 4.60	9.58 ± 5.30	6.32 ± 1.65	4.62 ± 2.64	3.89 ± 2.74
4	12.93 ± 4.78	9.52 ± 6.50	7.02 ± 7.89	8.27 ± 7.99	8.24 ± 5.19
5	13.56 ± 8.50	4.38 ± 2.20	5.91 ± 3.74	6.97 ± 5.31	4.81 ± 3.11
6	6.93 ± 8.11	1.36 ± 0.55	1.14 ± 0.54	0.90 ± 0.55	1.45 ± 0.94
7	9.56 ± 5.56	6.24 ± 3.88	9.34 ± 1.25	6.69 ± 1.28	7.97 ± 4.26
8	5.85 ± 3.07	1.17 ± 0.91	1.61 ± 1.16	1.41 ± 1.00	1.34 ± 0.97
9	3.04 ± 4.03	4.70 ± 0.75	4.45 ± 0.57	4.86 ± 2.21	4.53 ± 0.80
10	7.56 ± 3.28	1.29 ± 0.65	2.69 ± 2.04	0.92 ± 0.47	1.40 ± 1.05

Table 4.15: Average orientation error when the robot is located in the real map of Figure 4.12. Errors in mean ± standard deviation (°).

Robot's Pose	L2	KL	IS	JS	DP
1	0.14 ± 0.11	0.13 ± 0.06	0.16 ± 0.09	0.14 ± 0.07	0.13 ± 0.06
2	0.42 ± 0.26	0.13 ± 0.05	0.37 ± 0.17	0.11 ± 0.06	0.16 ± 0.11
3	0.99 ± 1.03	0.81 ± 0.33	0.58 ± 0.21	0.50 ± 0.38	0.39 ± 0.22
4	1.92 ± 0.77	2.09 ± 0.67	2.49 ± 0.82	2.21 ± 0.88	1.64 ± 0.78
5	0.75 ± 0.70	0.20 ± 0.13	0.36 ± 0.25	0.53 ± 0.50	0.53 ± 0.37
6	0.47 ± 0.32	0.28 ± 0.52	0.12 ± 0.07	0.14 ± 0.09	0.26 ± 0.13
7	0.70 ± 0.72	0.49 ± 0.46	0.62 ± 0.13	0.41 ± 0.12	0.49 ± 0.17
8	0.25 ± 0.20	0.11 ± 0.06	0.12 ± 0.09	0.15 ± 0.08	0.12 ± 0.08
9	0.33 ± 0.47	0.23 ± 0.09	0.7 ± 0.08	0.38 ± 0.48	0.17 ± 0.11
10	0.23 ± 0.17	0.08 ± 0.04	0.19 ± 0.15	0.07 ± 0.06	0.15 ± 0.11

are over 65% for all cases, with no remarkable distinctions between metrics although the L2-norm presents slightly lower values. In terms of average orientation, the estimation is optimal, with very low values of angular error for all cost functions (Table 4.18).

These experiments were performed with a default 2% sensor noise. Another important aspect, as tested in the previous section for 2D environments, is the response of the GL filter to the increase in sensor noise. This is an important quality, as different possible laser range finders utilized for this task may not present a reliable indication in this matter, or the amount of possible noise is highly dependent on the sensor chosen. Following the scheme of section 4.1 this noise is modeled as a Gaussian with deviation σ from the real value. This deviation is increased in the next

Figure 4.13: GL in semi-simulated 3D map. Laser measures in red. Sensor noise:2%.

Table 4.16: Percentage of GL success when the robot is located in the semi-simulated map of Figure 4.13.

	Robot's Pose	L2	KL	IS	JS	DP
1	(100,333,15,0,0,0)	100	100	100	100	100
2	(250,525,10,-4,2,45)	65	93	89	92	85
3	(500,125,20,4,4,90)	85	84	79	87	91
4	(150,500,5,0,3,270)	98	100	100	100	100
5	(95,308,9,-4,-5,0)	100	100	100	100	100
6	(125,380,5,-4,-1,0)	100	100	100	100	100
7	(120,320,8,-3,-2,10)	83	100	100	100	100
8	(520,75,13,-1,3,3)	100	94	100	95	91
9	(122,132,20,1,1,15)	65	88	70	99	100
10	(60,180,23,2,0,0)	100	97	95	100	76

experiment to test the reaction of the different metrics in the DE implementation.

A smaller section of the building map in Figure 4.12 is considered to reduce the computational cost. The location selected is chosen from Table 4.13. The robot is at location $(50, 400, 5, 0, 0, 0)$, a favorable position to reduce the computational cost of this experiment. Figure 4.14 shows the simulated partial map considered and a resulting localization performed with a 15% Gaussian noise level. The influence of this

Table 4.17: Translation GL error when the robot is located in the semi-simulated map of Figure 4.13. Errors in mean ± standard deviation (cm).

Robot's Pose	L2	KL	IS	JS	DP
1	9.38 ± 3.22	0.90 ± 0.77	0.79 ± 0.35	1.44 ± 0.87	1.23 ± 1.04
2	16.89 ± 8.23	5.33 ± 3.76	1.68 ± 1.34	3.00 ± 0.00	1.42 ± 1.33
3	26.67 ± 2.91	1.14 ± 1.04	1.98 ± 1.38	1.88 ± 1.46	1.63 ± 1.25
4	24.72 ± 4.28	1.01 ± 0.69	0.82 ± 0.46	1.07 ± 0.75	1.11 ± 0.88
5	1.50 ± 1.98	0.97 ± 0.95	0.72 ± 0.64	0.83 ± 0.80	1.60 ± 1.12
6	17.10 ± 9.99	0.71 ± 0.44	0.74 ± 0.46	0.73 ± 0.43	0.68 ± 0.31
7	7.79 ± 5.81	0.75 ± 0.54	1.18 ± 0.71	0.88 ± 0.68	0.71 ± 0.46
8	6.61 ± 4.31	7.43 ± 3.72	8.26 ± 4.74	7.40 ± 5.06	9.92 ± 9.26
9	23.68 ± 5.42	8.91 ± 8.99	9.55 ± 5.74	9.43 ± 8.04	9.83 ± 8.58
10	0.76 ± 0.39	4.69 ± 3.41	6.36 ± 4.61	4.98 ± 3.77	5.30 ± 3.49

Table 4.18: Average orientation error when the robot is located in the semi-simulated map of Figure 4.13. Errors in mean ± standard deviation ((°).

Robot's Pose	L2	KL	IS	JS	DP
1	0.40 ± 0.28	0.05 ± 0.04	0.07 ± 0.02	0.10 ± 0.05	0.09 ± 0.07
2	1.68 ± 0.76	0.38 ± 0.27	0.12 ± 0.06	0.11 ± 0.00	0.11 ± 0.09
3	1.50 ± 0.81	1.21 ± 1.77	2.79 ± 1.85	1.97 ± 1.80	2.47 ± 0.95
4	1.06 ± 0.47	0.11 ± 0.07	0.10 ± 0.06	0.11 ± 0.05	0.11 ± 0.07
5	0.09 ± 0.09	0.06 ± 0.05	0.05 ± 0.03	0.05 ± 0.03	0.07 ± 0.05
6	0.20 ± 0.17	0.03 ± 0.01	0.03 ± 0.03	0.04 ± 0.01	0.04 ± 0.02
7	0.50 ± 0.26	0.07 ± 0.05	0.09 ± 0.04	0.07 ± 0.04	0.07 ± 0.05
8	0.46 ± 0.17	0.69 ± 0.37	0.96 ± 0.62	0.62 ± 0.36	1.16 ± 0.91
9	2.74 ± 0.72	2.37 ± 0.87	2.41 ± 0.83	2.05 ± 0.62	2.17 ± 0.72
10	0.13 ± 0.08	0.59 ± 0.29	0.48 ± 0.21	0.68 ± 0.25	0.52 ± 0.19

noise can be observed in the deviation from laser perception shown in the expected measures over the surrounding map. The results of the positioning error and success rate with the increase of this noise from 0-20% are shown in the chart of Figure 4.15. The top of the figure shows the mean positioning error with the deviation for each increase of 1% in Gaussian perception noise, from 0-20%. The different performance between the L2-norm and the rest of the divergences is remarkable. The positioning error rises from close to 0 cm to an approximate value of 9cm for all KL, DP, IS, and JS metrics when a 20% noise level is induced. JS and DP present an almost linear behavior while KL and IS form peaks where the positioning mean error rises up to 10cm. None of the metrics stand out, but it is interesting to remark on the lineal behavior of DP. In contrast, the L2-norm presents a fast impoverishment for

Figure 4.14: GL example in partial map. Sensor noise:15%.

relatively low levels of noise(3-4%) and reaches a maximum of 33 cm. The bottom chart presents the percentage of success where it can be observed that the algorithm finds an acceptable solution in more than 95% of the cases up to 20% level of noise for all divergences. The success rate of RMSE fluctuates between 74-100% in this interval.

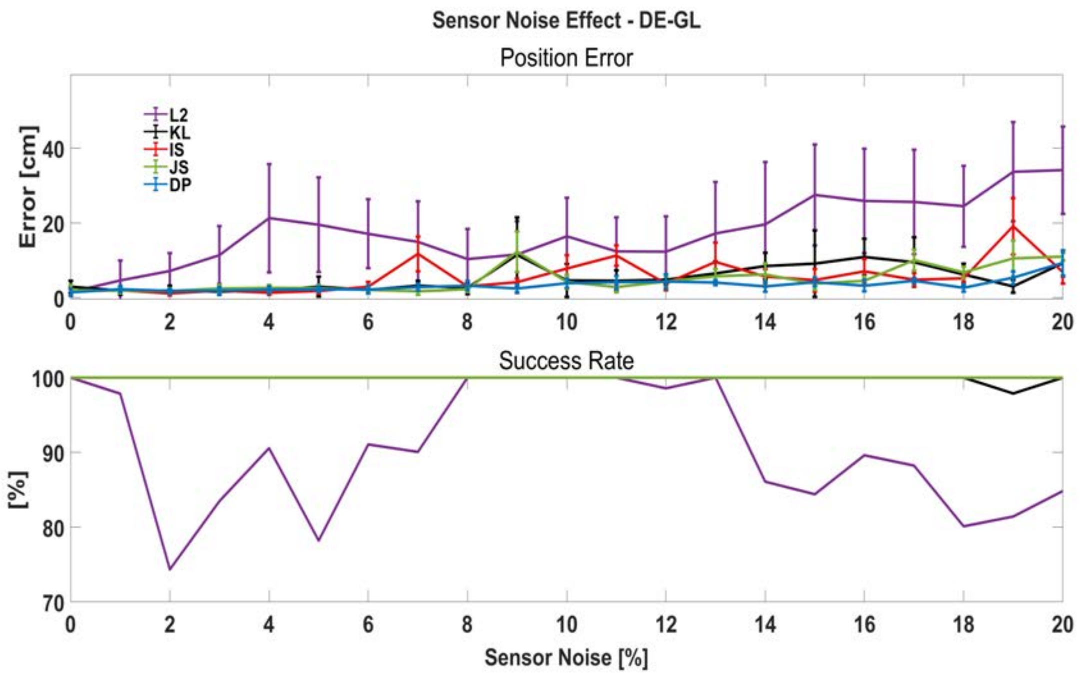

Figure 4.15: Sensor noise influence. Top: position error. Bottom: Success rate.

Comparing the accuracy results obtained with other 3D GL methods the results show that similar if not more accurate results can be obtained with DE-GL filter if considering the divergence-based fitness function. The accuracy achieved by Kümmerle *et al.* is about a few centimeters [18]. Ho *et al.* [41] report a 9 cm and 0.49 ° error in the best case. In [24] a combination of HS and DE algorithms is used for re-localization, the experiments report a minimum error of between [1-5]cm, an experiment showing the response to Gaussian sensor noise where the error rises up to 16 cm for a 30% not considering GL but only pose tracking.

4.2.2 Occlusions: Uniform Noise

Following the casuistry presented in section 4.1 for 2D environments, the performance of the algorithm before uniformly distributed noise is evaluated. These occlusions, as stated, represent the presence of small mobile objects or people in the sight of the laser scan from the real position and are modeled following Equation 4.1. A contamination level (ϵ) is modified to perturb a percentage of the actual measures of the sensor. This reduction is within the range of 25-75% of the actual distance through a uniformly distributed random number $U(0.25_{zk}, 0.75_{zk})$, where z_k is the actual perception of beam k. The considered pose for this test is $(x, y, z, \alpha, \beta, \gamma) = (50, 400, 5, 0, 0, 0)$ in Figure 4.14 and the sensor noise σ is 2%. An example illustrating a comparison between the ideal perception and another with a 25% level of contamination for the considered pose is shown in Figure 4.16. It can be appreciated that the uniform noise causes significant changes in the laser scan.

Figure 4.16: Effect of uniform noise over simulated laser scan. Contaminated measures: 25%. All units in cells.

The study of the influence of the percentage of contamination level (ϵ) is presented in Figure 4.17. The horizontal axis corresponds to the increasing percentage of contaminated measurements. The top chart presents the positioning error for all the different fitness functions. In the bottom chart the success rate can be analyzed.

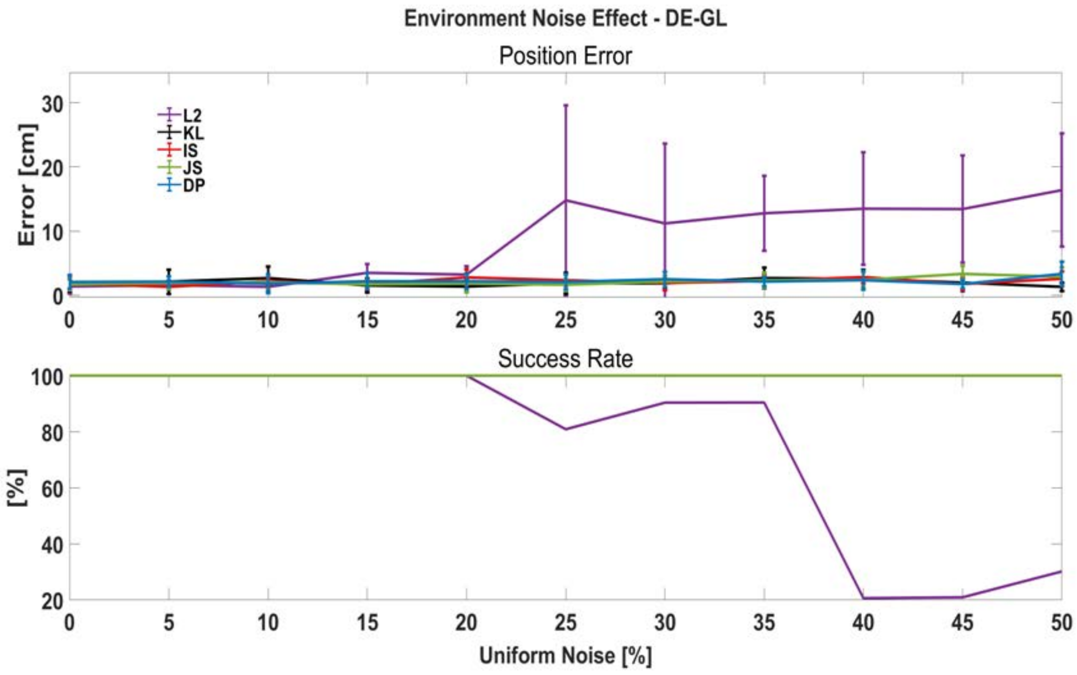

Figure 4.17: Top: Position error vs. uniform noise (percentage of contaminated measurements) in the simulated map of Figure 4.16. Bottom: Success rate.

