Laboratório de Sistemas de Computação e Robótica
Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627, 31270-901 Belo Horizonte, MG Brasil
Fone: +55 31 3409-3470

Title:
Multi-Objective Approach for Robot Exploration

Kossar Jeddisaravi

Supervisor: Prof. Luciano Cunha de Araujo Pimenta

Belo Horizonte, June, 2015

Laboratório de Sistemas de Computação e Robótica
Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627, 31270-901 Belo Horizonte, MG Brasil
Fone: +55 31 3409-3470

Multi-Objective Approach for Robot Exploration

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica

Kossar Jeddisaravi

Supervisor: Prof. Luciano Cunha de Araujo Pimenta

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
Belo Horizonte
June, 2015

Acknowledgement

Foremost, I am grateful to the God for giving me health, well-being and strength that were necessary to complete this dissertation.

My special thanks is to my advisor, *Prof.Luciano Pimenta* for the continuous support of my master study and research, for his patience, motivation, helpful suggestions and discussions, and immense knowledge.

I wish to express my sincere thanks to *Prof.Frederico Guimaraes*. I am extremely thankful to him for sharing expertise, and sincere and valuable guidance when I was writing my paper.

Prof.Felipe Campelo is one of the best teachers that I have had in my life. I am also thankful to him for his practical advice. He introduced me to Design and Analysis of Experiments and his teachings inspired me to work on Chapter 5 of this dissertation.

I would like to thank *Prof.Guilherme Pereira* for offering me to use devices and equipments of robotic lab (CORO).

I appreciate the financial support from CNPq, CAPES, and FAPEMIG, Brazil that funded the research discussed in this dissertation.

I also wish to express my gratitude to my parents, brother and sister who always supported me during this period. None of this would have been possible without their prayer and patience of my family.

And finally, my heartfelt acknowledgment especially to my dear *Reza*. My husband to whom this dissertation is dedicated to, has been a constant source of love, concern, support and strength all these years. I am deeply indebted to him for his continuous encouragement and guidance as a kind teacher.

Abstract

This work addresses the problem of single robot coverage and exploration of an environment with the aim of finding a specific previously known object. As limited time is a constraint of interest we cannot search for an infinite number of points. Thus, we propose to find good points (also called search points) to place the robot sensors in order to acquire information from the environment and find the desired object. Given the interesting properties of the Generalized Voronoi Diagram (GVD), we define the search points along this roadmap. We redefine the problem of finding these search points as a multi-objective optimization one. NSGA-II is used as optimizer and ELECTRE I is applied as a decision making tool. We also solve a Chinese Postman Problem to optimize the path followed by the robot in order to visit the computed search points.
To identify the desired object in environment, we used a fast and robust object recognition application which is called Speeded Up Robust Features (SURF) algorithm. Simulations on Stage with implementation in ROS are also presented. The proposed approach tested on an real robot in a real world situation that indicates the applicability of our method. Lastly, statistical analysis shows a comparison between the solution found by our method and two others.

Contents

List of Figures

List of Tables

List of Symbols

A	Map
X	Set of site
x_i	Region site
R_j	Voronoi region
p	Point in each region
$d(x_i, p)$	Distance between two point x_i and p
T_{ij}	two-equidistant face
$\mathcal{QO}_i$	Configuration Obstacle i
$\mathcal{Q}_{free}$	Configuration free space
q_{start}	Start configuration
q_{free}	Free configuration
q_{goal}	Target configuration
$f()$	Vector function
s_i	Solution i
F_1	First objective
F_2	Second objective
v	Number of SPs
P_i	Position on the GVD
G	Graph
V	Set of vertices
(v_i, v_j)	Edge between vertices v_i and v_j
c_{ij}	Cost of edge(v_i, v_j)
P_c	Crossover probability
P_m	Mutation probability
S_1	First strategy

S_2	Second strategy
S_3	Third strategy
TTF	Time to find object or elapsed time
α	Significance level
$1 - \beta$	Power level
μ_i	mean of TTF in strategy i
H_0	Null hypothesis
H_1	Alternative hypothesis
y_{ij}	Data in linear statistical model
ϵ_{ij}	Residual for strategy i and observation j
sd	Standard deviation
$delta$	Minimally interesting effect
CI	Confidence interval
S_p^2	Pooled variance

List of Abbreviations

GVD	*Generalized Voronoi Diagram*
SP	*Search Point*
CPP	*Chinese Postman Problem*
RM	*Roadmap*
MOEAs	*Multi objective evolutionary algorithms*
NSGA	*Non-dominated Sorting Genetic Algorithm*
NSGA-II	*Non-dominated Sorting Genetic Algorithm-II*
ELECTRE	*Elimination And Choice Corresponding to Reality*
SURF	*Speeded Up Robust Features*
ROS	*Robot operating system*
Knn	*K nearest neighbor*

1 Introduction

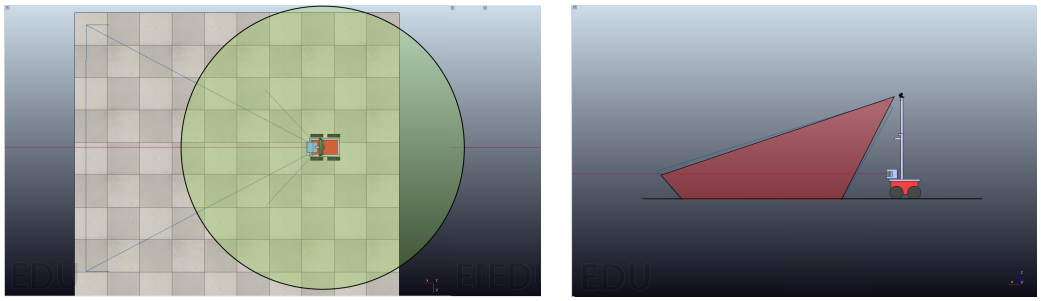
Exploration is an important task in different applications of robotics such as surveillance, cleaning, map building, coverage and search and rescue operation. Exploring a given area with robots requires the specification of paths that cover the whole environment. This type of problem is well known as Coverage Path Planning. In Coverage Path Planning the robot must traverse all points of the area while avoiding collisions (Galceran and Carreras, 2013). The requirements of a coverage path planning algorithm are:

- All points of the environment must have been seen by the robot sensors by the end of the task.
- The region must be covered by the sensors without overlapping.
- The robot must avoid collisions with the obstacles.
- The operation must be consecutive and sequential with no repetition of the path.

However, it is not always possible to satisfy all these requirements in complex environments because of limitations such as time, sensor range and energy consumption. Therefore, sometimes a priority should be considered. In this work, we address the problem of exploring an indoor environment with the aim of finding a previously known object with a robot equipped with sensors in limited time. However, we assume the robot may not be powerful enough to guarantee that the whole environment can be accurately inspected by the sensors in the given time as the robot moves. Thus, in our approach we define some stationary points, called search points (SPs), from which the robot carefully analyze the sensor readings to find the object. The goal of the robot is to visit as many search points as possible (given the limited time) to maximize the chance of finding the object. By the expression "visiting a search point", we mean: stay at the point and accurately acquire measurements from the sensors, rotating and moving the sensors if necessary, with the aim of finding the object.
We consider that the desired object is placed in a cluttered environment. Thus, the robot

should traverse safe paths to avoid collisions with obstacles. In this work we use the Generalized Voronoi Diagram (GVD) as a roadmap, so that the motion of the robot is constrained to this structure. Since the GVD maximizes the distance from obstacles, this roadmap allows for collision avoidance. Thus, we propose to distribute the *SPs* over the GVD, constraining the motion to a safe path as desired. The distribution of these points will be defined by the solution of a multi-objective optimization problem.

In order to evaluate the objective functions to be designed in this work, we considered a circular footprint sensor model. This model assumes that inside a circle centered at the robot position with radius r, the object can be seen. In fact, we use a combination of Laser Range Finder and Kinect for navigation and Kinect sensor for object recognition. Clearly, a Kinect is not a sensor with circular footprint. However, we associate robot and sensor motion so that the abstract sensor model can be emulated. This means that at every *SP* the robot will rotate in place and move the Kinect so that the object can be found if it is located inside the defined circular footprint (see Figure 1.1).



(a) Top view of the robot circular footprint.

(b) Side view of the area covered by a Kinect sensor.

Figure 1.1: A robot with its footprint and equipped sensor.

1.1 Motivation

For the searching purpose, when limited time is a constraint of interest, we cannot search for an infinite number of points. Therefore, having a strategy which leads the robot to explore the whole area within a specific time limit, can be very useful. For this reason, we propose a technique to find good points to place the robot sensors in order to acquire information from the environment and find the desired object. This technique guides the robot to search just at some stationary points instead of searching at all possible configurations in the map.

This work can be suitable for different applications such as RoboCup@Home, specially

when time limitation is one of the important criteria. In RoboCup@Home competition, tests are classified in two parts: predefined procedures and open demonstrations (Stückler et al., 2014). The robot must carry out the tasks within a limited amount of time in the predefined tests and teams present their abilities in open demonstrations. Our proposed method is applicable in this example to minimize the overall exploration time.

1.2 Objectives

Since obstacle avoidance is an important requirement, the first objective of this work is to consider safety in exploration. As in home environment, we are interested in indoor cluttered environments. We use the GVD of the map as a roadmap to guarantee safety.
The second objective is the reduction of the overall exploration time. In fact, the main goal of this work is to find a pre-specified object in limited time.
It is also objective of this work to simulate the proposed technique and compare it with other techniques statistically.
We also implement the proposed strategy in a real robot to show the performance and applicability of our approach.

1.3 Thesis Organization

The remainder of this dissertation is organized as follows: in the next section, a literature review of robot exploration is discussed.
In Section 3, the proposed solution is introduced. Topics included are Generalized Voronoi Diagram, GVD induced graph, multi-objective optimization, chinese postman problem and Speeded Up Robust Features algorithm. The experimental result (Section 4) includes computing the GVD, multi-objective solution in MATLAB, simulation on Stage with ROS, object recognition with SURF and test results on a real robot.
In Section 5, experimental statistical analysis comparing our proposed method with other methods is presented. There is also detailed information of the method for determining appropriate sample sizes and hypothesis tests. The graphical data analysis is emphasized.
Finally, conclusion and future work are drawn in Section 6.

2 Literature Review

In this work, searching and exploring an area to find a specific object is defined as an optimization problem. Our goal is to find the object as fast as possible. In the literature, many approaches have been proposed to solve the exploration problem with different purposes such as to minimize exploration time and traveled distance, build maps (Amigoni, 2008), (Gonzales-banos and Latombe, 2002), (Amigoni and Gallo, 2005), find object or search and rescue, (DasGupta, 2004), (Marjovi et al., 2010), (Amanatiadis et al., 2013), (Grady et al., 2012), (Davoodi et al., 2013), etc.

In (DasGupta, 2004), the authors investigate the problem of searching for a hidden target in a bounded region assuming the knowledge of a-priori probability density function. They consider an autonomous agent that is only able to use limited local sensory information. Their goal is to find a path that maximizes the probability of finding the object, given constraint on the time or fuel spent by the searcher. Their solution relies on partitioning the environment into a finite collection of regions on which they define a discrete search problem. However, in general, their solution is not optimal.

Authors in (Grady et al., 2012) studied the problem of multi-objective mission planning for car-like robots. They consider two objectives: a primary objective (moving from point A to point B) and secondary objective (collecting information about a target T found while executing the primary objective). In fact, when a target is discovered, the robot replans a new trajectory to visit the target along its way to the goal region. Furthermore, they take into account the differential constraints on the robot's motion and obstacles.

In (Amanatiadis et al., 2013), a method which is appropriate for real-time search and rescue application was proposed. The authors addressed a twofold challenge of realistic robotic exploration operations that is the ability to efficiently handle multiple temporal goals while satisfying the mission constraints. This paper describes a system that focuses on two different objectives, which are essential for the navigation of mobile robots in unexplored hazardous environments: (i) the development of an accurate 3D reconstruction and registration algorithm suitable to produce dense 3D maps and precise estimations of the robot 's motion and (ii) the integration of a path planning algorithm within the resulted

3D map in order to produce a collision free trajectory. In their approach, they model the environment constraints with cost functions and use the cognitive-based adaptive optimization algorithm to meet time critical objectives during the trajectory calculation.
Some researchers tried to solve the exploration problem with the objective of building maps of unknown environments. A good experimental comparison between different strategies is detailed in (Amigoni, 2008). A common approach for this type of search problem is an iterative solution in which the question "where should the robot go next or where is the Next-Best View (NBV)?" is answered every time step. In (Gonzales-banos and Latombe, 2002), the authors tried to find the Next Best View by incorporating two main features: safe navigation and avoidance of overlap between each new local model in the current map. They also proposed the concept of safe regions: the largest region, which is free of obstacle. Therefore, the candidate for Next Best View is generated across the edge of the explored regions, in which the robot is guaranteed to move without collision risks. This real-time method is built for unknown environments and the result can be considered as a solution of the well-known SLAM (simultaneous localization and mapping) problem. This work is similar to our work in terms of safe navigation and finding points for searching with minimum overlap. In our work, we use a traditional Generalized Voronoi Diagram (GVD) as a roadmap to guarantee safe motion for the robot. The GVD is created offline in the context of path planning considering a known environment. Our proposed approach tries to find points called search points, *SPs*, located at the GVD. Although in (Gonzales-banos and Latombe, 2002), they also aim to guarantee that NBV is found in a safe region, sometimes the points are too close to an obstacle. As one can see in the Figure 2.1, the third NBV is selected in a region free of obstacles but it is too close to the wall.

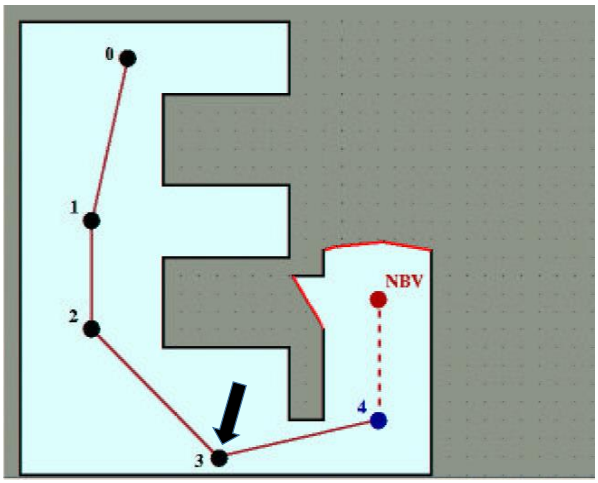


Figure 2.1: NBV computation (Gonzales-banos and Latombe, 2002).

By using the GVD we can guarantee maximization of distance from closest obstacles.
In (Amigoni and Gallo, 2005), a multi-objective exploration strategy for mobile robots has been proposed. This method determines the next best observation position considering three features: the traveling cost, the information gain, and the precision of the localization of the robots. The best observation position is selected by using the concept of distance from ideal solution.
Another common exploration strategy is the so-called frontier based exploration, which was proposed by Yamauchi in (Yamauchi, 1997). Frontiers are defined as the regions on the boundary between the open space and unexplored space (see Figure 2.2). This exploration strategy directs the robot to keep moving towards the nearest unvisited frontier to extend the prior map. Several researchers have tried to improve this method, such as (Dornhege and Kleiner, 2013) and (Juliá et al., 2010). For instance, (Dornhege and Kleiner, 2013) further develop this approach to search for an object in unknown 3D spaces. The proposed method detects frontiers and voids (unexplored volumes) in 3D space to compute next best viewpoints.

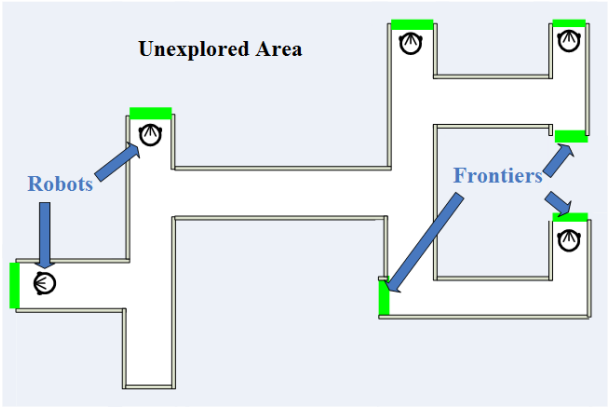


Figure 2.2: Frontier based search and exploration (Marjovi et al., 2010).