A remarkable performance of the divergence-based fitness functions in contrast with the L2-norm has been noticed. L2 commences to be affected from a 20% of occluded measurements, and the error increases from a range of [1-3]cm to [10-17]cm. The percentage of accurate locations within our specified threshold is slightly affected in the interval from 20-30% of contamination levels and falls into a 20% localization rate from 40% contamination. On the other hand, this type of occlusion does not affect probability-based fitness functions. The percentage of success in a favorable position (a singular location on the map) remains optimum, and the positioning error remains constant in an interval of [0.9-2.5]cm. When considering the same experiment on the semi-simulated environment of Figure 4.13 equivalent response is obtained. The position selected is $(x, y, z, \alpha, \beta, \gamma) = (150, 500, 5, 0, 3, 270)$ and the chart is omitted for simplification. Again, divergence-based evaluation stands out before the L2-norm in terms of GL error and percentage of success. Contamination levels of $\epsilon > 40\%$ significantly decrease success rate when applying the L2-norm.

4.2.3 Occlusions: Unmodeled Obstacles

Analogous situations to those presented for 2D environments are considered regarding unmodeled obstacle presence in 3D occupancy grid maps. Two different experiments are implemented to provoke laser beam occlusions. A single remarkable obstacle and multiple obstacles. The objective is to test if a distinguishable behavior can be extracted from the source of occlusions. In the first consideration, a single obstacle presence is tested at different distances from the robot's location. The percentage of occluded measures increases proportionally as the distance from the obstacle is reduced. In a second experiment, multiple obstacles are placed around the robot, and the percentage of occluded laser beams increases with the quantity of these obstacles. Both situations are presented in Figure 4.18.

Figure 4.18: Unmodeled obstacle possibilities. Left: Single. Right: Multiple. Location of the robot: (50,400,5,0,0,0) in the map of Figure 4.14.

The results of these two experiments are presented in Figures 4.19 and 4.20. Similar conclusions can be extracted depending on the metric implemented when comparing both situations. In both cases, the success rate of the L2-norm moves away from the optimum 100% when any unmodeled occlusion is introduced. This behavior can be extended to any type of perturbation analyzed in this section. For 4-6% of occlusions, RMSE performance falls under the 50% of success, and the influence of a single obstacle is greater than multiple occlusions. The DE-based method using RMSE is unable to localize the robot when more than 6% of the measurements are occluded by a single obstacle or more than 10% by multiple intrusions. Regarding the divergence-based metrics, the performance is considerably better, but it can be

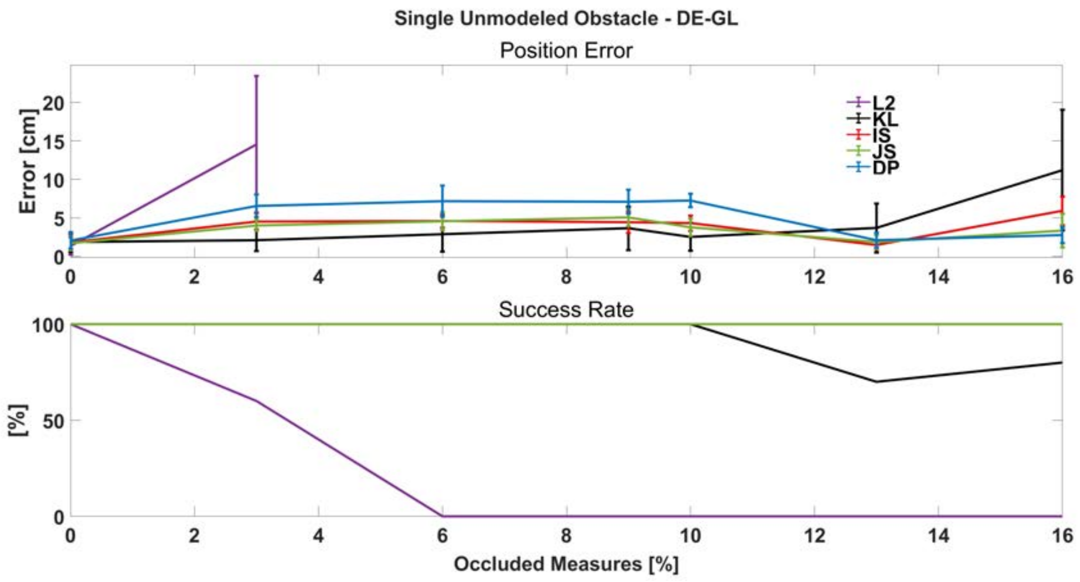

Figure 4.19: Unmodeled obstacle response. Top: Position error vs. % of occluded measurements in the simulated map of Figure 4.14. Bottom: Success rate.

extracted that this type of occlusions affects the performance in a more significant manner than the sensor and uniform noise. In contrast with the L2-norm, divergences' performance is more affected by multiple obstacles. In terms of success rate, only KL is slightly affected by single-source occlusions. The rest remain at a 100% success rate until 16% of occluded measures. Analyzing the top chart of Figure 4.19, the positioning error for a single unmodeled obstacle increases from values close to 0 until 10cm using KL, while the rest of the divergences remain in a [0-5]cm range and a 100% localization rate.

For multiple unmodeled sources (Figure 4.20), the behavior of DP and JS in this experiment was remarkably better. The success rate is maintained until a 30% of occlusions, and the positioning error stays within an interval of [0-10]cm. The worst results among divergences correspond to the KL-divergence, with a fast increase of the positioning error up to 40cm with a 30% of occlusions. IS in an intermediate position maintains a relatively low positioning error, comparable to DP and JS, but instead fails to localize the robot at a 30% occlusion rate.

It must be considered that these kinds of tests are not simple to perform as there are many variable aspects that can modify the outcome. The location of the robot within the map but, more important, the location and orientation of the obstacle(s). Occluding distinctive sections of the map may be more transcendent than the % of occlusion.

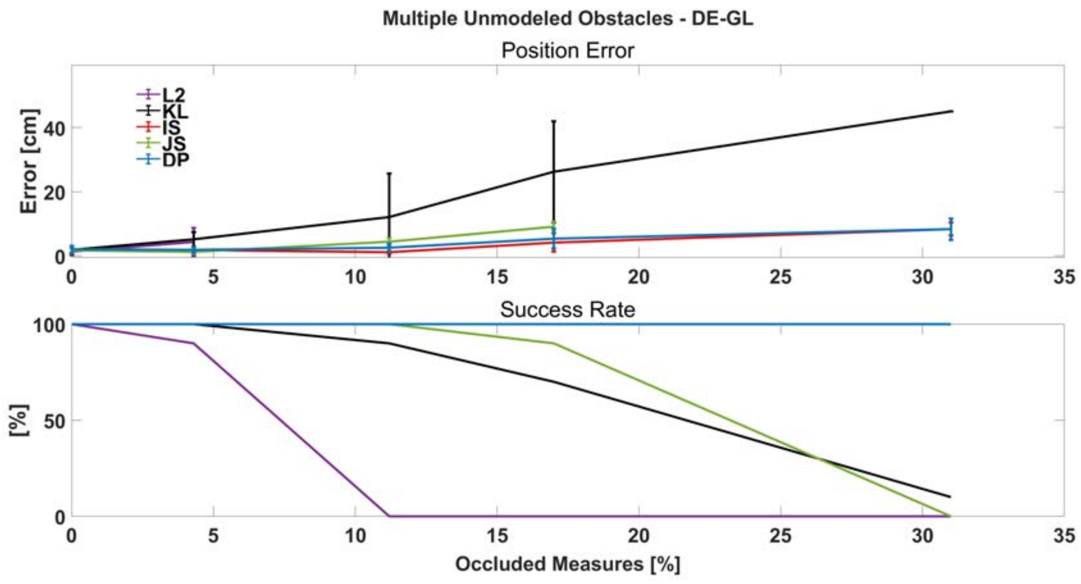

Figure 4.20: Unmodeled multi-obstacle response. Top: Position error vs. % of occluded measurements in the simulated map of Figure 4.14. Bottom: Success rate.

4.2.4 Population requirements and computational cost

A simple study of the population requirements and the consequent computational cost has been carried out over the map representation in Figure 4.12. The objective of this test is to clarify if for a certain set of considered positions the positioning error and the success rate are directly related to the population number.

The resulting comparison is shown in Figure 4.21. In this experiment, the number of candidates considered in the DE-GL filter developed is increased from 10 to 450 members. The top chart represents the mean positioning error considering only successful localizations. As it can be extracted from the results, this error is not dependent on the population number. Once the algorithm has converged into the global minimum, the accuracy obtained does not depend on the number of candidates. On the other hand, the bottom chart shows the success rate for each number of particles. Extracted from this experiment, the maximum localization rate tends to saturate at a given number, and increasing the population does not lead to a more reliable outcome. In this particular case, a population size of over 300-350 candidates does not improve the percentage of success significantly. The computational cost and the time needed to obtain an optimum solution increase proportionally with the amount of population. The middle chart of Figure 4.21 shows this situation. It is interesting to highlight that this computational cost remains approximately constant in the interval between 300-450 candidates. This is due to the fact that the convergence

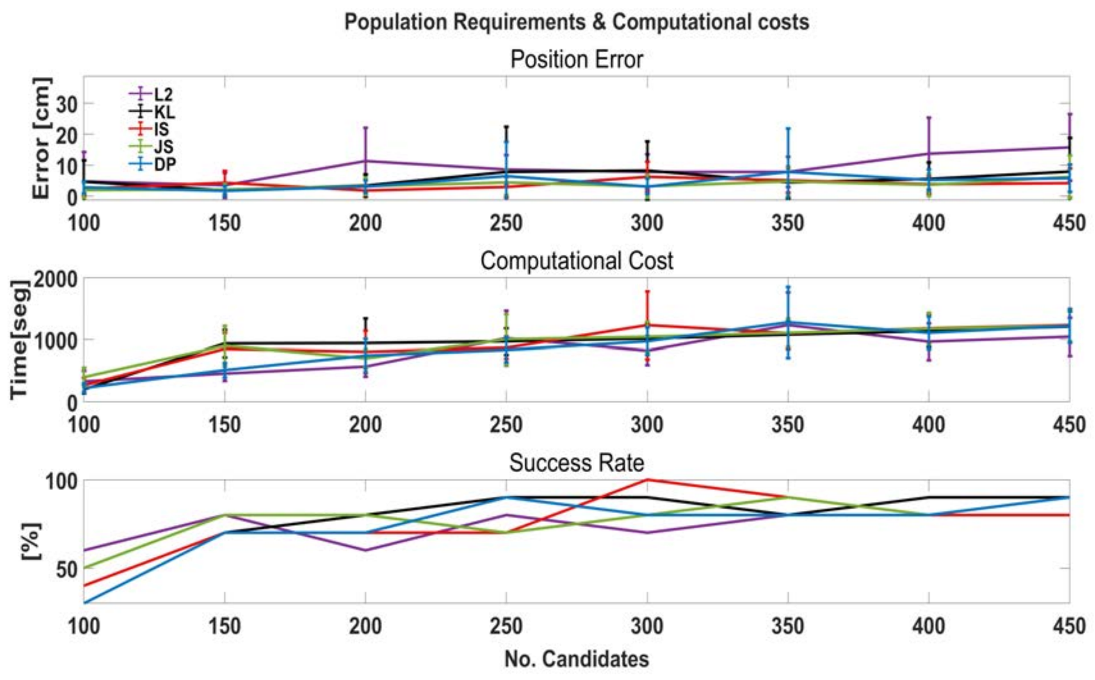

Figure 4.21: Performance and computational time response to increasing population.

conditions described in section 3.3 terminate the stochastic search at a lower number of iterations with a higher population number.

4.3 3D Point Cloud Environments

This third and last section presents the experimental results for Evolutionary-based GL filters. In this last case, Point Cloud-based maps are considered. As described in Chapter 3, these environments were built from 3D LiDAR real data using the ADAM platform (1.1) developed in the Robotics Lab at Universidad Carlos III de Madrid. The slam technique implemented is described in [72] and [152]. Different Meta-heuristics were applied for PC Scan Matching in this work to correct the odometry information provided by the encoders during the SLAM process. In this section, the resulting map from a Harmony-Search-ICP method [152] is considered as the global known map in which the mobile robot is localized. This map represents the DISA department and covers an approximate volume of 40x20x4=3200m^3. The environment representation is shown in Figure 4.22. It should be highlighted that this representation is a horizontal cut parallel to the xy plane for a clearer interpretation. The complete map includes the ceiling and floor.