Most of the reviewed works can be classified as frontier-based methods. In fact, this exploration method is a good strategy to explore unknown environments. The main idea behind the frontier-based exploration is to gain new information about the environment to create a map of the unknown environment.
Other works in the literature used topological maps (Maohai et al., 2013) or roadmaps, such as (Oriolo et al., 2004), (El-Hussieny et al., 2013), (Freda and Oriolo, 2005), (Franchi et al., 2007) that used a Sensor-based Random Tree (SRT) technique to explore an area. The SRT method is a randomized strategy used to explore unknown environments with mobile robots equipped with range finders. The SRT is a data structure that represents a roadmap of the explored area with an associated safe region. This safe region is built

based on sensor readings (Oriolo et al., 2004). Nodes of this tree are robot configurations or visited explored locations and the Local Safe Region (LSR) found out from that location, while an arc between two nodes represents a collision-free path between the two associated configurations. The SRT is incrementally built by random selection of robot configurations inside the LSR. At first, current configuration q_{curr} and LSR are added to the tree. Then, the direction of movement is determined by generation of a random angle θ_{rand} along distance r. This distance is computed based on LSR, q_{curr} and θ_{rand}. According to this random direction θ_{rand} and distance r, a random candidate configuration q_{cand} will be picked out inside LSR.

In contrast to the mentioned works, in our problem the map is known and all the computation is done off-line. Furthermore, the purpose of exploration is to find an object in an indoor environment. The important requirements in indoor environment exploration are safety and also time limitation to explore the entire map. In fact, we aim to find optimal and safe routes for the robot considering perfectly known environments and a multi-objective framework.

3 Methodology

As it was mentioned before, coverage and exploration of an environment for searching purpose is one of the most important tasks in mobile robotics. This chapter further explains our strategy. In section 3.1, the overall proposed methodology is explained. In the next section (section 3.2), we show about an efficient and robust algorithm for computing safe paths for a mobile robot. This section also presents the proposed method to construct the GVD. Graph representation of the GVD is discussed in section 3.3. The multi-objective solution is introduced to find the good locations for the search points in section 3.4. In section 3.5, we describe a routing strategy to define in which order the search points will be visited. Finally, the SURF algorithm is presented as an object recognition algorithm in section 3.6.

3.1 Proposed Method

We assume we have a robot with some sensors installed on it and the two-dimensional map A of the static environment. As mentioned before the objective is to find a pre specified object in the environment with the given sensors in minimum limited time. We assume sensors with limited sensing range r.
We propose the transformation of the original problem into the one of navigating through some stationary search points (SPs) located at the GVD of the given map. A limited number of search points is important due to the time limitation and the choice of points at the GVD is interesting since this maximizes the covered area by the sensors, as the GVD points maximize the distance from obstacles and consequently minimizes occlusion. Furthermore, GVD provides safe routes for the robot.
The SPs are not placed at the GVD randomly. We solve a multi-objective problem to find the best or near best location of these points. After this step, we compute the route to be tracked by the robot by solving a Chinese Postman Problem (CPP). Since the SPs are placed at the edges of the GVD, by visiting every edge, we can guarantee that all the

SPs will be visited. Whenever the robot reaches a SP, the robot stops moving according to the planned route and perform a careful search for the object in the area.
In general, the whole exploration algorithm can be described by the following algorithm:

Algorithm 1: *Exploration Algorithm*

Input: map, p *//map is the input map, p is the initial configuration of the robot.*

1 $GVDmap \leftarrow Create\_GVD(map)$ // Read the input map and compute its corresponding GVD.
2 $G \leftarrow Create\_Graph(GVDmap)$ // Create the corresponding graph G from GVD.
3 $SPs \leftarrow Find\_SP(GVDmap, G)$ // Run NSGA II and ELECTRE I to find the best *SPs*.
4 $Q \leftarrow CPP(G, p)$; // Solve the Chinese Postman Problem.
5 **while** $(Q \neq \emptyset)$**do**
6 $T \leftarrow Pop(Q)$; *// Pick a single edge from Q* $(Q \leftarrow Q - T)$.
7 **while** *(!End_Of_Edge())* **do**
8 $Move()$; *// Move along the edge in T and update p.*
9 **if** *(Robot reaches a SP and the SP has not been visited)* **then**
10 *Execute_Search // Execute a precise searching with SURF algorithm on the current SP.*
11 **if** *(Object_Is_Found())* **then**
12 *Exit*(); *// The algorithm is finished.*
13 *Return_Failure*; *// Object was not found.*

The inputs of the algorithm are a map and the initial robot configuration. After computing the corresponding GVD of the input map, the graph G is constructed by defining the GVD meet points and end points as the graph nodes and the GVD edges as the graph edges. Next, NSGA II and ELECTRE I are applied to find the set of search points. In line 4 the function $CPP()$ solves the chinese postman problem by using the algorithm proposed in (Pearson and Bryant, 2004) and as a result a route Q is designed. This solution could be improved by using a solution of a Traveling Salesman Problem (TSP) to obtain directly the optimal sequence of search points to be visited. However, due to the exploration time limitation we intend to use a time efficient route planner which works even for the case where the number of *SPs* is large. Thus, we propose to solve the CPP instead of a TSP since this can be done in polynomial time using Edmonds'matching algorithm (Edmonds and Johnson, 1973).
In the loop starting in line 5, the robot moves from edge to edge, searching for the object

at every *SP*, until the object is found or the list of edges is empty. It should be clear that in line 8 the function $Move()$ enforces the robot to move constrained to the GVD edges to guarantee safety. In line 9, when the robot reaches an unvisited *SP*, it executes a careful search by rotating in place and moving its sensors accordingly. It should be noted that the input graph may not be Eulerian, which means that in the CPP route there might be an edge that is visited more than once. Hence in order to avoid repeating precise searching on a *SP*, we check if the *SP* has been explored before or not.

3.2 Generalized Voronoi Diagram

The Generalized Voronoi diagram (GVD) is one of the most famous roadmaps for navigation. One of the main advantages of this roadmap is safety that can be applicable in exploration of cluttered environments. Indeed in this kind of environment collision avoidance can be vital when the robot must move through the map to find the object. The definition of GVD is given in the next lines.

Let the set A defines the free configuration space as defined in (Choset, 2005). The Voronoi diagram is a structure that induces a partition of A dividing this map into zones, called Voronoi regions. Each region has a corresponding point inside which is called seed or site. The set of these points will be given by $X = \{x_1, x_2, ..., x_n\}$. The formal definition of Voronoi region is:

$$R_i = \{p \in A | d(x_i, p) \leq d(x_j, p), \forall i \neq j\} \tag{3.1}$$

where $d(x_i, p)$ is the distance between p and x_i which is the region site.

The ordinary definition of Voronoi region can be extended by considering the seeds to be sets instead of single points. More specifically, we consider the sets induced by the obstacles. The GVD is defined as the set of points where the distance to two closest obstacles is the same (Choset, 2005).

The so-called two equidistant face is given by:

$$T_{ij} = \{p \in SS_{ij} | d(p, \mathcal{QO}_i) \leq d(p, \mathcal{QO}_h), \forall h\}, \tag{3.2}$$

where $d(p, \mathcal{QO}_i)$ represents the distance between the point p and the closest point of the obstacle $\mathcal{QO}_i$ and SS_{ij} is defined by:

$$SS_{ij} = \{p \in A | d(p, \mathcal{QO}_i) = d(p, \mathcal{QO}_j) \; and \; \nabla d(p, \mathcal{QO}_i) \neq \nabla d(p, \mathcal{QO}_j)\}. \tag{3.3}$$

Now, the definition of the GVD can be given:

$$GVD = \bigcup_i \bigcup_j T_{ij}. \tag{3.4}$$

An example of a simple map with its corresponding GVD (green lines) is shown in Figure 3.1.

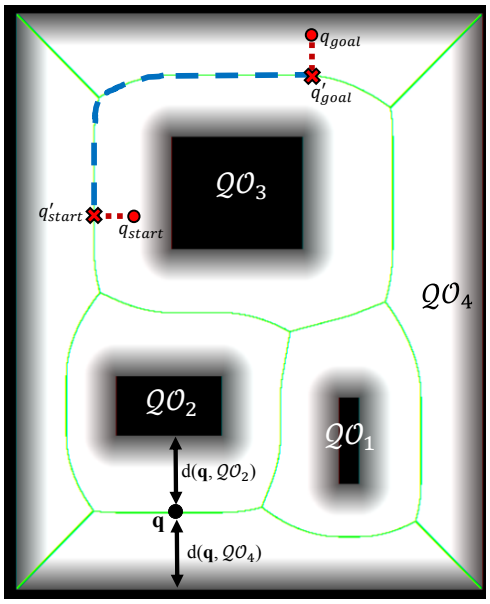


Figure 3.1: An input image, corresponding GVD which is formed by green lines. The black shapes show the obstacles ($\mathcal{O}$) and the remaining space is free space ($\mathcal{Q}_{free}$).

Let q_{start} be a start configuration and q_{goal} be a target configuration. Let also $\mathcal{Q}_{free}$ be the free configuration space. As any roadmap (RM), the GVD has three main properties as follows (Choset, 2005):

- Accessibility: there exists a path between any $q_{start} \in \mathcal{Q}_{free}$ and some $q'_{start} \in RM$.
- Departability: there exists a path from some point on the RM,$q'_{goal} \in RM$, to $q_{goal} \in \mathcal{Q}_{free}$.
- Connectivity: there exists a path in RM between q'_{start} and q'_{goal} .

The GVD can be used as a safe route (since it maximizes distance between obstacles) to connect any two configurations in the free space. This can be done easily by enforcing the robot to move from an initial configuration q_{start} to a point in the GVD, q'_{start}, finding a route to the point q'_{goal} also in the GVD and finally guiding the robot from q'_{goal} to the target q_{goal} (see Figure 3.1).

3.2.1 GVD Construction

There are several techniques to compute a GVD. For example, in a discrete grid, Brushfire (Choset, 2005) and wavefront (Zelinsky et al., 1993) are two useful methods. In this work, the environment map (2D) is considered as an image, therefore in order to create the GVD on this image, a morphological approach is used. Morphological operators include a set of operators such as Dilation, Thinning, Skeletonization, Erosion and so on (Gonzalez and Woods, 2001). Usually the combination of these operators are used to give different outputs. But before creating the GVD, it is necessary to compute free configuration space ($\mathcal{Q}_{free}$), where the robot can move freely without colliding with obstacles. The common solution for computing the free configuration space is to construct the *configuration space obstacle*, $\mathcal{QO}$. This is done by growing the obstacles by the size of the robot. As an example Figure 3.2(left) shows a two-dimensional workspace, $\mathcal{Q}$, includes an obstacle, $\mathcal{O}$. Also the result of growing a polygonal obstacle by the size of a circular robot is depicted in this figure (middle and right). After computing $\mathcal{QO}$, $\mathcal{Q}_{free}$ is given by:

$$\mathcal{Q}_{free}=\mathcal{Q}\setminus(\bigcup_i \mathcal{QO}_i).$$

It should be mentioned that circular robot in the workspace is equivalent to a point robot in the configuration space.

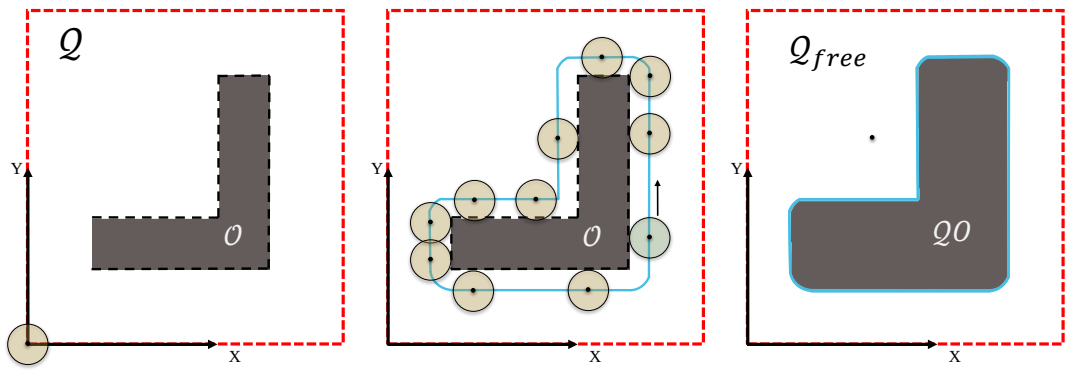


Figure 3.2: Result of the growth of a polygonal obstacle by the size of a circular robot.

Since the map is represented as an image, in our work in order to compute ($\mathcal{Q}_{free}$), we use dilation operator to grow obstacles with size of the robot. In dilation, the structuring element has a vital role in the result. The robot shape is considered as the structuring element to compute the free configuration space ($\mathcal{Q}_{free}$). In this work a circular robot is defined (see Figure 3.3).

Figure 3.3: Disk shape robot.

Before representing more example of the dilation operator on an input map with circular structure element, a definition of the dilation operator is explained as following: Suppose A is a binary image and B is the structural element. Dilation of A by B is defined as:

$$A \oplus B = \{x|(\hat{B})_x \cap A \neq \emptyset\}. \tag{3.5}$$

The basic effect of the dilation operator on a binary image is to gradually enlarge the boundaries of regions, which is applicable to compute $\mathcal{QO}$.
As it can be seen in figure 3.4, in the "original" image, the black regions show obstacles. The image "complement" or negative is computed to be used as the input of dilation operation. In the "configuration space obstacle" image, the white region represents the configuration space obstacle ($\mathcal{QO}$) where robot collides with obstacle. By complementing this image the final result of "free configuration space" is achieved, where the robot can freely move in the map, without having collision with obstacles.

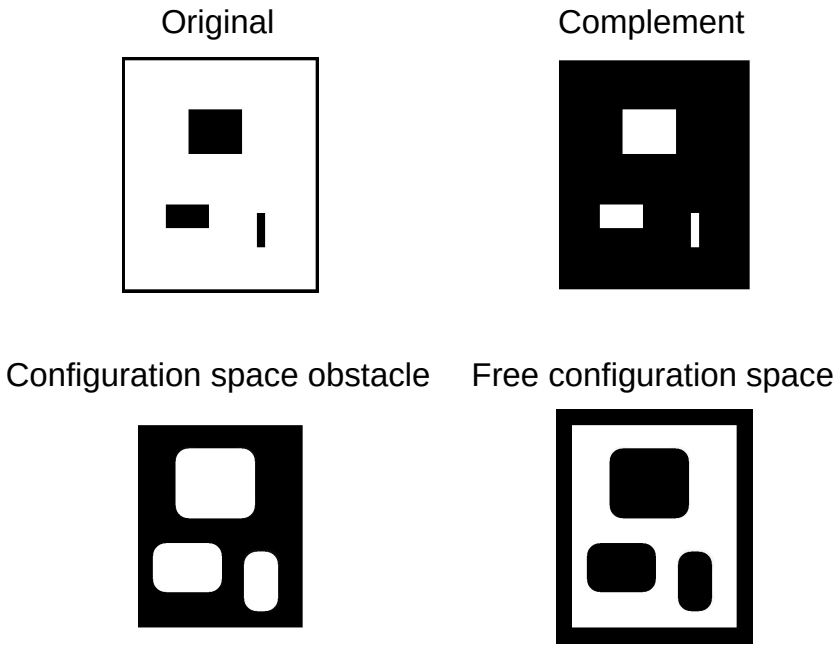


Figure 3.4: An example of input map and its corresponding $\mathcal{Q}_{free}$

Afterward, skeletonization operator is used to construct the GVD from $\mathcal{Q}_{free}$. The informal definition of a skeleton is a line representation of an image that is one pixel thick, through the "middle" of the image, and preserves the topology of the image. Figure 3.5 shows the skeletons for the polygonal image. As can be seen, different size of disk is used to create circular structuring element, therefore the result of skeleton are different respectively. More information about this operator can be find in (Gonzalez and Woods, 2001).

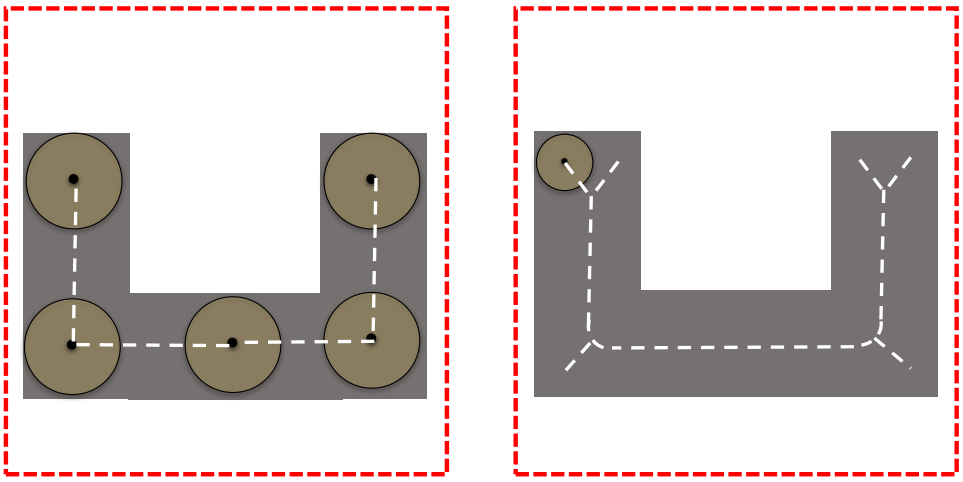


Figure 3.5: The white lines represent skeleton of the image.