Three different Metaheuristic approaches have been tested in this environment,

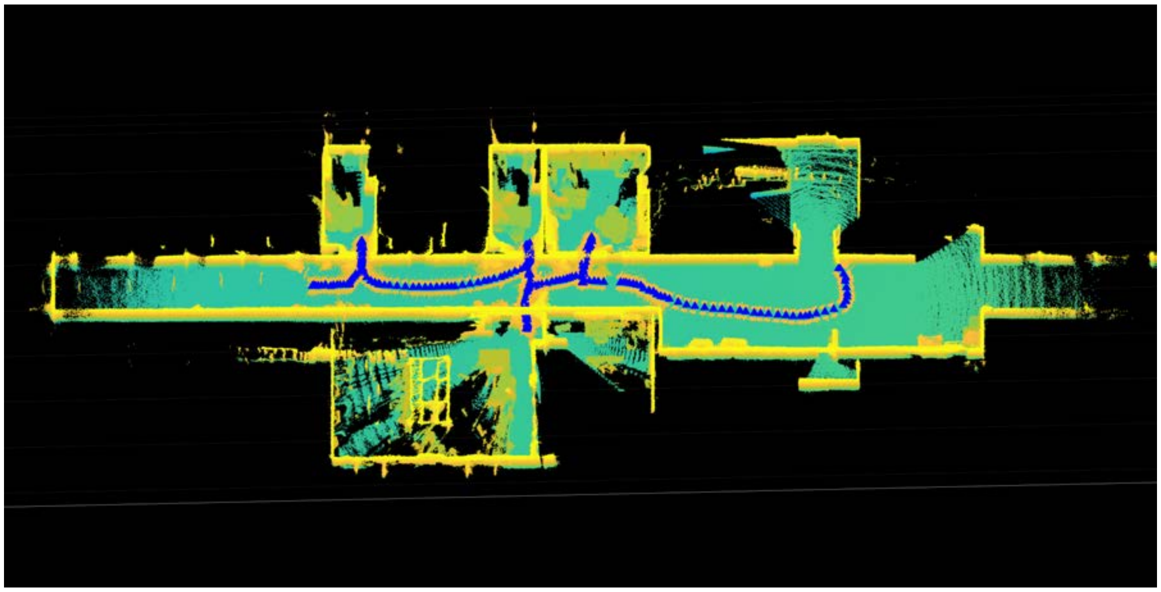

Figure 4.22: PC-based map. Blue triangles indicate the location of the mobile robot during the mapping process.

DE, PSO, and IWO optimization. A comparative of the performance of these GL-filters when considering the different metrics (L2, KL, IS, JS) is presented following homologous scenarios to previous sections 4.1 and 4.2, regarding the accuracy, the influence of sensor and environmental uniform noise and unmodeled obstacles. The objective is to estimate the correct position and orientation of the robot covering the possible 6DOF of the three-dimensional space $(x, y, z, \alpha, \beta, \gamma)$. As in previous sections, a successful outcome is considered when the difference between the estimated and real position is less than 50cm. The sensor noise is modeled as a Gaussian distribution over the real distance measured with standard deviation $\sigma = 2\%$ when another value is not specified.

Configuration parameters for all stages presented in sections 3.2.2 and 3.3 are:

- Pre-processing: Global cloud and local scan are reduced to a 0.5% and 1% of original points respectively. Random downsampling is applied

- DE parameters: $F = 0.9, CR = 0.75$. Discarding: worst 5% substituted by best 20%. Thresholding: new members must reduce the cost value below 98% of its competitors.

- PSO parameters: $w = 1, w_{damp} = 0.99, c_1 = 1.25, c_2 = 1.75, VelMax_i = 0.1D_i$ where D_i is each dimension of the global map.

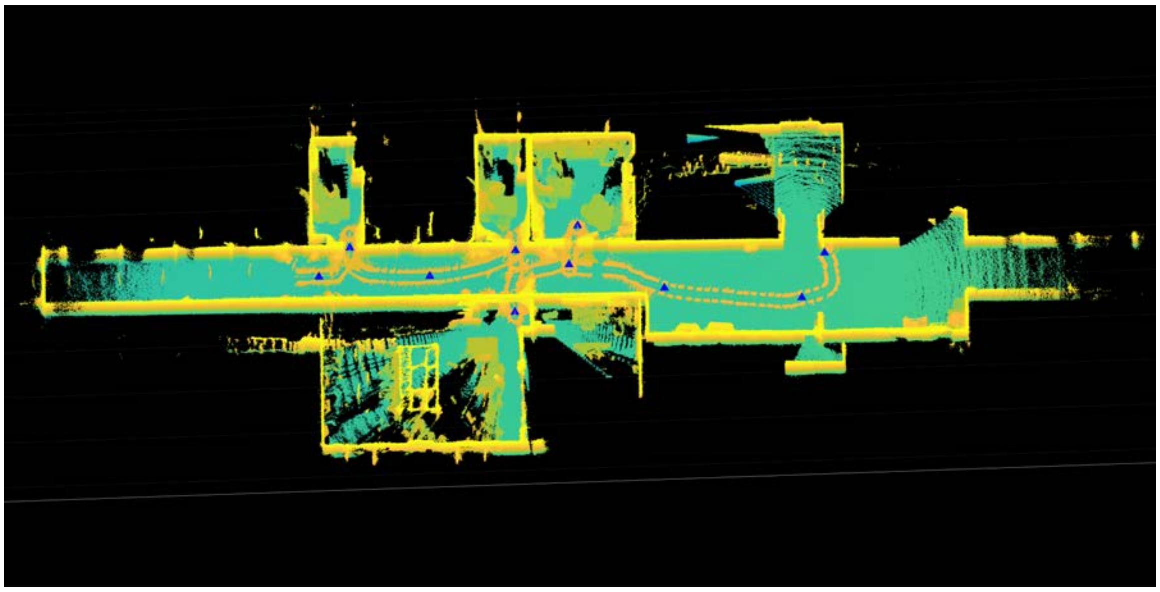

Figure 4.23: Selected poses for accuracy test.

- IWO parameters: $S_{max} = 6, S_{min} = 2, \sigma_{ini} = 0.8, \sigma_{fin} = 0.1, exp = 2, N_{Pini} = N_{Pfin}/2$
- Convergence conditions:
  - Maximum iterations: 500.
  - Total convergence: Best cost= Avg.Cost=Worst Cost. Minumum iterations:50.
  - Normal convergence: Avg.Cost/Best Cost < 1.05 and Worst Cost/Best cost < 1.05
  - Invariant: No changes in best, average or worst cost for 10 consecutive iterations.

4.3.1 Performance and Accuracy

The first set of experiments is addressed to test the accuracy of the implemented methods in solving the GL task over sparse PC-based maps. The results for ten different positions are presented as an extract of the evaluated localization tests. These selected locations can be appreciated in Figure 4.23. These positions have been tested for every combination of Meta-heuristic (DE, PSO, and IWO) and fitness function implementation (L2, KL, IS, JS) presented in Chapter 3. The data extracted from the SLAM solution in [152] contains the global map, the individual scans that are used to build that map, and the absolute positions of each scan in reference to the initial position of the followed path (Figure 4.22). Therefore the actual position of the robot within that map is known for each individual scan considered and will serve as a reference to evaluate a successful localization. As in previous sections, GL is achieved if the distance between actual and estimated positions is lower than 50cm.

Table 4.19: Percentage of GL success when the robot is located in the real map of Figure 4.23.

	DE-GL					PSO-GL				
Robot's Pose	Iter.	L2	KL	IS	JS	Iter.	L2	KL	IS	JS
1	71	100	100	100	100	95	100	100	100	100
2	62	100	77	85	100	110	100	100	100	100
3	87	96	100	100	100	86	88	93	94	91
4	64	74	69	86	84	92	95	100	100	100
5	73	76	86	78	97	91	100	100	100	100
6	52	73	98	61	100	98	100	100	99	100
7	56	95	100	100	100	93	82	87	92	82
8	50	61	59	54	63	109	79	100	100	85
9	59	49	47	62	45	132	85	80	88	74
10	73	63	65	63	61	129	66	79	95	100
	IWO-GL									
Robot's Pose	Iter	L2	KL	IS	JS					
1	380	100	100	100	100					
2	388	100	100	100	100					
3	390	100	100	100	100					
4	373	94	100	96	100					
5	401	83	100	92	100					
6	350	94	100	87	100					
7	391	82	86	100	100					
8	260	56	98	39	90					
9	304	63	55	52	55					
10	410	55	67	59	59					

Qualitative results are shown in Table 4.19 for the different poses. These locations are indexed from 1-10 in the table, referring to the positions (blue triangles) from left to right in Figure 4.23. The performance of each EA for all the different metrics is shown in this table, DE-Gl on the top left corner, PSO on the top right corner, and IWO on the bottom left side. The number of iterations represented in this table (Iter.) specifies the average number of iterations necessary for each algorithm to converge into an optimal solution for each location. Unsuccessful localizations are excluded, and the average is calculated for all tests for all cost functions. No metric shows any remarkable difference in this aspect.

Several conclusions can be extracted from Table 4.19. Regarding the convergence velocity, DE shows the fastest results with a minimum number of iterations for every position considered, a 30% faster than PSO and 80% faster than IWO (in terms of No. of iterations). On the contrary, it shows a weak performance in terms of success rate, with acceptable values over 70% of successful estimations for the majority of positions evaluated. The parameters selected to tune the DE implementation are set to improve the inherent global search qualities of this algorithm. A variable mutation factor F has been implemented, decreasing from 0.9-to 0.3 depending on the best/worst and best/average cost ratio of the population. This implementation may have caused premature convergence in some situations. The success rate using PSO and IWO GL-filters is higher and considerably better in the case of PSO, with a localization ratio over 75% for all cases considered. The iterations needed for IWO-based localization are remarkably higher in contrast with DE and PSO. This outcome was expected as IWO exploits the local search by spreading a localized amount of seeds within a distance of each plant. Further analysis of population requirements and computational costs is presented later in this section.

Comparing the different fitness functions implemented, in general terms, KL and IS present a higher success rate for the three different algorithms.

An important consideration regarding the locations chosen on the map of Figure 4.23 must be made. The final positions of the chosen set present worse results for every cost function and every algorithm. By observing this figure, these locations correspond to non-singular positions in open spaces, the sensor noise can affect the laser measurements to a greater extent, and the basin of attraction of the global minimum in that region is low. A tracking experiment was performed in that region to explore this situation. Results are presented later in this section.

Regarding the positioning error, the results of Table 4.20 were obtained with the different implementations. In a first general conclusion, regardless of the optimization algorithm or the fitness function utilized, the average positioning error is in an interval between [0.5-11.82]cm for all positions considered. This range is accurate enough to consider that Evolutionary-based GL filters solve the GL issue in sparse PC-based map representations. In a deeper analysis, the most accurate results are obtained using the IWO implementation, with the lowest error in 5 out of 10 positions, although the difference is not outstanding. This could be expected as if converged, IWO exploits the local search around the minimum deeper than the competitors. In contrast, computational costs are elevated. All different metrics present similar results, and no behavior pattern can be established. L2 tends to present minimum errors, but the difference is not conclusive.

Table 4.20: Translation GL error when the robot is located in the real map of Figure 4.23. Errors in mean ± standard deviation (cm).

DE-GL

Robot's Pose	L2	KL	IS	JS
1	8.40 ± 9.45	11.82 ± 7.24	11.77 ± 5.59	3.48 ± 4.95
2	1.12 ± 0.42	9.24 ± 6.66	5.69 ± 4.91	2.51 ± 3.66
3	7.62 ± 6.35	1.87 ± 1.79	4.17 ± 2.12	5.52 ± 3.04
4	4.51 ± 6.65	5.96 ± 4.77	4.30 ± 2.06	6.14 ± 11.75
5	4.05 ± 4.53	5.73 ± 1.16	6.48 ± 5.72	6.01 ± 1.45
6	2.17 ± 1.04	3.90 ± 6.58	3.16 ± 2.21	1.38 ± 0.69
7	4.91 ± 3.25	8.78 ± 2.19	7.77 ± 3.46	7.45 ± 8.93
8	0.88 ± 0.61	5.51 ± 1.47	5.59 ± 1.79	3.64 ± 6.38
9	2.28 ± 0.78	3.83 ± 1.22	15.54 ± 15.16	5.08 ± 8.72
10	1.16 ± 0.54	11.67 ± 14.95	10.19 ± 1.86	2.55 ± 3.83

PSO-GL

Robot's Pose	L2	KL	IS	JS
1	1.93 ± 0.86	2.55 ± 0.82	2.08 ± 1.07	3.64 ± 2.11
2	1.21 ± 0.57	6.26 ± 2.81	3.94 ± 1.77	2.47 ± 1.36
3	3.7 ± 4.21	3.35 ± 2.02	3.45 ± 2.76	5.76 ± 1.59
4	8.63 ± 15.14	5.78 ± 1.64	4.58 ± 2.34	5.46 ± 2.97
5	6.11 ± 0.9	5.16 ± 1.42	4.62 ± 2.53	5.54 ± 2.12
6	1.41 ± 0.52	3.36 ± 1.89	4.47 ± 1.88	3.93 ± 2.32
7	8.65 ± 8.47	8.87 ± 1.86	9.97 ± 3.7	9.23 ± 1.41
8	0.67 ± 0.22	5.14 ± 0.61	4.18 ± 0.65	2.72 ± 1.15
9	0.95 ± 0.63	3.21 ± 1.14	3.27 ± 0.74	3.23 ± 1.43
10	2.06 ± 3.57	5.12 ± 3.98	5.3 ± 3.22	4.13 ± 3.29

IWO-GL

Robot's Pose	L2	KL	IS	JS
1	2.34 ± 1.22	5.25 ± 2.98	7.12 ± 2.17	9.11 ± 4.63
2	1.73 ± 0.73	5.31 ± 1.90	3.86 ± 1.97	6.01 ± 2.03
3	4.76 ± 3.30	1.90 ± 1.47	3.91 ± 1.26	3.62 ± 1.59
4	2.16 ± 1.23	4.25 ± 1.92	4.83 ± 2.15	7.30 ± 3.33
5	6.55 ± 0.95	4.94 ± 1.03	6.79 ± 3.34	5.99 ± 2.36
6	1.78 ± 0.89	1.85 ± 0.86	3.34 ± 2.44	3.34 ± 1.84
7	6.63 ± 10.42	4.24 ± 2.32	8.80 ± 3.49	6.50 ± 3.19
8	0.96 ± 0.21	3.73 ± 0.77	5.00 ± 1.22	6.12 ± 3.37
9	4.56 ± 3.99	3.13 ± 0.88	6.12 ± 0.62	5.99 ± 3.29
10	0.85 ± 0.36	7.34 ± 1.51	7.30 ± 3.11	8.13 ± 3.94

Table 4.21: DE-GL. Average orientation error when the robot is located in the real map of Figure 4.23. Errors in mean ± standard deviation (°).