Given the above explanation, skeletonization operator is applied to find the skeletal of the map which is demonstrated to be the GVD (Garrido et al., 2011). The Pseudo code of making the GVD is shown below:

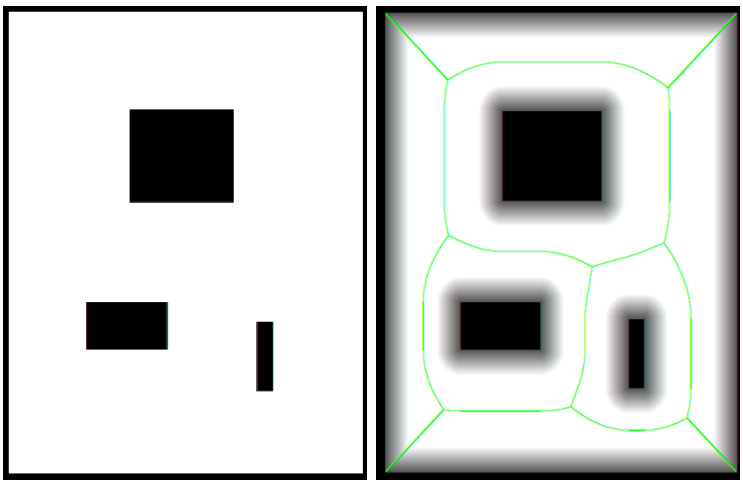
Algorithm 2: GVD maker

Input: M, R // where M is input map and R contains robot shape and size

Output: GVD

```
M2 = Complement Image(M)
M3 = Dilate image(M2,R);
M4 = Complement Image(M3)
GVD = Image Skeleton(M4)
```

Figure 3.6 shows the results obtained from skeletonization techniques of the given map. In this figure, the junction of obstacles and the shaded area represent configuration space obstacle. Moreover, the skeleton of the input map or the GVD is depicted in figure 3.6 (b) with green color.



(a) Input map (b) GVD demonstration

Figure 3.6: The green lines are the skeleton of the map (GVD).

3.3 GVD Induced Graph

A graph $G = \{V, E, C\}$ is induced from the GVD by considering Meet Points (where more than 2 GVD curves meet each other) or End Points (where curves terminates) as graph nodes and curves as graph edges. The set of vertices (nodes) is given by V, the set of edges by $E \subseteq V \times V$, and a cost function is denoted by $C : E \to R$. The cost represents the distance between two vertices.
In Figure 3.7 the corresponding graph of the map in Figure 3.6 is illustrated.

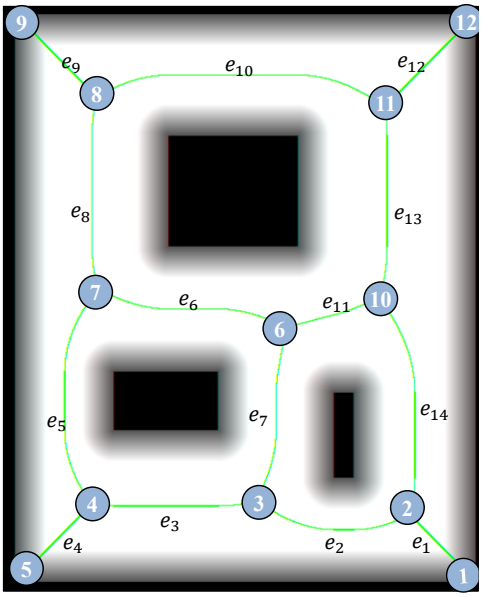


Figure 3.7: A graph with 12 nodes and 14 edges.

3.4 Multi-Objective Optimization

A multi-objective optimization problem involves multiple objective functions subject to a set of constraints. Such a problem can be described in mathematical terms as follows:

$$\begin{aligned} min(f_1(x), f_2(x), ...f_m(x)) \\ s.t, x \in S \end{aligned} \tag{3.6}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a vector of variables, $f_i(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ are scalar functions, $m > 1$ is the number of objectives, and S represents the set of feasible solutions, which is defined by the satisfaction of the problem constraints:

$$S = \{x \in R^n : h_j(\mathbf{x}) = 0, g_k(\mathbf{x}) \geq 0\}$$

In contrast to the single objective problem, the scalar concept of "optimality" does not apply directly in the multi-objective setting. In fact, instead of a scalar value solution, there exists a set of so-called Pareto optimal solutions. Decision making methods are needed to select one of the solutions in the Pareto set.
Different interactive and evolutionary based algorithms have been proposed to solve multi-objective problems (Branke et al., 2008). An interesting multi-objective evolutionary algorithm (MOEAs) is the Non Dominated Sorting Genetic Algorithm II (NSGA-II) (Deb and Pratap, 2002). As its name is evident, this algorithm has two parts: the first part is related to the use of non-domination rules and the second part is related to sorting a genetic population according to different preferred levels.
Domination will happen if a solution is better than the other in at least one objective and equals in other objectives. Mathematically,

$$f(s_1) \prec f(s_2),$$

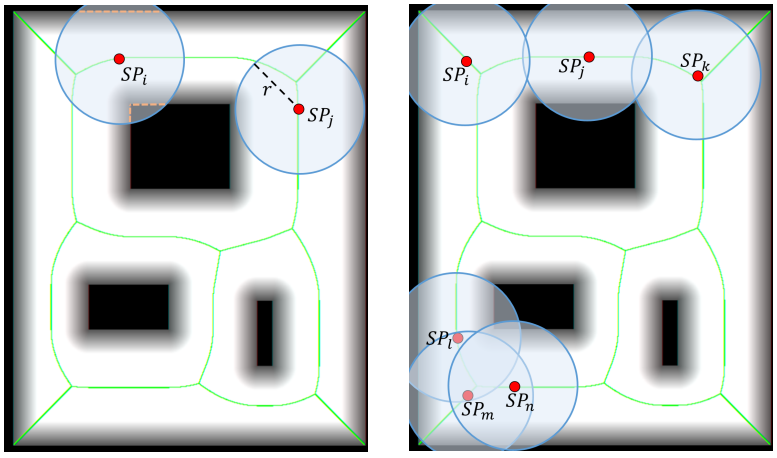
where $f()$ is a vector function $f = [f_1, f_2, ..., f_n]^T$and $s_1, s_2 \in S$, which is the parameter space. In this case: $f_i(s_1) \leq f_i(s_2)$,$\forall i$ and $f_j(s_1) < f_j(s_2)$ for at least one index j.
If the solutions do not dominate each other, we say that they are non-dominated or incomparable.
NSGA has an important advantage which is the fact that it typically generates sets of solutions, allowing the computation of an approximation of the entire Pareto front. The

disadvantages are the high computational complexity of non-dominated sorting and the lack of elitism.
NSGA-II is an improved version of NSGA and in this version, the disadvantages of NSGA is improved. Therefore, we used NSGA-II in this work.

3.4.1 Multi-objective Problem

As previously mentioned, in the present work, a robot must search for a specific object on the map. The main problem here is that we assume we cannot wait for the complete search of the given environment. Therefore, our solution is to compute a finite number of good view points in the environment and then move the robot through these points in order to find the desired object. We define the multi-objective problem to define where the SPs should be placed. We consider the following objectives:

1. First objective: maximum covered area from the SPs.
   As a first objective, we try to maximize the area of the map viewed by the sensors. As a result, since the object can be anywhere in the free environment, the probability of finding the object is maximized in this objective. We assume the robot is equipped with sensors and is able to move so that the sensor footprint can be modeled by a circle centered at the robot position with radius r (see Figure 1.1).



(a) In contrast to SP_i, in SP_j the robot has a maximum visible range.

(b) SPs i, j, k have less overlap (Higher F2 value) than SPs l, m, n.

Figure 3.8: A representation of objective functions on the map.

We want to find SPs in places where the robot's sensor has large visible range. The

visible range is determined by the area viewed by the sensor after subtracting the portion occupied and occluded by obstacles (see Figure. 3.8 (a)). The first objective is given by:

$$F_1 = \sum_{i=1}^{n} A(SP_i), \tag{3.7}$$

where n is the number of *SPs* and $SP_i \in GVD$. $A(SP_i)$ is the covered area from SP_i i.e, it is the visible range. This objective function must be maximized.
Since the GVD is the set of points that maximizes the distance from obstacles it is interesting to note that by constraining the feasible set to this one dimensional set, this not only reduces the dimension of the search space, but also helps in the visible range maximization and provides safety in the robot motion.

2. Second objective: good distribution of search points.
Overlapping is also another problem to be avoided when defining the positions of the *SPs* in order to provide efficiency in the search. Thus, we require that the distance between *SPs* be maximized. In order to deal with this problem, we define the second objective as follows:

$$F_2 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \parallel SP_i - SP_j \parallel . \tag{3.8}$$

In the equation above, $F2$ is the sum of Euclidean distances between all pairs of SPs. Likewise the first objective, our second objective function ($F2$) must be maximized. According to the definition of good distribution, SPs i,j,k are better distributed than l,m,n in Figure 3.8.(b). In fact, the second objective aims to provide a more uniform distribution of points over the map which is useful to avoid sensor footprint overlaps.

3.4.2 Multi-objective Solution

As mentioned before, NSGA-II applies the principles of non-dominated sorting to direct the population toward the Pareto-optimal regions. The entire process to achieve the Pareto-front is shown in the flowchart of Figure 3.9.

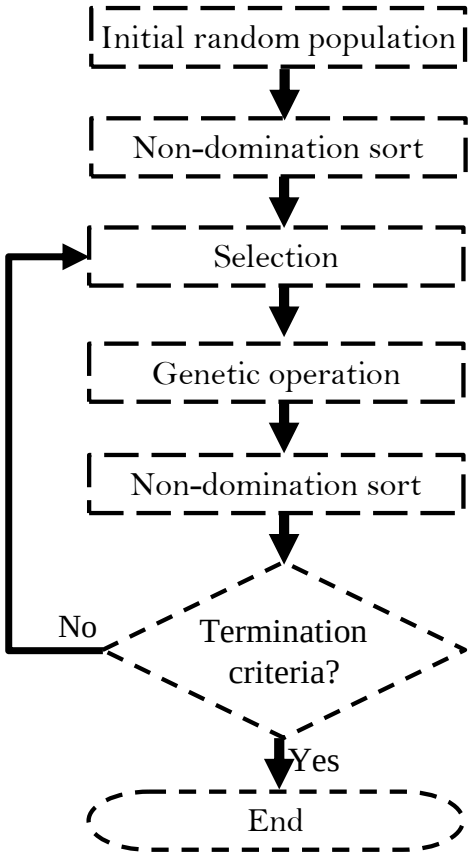


Figure 3.9: Flowchart of NSGA II algorithm.

In the following, we show some details.

a)Representation

Each chromosome contains a set of SPs, and each gene refers to a position on the GVD, $p_i \in GVD$. The table below illustrates the chromosome representation.

p_1	p_2	...			p_n	F_1	F_2

Table 3.1: Chromosome representation.

In Table 3.1, $F1$ and $F2$ are the values of the objective functions and n is the maximum number of points in the chromosome.

b) Initialization

In the first step, an initial population is created randomly. Indeed, the first population is a set of SPs that is sampled randomly over the GVD. Figure 3.10 presents an example of initialization.

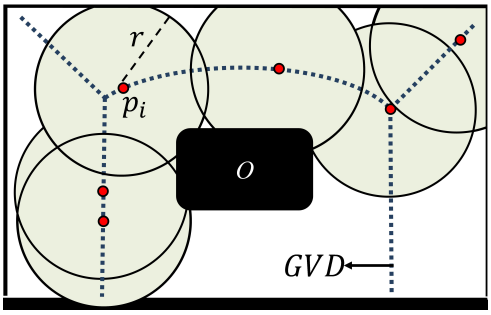


Figure 3.10: Red points represent random *SPs*.

c)Evaluation and non-dominated sorting

Each chromosome is evaluated by computing its objective functions. Then, non-dominated sort ranks chromosomes based on their objectives. In addition the crowding distance is computed. The crowding distance is relative to the closeness of each individual to its neighbor (Deb and Pratap, 2002).

d) Operators

Genetic operators are usually applied to generate children. Genetic algorithm includes two basic operators: Crossover and Mutation.

In this work, we used two point crossover, in order to create new children from inheriting and merging the properties of two parents. This is illustrated in Figure 3.11.

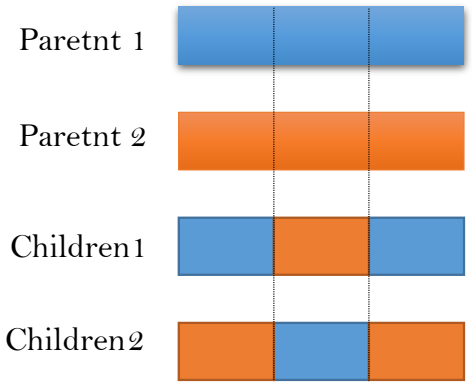
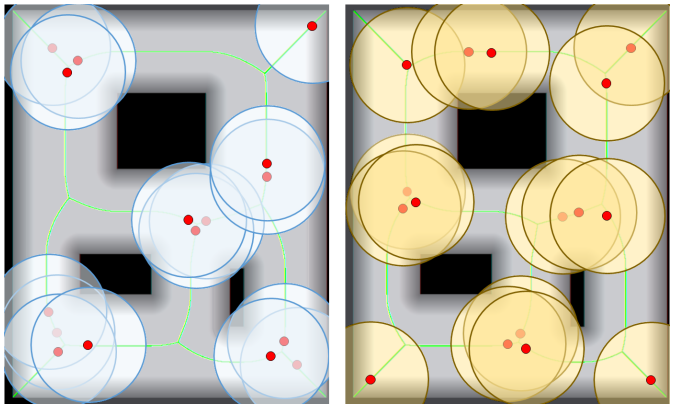
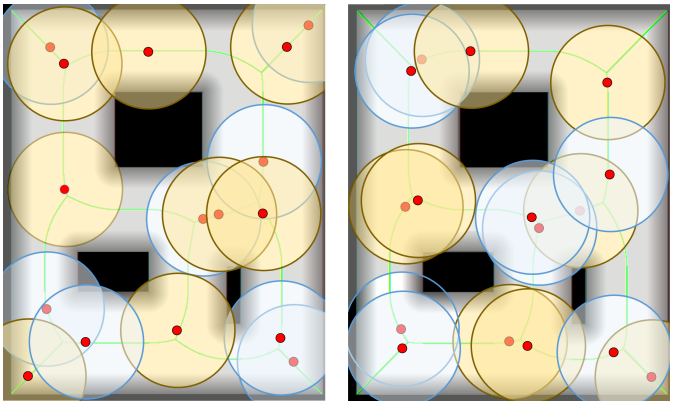


Figure 3.11: Two point crossover.

An example of two point crossover operator is depicted in Figure 3.12, where two random parents are selected and after applying the crossover two new children are created. In this example we also show how better are the new children since the uncovered area is minimized.



(a) Two random parents.



(b) Two created children after operating crossover.

Figure 3.12: An example of crossover. Uncovered area has changed (decreased) in the new generation.

Different versions of mutation operator have been proposed for different situations. Here, in order to have a good exploration through the map, one gene is selected randomly and it is replaced by a new random point *SP*. This operator guarantees the variety of new generation.

e) Selection

Before selection, the parent population and children concatenate together and they are sorted based on the non-dominated sort algorithm. After ranking, just the N chromosomes are selected as elitism for the next generation. Then, if the output population satisfies the stopping criteria, then the final population can be reported. If none of these conditions are verified, a new iteration starts.

3.4.3 ELECTRE I

Choosing one solution over the set of solutions is also a challenge in MOEAs. In order to solve this challenge, decision making techniques are usually applied. ELECTRE I is the multi-criteria decision making method applied to choose the best solution in our work (Shanian and Savadogo, 2006). The Elimination and Choice Translating Reality (ELECTRE) method was first introduced in (Shanian and Savadogo, 2006). It is one of the most extensively used outranking methods reflecting the decision makers preferences in many fields. The ELECTRE I approach was then developed by a number of variants (Bojković et al., 2010).