Robot's Pose	L2	KL	IS	JS
1	0.84 ± 0.67	0.31 ± 0.25	0.34 ± 0.28	0.93 ± 0.39
2	0.80 ± 0.25	1.83 ± 0.47	1.37 ± 0.72	1.36 ± 0.42
3	1.41 ± 1.04	1.10 ± 0.29	0.98 ± 0.36	1.02 ± 0.27
4	0.82 ± 0.55	1.13 ± 0.65	0.87 ± 0.31	1.13 ± 1.12
5	1.40 ± 0.55	1.10 ± 0.90	1.26 ± 0.83	1.04 ± 0.51
6	1.69 ± 1.32	1.10 ± 0.69	1.46 ± 0.47	0.72 ± 0.42
7	1.06 ± 0.54	1.08 ± 0.47	1.00 ± 0.36	0.99 ± 0.18
8	0.35 ± 0.19	1.75 ± 1.57	1.09 ± 1.05	0.95 ± 1.27
9	0.95 ± 0.93	1.43 ± 1.06	1.64 ± 1.87	1.28 ± 1.13
10	0.40 ± 0.30	1.28 ± 1.14	1.09 ± 1.61	0.43 ± 0.28

The average orientation error is summarized in Table 4.21 showing the average error of the three different Euler angles. As no singular behavior was denoted by any specific algorithm or metric implemented and no remarkable differences are encountered between different angles α, β or γ, no further analysis is presented to simplify the results. Equivalent results are shown by PSO and IWO. The orientation error is close to zero for all positions and fluctuates in an optimum interval between [0.27-1.87]°.

The systems' response to the presence of sensor noise is also evaluated using the same Gaussian model as in previous sections. An increasing level of noise is considered, rising from 0-20% in the laser beam direction, modifying the local scan PC. Position number 4 on the map of Figure 4.23 was selected to perform these tests for all algorithms.

Figure 4.24 shows the results when the DE-GL implementation is utilized. The position error is evaluated in the top chart. As expected, the average error increases in an approximate linear behavior with the level of Gaussian noise. In terms of error, the divergences' behavior is similar. The error interval is between [0-16]cm. Regarding orientation, on the middle chart, the same conclusion is extracted. The difference of angular error between metrics for any given noise level is no more than 1-2°. The orientation error is the interval between [0-4]°. Important conclusions are drawn as the behavior of the different metrics diverges from occupancy grid representations. The success rate of divergence-based evaluation functions is much more dependent on sensor noise than the L2-norm, which maintains more than a 50% percentage of success at high levels of noise (18%) while divergence-based functions fail at 12-14%. JS divergence presents the worst performance regarding success rate, failing at an 8% level and higher. Comparing these results with those in Figure 4.15, the performance of the L2-norm maintains similar values compared to 3D occupancy grid maps. The

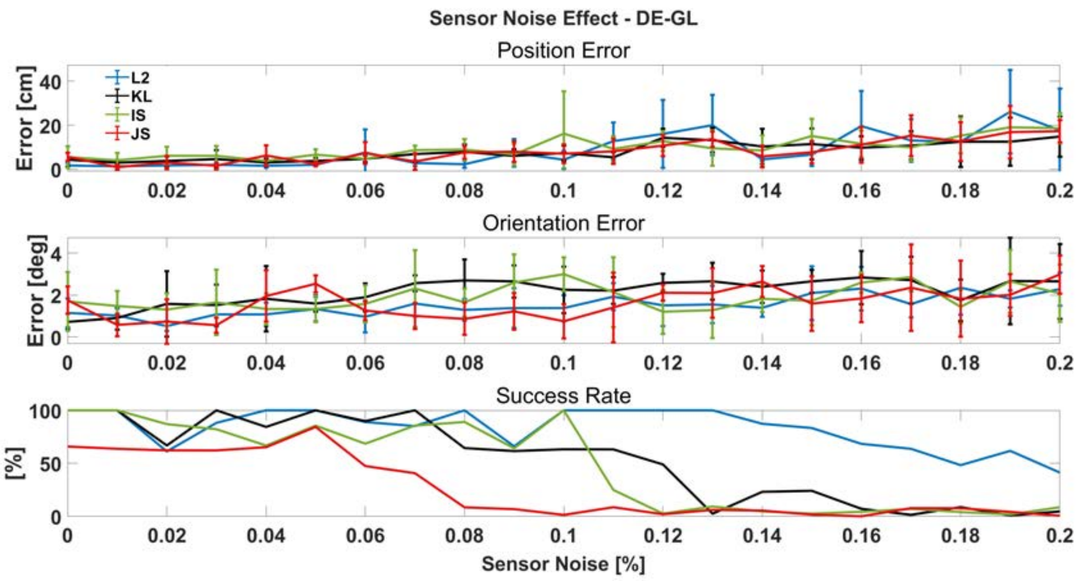

Figure 4.24: DE-GL filter response to increasing sensor noise.

divergences, in contrast, depend on the spatial relation between individual laser beams and this quality is not exploited in sparse scan-matching based approach.

The next implementation considered is PSO-GL. The results of the influence of increasing senor noise are shown in Figure 4.25. The outcome of this experiment in terms of positioning error is similar to the one obtained using DE-GL. The average error increases linearly in a [0-18]cm. The same behavior is observed in the orientation error but with a smaller maximum deviation of 2.8 ° from the actual orientation. In both values, the response of the L2 norm presents a more constant result compared with divergences, with a maximum error of 5cm and 1.9° for a 20% sensor noise.

From a qualitative point of view, the success rate presented by the L2 norm outstands the rest of the metrics, and, as occurred with DE-GL, divergences success rate falls under a 50% from a 12% noise level while L2 maintains an almost optimal performance up to 17% noise, with a minimum ratio of 60%.

Finally, results when applying the IWO-GL filter are shown in Figure 4.26. The same conclusions extracted for the PSO method could be applied in this case. The position error on top of the figure shows a linear response on the interval from [0-20]cm and an orientation error between [0-3.7]°. The L2 again implies a lower error in position and orientation in most noise intervals, although the difference with the rest of the metrics is lower than in the case of PSO. The success rate shows the same behavior. The L2-norm locates the robot 90% of the time with 17% noise, while the rest of the divergences fail completely from a 14%.

These results show accurate location for all GL-filters considered, with better

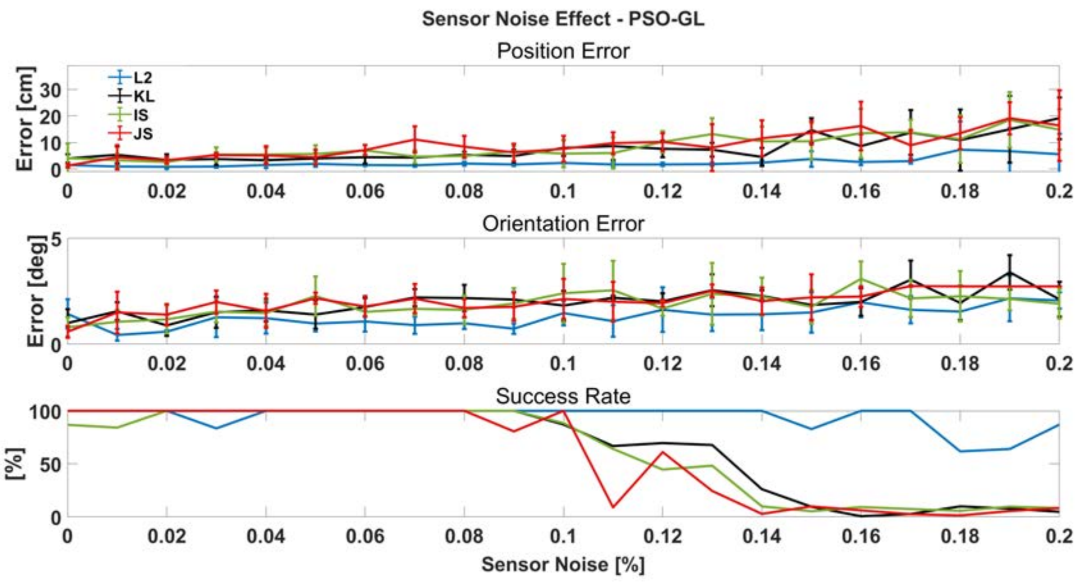

Figure 4.25: PSO-GL filter response to increasing sensor noise.

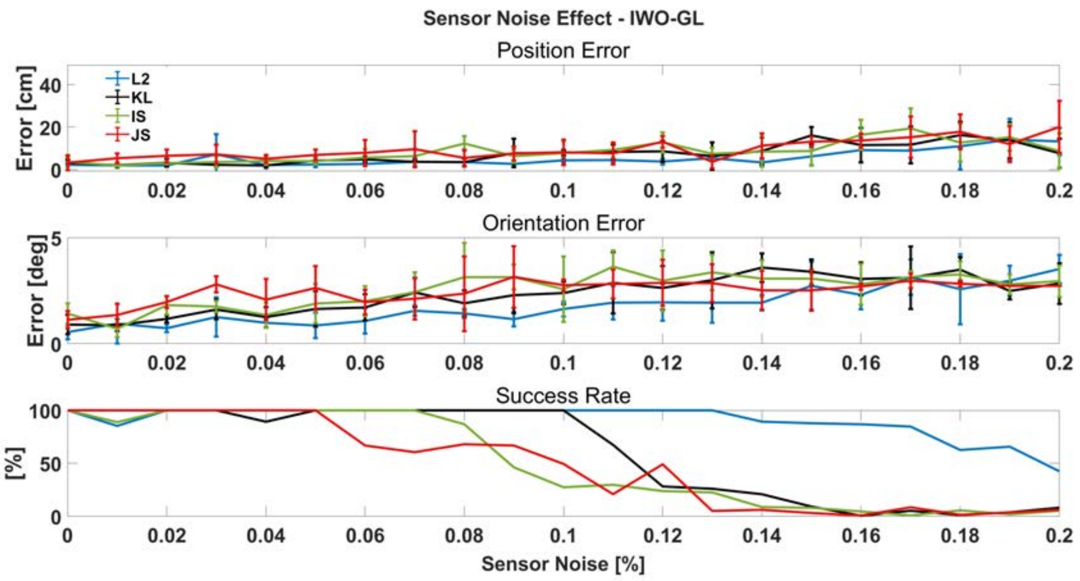

Figure 4.26: IWO-GL filter response to increasing sensor noise.

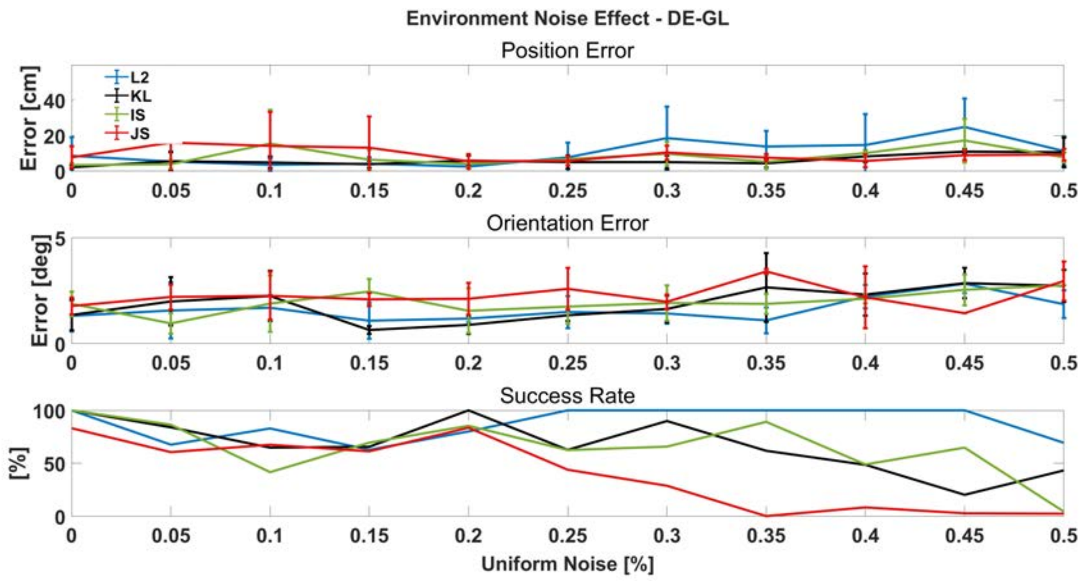

Figure 4.27: DE-GL filter response to increasing uniform noise.

qualitative results using PSO and IWO. In a comparison with other researches it can be concluded that our method shows similar translation and orientation errors as ICP-based Scan Matching techniques, which is meritorious since these techniques use a smaller search space. In [152], Gonzalez-Prieto *et al.* report an error between [0-6]cm for a HS-based Scan Matching. Vision based techniques like [17] also fluctuate in a 0-10cm accuracy. A similar size PC environment is considered in [153], a feature-based ICP algorithm is proposed as the global/local pair matching. Results show and average error around 12cm.

4.3.2 Occlusions: Uniform Noise

This section presents the experiments regarding the inclusion of environment noise, representing small mobile obstacles or people. These unmodeled occlusions are considered small enough to model them as random contamination of measures between 25-75% of the actual measure. This was explained in sections 4.1 and 4.2, but as a reminder, the response against the increase of a contamination level ϵ is evaluated. This parameter represents the percentage of actual measures that are reduced to a lower distance from the reference location. This reduction is performed through a random number over the uniform distribution $U(0.25_{zk}, 0.75_{zk})$, where z_k is the actual distance of point k in the local PC.

Results of this effect when considering the DE-based GL filter are shown in Figure 4.27 including position error, orientation error, and success rate from top to bottom

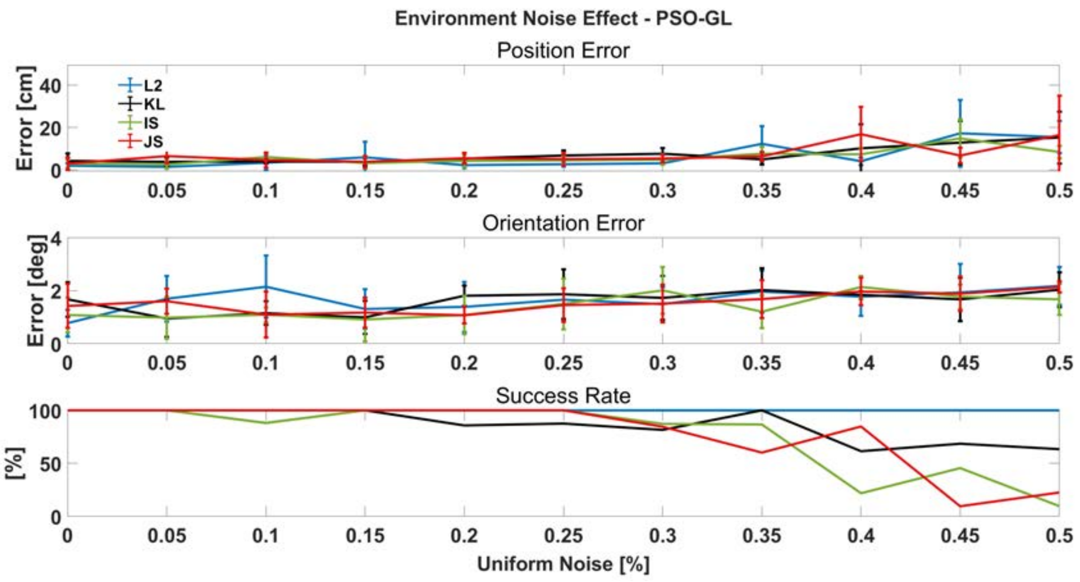

Figure 4.28: PSO-GL filter response to increasing uniform noise.

charts. The abscissa axis represents the percentage of points of the local scan affected by this noise. In this case, all metrics present a similar behavior in terms of positioning and orientation error. JS divergence presents a worse performance at low levels (5-20%) while L2 provides the highest error at high levels (30-50%). The position error is among the interval between [0-18]cm for the rest of the metrics. The orientation error fluctuates between [1-4]°. When considering the successful localization ratio, the L2 norm presents optimum values for almost the entire interval. KL and IS divergences decrease to a 50% rate at 40% of contamination level while JS at a 25%.

Figure 4.28 introduces the results when applying PSO-based localization. This method presents the best response in this situation with a constant optimum value in position, orientation, and success rate for all fitness functions up to 30% of contaminated measurements. The error in that interval is below 5cm and 2° with more than a 90% success in all cases. From 30-50% of occluded measurements, the L2-norm shows a steady 100% localization rate while the divergences decrease to a 50% around 35-40%.

Finally, IWO-GL shows a similar reaction to uniformly distributed noise when compared with DE. The response is almost optimal for all metrics until a 15% noise, with low positioning (0-6cm) and orientation (0-3°) errors. The success rate stays between 90-100% except when applying IS divergence, which shows a poor performance at 10%. The evolution for higher contamination levels evolves differently depending on the chosen function. JS and IS fall into a 50% of successful localizations with a 25% of contamination level, together with L2. KL maintains a good ratio until 35%.

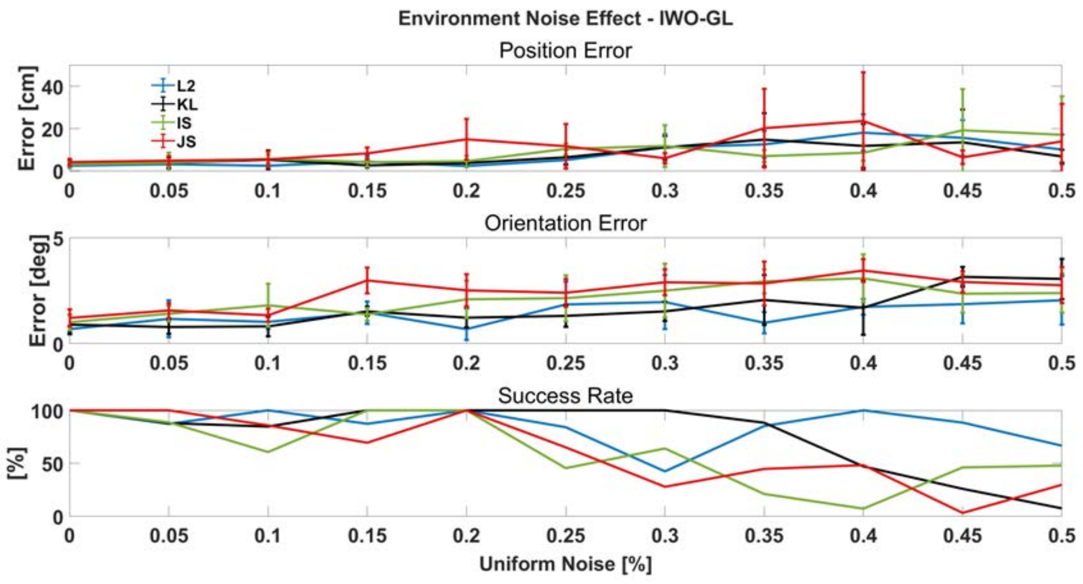

Figure 4.29: IWO-GL filter response to increasing uniform noise.

In terms of error, except for JS, all metrics show similar behavior.

Regarding the implementation of probability-based fitness functions for PC-based environments, the results show that if this problem is approached as a Scan Matching solution, probability profiles do not suppose a drastic improvement as in occupancy maps. The difficulty in establishing a spatial relation for each point of the local laser scan with a point in the global map prevents the weighted divergence-based fitness function from exploiting its capabilities. A quadratic consideration addresses the optimization problem in a more accurate manner.

4.3.3 Occlusions: Unmodeled Obstacles

This section presents the results for the implemented methods regarding GL before the presence of unmodeled obstacles. These occlusions are simulated by reducing the distance of a percentage of contiguous points of the local scan considered, forming groups in various directions surrounding the laser sensor. Figure 4.30 illustrates this concept. On the left side of the figure, an example of successful localization of the selected position is shown. The red dots indicate the local point cloud. On the middle and right part of the image, localization before the presence of unmodeled obstacles is presented. Notice the grouping of red points surrounding the robot forming unmodeled occlusions.

Figure 4.32 shows the results when applying DE-based GL. The position error (top chart of the figure) increases proportionally with the number of points occluded

Figure 4.30: Multiple obstacle occlusions. From left to right: 0%, 15% and 20% points occluded.

by the obstacles. In this case, the L2-norm presents the best qualitative results, with a 100% success rate regardless of the occlusions. In contrast, the positioning error is higher compared with divergence-based functions, especially in the interval from 25-35% occlusions. Probability-based fitness functions are more affected in terms of localization rate, but the resulting error is lower.

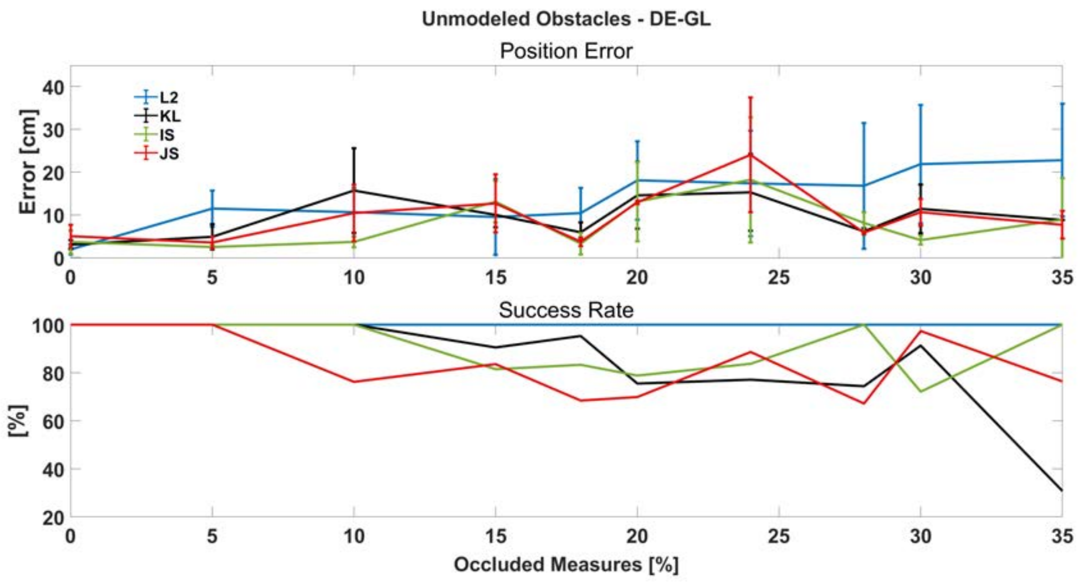

Figure 4.32: DE-GL response to multiple unmodeled occlusions.

Observing Figure 4.31, similar behavior can be concluded when the PSO algorithm is applied. In this case, the difference between L2 and KL, IS, and JS is most notable.

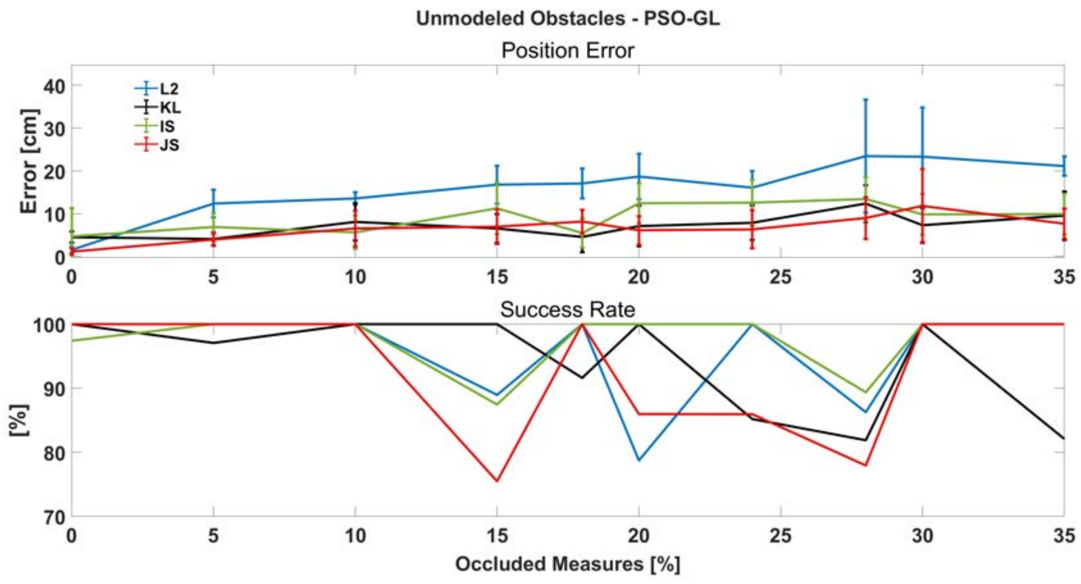

Figure 4.31: PSO-GL response to multiple unmodeled occlusions.

The positioning error is between 5-10 cm higher in regard to divergences. The success rate is similar for all metrics, and no conclusions can be extracted. Using PSO, the GL percentage remains over 75% for all the intervals of occlusions considered.

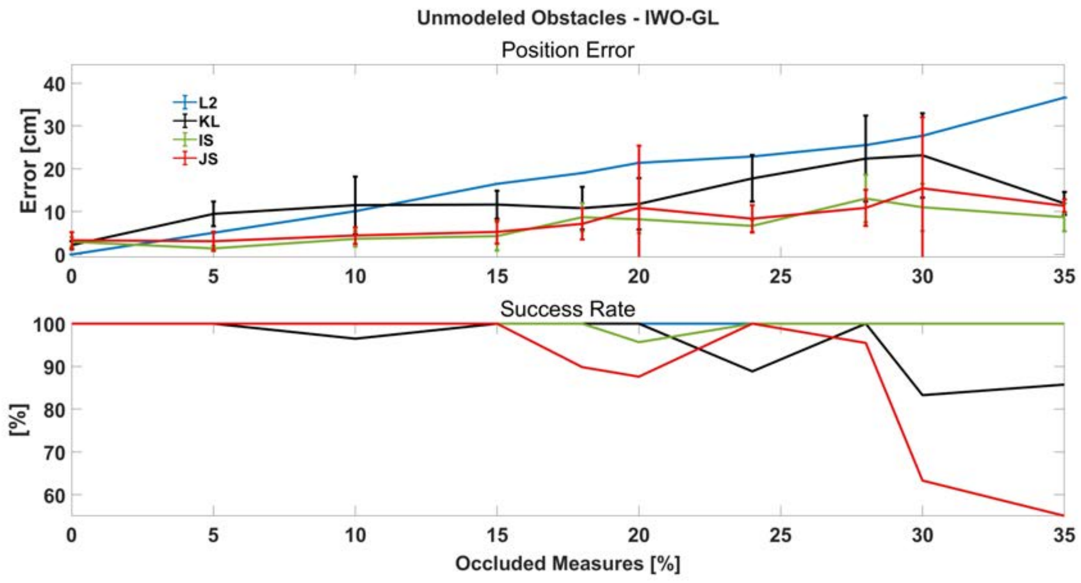

Figure 4.33: IWO-GL response to multiple unmodeled occlusion.

The last test was performed using the IWO-based filter. In this case, translation error remains in similar intervals as previous methods, within 0-18cm, except when the L2-norm is applied to the fitness function. The positioning error using this metric increases exponentially up to 35 cm. In contrast, a 100% success rate is obtained when divergence-based functions start to fail with a percentage of occlusion of 25-30% except for IS, which stays at the maximum level for all the tests performed.

4.3.4 Tracking

Figure 4.34: DE-GL tracking experiment. Red dots indicate the followed path.

The last experiment is conducted to test the tracking performance when the mobile robot is following a path in the map of Figure 4.34. In this experiment, the last part of the complete path shown in Figure 4.22 is performed as accuracy tests have shown that the wide corridor on the right side of the map presented the greater challenge in terms of GL. Red dots in Figure 4.34 indicate the considered trajectory extracted from the SLAM results. In this test, a sensor noise of 2% and uniform noise of 5% are introduced to emulate a possible real situation. The algorithm selected for this task is the DE-GL implementation due to its convergence speed and the simplicity of parameter tuning. After the first iteration, once the GL is performed, the mutation factor F is reduced to 0.8 and the CR to 0.75. The initial population considered is $N_{Pini} = 160$ while in tracking is reduced to $N_{Pini} = 25$. Better accuracy results can be obtained if the population size is increased but with the consequent rise of computational costs. The search space has been reduced into a volume of ± 1 meter in directions x, y from the previously estimated position and ± 0.5 in z. In

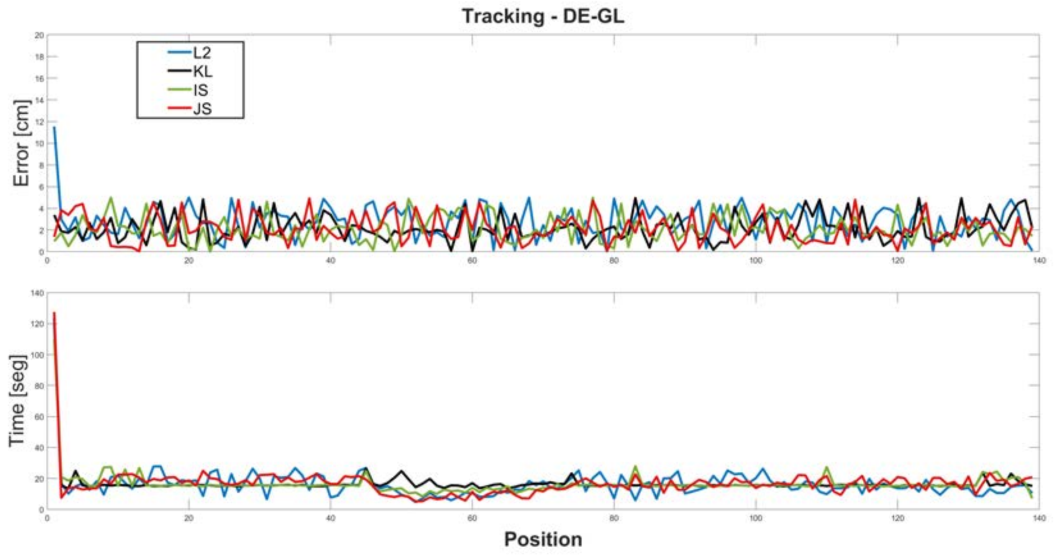

Figure 4.35: Tracking results for the bot path in Figure 4.34. Top chart: translation error. Bottom chart: computational time.

terms of rotation, the possible variation considered between consecutive positions is $(\pm 5, \pm 5, \pm 90)$ for roll, pitch, and yaw, respectively.