This method is used to rank a set of alternatives and also to analyze the data of a decision matrix (Shanian and Savadogo, 2006). This method is based on comparisons pairs of all alternatives, which has lead to the make a partial ordering of options according to preference of decision maker.The method to form the final rankings uses two matrices:concordance matrix and discordance matrix.

In concordance matrix, values calculated for each pair of criteria that inform the extent to which one alternative is at least as good as the other;

In discordance matrix, values calculated for each pair of criteria that inform the extent to which one alternative is worse than the other.

At the end, we define the relationship between the alternatives. We create a ranking based on the difference amount of alternatives that exceeds alternative, and those that exceed it. Ranks the difference from the highest to the lowest.

3.5 Chinese Postman Problem

As previously mentioned, in our solution the robot must visit the search points to find the desired object. Therefore, it should be clear the necessity of efficiently solving this routing problem. In this work, we are going to use an algorithm that solve the so-called Chinese Postman Problem.

Meigu Guan(or Kwan Mei-Ko), was a Chinese mathematician who proposed the Chinese postman problem(CPP) (Eiselt et al., 1995). Guan was interested in finding out how a postman could cover assigned segments at least once with minimum walking distance.

Considering a graph G, in the Chinese postman problem, the main objective is to find the shortest tour such that each edge is traversed at least once. If the graph has an Eulerian cycle (a closed path that visits every edge once), that route is an optimal solution.

In the case of an undirected connected graph, a necessary and sufficient condition for an Euler cycle to exist is that the graph contains no node of odd degree. Such a graph is called *Eulerian graph* (Pearson and Bryant, 2004). Given an Eulerian graph, it is possible to find an Euler cycle in linear time by using the Hierholzer's algorithm (Fleischner, 1991). In the general case of non Eulerian graph, optimal routes for the Chinese Postman Problem can be found with complexity $O(N^3)$, where N is the number of nodes. The algorithm in this case consists of finding the nodes of odd degree, finding a minimum length pairwise matching of the odd-degree nodes, adding to the graph the new edges of the shortest paths between the two nodes in each of the pairs given by the minimum length pairwise matching, and finally finding the Euler tour in the modified graph which is now Eulerian (Larson and Odoni, 1981).

A common way to formulate the CPP is to seek a least-cost augmentation of G into $G\prime$ such that all nodes of $G\prime$ have an even degree. Consider x_{ij} as the number of repeat of (v_i, v_j) required to add to G and let $T \subseteq V$ be the set of odd nodes of V and define $\delta(i)$ as the set of edges incident to v_i. The formulation is as follow (Eiselt et al., 1995):

$$Minimize \sum_{(v_i,v_j)\in A} c_{ij}x_{ij}, \tag{3.9}$$

subject to

$$\sum_{(v_i,v_j)\in\delta(i)} x_{ij} \equiv \begin{cases} 1 \text{ (mod 2) if } \mathrm{v}_i \in T \\ 0 \text{ (mod 2) if } \mathrm{v}_i \in V \setminus \mathrm{T} \end{cases} \tag{3.10}$$

$$x_{ij} \quad \in 0, 1((v_i, v_j) \in A) \tag{3.11}$$

Algorithm 3 represents the sequence of functions to find the final route:

Algorithm 3: Chinese postman algorithm (Pearson and Bryant, 2004)

Input: G // G is the graph

Output: Q //shortest closed route

1 Find and list all odd vertices in G.

2 Find and list all possible pairings of odd vertices (from step 1).

3 Find an edge for each pairing that connect the vertices with the minimum weight.

4 Find the pairings such that the sum of the weights is minimized.

5 On the input graph G add the edges that have been found in step 4. // The length of an optimal chinese postman route is the sum of all the edges added to the total found in step 4.

6 A route Q corresponding to this minimum weight can be found with Fleury 's algorithm.

After building the new graph, the Fleury 's algorithm (Flurt, 1883) may find an Eulerian cycle which is the optimal solution for the CPP.

Fleury 's algorithm is a method which, if followed, is guaranteed to produce an *Eulerian tour* in a connected graph with no vertices of odd degree. In a connected graph, a *bridge* is defined as an edge which, if picked up, a disconnected graph is generated. (See Algorithm4)

Algorithm 4: Fleury's Algorithm for finding an Eulerian tour (Wilson and Watkins, 1990)

Input: G // where G is an *Eulerian graph.*

Output: //*current trail* is an *Eulerian trail.*

1 Select an arbitrary vertex v_i of G;
set *current vertex*=$\{v_i\}$; *current trail* =$\{\}$.

2 cnt=1 // cnt is counter for edge number(edgenum).

3 **while** *(cnt* $\leq$ *edgenum)* **do**

4 Choose any edge (v_i, v_j) incident at v_i of *current vertex* which is not *bridge* unless there is no alternative.

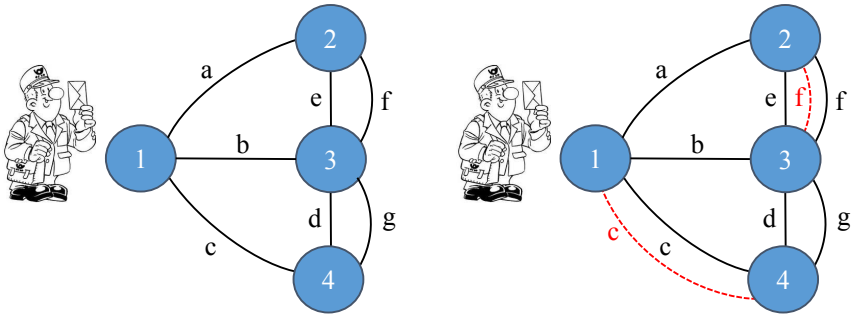
5 Add (v_i, v_j) to the *current trail*;
$cnt = cnt + 1$;
set the *current vertex* $= \{v_j\}$;

6 Remove (v_i, v_j) from the graph; Remove any isolated vertices.

After Initialization, while loop repeats until all edges have been deleted from G. The final *current trail* is an *Eulerian trail* in G.

As an example in Figure 3.13, suppose a robot is initialized on node 1, by applying CPP

on this graph , a possible result can be as follows:



(a) A graph with 4 nodes.

(b) Adding two virtual edges to two odd nodes.

Figure 3.13: Postman problem example.

$Q = \{a, e, f, f, b, c, d, g, c\}$,
where, Q includes a sequence of edges that indicates the Eulerian cycle.

3.6 Speeded Up Robust Features Algorithm

Object recognition is also an important problem in robotics. Hence applications of mobile robots require not only the ability to move around in the environment and avoid obstacles, but also the ability to detect and recognize objects and interact with them.
Most object recognizer/detector methods are based on two terms: training images and query images. Training images are the images which the detector uses to learn information. Query images are the images from which the detector, after learning, is supposed to detect object(s). In this work, we want to use a robust object recognition method to be invariant in terms of scale and rotation. It means when the object in query image is at a different size or angle from the training images, the method can still recognize the object.
The Speeded Up Robust Features, in short (SURF) is a robust object recognition method that is a scale- and rotation invariant detector and descriptor (Bay et al., 2008). The SURF algorithm is mainly divided into three phases (Huijuan and Qiong, 2011): interest point detection, interest point descriptor and interest point matching. The main motivation of this work to select SURF is its fast interest point detection , its distinctive interest point description, and its speeded-up descriptor matching. SURF algorithm is also invariant to common image transformations such as: image rotation, scale changes, illumination change, small change in viewpoint (Adel et al., 2014). Next, we will explain the three

phases of SURF algorithm.

- **Interest point detection:** In this phase SURF algorithm finds points which are in a special position in image such as corners, blob or spot and T-junction. The most valuable feature of interest point detection is repeatability which represents the reliability of detector in terms of finding same physical interest points in different viewpoints. In order to detect the interest points in image, the algorithm uses Hessian matrix approximation because of its good performance in accuracy (Bay et al., 2008).
  Suppose x=(x, y) is a point in image I, the Hessian matrix $H(\sigma, x)$ in x at scale σ is defined as follows:

$$H(\sigma, x)= \begin{bmatrix} \mathrm{L}_{xx}(\sigma, x) & \mathrm{L}_{xy}(\sigma, x) \\ \mathrm{L}_{xy}(\sigma, x) & \mathrm{L}_{yy}(\sigma, x) \end{bmatrix},$$

  where $L_{xx}(\sigma, x)$ is the convolution of the Gaussian second order derivative with the image I in point x, and similarly for $L_{xy}(\sigma, x)$ and $L_{yy}(\sigma, x)$.

- **Interest point descriptor:** This phase is divided in two steps: The first step consists in determining the orientation of interest points. Then in second step, descriptor uses Haar wavelets in a suitably oriented square region around the interest points to find intensity gradients in the X and Y directions. As the square region is divided into 16 squares for this, and each such sub-square yields 4 features, the SURF descriptor for every interest point is 64 dimensional.

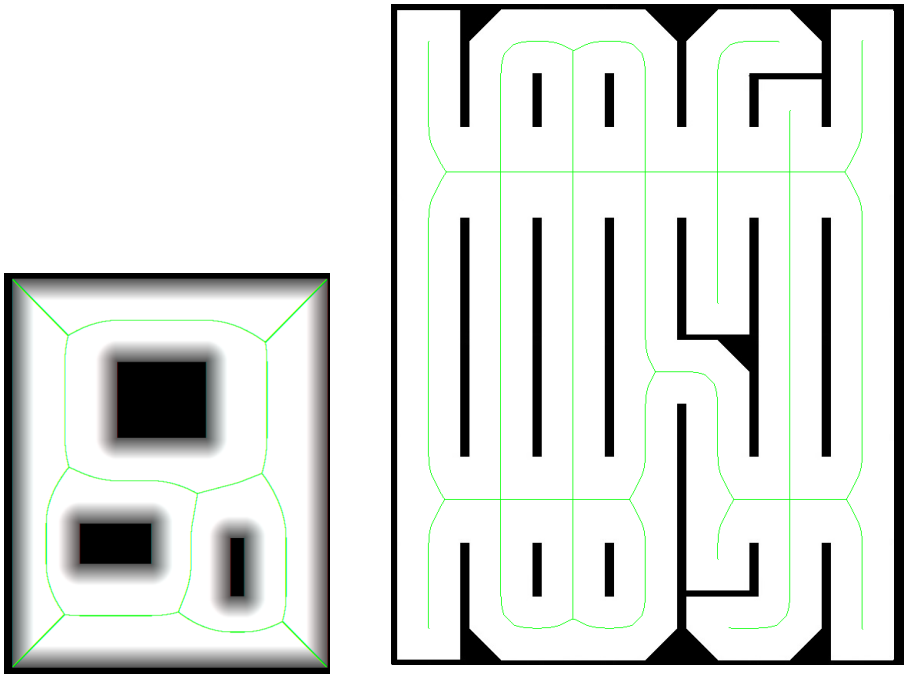
- **Interest point matching:** SURF algorithm uses K nearest neighbor (KNN) search for matching. A KNN is based on distance between a query descriptor and all of the training descriptors, and returns the K pairs with lowest distance.

4 Simulation and Experimental Results

In this section, first we show the two maps considered for our tests and also the corresponding GVD. Second we show the multi-objective solution to find the *SPs*. Simulation on ROS stage is presented in section 4.3. Next, we show how SURF algorithm could recognize the object in different tests. Finally, an experiment with a real robot is presented to validate the method.

4.1 Computing the GVD

As explained before, in order to create the GVD on input image, we used morphological operators such as dilation, skeletonization. We consider two different maps, a simple one and a more complex map (see Figure 4.1). As it can be seen, the corresponding GVD of the two maps are depicted with green lines. The simple map is illustrated in Figure 4.1.(a) and its size is : 530×640 *units*. The complex map is shown in Figure 4.1.(b) and its size is 926×1194 *units*.



(a) A simple map. (b) A complex map.

Figure 4.1: Two scenarios with their corresponding GVD.

4.2 Multi-Objective Solution in MATLAB

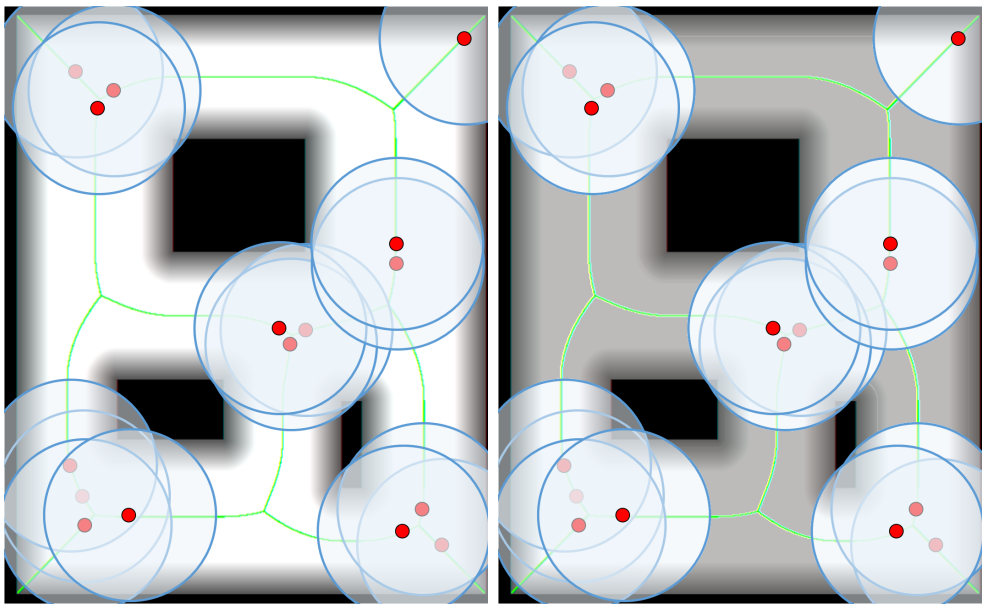
Our solutions were found using MATLAB on a computer with 4 GB RAM and CPU Core i 7.

In our simulation, we set the population size or chromosome number to 30. The dimension of each chromosome for simple map and complex map are respectively 16 and 60. It is significant to mention that the dimension of chromosomes or the number of SPs is related to the maximum sensor range. In other words, when this range (r) is small, the number of *SPs* should be large to cover the map. We defined a robot equipped with a laser sensor with maximum range equal to 10 meters ($r = 10$).

In each generation, 50% of the population is selected as elitism. The crossover probability, P_c, is a noticeable parameter in this test and it is set to 70% (mutation ratio or P_m is equal to 30%). P_m is high because the only way of generating new points along the GVD. And also the maximum number of iterations is equal to 50. In order to use ELECTRE I, we must assign weights to the objective functions. In this experiment we considered that the two objectives are equally important and due to this we assigned equal weights 0.5.

4.2.1 Simulation Results:

This section presents one of the initial populations, which is generated randomly in a simple map. In Figure 4.2.(a) , the *SPs* were not well distributed and there are overlaps. Figure 4.2.(b) shows that a big area could not be covered by the robot. Hence, if the object is placed in this big area, the robot can not find it.

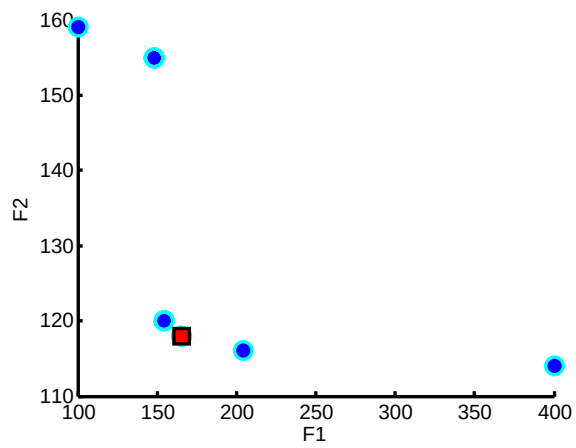


(a) Initial population.

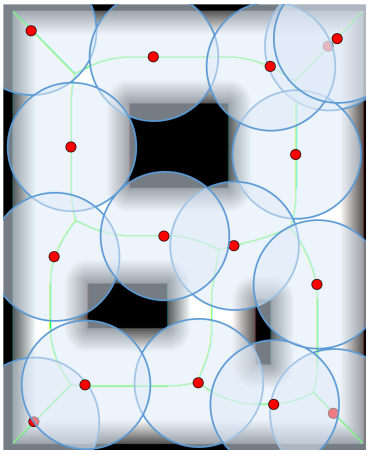
(b) Uncovered area from initial population is highlighted in gray color.

Figure 4.2: Initial population, its covered and uncovered region.