The results of the experiment are shown in Figure 4.35, where positioning error (cm) and computational time (seg) are displayed for each position. The localization is successful all along the performed path, with a translation error between 0-5cm. Computational time follows a stable curve around 20seg for every metric evaluated except for the initial GL task, which takes 125 segs. In contrast with occupancy grid representations, no distinctive behavior can be concluded between quadratic and divergence-based approach. Regarding the positioning error, all fitness functions perform in similar behavior. It could be worth mentioning that L2-norm presents the most significant error in the initial GL but not as conclusive as occupancy models.

4.3.5 Population Requirements

A homologous study regarding population requirements and computational cost to the one explained in section 4.2 is presented in this section. In this experiment, all available known positions shown in Figure 4.22 were considered, increasing the population number from 20 to 200 members. In the case of the IWO algorithm, as explained in section 3.3.3, the population size increases in each iteration to a maximum number. The maximum population is indicated in this experiment. The objective is to empirically test the necessary population for each algorithm by testing

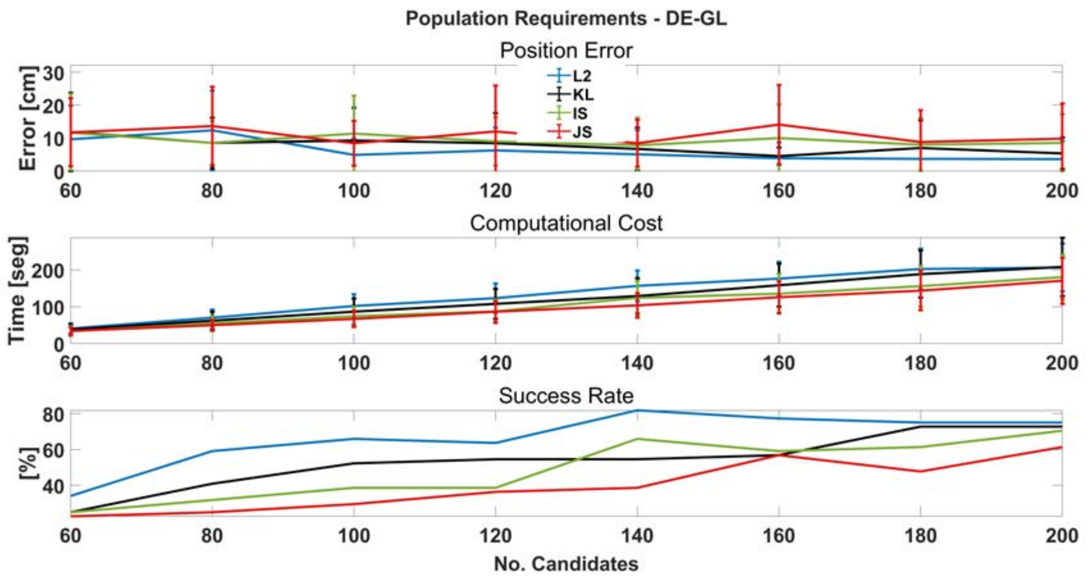

Figure 4.36: DE-GL Population requirements.

the evolution of success rate and position error in all known locations, where ground truth is obtained from the SLAM process.

Figures 4.36, 4.37 and 4.38 show the results in terms of position error, time to convergence, and success rate depending on the population number for each algorithm implemented, DE, PSO, and IWO respectively. Several interesting conclusions can be drawn from this experiment. The initial objective was to empirically select an optimum population number to perform the tests in the environment of Figure 4.22. As a general reflection, applicable to the three different methods, it can be determined that position error is not influenced by the population number. However, this is not totally true. IWO requires a determined minimal maximum population N_{Pmax} to be able to increase the size of the colony and perform a local search that optimizes the error. But, from a very low population number considering the search volume, the resulting average error for all possible positions in the map remains practically invariant even if N_p is increased. Comparing the average errors between algorithms, IWO presents the best results with a minimum average error of 4.5cm with $Np = 200$, in comparison with 5.7cm for PSO and 7.1 for DE. All metrics obtain similar errors applied to the same algorithm.

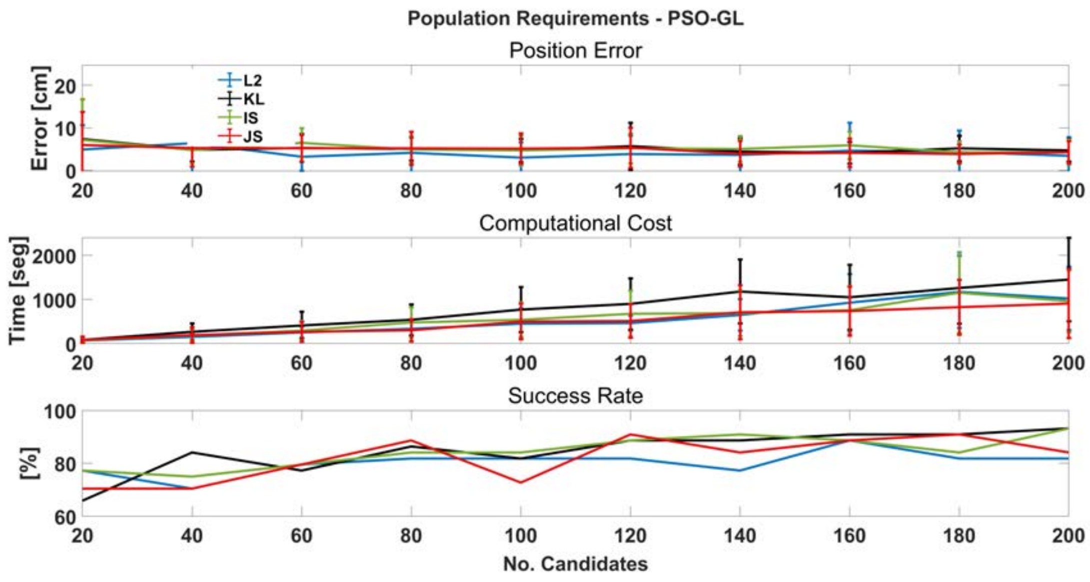

Figure 4.37: PSO-GL Population requirements.

In terms of computational cost, the increase in processing time is linear with the size of the population evaluated. The bottleneck in terms of efficiency for these implementations is the NN-search to establish each homologous point in the global map for each point in the local scan of each candidate, as explained in section 3.2.2. If a higher amount of candidates is evaluated, the computational cost is consequently increased. Comparing the processing times for each algorithm, it is remarkable that DE is 10 times faster than IWO. The parameter settings of the DE Metaheuristic allow to perform an exploratory search by amplifying the mutation factor F, and conversion is achieved in fewer iterations. In contrast, the possibility of premature convergence augments. IWO is slower by definition when performing global searches, with multiple local minimums like this case. For each plant, 2-6 new seeds are generated in this implementation, increasing the number of evaluations while in DE and PSO, only N_p evaluations are performed in each iteration. This fact is reflected in the middle chart of Figure 4.38, with almost 1200segs needed to converge when the colony size is set to 200 members. No metric shows any remarkable behavior in terms of computational cost.

Finally, regarding the comparison between success rate and population number, it can be concluded that the success rate follows an approximation to a sigmoid function when the population is increased. From a certain number of members, the localization rate does not improve. This can be noticed by observing the bottom charts of Figures 4.36, 4.37 and 4.38. Optimal population sizes for this map and each implementation are extracted from this experiment. The population was set to 160 members for all

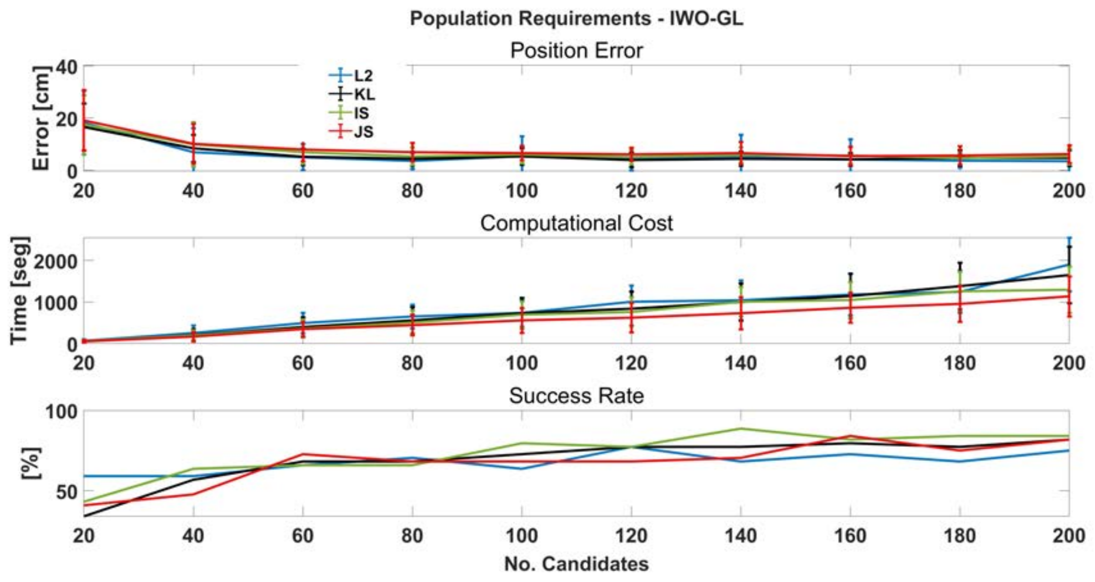

Figure 4.38: IWO-GL Population requirements.

implementations, observing that from that point, the success rate did not improve significantly to compensate for the computational cost.

The optimum population depends on numerous factors such as symmetries in the environment, size of the search space, or the basin of attraction of the global optimum for each position of the map; therefore, a more thorough study is necessary to extract general conclusions that are applicable to different environments, augmenting the flexibility of these methods.

Chapter 5

Conclusions and Future Works

This chapter describes the contributions and extracted conclusions from this dissertation, describing the main achievements, solved and unsolved objectives, and presents the future improvements and considered work.

In this thesis, various solutions have been studied, developed, implemented, and tested to solve the Global Localization issue through evolutionary techniques, in contrast with probabilistic or sensor fusion approaches. Two types of maps have been considered, covering the most common format of environment geometric representation: occupancy grid maps and Point Cloud representations. These algorithms were selected among the population-based algorithms for the simplicity and low level of specific parameters required to be tuned. The main objective of this work was to solve the GL problem in large volume maps and different types of representations with only laser-based information, a remarkable quality considering indoor and underground spaces.

In addition, a novel approach for fitness function implementation for laser perception comparison is the main contribution of this thesis. Probability-based cost functions introduce an asymmetry into the evaluation, favoring a biased comparison on "how" the scans differ, in contrast with symmetric L1 and L2 norms. These probability-based functions enable a different perspective when comparing laser scans by punishing or favoring different situations through a weighted comparison. The necessary probability model of the laser sensor and the weighted comparison designed and implemented as a fitness function were presented. This implementation has proven to manage modeled and unmodeled types of perturbations in the perception phase from the actual position.

The different Meta-heuristics have been tested in various situations. The accuracy of the algorithms is enough to consider the Global Localization solved in both types of environments and 2D and 3D spaces. The stochastic nature of the search can deal with the non-linearity and arbitrariness of the dynamics and the introduction of non-Gaussian perturbations without the constraints of posterior density approximations. Population-based Meta-heuristics have proven to cope with the expected quantity of local minimums caused by the symmetries and non-singular positions in indoor maps.

A full 6DOF DE-based GL filter has been developed. This implementation is an easy-to-tune tool for localization tasks. Its mutation and cross-over characteristics encompass an inherent exploratory nature very suitable for large search domains. Thresholding and discarding mechanisms are implemented to adjust premature convergence to local minimums and computational costs. DE-GL filter was tested over simulated and real 3D occupancy grid maps with an average error of around 5cm obtained in a search space encompassing thousands of cubic meters. The orientation error is insignificant, and the successful localization rate is over 75% in most cases. Several experiments have been conducted to test the DE-GL filter before perception perturbations, including Gaussian noise, uniform noise, and unmodeled obstacles. A comparison between divergence-based functions and L2-norm shows outstanding results in favor of the probabilistic approach. The robustness of the weighted divergence-based fitness function can deal with up to 60% occluded laser measures.

Four different statistical distances have been implemented, KL, IS, DP and JS. Further analysis of their behavior as a fitness function has been accomplished in this work. We can conclude that no remarkable differences are found before the different situations addressed in the experimental tests. To conclude, the outstanding overcome of statistical distances compared with quadratic functions enforces the idea that a probabilistic approach bears a significant advantage compared to symmetric functions regardless of the divergences chosen. This approach can deal with higher levels and different sources of unexpected information from the laser scan.