After running the NSGA-II on the simple map, a non-dominated set is obtained. In order to select one of the solutions, ELECTRE I is applied as our decision making tool. The non-dominated set and selected solution by using this technique are presented in Figure 4.3.(a). Because of the overlaps between solutions, just six solutions (out of 15 solutions) are visible in this figure. Since the algorithm selects *SPs* among the set of GVD points, these overlaps happen due to the similarity and limitation of selection in each iteration. In other word, the algorithm converges on these set of solutions. Accordingly, Figure 4.3.(b). shows the distribution of the *SPs* of this selected solution, which has minimum overlap and are well distributed as desired.



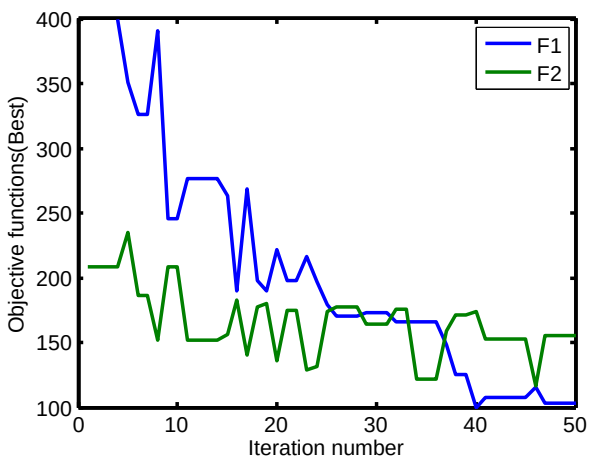
(a) Red square is selected as a Pareto optimal point by ELECTRE I.



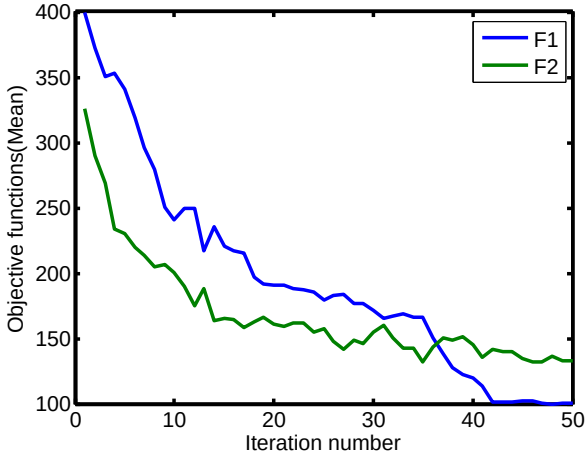
(b) Final population with minimum uncovered area

Figure 4.3: Result of multi-objective optimization for the simple map.

Figure 4.4.(a) presents the cost of objective functions $F1$ and $F2$ with respect to iteration number. At each iteration, the values of objective functions for the best solution are selected. This best solution is selected based on ELECTRE I. Figure 4.4.(b) also shows the mean values of objective functions in each iteration. Both plot indicate the progress of the algorithm in minimizing the objectives during iterations.



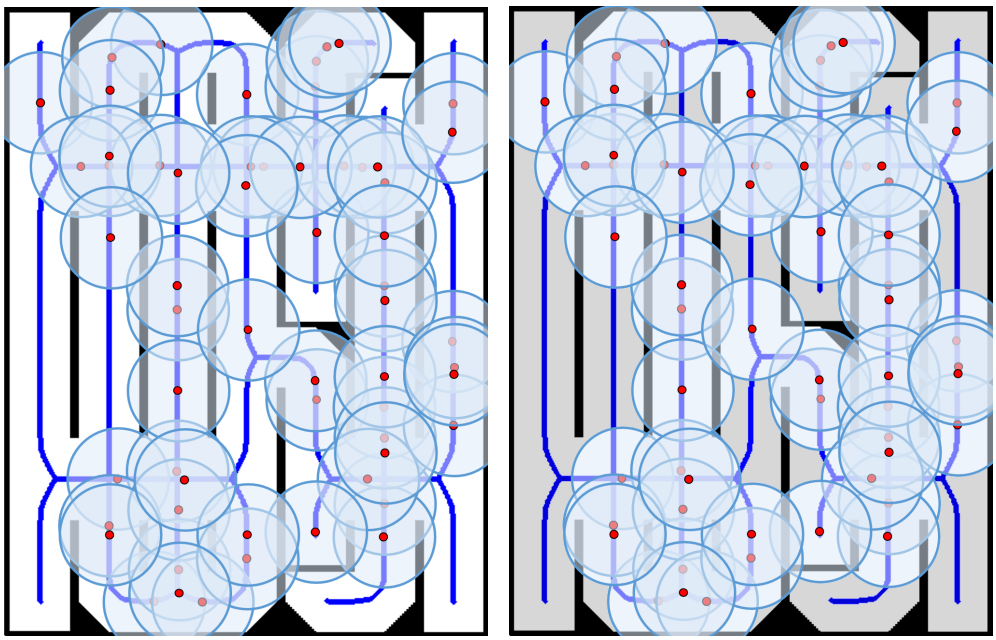
(a) The values of objective functions for best solution with respect to iteration number.



(b) The mean values of objective functions with respect to iteration number.

Figure 4.4: The value of objective functions in iterations.

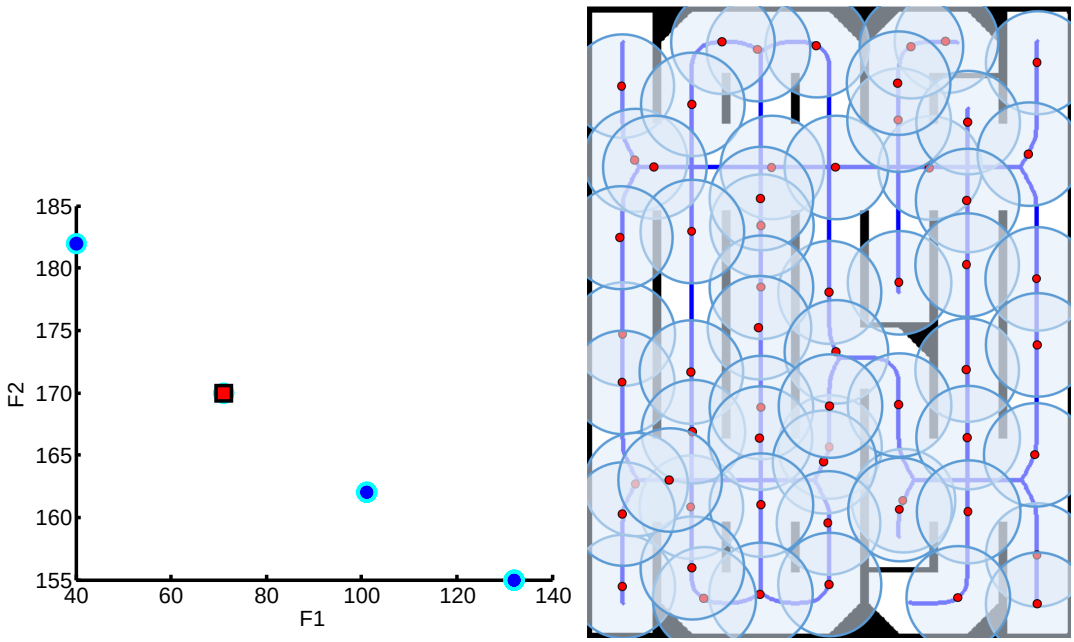
The corresponding results for the complex map are shown in Figs. 4.5, 4.6 and 4.7.



(a) Initial population.

(b) Uncovered area from initial population is highlighted in gray color.

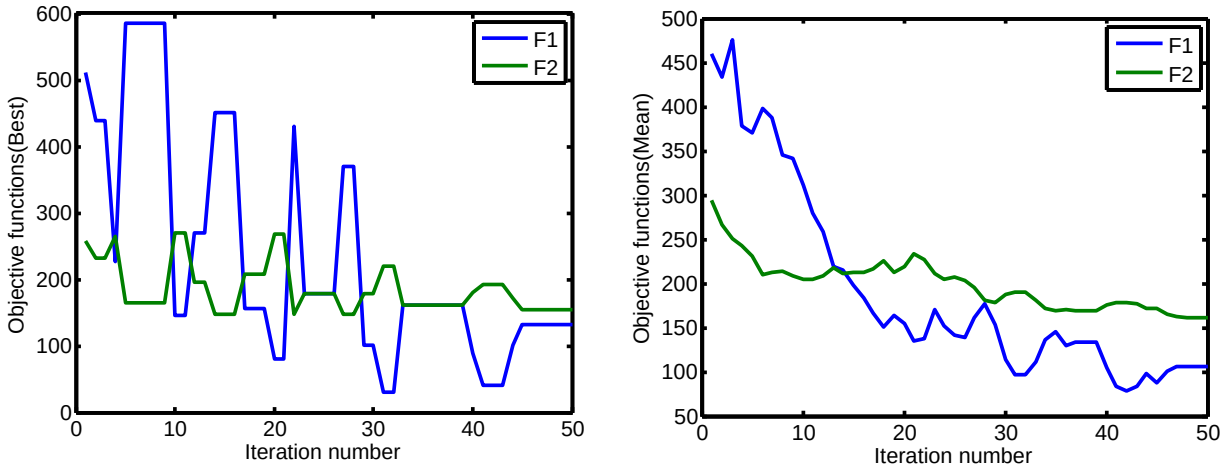
Figure 4.5: Initial population, its covered and uncovered region.



(a) Red square is selected as a Pareto optimal point by ELECTRE I.

(b) Final population with minimum uncovered area

Figure 4.6: Result of multi-objective optimization for the complex map.



(a) The values of objective functions for best solution with respect to iteration number.

(b) The mean values of objective functions with respect to iteration number.

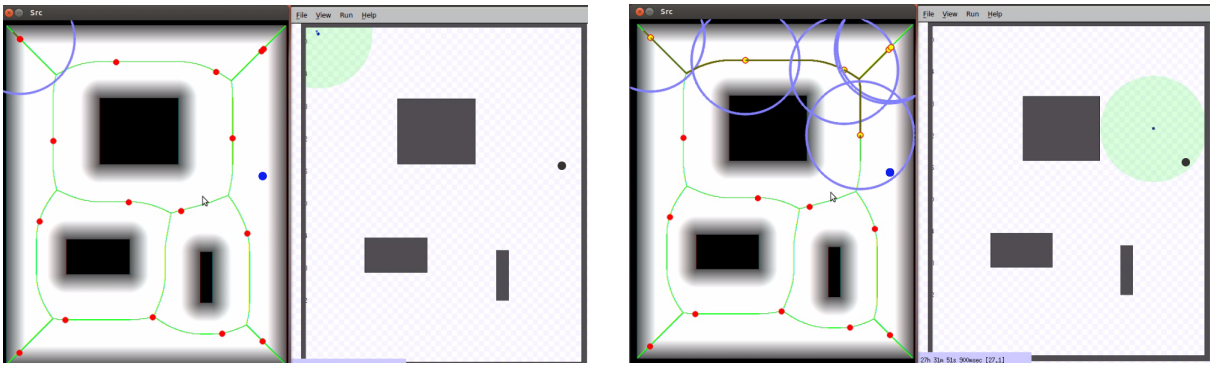
Figure 4.7: The value of objective functions in iterations.

4.3 Simulation on ROS/Stage

The Robot Operating System (ROS) (Quigley et al., 2009) is a framework to help in the development of robotic software. It consists of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior. Stage ROS is a useful package which allows for $2D$ robotic simulation. In order to verify the proposed exploration strategy, we tested it first on Stage ROS. This is an important step before implementing in the real robot to verify if the code is running as expected. In section 4.5 we show experiments with a real robot.

We defined a robot equipped with a laser sensor with maximum range equal to 10 meters ($r = 10$). The robot starts exploration in a random position in the map and move according to the route found by CPP in order to visit the *SPs* computed by the NSGA II with the help of ELECTRE I. At each *SP* the robot captures data from the laser scanner to detect the object.

In Figure 4.8, the initial configuration and the final configuration when finally the object was detected are shown.



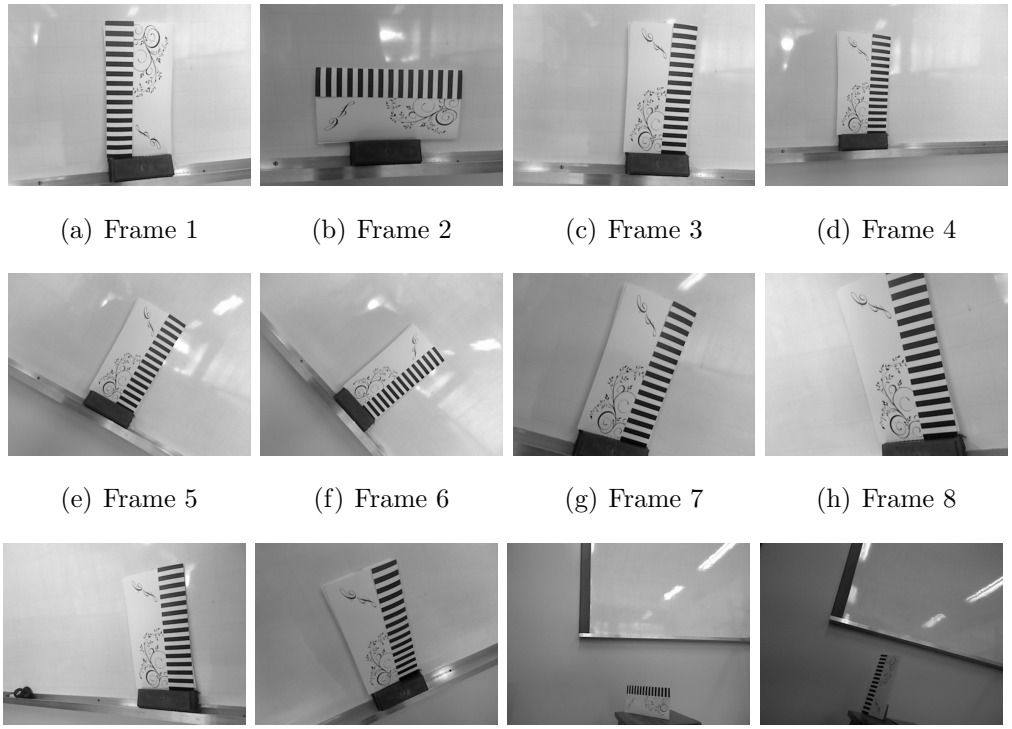
(a) Initial configuration

(b) Object is found.

Figure 4.8: Snapshots of the exploration. Red points show the *SPs* on the GVD, yellow points show the points which have been explored so far by the robot.

4.4 Object Recognition with SURF

In this part, several image matching experiments are presented to evaluate the method in our work. The test object used in the experiments is a card. We chose a set of images of the specified object as the training sample which are shown in Figure 4.9, and the whole as the test samples. Then the object from different positions should be recognized.



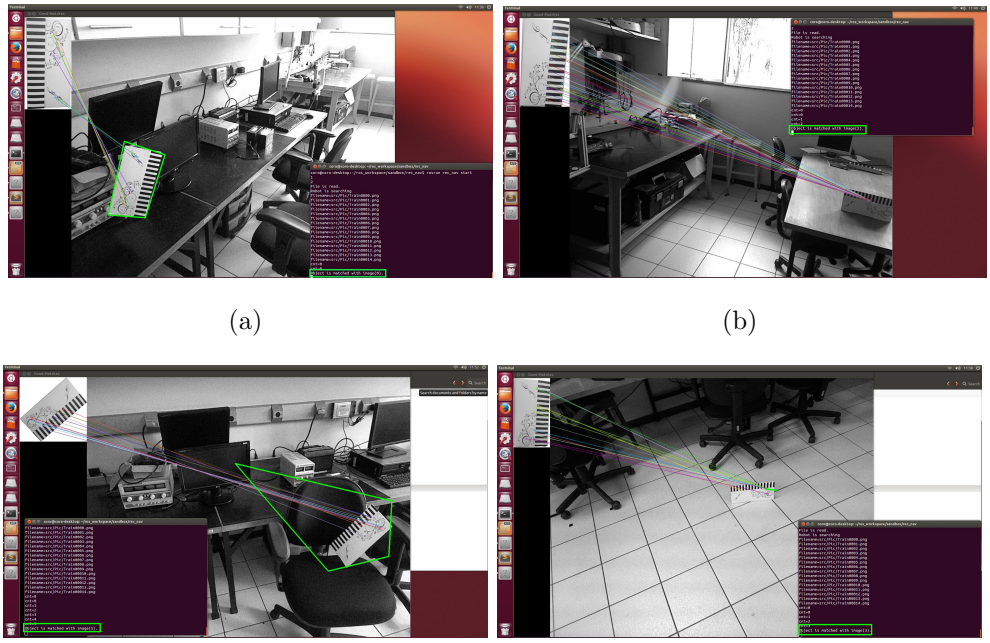
(a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Frame 4

(e) Frame 5 (f) Frame 6 (g) Frame 7 (h) Frame 8

(i) Frame 9 (j) Frame 10 (k) Frame 11 (l) Frame 12

Figure 4.9: Training image set.

The screenshots below (see Figure 4.10) are the results of the tests that we performed using SURF. As it can be seen, there are three windows in the screenshot. The one on the left-up is the image representing the object of interest, that is, the object that we want the robot to find. The window on the center is an actual picture of the lab where the object of interest is present. In other words, the first window can be seen as a training image, where as the second window can be seen as what the robot sees. In addition, the third window shows that the object could match with which one of the training images set.



(a) (b)

(c) (d)

Figure 4.10: The matched results of the SURF method.

4.5 Real Robot Results

The proposed method has been tested in a realistic scenario using the Pioneer platform[1] shown in Figure 4.11. The name of this service robot is Maria and it is equipped with all the necessary sensors for navigation and object recognition such as laser scanner and Kinect.

[1]http://www.mobilerobots.com/ResearchRobots/P3AT.aspx

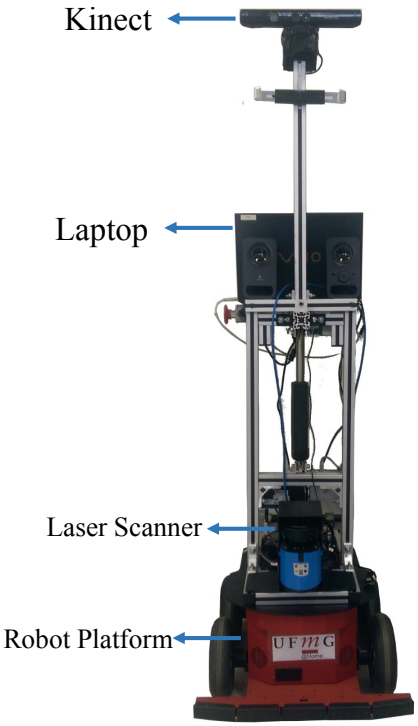
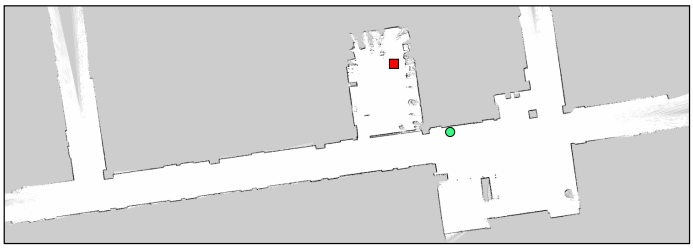
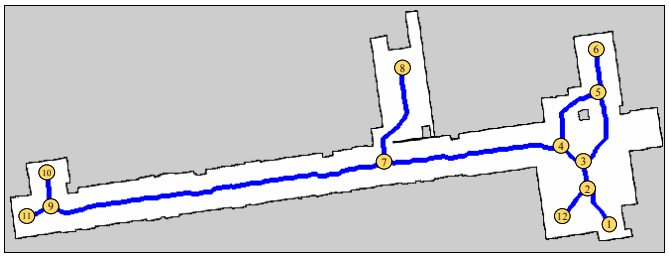


Figure 4.11: The structure of Maria and its sensors.

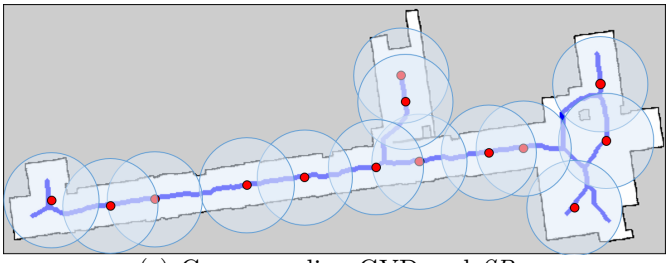
In our experiment, mapping and localization was provided by ROS packages. In order to control the robot 's linear and angular velocities v and w, respectively, so that the robot moves along the edges of the GVD, we used a static feedback linearization scheme (Desai et al., 1998). A real map of part of a building floor is considered for this experiment (see Figure 4.12.(a)). The size of this environment is 79.89×4.04 $meters$. A predefined object is placed in the map and the robot must find this object in minimum time. The initial configuration for the robot and also the position of the object are shown in Figure 4.12(a). The task of object recognition was done by using the local feature detection named Speed Up Robust Features (SURF) which is robust to rotation, scaling and affine transformation (Bay et al., 2008).



(a) Input map, red square: robot initial configuration, green circle: position of the object.



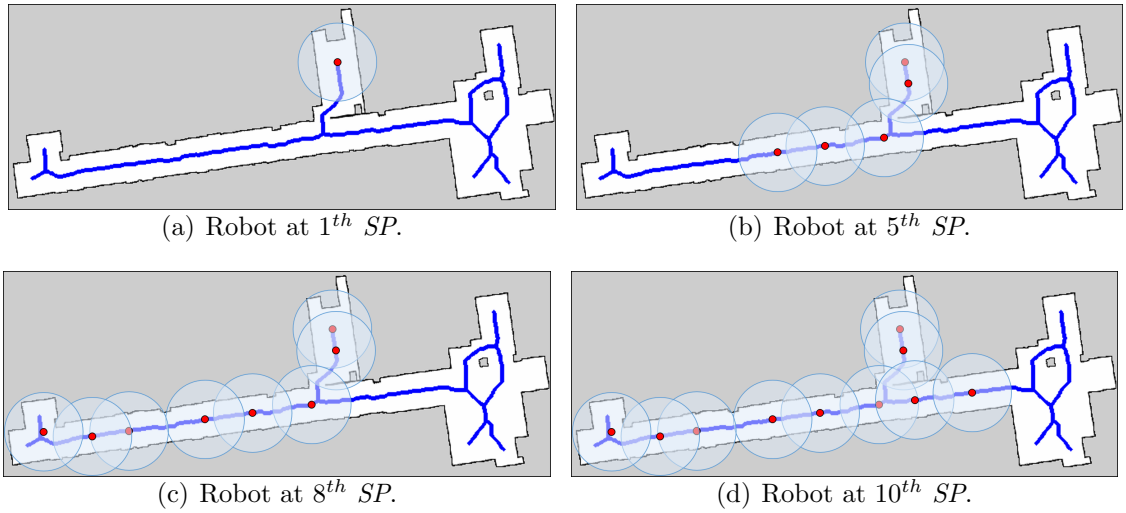
(b) Corresponding graph.



(c) Corresponding GVD and *SPs*.

Figure 4.12: Input map, GVD and search points.

After constructing the GVD and the graph, we used NSGA II and ELECTRE I to find good locations for *SPs*. In Figures 4.12.(b) and (c), the graph and the distribution of *SPs* over the GVD are depicted, respectively.
Figure 4.13 shows the sequence of SPs visited by the robot. The object has been found when the robot visits the 10^{th} SP (see Figure 4.13 (d)).



(a) Robot at 1^{th} *SP*.

(b) Robot at 5^{th} *SP*.

(c) Robot at 8^{th} *SP*.

(d) Robot at 10^{th} *SP*.

Figure 4.13: The sequence of visiting *SPs*.

Figure 4.14 represents the GVD and the trajectory executed by the robot according to the Chinese Postman Problem solution. The robot stopped exploration when the object was found. A video of this experiment can be found in https://youtu.be/TC3TJDoX2C4/.

Figure 4.14: Blue thicker line is the GVD and red line is the robot trajectory.

5 Statistical Analysis and Comparison of Strategies

In the case of large maps in which a large number of search points is required, the use of our proposed automatic approach to distribute these points is clear when compared to a normal distribution of points since the latter is unfeasible in such a scenario. However, in order to verify if the method is comparable or even better than an intuitive distribution of points done by a human specialist, we present in this chapter a comparison between the automatic proposed distribution and the manual distribution in the case of the small maps previously shown.

In fact, this chapter focuses on the comparison of our proposed method with two other methods based on hypothesis testing. Before comparison of strategies, we give an example to describe some basic statistical concepts.

Consider our exploration problem described in the introduction. Suppose that we are interested in the time to find the object as a random variable that can be described by a probability distribution. Suppose that our interest focuses on the mean time to find the object (a parameter of this distribution). Specifically, we are interested in deciding whether or not the mean time to find is 50 time units. We may express this formally as:

$$\begin{cases} \mathrm{H}_0 : \mu_1 = 50 \\ \mathrm{H}_1 : \mu_1 \neq 50 \end{cases} \tag{5.1}$$

The statement $H_0 : \mu_1 = 50$ time units in above equation is called the null hypothesis. This is a claim that is initially assumed to be true. The statement $H_1 : \mu_1 \neq 50$ time units is called the alternative hypothesis and it is a statement that contradicts the null hypothesis. Testing the hypothesis involves taking a random sample, computing a test statistic from the sample data, and then using the test statistic to make a decision about the null hypothesis. It is important to remember that hypotheses are always statements about the population or distribution under study, not statements about the sample.

Consider that a sample of n specimens is tested and that the sample mean time to find

$\bar{x}$ is observed. The sample mean is an estimate of the true population mean μ. Suppose that if $48.5 \leq \bar{x} \leq 51.5$, we will not reject the null hypothesis $H_0 : \mu_1 = 50$, and if either $\bar{x} < 48.5$ or $\bar{x} > 51.5$, we will reject the null hypothesis in favor of the alternative hypothesis $H_1 : \mu_1 \neq 50$. This is illustrated in Figure 5.1.

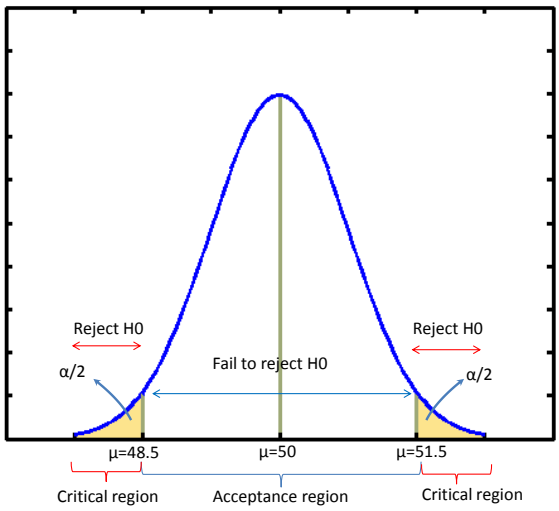


Figure 5.1: Decision criteria for testing $H_0 : \mu_1 = 50$ versus $H_1 : \mu_1 \neq 50$.

The acceptance region is a region where all values of $\bar{x}$ are in the interval $48.5 \leq \bar{x} \leq 51.5$ and we will fail to reject the null hypothesis; The values of $\bar{x}$ that are less than 48.5 and greater than 51.5 constitute the critical region for the test. We reject H_0 if the test statistic falls in the critical region and fail to reject H_0 otherwise.
This decision procedure for testing null hypothesis can lead to two wrong conclusions (Campelo, 2014): **type I error** and **type II error**.
Type I error is rejection of the null hypothesis H_0 when it is true.
The probability of occurrence of type I error is called *significance level* (α):

$$\alpha = P(type\ I\ error) = P(reject\ H_0 | H_0\ is\ true) \tag{5.2}$$

The selected value of α defines the critical threshold for the rejection of H_0. In our example (see Figure 5.1), a type I error will occur when either $\bar{x} < 48.5$ or $\bar{x} > 51.5$ when the true mean time to find the object really is $\mu_1 = 50$ time units. This probability $100(1-\alpha)\%$ is also usually known as confidence level. The interval contains only values of μ that are not rejected by the level-α test. For example, if this were a 95% confidence interval, in the long run only 5% of the intervals would fail to contain μ.

Failing to reject the null hypothesis when it is false is defined as a **type II error**. The probability of occurrence of a **type II error** in any test of hypotheses is generally represented by β:

$$\beta = P(type\ II\ error) = P(fail\ to\ reject H_0 | H_0\ is\ false) \tag{5.3}$$

The quantity $(1-\beta)$ is known as *power* of the test, and quantifies its sensitivity to effects that violate the null hypothesis.

5.1 Comparison of First and Second Strategies in Robot Exploration Experiment with Simple Map

To compare two strategies statistically, we divided the necessary tasks in four steps as follows: problem description, experimental design, analysis of experiment, and discussion and conclusion.

5.1.1 Problem Description

In this part, we compare the average time to find (TTF) the object of our strategy ($S1$) with another strategy($S2$) which will be explained below. More specifically, we are interested in knowing whether mean TTF measured by each strategy differs by more than 10 time units as **minimally interesting difference**.

In this section strategy ($S1$) refers to our proposed method where the robot moves along the GVD and execute the search at the *SPs*. In this strategy all *SPs* are found by NSGA II. On the other hand, in second strategy($S2$), robot explores the created GVD of input map and stops at specific points namely *SPs* to find the object similary to $S1$. However there is a constraint in $S2$ such that every graph edge of GVD has at least one *SP* placed on it.

In this experiments, the input map is the simple one in Figure 3.6(b) In order to have available information, experiments are performed to generate initial data for the two strategies. It means that the two different strategies ($S1, S2$) are run to save elapsed time for finding the object. For this reason, the object positions is changed 20 times for each strategy.

The result is considered relevant when it can generate effects greater than the minimally interesting difference(δ) of 10 time units. The other desired characteristics for the experiment are:

- Significance level(α): 0.1
- Power level($1 - \beta$): 0.8

The selected value of significance level defines the critical threshold for the rejection of H_0 and the power level of the test quantifies its sensitivity to effects that violate the null hypothesis.

5.1.2 Experimental Design

In this subsection, we divided the design of experiment in three parts as follows: first, we illustrated how to construct statistical hypothesis testing . In the next part we describe the results of an experiment with a linear statistical model. Lastly we determine the sample size required for hypothesis testing.

5.1.2.1 Statistical Hypothesis

The experimental design is primarily defined by the establishment of the null and alternative hypotheses, H_0 and H_1 respectively. Consider robot exploration experiment introduced earlier, we may think that the mean of TTF in $S1$ is equal to mean of TTF in $S2$. This may be stated formally as equation below:

$$\begin{cases} \mathrm{H}_0 : \mu_1 = \mu_2 \\ \mathrm{H}_1 : \mu_1 \neq \mu_2 \end{cases} \tag{5.4}$$

where μ_1 is the mean of TTF in $S1$ and μ_2 is the mean of TTF in $S2$. The null hypothesis is that the mean of TTF in $S1$ (μ_1) is equal to mean of TTF in $S2$ (μ_2) and alternate is that μ_1 is different from μ_2.

5.1.2.2 Representation of Observations

The null hypothesis H_0 represents that there are no differences between means. In contrast, alternative hypothesis H_1 states clearly that μ_1 and μ_2 are different. We characterize the results of an experiment with a model. A linear statistical model that describes

the data from our experiment is (Montgomery and Runger, 2013):

$$y_{ij} = \mu_i + \epsilon_{ij} \begin{cases} i = 1, 2 \\ j = 1, .., n_i \end{cases} \tag{5.5}$$

where y_{ij} is the j_{th} observation of i_{th} strategy; μ_i is the mean of each strategy; And ϵ_{ij} stands for the residual associated with i_{th} strategy at the j_{th} observation.

5.1.2.3 Choice of Sample Size

An important part of any experimental design is determining appropriate sample sizes (Montgomery, 2012); We need to determine how many observations are enough in order to draw conclusions about a population using a sample from that population. That is deciding the number of replicates to run. Choice of sample size depends on some parameters such as: minimally interesting effect (δ) or true difference between means , standard deviation(sd), significance level(α) and power of test.

To perform statistical tests, there is a high level programming language for data analysis and graphics which is called "R". This language includes various functions in statistics area,mathematics, graphics and etc that is used as statistical computing tools (Crawley, 2012).

We used a powerful command in R implementation which can be used to compute sample size considering target power. The command is known as *power.t.test(...)*. To compute the required sample size, we need to have the necessary parameters which are mentioned above and also know about type of t.test which is two.sample and type of alternative that is two.sided.

Since the variance is not known and it is essential for computing sample size, it will be estimated from the data. As we are assuming $\sigma_1^2 \approx \sigma_2^2$, we can use the pooled variance S_p^2 (Montgomery, 2012):

$$S_p^2 = (sd1^2 + sd2^2)/2 \tag{5.6}$$

where $sd1$ and $sd2$ are the standard deviation of two sample data ($S1$,$S2$). As a result, S_p equals to 36.

A number of 160 observations for each group was achieved as a sample size based on the desired characteristics, see Table 5.1 .