The bio-inspired optimization solutions have been extended to Point Cloud maps, the current state-of-the-art 3D space representation technique. Sparse laser information presents a more challenging issue as more information is obtained with an increase in computational costs. In addition, no spatial relation can be ensured between the points that represent a global map and a local scan. Hence, this sparsity is addressed by solving the optimization as a Scan Matching procedure. Two more Meta-heuristics are presented based on PSO and IWO algorithms that present different qualities but the same simplicity regarding DE. These are population-based algorithms that introduce a more local-focused search of the domain. Higher accuracy is expected if the convergence of the algorithm is achieved. The implementation, the considered parameters, and its particularization for the GL task are presented and tested. In a general conclusion, the GL issue has been solved for sparse metric maps and all Meta-heuristic implementations. Differences can be observed that lead us to select PSO-GL for the trade-off between precision and computational time. The local search nature of IWO provides high accuracy. However, the exponential increase in population in each iteration leads to a high computational cost, the main drawback of this implementation. The exploratory nature of DE leads to satisfactory results but even reducing it close to converging still leads to a lower success rate. PSO shows a more stable performance in terms of accuracy and localization rate. The parameter adjustment to establish a trade-off between local and global search is more intuitive, and accurate localization is obtained in fewer iterations when compared to IWO.

Regarding the implementation of probability-based fitness functions, the results show that if this problem is approached as a Scan Matching solution, probability profiles do not suppose a drastic improvement as in occupancy maps. The difficulty in establishing a spatial relation for each point of the local laser scan with a point in the global map prevents the weighted divergence-based fitness function from exploiting its capabilities. Based on the experiments in Point Cloud environments, the probabilistic approach still shows some advantages over the L2-norm in positioning accuracy before unmodeled obstacles. However, in a general conclusion, the quadratic function is more suitable in this type of representation if no spatial relation is achieved, showing a more robust performance with a higher success rate.

The main drawback of these methods is the computational cost and initialization parameters. The high amount of time elapsed is motivated by the large size of the search space, with a significant population required, which leads to a high number of evaluations to converge to an optimum solution. As shown in the experimental tests, posterior tracking performance alleviates this situation. The search space and the necessary population number can be significantly reduced, and accurate localization can be obtained at reasonable times. However, stochastic algorithms are still challenging to apply online. Population initialization and adjustment are important factors in the computational efficiency of these methods. Empirical tests performed prove that an increase in population does not necessarily improve the performance of the different solutions.

The different objectives proposed in this thesis have been satisfactorily resolved. Furthermore, the several tools developed in this work provide a benchmark for LiDAR only Global Localization using evolutionary optimization, providing a set of algorithms and function implementations that can be easily scaled and flexible to apply in other mapping examples for different indoor or outdoor environments.

5.1 Future Developments

Although the experimental section has validated the proposed methods, contributions and developments clarify that there are still many pending works and still room for improvement. In addition, this dissertation sets out several research areas for further development:

- An immediate consideration is to convert PC-based maps into occupancy grid maps and explore the possibilities of a divergence-based approach for our mapping results.
- Experiments can be expanded to several other environments, including underground and mining situations or outdoor localization, where external sources of information can be very helpful in improving accuracy and computational costs.
- Although several tests have been performed regarding the introduction of simulated non-modeled obstacles, GL using laser scan information, including real obstacle perception, would be an interesting consideration to validate the methods. In addition, further study must be carried out to parameterize these situations, considering the many particular possibilities: distance to the obstacle, orientation, and specific map features occluded.
- Further study is necessary regarding convergence conditions of the algorithms, considering different factors depending on the cost function and characteristics of the map.
- The improvement of population requirements analysis is needed.
- Development of an optimization method to select the different parameters that control each Meta-heuristic.
- The main drawback of the implemented methods is their computational cost. Although GL is a particular task that does not need to be performed recursively and can be relaxed by tracking, this factor must be reduced for real-time applications. Considering parallel computing would be an interesting starting point.
- Inclusion of geometric feature detection of singular points in the global map. This could lead to increase accuracy by implementing a multi-objective cost function or reducing the amount of information processed.
- Application of the probabilistic fitness function approach to dynamically challenging SLAM situations. The robustness before unexpected perceptions proven in the experiments could be utilized to address dynamically changing or noisy environment mapping.
- Implementation of other stochastic optimization possibilities among the many existing. An ambitious consideration is to design a specific algorithm for localization and mapping tasks that can improve the results of the methods presented.

Bibliography

[1] P. Jensfelt, *Approaches to mobile robot localization in indoor environments.* PhD Thesis, Royal Institute of Technology, Sweden, 2001.

[2] D. Fox, W. Burgard, and S. Thrum, "Markov localization for mobile robots in dynamic environments," in *Journal of Articial Intelligence Research, vol. 11, no. 11*, pp. 391–427, 1999.

[3] W. Neto, M. Pinto, and A. Marcato, "Mobile robot localization based on the novel leader-based bat algorithm," in *J Control Autom Electr Syst 3,vol. 30*, p. 337–346, 2019.

[4] J. Weingarten and R. Siegwart, "Ekf-based 3d slam for structured environment reconstruction," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3834–3839, 2005.

[5] M. Begum, G. Mann, and R. Gosine, "A fuzzy-evolutionary algorithm for simultaneous localization and mapping of mobile robots," in *2006 IEEE International Conference on Evolutionary Computation*, pp. 1975–1982, 2006.

[6] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots.* MIT press, 2011.

[7] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," in *Journal of Global Optimization vol. 11*, pp. 341–359, 1997.

[8] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the sixth international symposium on micro machine and human science*, pp. 39–43, Ieee, 1995.

[9] A. R. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecological informatics*, vol. 1, no. 4, pp. 355–366, 2006.

[10] I. Cox, "An experiment in guidance and navigation of an autonomous robot vehicle," in *IEEE Transaction on Robotics and Automation, vol. 7*, pp. 193–204, 1991.

[11] J. L. Crowley, "World modelling and position estimation for a mobile robot using ultrasonic ranging," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'89)*, 1989.

[12] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[13] W. Burgard, D. Fox, D. Henning, and T. Schmidt, "Estimating the absolute position of a mobile robot using position probability grids," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'96)*, 1996.

[14] J. Reuter, "Mobile robot self-localization using pdab," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 4, pp. 3512–3518 vol.4, 2000.

[15] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1, pp. 99–141, 2001.

[16] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, pp. 1322–1328 vol.2, 1999.

[17] J. Klaess, J. Stueckler, and S. Behnke, "Efficient mobile robot navigation using 3d surfel grid maps," in *ROBOTIK, 7th German Conference on Robotics*, pp. 1–4, 2012.

[18] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard, *Monte Carlo Localization in Outdoor Terrains Using Multi-Level Surface Maps*, pp. 213–222. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[19] T. Back, D. Fogel, and Z. Michalewicz, *Evolutionary Computation I: Basic Algorithms and Operators.* Bristol, UK: IOP Publishing Ltd., 2000.

[20] T. Back, D. Fogel, and Z. Michalewicz, *Evolutionary Computation II: Basic Algorithms and Operators.* Bristol, UK: IOP Publishing Ltd., 2000.

[21] A. R. Vahdat, N. NourAshrafoddin, and S. S. Ghidary, "Mobile robot global localization using differential evolution and particle swarm optimization," in *2007 IEEE Congress on Evolutionary Computation*, pp. 1527–1534, 2007.

[22] M. Lisowski, Z. Fan, and O. Ravn, "Differential evolution to enhance localization of mobile robots," in *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, pp. 241–247, 2011.

[23] Z. W. Geem, J. H. Kim, and G. Loganathan, "A new heuristic optimization algorithm: Harmony search," *SIMULATION*, vol. 76, no. 2, pp. 60–68, 2001.

[24] M. Mirkhani, R. Forsati, A. M. Shahri, and A. Moayedikia, "A novel efficient algorithm for mobile robot localization," *Robotics and Autonomous Systems*, vol. 61, no. 9, pp. 920–931, 2013.

[25] L. Moreno, S. Garrido, and M. Muñoz, "Evolutionary filter for robust mobile robot localization," in *Robotics and Autonomous Systems, vol. 54, no. 7*, 2006.

[26] L. Moreno, S. Garrido, F. Martín, and M. Muñoz, "Differential evolution approach to the grid-based localization and mapping problem," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*, pp. 3479–3484, 2007.

[27] F. Martín, L. Moreno, S. Garrido, and D. Blanco, "High-accuracy global localization filter for three-dimensional environments," in *Robotica, vol. 30*, pp. 363–378, 2011.

[28] L. Ronghua and H. Bingrong, "Coevolution based adaptive monte carlo localization (ceamcl)," *International Journal of Advanced Robotic Systems*, vol. 1, no. 3, p. 19, 2004.

[29] C.-H. Chien, C.-C. Hsu, W.-Y. Wang, W.-C. Kao, and C.-J. Chien, "Global localization of monte carlo localization based on multi-objective particle swarm optimization," in *2016 IEEE 6th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, pp. 96–97, 2016.

[30] K. Arras, J. Castellanos, and R. Siegwart, "Feature-based multi-hypothesis localization and tracking for mobile robots using geometric constraints," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'02*, pp. 1371–1377, 2002.

[31] S. Roumeliotis and G. Bekey, "Bayesian estimation and kalman filtering: a unified framework for mobile robot localization," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 3, pp. 2985–2992 vol.3, 2000.

[32] I. J. Cox and J. J. Leonard, "Modeling a dynamic environment using a bayesian multiple hypothesis approach," *Artificial Intelligence*, vol. 66, no. 2, pp. 311–344, 1994.

[33] D. Austin and P. Jensfelt, "Using multiple gaussian hypotheses to represent probability distributions for mobile robot localization," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, pp. 1036–1041 vol.2, 2000.

[34] G. Vasiljević, D. Miklić, I. Draganjac, Z. Kovačić, and P. Lista, "High-accuracy vehicle localization for autonomous warehousing," *Robotics and Computer-Integrated Manufacturing*, vol. 42, pp. 1–16, 2016.

[35] L. Chen, H. Hu, and K. McDonald-Maier, "Ekf based mobile robot localization," in *2012 Third International Conference on Emerging Security Technologies*, pp. 149–154, 2012.

[36] G. Jochmann, S. Kerner, S. Tasse, and O. Urbann, "Efficient multi-hypotheses unscented kalman filtering for robust localization," in *RoboCup 2011: Robot Soccer World Cup XV* (T. Röfer, N. M. Mayer, J. Savage, and U. Saranlı, eds.), (Berlin, Heidelberg), pp. 222–233, Springer Berlin Heidelberg, 2012.

[37] K. Lingemann, A. Nüchter, J. Hertzberg, and H. Surmann, "High-speed laser localization for mobile robots," *Robotics and Autonomous Systems*, vol. 51, no. 4, pp. 275–296, 2005.

[38] C. Tsai, H. Lin, and S. Lai, "Multisensor 3d posture determination of a mobile robot using inertial and ultrasonic sensors," *Journal Intelligent Robot Systems*, vol. 42, p. 317–335.

[39] L.-C. Lai, T.-L. Lee, H.-T. Fan, and C.-J. Wu, "A nonlinear programming method for 3d localization of mobile robots," in *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, pp. 250–255, 2005.

[40] M. Agrawal and K. Konolige, "Real-time localization in outdoor environments using stereo vision and inexpensive gps," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, pp. 1063–1068, 2006.

[41] N. Ho and R. Jarvis, "Vision based global localisation using a 3d environmental model created by a laser range scanner," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2964–2969, 2008.

[42] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 404–417, Springer Berlin Heidelberg, 2006.

[43] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer Handbook of Robotics, Springer Berlin Heidelberg, 2008.

[44] S. Thrun, D. Hahnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker, "A system for volumetric robotic mapping of abandoned mines," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 3, pp. 4270–4275 vol.3, 2003.

[45] E. Zalama, G. Candela, J. Gomez, and S. Thrun, "Concurrent mapping and localization for mobile robots with segmented local maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 546–551 vol.1, 2002.

[46] H. Durrant-Whyte, S. Majumder, S. Thrun, M. de Battista, and S. Scheding, "A bayesian algorithm for simultaneous localisation and map building," in *Robotics Research* (R. A. Jarvis and A. Zelinsky, eds.), (Berlin, Heidelberg), pp. 49–60, Springer Berlin Heidelberg, 2003.

[47] M. Kaess and F. Dellaert, "A markov chain monte carlo approach to closing the loop in slam," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 643–648, 2005.

[48] T. Zhang, K. Wu, J. Song, S. Huang, and G. Dissanayake, "Convergence and consistency analysis for a 3-d invariant-ekf slam," *IEEE Robotics and Automation Letters*, vol. 2, pp. 733–740, 2017.

[49] H. Taketomi, T.; Uchiyama and S. Ikeda, "Visual slam algorithms: a survey from 2010 to 2016," in *IPSJ Transactions on Computer Vision and Applications*, 2017.

[50] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[51] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 International Conference on Computer Vision*, pp. 2320–2327, 2011.

[52] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *2013 IEEE International Conference on Computer Vision*, pp. 1449–1456, 2013.

[53] J. Reuter, "Mobile robot self-localization using pdab," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00)*, 2000.

[54] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *2013 IEEE International Conference on Computer Vision*, pp. 1449–1456, 2013.

[55] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

[56] S. Zhao and Z. Fang, "Direct depth slam: Sparse geometric feature enhanced direct depth slam system for low-texture environments," *Sensors*, vol. 18, no. 10, 2018.

[57] S. Thrun, W. Burgard, and D. Fox, "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, pp. 321–328 vol.1, 2000.

[58] H. Zhao and R. Shibasaki, "Reconstructing a textured cad model of an urban environment using vehicle-borne laser range scanners and line cameras," in *Machine Vision and Applications*, no. 14, p. 35–41, 2003.

[59] M. Montemerlo and S. Thrun, *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, vol. 27. Springer, 2007.

[60] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[61] D. Valiente, A. Gil, L. Fernández, and Óscar Reinoso, "A comparison of ekf and sgd applied to a view-based slam approach with omnidirectional images," *Robotics and Autonomous Systems*, vol. 62, no. 2, pp. 108–119, 2014.

[62] L. D'Alfonso, A. Griffo, P. Muraca, and P. Pugliese, "A slam algorithm for indoor mobile robot localization using an extended kalman filter and a segment based environment mapping," in *2013 16th International Conference on Advanced Robotics (ICAR)*, pp. 1–6, 2013.

[63] F. A. A. Cheein, J. M. Toibero, F. di Sciascio, R. Carelli, and F. L. Pereira, "Monte carlo uncertainty maps-based for mobile robot autonomous slam navigation," in *2010 IEEE International Conference on Industrial Technology*, pp. 1433–1438, 2010.

[64] J. Andrade-Cetto, T. Vidal-Calleja, and A. Sanfeliu, "Unscented transformation of vehicle states in slam," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 323–328, 2005.