Two-sample t test power calculation
n = 160.9326
delta = 10
sd = 36
sig.level = 0.1
power = 0.8
alternative = two.sided

Table 5.1: Choice of sample size.

In addition, we consider that two sample sizes are equal ($n_1 = n_2 = n$)

5.1.3 Analysis of Experiment

This subsection focused on the test of statistical hypothesis including interpreting visual data displays with box plot, computing a test statistic for the two sample data and making a decision about the null hypothesis.

5.1.3.1 Test of Statistical Hypothesis

To describe several important features of a data set from an experiment, box plot can be used as a graphical method to assist us. Hence, box plot depicts significant characteristics of data such as median, spread or variability, departure from symmetry, and identification of unusual observations or outliers (Montgomery and Runger, 2013). A box plot displays the minimum and maximum of data, first and third quartiles, and the median or second quartile on a rectangular box. The box extends from the first quartile to third quartile, and a band inside the box shows the median. Lines extend from end of the box to minimum and maximum. The lower line extends from first quartile to smallest data point and the upper line extends from third quartile to largest data point. Any data not included between the lines is called outlier. The outlier is plotted with a circle or star(see Figure 5.2).

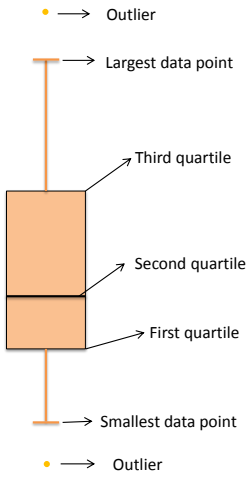


Figure 5.2: Description of a box plot.

Figure 5.3 shows the box plots for two sample data of TTF in robot exploration experiment based on $S1$ and $S2$. Each box plot presents a sample of 160 observations (this number of observations is calculated as sample size in previous subsection). The lines connect paired samples, that is, the samples taken by each strategy for the same object placement. The variability in the sample data from all two strategies seems very similar. The result is an indication that the median of data (which generally will be close to the mean) in $S1$ is smaller than that one in $S2$.

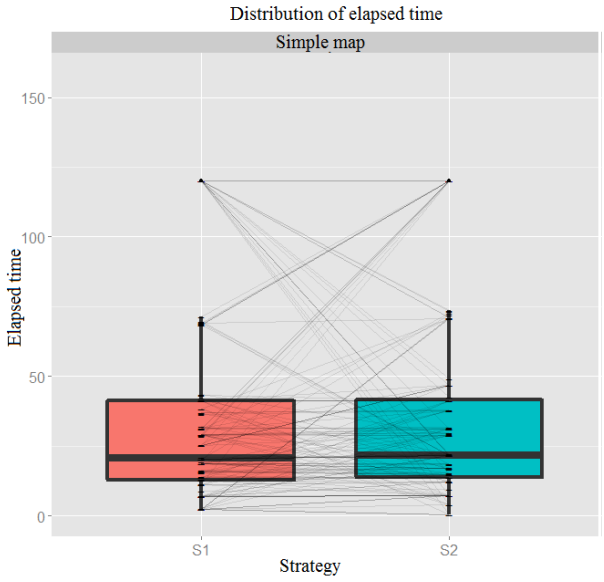


Figure 5.3: Comparative box plots of TTF in $S1$ and $S2$.

There is a built-in function in R which is called **t-test**. Computationally, we perform the **Two Sample t-test** as a smart solution for comparison of two means:

$> t.test(Elapsed.time \sim Strategy, alternative = "two.sided", mu = 0, var.equal = TRUE, conf.level = 0.9)$

Tables 5.2 shows the result of applying the Two Sample t-test for comparison between $S1$ and $S2$.

Two Sample t-test
data: Elaps.time by Strategy
t = -0.249, df = 318, p-value = 0.805
alternative hypothesis: true difference in means is not equal to 0
90 percent confidence interval:
-7.348978 5.434591
sample estimates:
mean in group $S1$ mean in group $S2$
35.32043 36.27762

Table 5.2: Two Sample t-test for comparing means of $S1$ and $S2$.

P-value is the lowest significance level that would lead to the rejection of $H0$ for the available data (Montgomery, 2012); In this experiment, because the p-value is big (> 0.1), the null hypothesis $H0$ would not be rejected. The evidence supports that the mean time elapsed with $S2$ equals the one in $S1$.

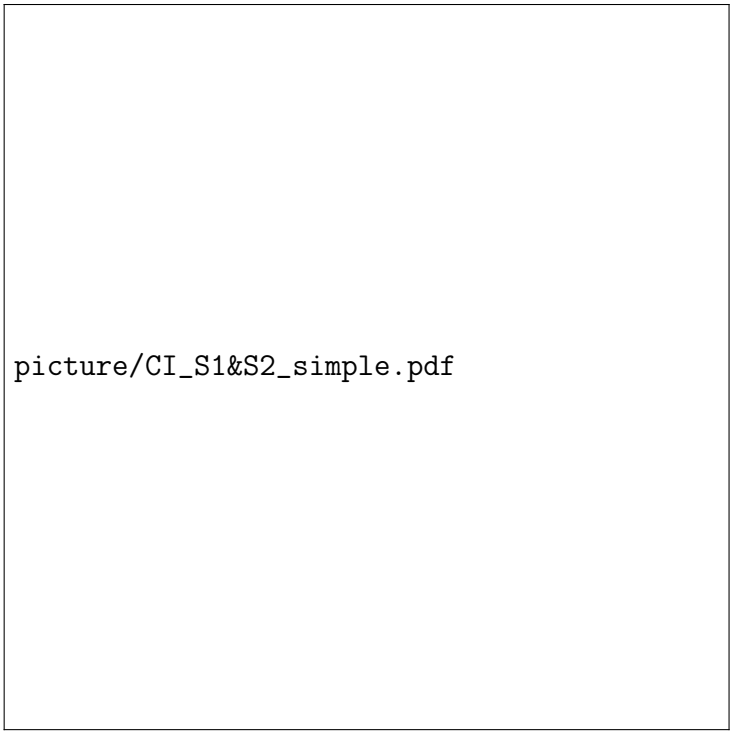


Figure 5.4: Difference between means(μ_1, μ_2) at a confidence of 90%.

Since the confidence intervals with a confidence of 90% include zero, the results presented in Figure 5.4 confirm that $S1$ provides results equal to $S2$.

5.1.4 Discussion

The performed statistical analysis has showed that the two strategies are statistically equivalent; the null hypothesis H_0 was not refuted.
It is recommended further investigation concerning current methods on an alternative map such as a complex map. Hence, more experiments will be needed to verify whether the means of TTF data for two strategies $(S1, S2)$ are equal or not.
In conclusion there are statistical evidences that support there are no significant divergences between the two strategies;

5.2 Comparison of First and Third Strategies in Robot Exploration Experiment with Simple Map

As it was mentioned before, this section includes four steps as: problem description, experimental design, analysis of experiment and discussion and conclusion.

5.2.1 Problem Description

The first method ($S1$) was presented previously and now we explain about the third strategy.
In the third strategy ($S3$), robot moves on created GVD of input map to explore the whole map. To search for the desired object, it stops at SPs. However there is a big difference between this strategy and $S1$. In $S3$, SPs are defined manually by an expert.
In the experiments the input map is the simple one, see Figure 3.6.(b). To have available information, experiments are performed to generate initial data based on two strategies. The result is considered relevant when it can generate effects greater than the minimally interesting effect of 10 time units. The other desired characteristics for the experiment are:

- Significance level(α): 0.1
- Power level($1 - \beta$): 0.8

5.2.2 Experimental Design

This subsection is divided in three parts as follows: statistical hypothesis, representation of observation and choice of sample size.

5.2.2.1 Statistical Hypothesis

We are interested in testing mean of TTF problem. The experimental design is primarily defined by the establishment of the null and alternative hypotheses, H_0 and H_1 respectively. The appropriate hypotheses are:

$$\begin{cases} \mathrm{H}_0 : \mu_1 = \mu_3 \\ \mathrm{H}_1 : \mu_1 \neq \mu_3 \end{cases} \tag{5.7}$$

The null hypothesis H_0 is defined as: mean of TTF in $S1$ is equal to mean of TTF in $S3$; This represents the case where there are no differences between two strategies in terms of mean of elased time. Furthermore, the alternative hypothesis is expressed as: mean of first strategy (μ_1) is different with respect the mean of third one (μ_3). This claim shows that differences between (μ_1) and (μ_3) is greater than the minimally interesting effect.

5.2.2.2 Representation of Observations

The data model can be described by the linear statistical model as:

$$y_{ij} = \mu_i + \epsilon_{ij} \begin{cases} \text{i=1,3} \\ \text{j=1,..,n}_i \end{cases} \tag{5.8}$$

where y_{ij} is the j_{th} observation from i_{th} strategy. μ_i is the mean of i_{th} strategy and ϵ_{ij} is the residual associated with i_{th} strategy at the j_{th} observation.

5.2.2.3 Choice of Sample Size

The experiment is consisted by selecting appropriate sample sizes; To determine how many replicates is enough in each of two samples, we applied *power.t.test(...)* with power = 0.8 when minimally interesting effect= 10,the significance level and standard deviation are 0.1 and 38 respectively. It is necessary to mention that the standard deviation is computed based on equation(5.6).
Table 5.3 shows that we need 180 replicates in each sample (360 replicates in all) to achieve a power of 0.8.

Two-sample t test power calculation
n = 179.2325
delta = 10
sd = 38
sig.level = 0.1
power = 0.8
alternative = one.sided

Table 5.3: Choice of sample size.

5.2.3 Analysis of Experiment

5.2.3.1 Test of Statistical Hypothesis

After running the designed experiment, it is a good idea to examine experimental data graphically. Consider Figure 5.5, which contains box plots for the TTF data of two

strategies $S1, S3$. The light gray lines connect paired observations, that is, the observations taken by each strategy for the same target object placement. As follows from the comparative box plots shown below, there is small differences between medians of the two strategies. The variability of data set in $S3$ is larger than $S1$.

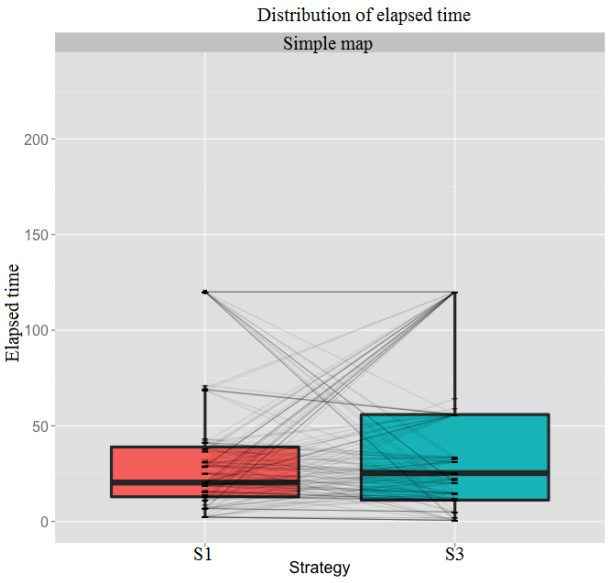


Figure 5.5: Comparative box plots of TTF in $S1$,$S3$.

The null hypothesis of equal means can be tested by the function **t-test** with equal variance for two samples. $> t.test(Elapsed.time \sim Strategy, alternative = "two.sided", mu = 0, var.equal = TRUE, conf.level = 0.9)$
The overall measurement results using **t-test** are summarized in Table 5.8.

Two Sample t-test
data: Elaps.time by Strategy
t = -1.8529, df = 358, p-value = 0.06471
alternative hypothesis: true difference in means is not equal to 0
90 percent confidence interval:
-14.2419845 -0.8289044
sample estimates:
mean in group $S1$ mean in group $S3$
34.87606 42.41150

Table 5.4: Two Sample t-test for comparing means of $S1$ and $S3$.

From the p-value 0.06471, the result is rejection of the null hypothesis $H0$. Note that

the p-value is smaller than significance level.
The result obtained shows that the intervals does not contain zero. Therefore, we can conclude that the null hypothesis can be rejected at 90% confidence level. In addition, the difference between $\mu_1 - \mu_3$ is negative so we can say that μ_1 is less than μ_3.

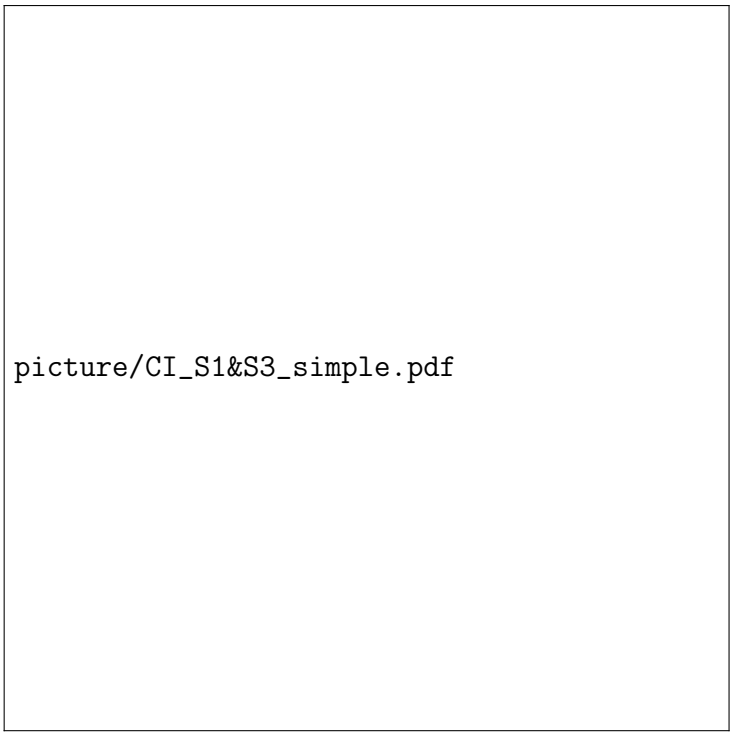


Figure 5.6: Difference between means(μ_1, μ_3) at a confidence of 90%.

5.2.4 Discussion

The results obtained have indicated that the two strategies ($S1, S3$) are not statistically equivalent; the null hypothesis was refuted.
It is recommended to use an alternative map such as a complex map. Therefore, further test of the issue is still required.
Summing up the results, it can be concluded that there are difference between the means of the two strategies ($S1, S3$); There are statistical evidences that support this statement.

5.3 Comparison of First and Second Strategies in Robot Exploration Experiment with Complex Map

In this section, our purpose is to compare $S1$ and $S2$ using a complex map. The following steps for this purpose are: problem description, experimental design, analysis of experiment and discussion and conclusion.

5.3.1 Problem Description

According to the results shown in sections 5.1 the second strategy($S2$)is not different from strategy ($S1$) in terms of average time to find the object. Thus, we decide to consider a complicated map containing more obstacles in which comparison of $S1$ with $S2$ should be done, see Figure 4.1.(b). Again, we test for differences in means considering significance level $(\alpha) = 0.1$, power $(1-\beta)= 0.8$. The result is considered relevant when it can generate effects greater than the minimally interesting effect of 40 time units.

5.3.2 Experimental Design

5.3.2.1 Statistical Hypothesis

The null and alternative hypotheses could be expressed as equation 5.4.

5.3.2.2 Representation of Observations

The statistical model for the data are defined such as equation 5.5.

5.3.2.3 Choice of Sample Size

The sample size of 22 observations for each group was achieved based on desired characteristics, see Table 5.5 .

Two-sample t test power calculation
n = 21.60658
delta = 40
sd = 52
sig.level = 0.1
power = 0.8
alternative = two.sided

Table 5.5: Choice of sample size.

5.3.3 Analysis of Experiment

5.3.3.1 Test of Statistical Hypothesis

Figure 5.7 presents the box plot. The result is an indication that the median of TTF data in $S1$ is significantly smaller than that one in $S2$ and also there is relevant difference between two strategies. The variability of data set in $S2$ seems bigger than in $S1$.

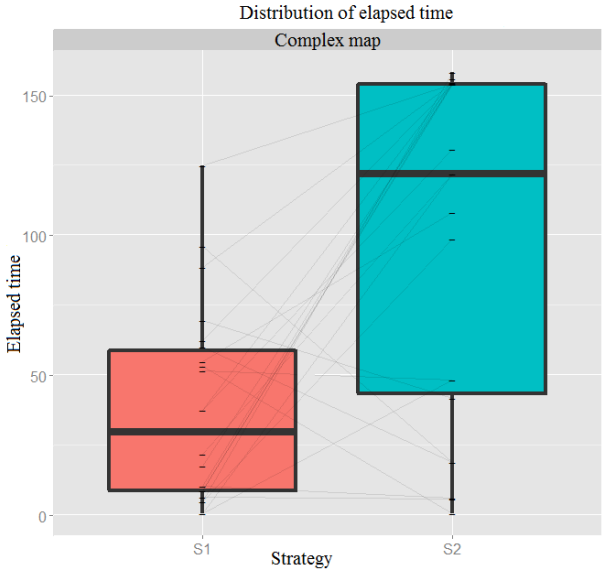


Figure 5.7: Comparative box plots of TTF in $S1$,$S2$.

The results of applying the Two Sample t-test to this experiment are as follows:

Two Sample t-test
data: Elaps.time by Strategy
t =-4.0228, df = 42, p-value = 0.0002346
alternative hypothesis: true difference in means is not equal to 0
90 percent confidence interval:
-85.83351 -35.22040
sample estimates:
mean in group $S1$ mean in group $S2$
37.62036 98.14732

Table 5.6: Two Sample t-test for comparing means of $S1$ and $S2$.

In this experiment, because the p-value is significantly less (< 0.1), the null hypothesis $H0$ would be rejected. The evidence supports that $S2$ elapses mean level of time is bigger than in $S1$.

As it can be seen in figure 5.8, since the interval does not contain zero then the null hypothesis can be rejected. whereas the difference between $\mu_1 - \mu_2$ is negative, we can claim that mean of TTF in $S1$ is less than $S2$.

picture/CI_S1&S2_Complex.pdf

Figure 5.8: Difference between means(μ_1, μ_2) at a confidence of 90%.

5.3.4 Discussion

The performed statistical analysis has showed that the two strategies are not statistically equivalent; the null hypothesis $H0$ was refuted.
In conclusion there are statistical evidences that support there are significant divergences between the two strategies;

5.4 Comparison of First and Third Strategies in Robot Exploration Experiment with Complex Map

We compared the strategies $S1$ and $S3$ statistically in four steps as follows: problem description, experimental design, analysis of experiment and discussion and conclusion.

5.4.1 Problem Description

In this experiments, the input map is the complex one, see Figure 4.1.(b). The desired characteristics for the experiment are:

- Significance level(α): 0.1
- Power level($1 - \beta$): 0.8
- minimally interesting effect(δ): 40 time units.

5.4.2 Experimental Design

5.4.2.1 Statistical Hypothesis

The null and alternative hypotheses are specified such as equation 5.7.

5.4.2.2 Representation of Observations

The data model can be described by the linear statistical model as equation 5.8:

5.4.2.3 Choice of Sample Size

Table5.7 shows that we need 23 replicates in each sample (46 replicates in all) to achieve a power of 0.8.

Two-sample t test power calculation
n = 23.24253
delta = 40
sd = 54
sig.level = 0.1
power = 0.8
alternative = two.sided

Table 5.7: Choice of sample size.

5.4.3 Analysis of Experiment

5.4.3.1 Test of Statistical Hypothesis

Consider Figure 5.9,which contains box plots for the TTF data of two strategies $S1, S3$. It is clear from the map that there is a large variability in the times to target, which is expected, given the random placement of the target points. Moreover, the light gray lines connecting the paired observations suggest that even for the same placement the times can exhibit a large variation between the two strategies. Larger differences are observed for the complex map, while there is a more apparent effect of the proposed approach in contrast with the manual placement of SPs in the simple one.

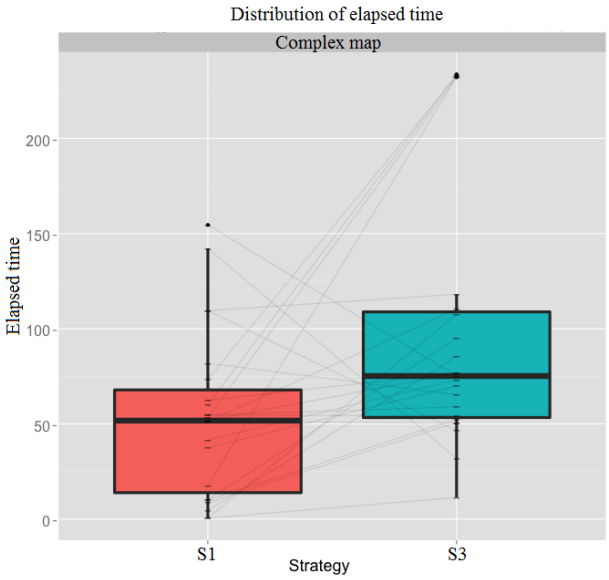


Figure 5.9: Comparative box plots of TTF in $S1$,$S3$.

The overall measurement results using **t-test** are summarized in Table 5.8.

Two Sample t-test
data: Elaps.time by Strategy
t = -2.5853, df = 44, p-value = 0.01312
alternative hypothesis: true difference in means is not equal to 0
90 percent confidence interval:
-71.73803 -15.22205
sample estimates:
mean in group $S1$ mean in group $S3$
53.93452 97.49994

Table 5.8: Two Sample t-test for comparing means of $S1$ and $S3$.

From the p-value 0.01312, the result is to reject the null hypothesis $H0$ of equal sample means. Note that the p-value is less than significance level and the difference between means $(\mu_2 - \mu_1)$ is greater than minimally interesting effect(δ).
As it can be seen from table 5.10, the interval does not contain zero then the null hypothesis of no difference between means of $S1$ and $S3$ can be rejected at confidence of 90%. Since the difference between $\mu_1 - \mu_3$ is negative, we can say that μ_1 is smaller than μ_3.

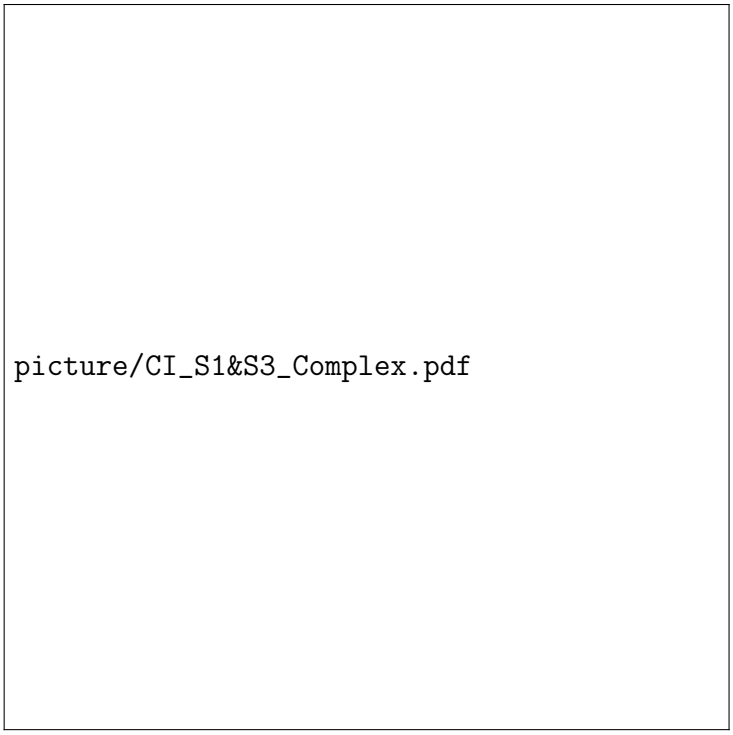


Figure 5.10: Difference between means(μ_1, μ_3) at a confidence of 90%.

5.4.4 Discussion

The results obtained from t-test have indicated that the two strategies ($S1, S3$) are not statistically equivalent; the null hypothesis was refuted.
Summing up the results, it can be concluded that there are difference between means of two strategies ($S1, S3$); There are statistical evidences that support this statement.

5.5 Conclusion

From the results of the experiments, we found that our method is comparable and even better than other two methods in terms of mean time of finding object. The advantage of the proposed approach is highlighted when it is applied on a large complex map. As shown in the results our method could distribute SPs better than an expert human, and accordingly minimized the elapsed time.

6 Conclusion and Future Work

This work has proposed a new strategy for robotic exploration with the aim of finding a specific object in a given static environment. We take into account some of the challenges such as time limitation and range of robot's sensors. In order to address these problems, we present a technique which guides the robot to search just at some stationary points instead of searching at all possible configurations of the map. These points are called search points (*SPs*). *SPs* are placed on a safe route defined by the GVD.

In this work, the placement of these *SPs* is provided by the solution of an optimization problem with two objective functions: maximum covered area and distribution of *SPs* on the map. NSGA-II is used as a tool to find the Pareto-front and ELECTRE I is used as a decision maker to choose a solution.

The problem of routing the robot to search the object at the computed search points is modeled as a Chinese Postman Problem so that it can be efficiently solved even in the case of large number of search points. For object recognition task, SURF algorithm is used because of its powerful attributes, including scale invariance, translation invariance, illumination invariance and rotation invariance.

Experimental results are in good agreement with simulation and theory. A real robot experiment is also presented to show that the proposed method can be used in real applications. In the statistical section, we have compared different strategies to show the efficiency of our method. Statistical results show that the proposed method is comparable with two mentioned methods and provide smaller average of the elapsed time in finding the pre specified object.

In conclusion, this work can be suitable for different applications, especially when time limitation is one of the important criteria.

The most obvious limitation of our work is that we need to have the map of the environment and the exploration can not be done in unknown environment.

Another limitation is that the proposed method can work in the static environment, but it is possible to extend this work to be applicable in the dynamic environment by updating the road map periodically. Moreover, some important aspects, including the effects of

uncertainty and the extension to outdoor environments, have not been fully considered in this work and deserve more attention.

As in single robot exploration, the goal is to minimize the overall time exploration, using a multi-robot solution helps to achieve this goal easier. Indeed, there are several advantages in multi-robot systems over single robot such as speed, accuracy and fault-tolerant. Therefore, this work can be extended to use cooperation of a multi-robot system instead of a single robot in future work.

Bibliography

Adel, E., Elmogy, M., and Elbakry, H. (2014). Image Stitching based on Feature Extraction Techniques: A Survey. *International Journal of Computer Applications*, 99(6):1–8.

Amanatiadis, A. A., Chatzichristofis, S. A., Charalampous, K., Doitsidis, L., Kosmatopoulos, E. B., Tsalides, P., Gasteratos, A., and Roumeliotis, S. I. (2013). A multi-objective exploration strategy for mobile robots under operational constraints. *Access, IEEE*, 1:691–702.

Amigoni, F. (2008). Experimental evaluation of some exploration strategies for mobile robots. *In Proc. of the IEEE International Conference on Robotics and Automation*, pages 2818–2823.

Amigoni, F. and Gallo, A. (2005). A multi-objective exploration strategy for mobile robots. *In Proc. of the International Conference on Robotics and Automation*, pages 3850 – 3855.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359.

Bojković, N., Anić, I., and Pejčić-Tarle, S. (2010). One solution for cross-country transport-sustainability evaluation using a modified ELECTRE method. *Ecological Economics*, 69(5):1176–1186.

Branke, J., Deb, K., Miettinen, K., and Slowinski, R. (2008). *Multiobjective Optimization - Interactive and Evolutionary Approaches*.

Campelo, F. (2014). Design and Analysis of Experiments. *Lecture notes*.

Choset, H. (2005). *Principles of robot motion: theory, algorithms, and implementation*, volume 12. Cambridge, Mass. [u.a.] MIT Press.

Crawley, M. J. (2012). *The R book*. John Wiley & Sons, 2nd edition.

DasGupta, B. (2004). Aggregation-based approaches to honey-pot searching with local sensory information. *In Proc. of the American Control Conference*, 2:1202 – 1207.

Davoodi, M., Panahi, F., Mohades, A., and Hashemi, S. N. (2013). Multi-objective path planning in discrete space. *Applied Soft Computing*, 13(1):709–720.

Deb, K. and Pratap, A. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Desai, J. P., Ostrowski, J., and Kumar, V. (1998). Controlling formations of multiple mobile robots. In *In Proc. of the IEEE International Conference on Robotics and Automation*, volume 4, pages 2864–2869. IEEE.

Dornhege, C. and Kleiner, A. (2013). A frontier-void-based approach for autonomous exploration in 3D. *Advanced Robotics*, 27(6):459–468.

Edmonds, J. and Johnson, E. L. (1973). Matching, Euler tours and the Chinese postman. *Mathematical Programming*, 5(1):88–124.

Eiselt, H. A., Gendreau, M., and Laporte, G. (1995). Arc routing problems, part I: The Chinese postman problem. *Operations Research*, 43(2):231–242.

El-Hussieny, H., Assal, S. F. M., and Abdellatif, M. (2013). Improved Backtracking Algorithm for Efficient Sensor-Based Random Tree Exploration. *2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks*, pages 19–24.

Fleischner, H. (1991). *Eulerian Graphs and Related Topics, Part 1, Volume 2*. Elsevier, Amsterdam.

Flurt, M. (1883). Deux problèmes de Géométrie de situation. *Journal de mathématiques élémentaires, 2nd ser. (in French)*, pages 257–261.

Franchi, A., Oriole, G., Reda, L., and Vendittelli, M. (2007). A Decentralized Strategy for Cooperative Robot Exploration. *In Proc. of First International Conference on Robot Communication and Coordination (ROBOCOMM)*, (April):1–8.

Freda, L. and Oriolo, G. (2005). Frontier-based probabilistic strategies for sensor-based exploration. *In Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2005(April):3881–3887.

Galceran, E. and Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276.

Garrido, S., Moreno, L., Blanco, D., and Jurewicz, P. (2011). Path planning for mobile robot navigation using voronoi diagram and fast marching. *International Journal of Robotics and Automation (IJRA)*, 2(1):42–64.

Gonzales-banos, H. H. and Latombe, J.-c. (2002). Navigation Strategies for Exploring Indoor Environments. *International Journal of Robotics Research*, 21(1):829—-ÂŰ848.

Gonzalez, R. C. and Woods, R. E. (2001). *Digital Image Processing*. Prentice Hall, 2nd edition.

Grady, D., Moll, M., Hegde, C., Sankaranarayanan, A., Baraniuk, R., and Kavraki, L. (2012). Multi-objective sensor-based replanning for a car-like robot. *IEEE Intl. Symp. on Safety, Security, and Rescue Robotics*, pages 1–6.

Huijuan, Z. and Qiong, H. (2011). Fast image matching based-on improved SURF algorithm. *2011 International Conference on Electronics, Communications and Control (ICECC)*, (1):1460–1463.

Juliá, M., Reinoso, O., Gil, A., Ballesta, M., and Payá, L. (2010). A hybrid solution to the multi-robot integrated exploration problem. *Engineering Applications of Artificial Intelligence*, 23:473–486.

Larson, R. C. and Odoni, A. R. (1981). *Urban Operations Research*. Prentice-Hall.

Maohai, L., Han, W., Lining, S., and Zesu, C. (2013). Robust omnidirectional mobile robot topological navigation system using omnidirectional vision. *Engineering Applications of Artificial Intelligence*, 26(8):1942–1952.

Marjovi, A., Nunes, J. G., Marques, L., and de Almeida, A. (2010). Multi-robot fire searching in unknown environment. *Field and Service Robotics*, 62:341–351.

Montgomery, D. C. (2012). *Design and Analysis of Experiments*. John Wiley & Sons, Inc., 8th edition.

Montgomery, D. C. and Runger, G. C. (2013). *Applied Statistics and Probability for Engineers*. John Wiley & Sons, 6th edition.

Oriolo, G., Vendittelli, M., Freda, L., and Troso, G. (2004). The SRT Method : Randomized strategies for exploration. *In Proc. of IEEE International Conference on Robotics and Automation*, 5:4688–4694.

Pearson, D. and Bryant, V. (2004). *Decision Math 1: Advancing Maths for AQA*, volume 01. Heinemann, advancing edition.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009). ROS: an open-source Robot Operating System. In *International conference on Robotics and Automation (ICRA)*, volume 3.

Shanian, A. and Savadogo, O. (2006). ELECTRE I Decision Support Model for Material Selection of Bipolar Plates for Polymer Electrolyte Fuel Cells Applications. *Journal of New Materials for Electrochemical Systems*, 199:191–199.

Stückler, J., Droeschel, D., and Gräve, K. (2014). Increasing flexibility of mobile manipulation and intuitive human-robot interaction in robocup@ home. *RoboCup 2013, LNCS*, 8371:135–146.

Wilson, R. J. and Watkins, J. J. (1990). *Graphs: an introductory approach : a first course in discrete mathematics*. illustrate edition.

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. *In Proc. of Intelligence in Robotics and Automation*, pages 146–151.

Zelinsky, A., Jarvis, R. A., Byrne, J. C., and Yuta, S. (1993). Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot. *In Proceedings of International Conference on Advanced Robotics*, pages 533–538.