[65] C. Schymura and D. Kolossa, "Potential-field-based active exploration for acoustic simultaneous localization and mapping," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 76–80, 2018.

[66] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct slam for omnidirectional cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 141–148, 2015.

[67] K.-T. Song, Y.-H. Chiu, L.-R. Kang, S.-H. Song, C.-A. Yang, P.-C. Lu, and S.-Q. Ou, "Navigation control design of a mobile robot by integrating obstacle

avoidance and lidar slam," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1833–1838, 2018.

[68] H. Chang, W. Yang, H. Zhang, X. Yang, and C.-Y. Chen, "An improved fast-slam using resmapling based on particle swarm optimization," in *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 229–234, 2016.

[69] Y. Toda and N. Kubota, "Self-localization based on multiresolution map for remote control of multiple mobile robots," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1772–1781, 2013.

[70] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[71] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6d slam—3d mapping outdoor environments: Research articles," *J. Field Robot.*, vol. 24, p. 699–722, aug 2007.

[72] P. G. Prieto, F. Martín, L. Moreno, and J. Carballeira, "Dendt: 3d-ndt scan matching with differential evolution," in *2017 25th Mediterranean Conference on Control and Automation (MED)*, pp. 719–724, 2017.

[73] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3, pp. 2743–2748, IEEE, 2003.

[74] A. Censi, "An icp variant using a point-to-line metric," in *2008 IEEE International Conference on Robotics and Automation*, pp. 19–25, 2008.

[75] D. Grant, J. Bethel, and M. Crawford, "Point-to-plane registration of terrestrial laser scans," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 72, pp. 16–26, 2012.

[76] A. Diosi and L. Kleeman, "Laser scan matching in polar coordinates with application to slam," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3317–3322, 2005.

[77] M. Greenspan and M. Yurick, "Approximate k-d tree search for efficient icp," in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pp. 442–448, 2003.

[78] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3212–3217, 2009.

[79] S. Salti, F. Tombari, and L. Di Stefano, "Shot: Unique signatures of histograms for surface and texture description," *Computer Vision and Image Understanding*, vol. 125, pp. 251–264, 2014.

[80] C.-S. Chen, Y.-P. Hung, and J.-B. Cheng, "Ransac-based darces: a new approach to fast automatic registration of partially overlapping range images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 11, pp. 1229–1234, 1999.

[81] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," *Journal of Intelligent and Robotic systems*, vol. 18, no. 3, pp. 249–275, 1997.

[82] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[83] M. Magnusson, A. Nuchter, C. Lorken, A. J. Lilienthal, and J. Hertzberg, "Evaluation of 3d registration reliability and speed - a comparison of icp and ndt," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3907–3912, 2009.

[84] T. Takubo, T. Kaminade, Y. Mae, K. Ohara, and T. Arai, "Ndt scan matching method for high resolution grid map," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1517–1522, 2009.

[85] C. Ulaş and H. Temeltaş, "A fast and robust scan matching algorithm based on ml-ndt and feature extraction," in *2011 IEEE International Conference on Mechatronics and Automation*, pp. 1751–1756, 2011.

[86] E. Einhorn and H.-M. Gross, "Generic 2d/3d slam with ndt maps for lifelong application," in *2013 European Conference on Mobile Robots*, pp. 240–247, 2013.

[87] J. C. Spall, *Introduction to stochastic search and optimization: estimation, simulation, and control*, vol. 65. John Wiley & Sons, 2005.

[88] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

[89] J. D. Bagley, *The behavior of adaptive systems which employ genetic and correlation algorithms.* University of Michigan, 1967.

[90] D. T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi, "The bees algorithm—a novel tool for complex optimisation problems," in *Intelligent production machines and systems*, pp. 454–459, Elsevier, 2006.

[91] A. Jevtic, A. Gutierrez, D. Andina, and M. Jamshidi, "Distributed bees algorithm for task allocation in swarm of robots," *IEEE Systems Journal*, vol. 6, no. 2, pp. 296–304, 2012.

[92] P. Bhattacharjee, P. Rakshit, I. Goswami, A. Konar, and A. K. Nagar, "Multi-robot path-planning using artificial bee colony optimization algorithm," in *2011 Third World Congress on Nature and Biologically Inspired Computing*, pp. 219–224, 2011.

[93] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[94] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[95] R. G. Reynolds, "An introduction to cultural algorithms," in *Proceedings of the third annual conference on evolutionary programming*, vol. 24, pp. 131–139, World Scientific, 1994.

[96] R. Y. Rubinstein, "Optimization of computer simulation models with rare events," *European Journal of Operational Research*, vol. 99, no. 1, pp. 89–112, 1997.

[97] S. H. Brooks, "A discussion of random methods for seeking maxima," *Operations research*, vol. 6, no. 2, pp. 244–251, 1958.

[98] F. Glover and C. McMillan, "The general employee scheduling problem. an integration of ms and ai," *Computers & operations research*, vol. 13, no. 5, pp. 563–573, 1986.

[99] S. Forrest and M. Mitchell, *Relative Building Block Fitness and the Building Block Hypothesis. Foundations of Genetic Algorithms 2.* Morgan Kaufmann Publishers Inc, 1993.

[100] R. Joshi and A. Sanderson, "Minimal representation multisensor fusion using differential evolution," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 29, no. 1, pp. 63–76, 1999.

[101] F. Neri and E. Mininno, "Memetic compact differential evolution for cartesian robot control," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 54–65, 2010.

[102] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L. C. Jain, and A. K. Nagar, "Realization of an adaptive memetic algorithm using differential evolution and q-learning: A case study in multirobot path planning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 4, pp. 814–831, 2013.

[103] S. Doctor, G. Venayagamoorthy, and V. Gudise, "Optimal pso for collective robotic search applications," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 2, pp. 1390–1395 Vol.2, 2004.

[104] J. Hereford, "A distributed particle swarm optimization algorithm for swarm robotic applications," in *2006 IEEE International Conference on Evolutionary Computation*, pp. 1678–1685, 2006.

[105] M. S. Couceiro, R. P. Rocha, and N. M. F. Ferreira, "A novel multi-robot exploration approach based on particle swarm optimization algorithms," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 327–332, 2011.

[106] H. Duan, Q. Luo, Y. Shi, and G. Ma, "?hybrid particle swarm optimization and genetic algorithm for multi-uav formation reconfiguration," *IEEE Computational Intelligence Magazine*, vol. 8, no. 3, pp. 16–27, 2013.

[107] K. Okada and Y. Fujimoto, "Grid-based localization and mapping method without odometry information," in *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, pp. 159–164, 2011.

[108] R. Havangi, H. D. Taghirad, M. A. Nekoui, and M. Teshnehlab, "A square root unscented fastslam with improved proposal distribution and resampling," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 5, pp. 2334–2345, 2014.

[109] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 132–141, 2013.

[110] E. Masehian and D. Sedighizadeh, "A multi-objective pso-based algorithm for robot path planning," in *2010 IEEE International Conference on Industrial Technology*, pp. 465–470, 2010.

[111] A. Sengupta, T. Chakraborti, A. Konar, and A. Nagar, "Energy efficient trajectory planning by a robot arm using invasive weed optimization technique," in *2011 Third World Congress on Nature and Biologically Inspired Computing*, pp. 311–316, 2011.

[112] D. R. Parhi and P. K. Mohanty, "Iwo-based adaptive neuro-fuzzy controller for mobile robot navigation in cluttered environments," *The International Journal of Advanced Manufacturing Technology*, vol. 83, no. 9, pp. 1607–1625, 2016.

[113] M. R. Panda, P. Das, and S. Pradhan, "Hybridization of iwo and ipso for mobile robots navigation in a dynamic environment," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 9, pp. 1020–1033, 2020.

[114] H. A. Ismail, M. S. Packianather, R. I. Grosvenor, and E. E. Eldhukri, "The application of iwo in lqr controller design for the robogymnast," in *2015 SAI Intelligent Systems Conference (IntelliSys)*, pp. 274–279, 2015.

[115] L. Moreno, D. Blanco, M. Muñoz, and S. Garrido, "L1-l2-norm comparison in global localization of mobile robots," in *Robotics and Autonomous Systems, vol. 59*, pp. 597–610, 2001.

[116] F. Donoso-Aguirre, J.-P. Bustos-Salas, M. Torres-Torriti, and A. Guesalaga, "Mobile robot localization using the hausdorff distance," *Robotica*, vol. 26, no. 2, pp. 129–141, 2008.

[117] M. Potter and K. DeJong, "A cooperative co-evolutionary approach to function optimization," in *Proceedings of parallel problems solving from nature 3, vol. 0*, pp. 249–257, 1994.

[118] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *International conference on evolutionary programming*, pp. 591–600, Springer, 1998.

[119] S. Kullback and R. Leibler, "On information and sufficiency.," in *The Annals of Mathematical Statistics,22*, pp. 79–86, 1951.

[120] J. Lin, "Divergence measures based on the shannon entropy," in *IEEE Transactions on Information Theory 37*, pp. 145–151, 1991.

[121] A. Basu, I. Harris, N. Hjort, and M. Jones, "Robust and efficient estimation by minimizing a density power divergence," in *Biometrika 85*, pp. 549–559, 1998.

[122] F. Itakura and S. Saito, "Analysis synthesis telephony based on the maximum likelihood method," in *Proceedings of the 6th International Congress on Acoustics, IEEE*, pp. C–17–C–20, 1968.

[123] T. de Laet, J. Schutter, and H. Bruyninckx, "Rigorously bayesian range finder sensor model for dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'08*, 2008.

[124] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'03*, 2003.

[125] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, and L. Kavraki, *Principles of Robot Motion: Theory, Algorithms, and Implementations.* Cambridge, 2005.

[126] P. Pfaff, W. Burgard, and D. Fox, "Robust monte-carlo localization using andpative likelihood models," in *European Robotics Symposium*, 2006.

[127] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.* Cambridge, 2005.

[128] A. Cichocki and S.-i. Amari, "Families of alpha- beta- and gamma- divergences: Flexible and robust measures of similarities," *Entropy*, vol. 12, no. 6, pp. 1532–1568, 2010.

[129] M. Hein and O. Bousquet, "Hilbertian metrics and positive definite kernels on probability measures," in *International Workshop on Artificial Intelligence and Statistics*, pp. 136–143, PMLR, 2005.

[130] J. Zhang, "Divergence function, duality, and convex analysis," *Neural computation*, vol. 16, no. 1, pp. 159–195, 2004.

[131] S. Eguchi and S. Kato, "Entropy and divergence associated with power function and the statistical application," *Entropy*, vol. 12, no. 2, pp. 262–274, 2010.

[132] S.-i. Amari and H. Nagaoka, *Methods of information geometry*, vol. 191. American Mathematical Soc., 2000.

[133] S. M. Ali and S. D. Silvey, "A general class of coefficients of divergence of one distribution from another," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 28, no. 1, pp. 131–142, 1966.

[134] M. Mihoko and S. Eguchi, "Robust blind source separation by beta divergence," *Neural computation*, vol. 14, no. 8, pp. 1859–1886, 2002.

[135] B. Jian and B. C. Vemuri, "A robust algorithm for point set registration using mixture of gaussians," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, pp. 1246–1251, IEEE, 2005.

[136] M. Kim and S. Lee, "Estimation of a tail index based on minimum density power divergence," *Journal of multivariate Analysis*, vol. 99, no. 10, pp. 2453–2471, 2008.

[137] Y. Sogawa, T. Ueno, Y. Kawahara, and T. Washio, "Active learning for noisy oracle via density power divergence," *Neural networks*, vol. 46, pp. 133–143, 2013.

[138] H. Jeffreys, "An invariant form for the prior probability in estimation problems," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 186, no. 1007, pp. 453–461, 1946.

[139] C. Manning and H. Schütze, "Foundations of statistical natural language processing," *MIT Press*, 1999.

[140] I. Dagan, L. Lee, and F. Pereira, "Similarity-based methods for word sense disambiguation," in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics*, pp. 56–63, 1997.

[141] D. Endres and J. Schindelin, "A new metric for probability distributions," 2003.

[142] J. Jensen, "Sur les fonctions convexes et les inégalités entre les valeurs moyennes," in *Acta Mathematica 30*, pp. 175–193, 1906.

[143] A. Majtey, P. Lamberti, and D. Prato, "Jensen-shannon divergence as a measure of distinguishability between mixed quantum states," *Physical Review A*, vol. 72, no. 5, p. 052310, 2005.

[144] J. A. Aslam and V. Pavlu, "Query hardness estimation using jensen-shannon divergence among multiple scoring functions," in *European conference on information retrieval*, pp. 198–209, Springer, 2007.

[145] P. Gajer, R. M. Brotman, G. Bai, J. Sakamoto, U. M. Schütte, X. Zhong, S. S. Koenig, L. Fu, Z. Ma, X. Zhou, *et al.*, "Temporal dynamics of the human vaginal microbiota," *Science translational medicine*, vol. 4, no. 132, pp. 132ra52–132ra52, 2012.

[146] S. Se, D. G. Lowe, and J. J. Little, "Vision-based global localization and mapping for mobile robots," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 364–375, 2005.

[147] Y. Wang, W. Chen, and J. Wang, "Map-based localization for mobile robots in high-occluded and dynamic environments," *Industrial Robot: An International Journal*, 2014.

[148] T.-Y. Tsou and S.-H. Wu, "A robust feature matching method for robot localization in a dynamic indoor environment," in *International Conference on Technologies and Applications of Artificial Intelligence*, pp. 354–365, Springer, 2014.

[149] L. Zhang, R. Zapata, and P. Lépinay, "Self-adaptive monte carlo localization for mobile robots using range sensors," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1541–1546, IEEE, 2009.

[150] M. Lisowski, "Differential evolution approach to the localization problem for mobile robots," Master's thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2009.

[151] L. Moreno, F. Martín, M. L. Muñoz, and S. Garrido, "Differential evolution markov chain filter for global localization," *Journal of Intelligent & Robotic Systems*, vol. 82, no. 3, pp. 513–536, 2016.

[152] P. Gonzalez, A. Mora, S. Garrido, R. Barber, and L. Moreno, "Multi-lidar mapping for scene segmentation in indoor environments for mobile robots," *Sensors*, vol. 22, no. 10, p. 3690, 2022.

[153] Y. Ma, Y. Guo, M. Lu, J. Zhao, and J. Zhang, "Global localization in 3d maps for structured environment," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 6680–6683, 2016.