ORTA DOĞU TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜ

# TEZ ŞABLONU ONAY FORMU
# THESIS TEMPLATE CONFIRMATION FORM

1. Şablonda verilen yerleşim ve boşluklar değiştirilmemelidir.
2. **Jüri tarihi** Başlık Sayfası, İmza Sayfası, Abstract ve Öz'de ilgili yerlere yazılmalıdır.
3. İmza sayfasında jüri üyelerinin unvanları doğru olarak yazılmalıdır. Tüm imzalar **mavi pilot kalemle** atılmalıdır.
4. **Disiplinlerarası** programlarda görevlendirilen öğretim üyeleri için jüri üyeleri kısmında tam zamanlı olarak çalıştıkları anabilim dalı başkanlığının ismi yazılmalıdır. Örneğin: bir öğretim üyesi Biyoteknoloji programında görev yapıyor ve biyoloji bölümünde tam zamanlı çalışıyorsa, İmza sayfasına biyoloji bölümü yazılmalıdır. İstisnai olarak, disiplinler arası program başkanı ve tez danışmanı için disiplinlerarası program adı yazılmalıdır.
5. Tezin **son sayfasının sayfa** numarası Abstract ve Öz'de ilgili yerlere yazılmalıdır.
6. Bütün chapterlar, referanslar, ekler ve CV sağ sayfada başlamalıdır. Bunun için **kesmeler** kullanılmıştır. **Kesmelerin kayması** fazladan boş sayfaların oluşmasına sebep olabilir. Bu gibi durumlarda paragraf (¶) işaretine tıklayarak kesmeleri görünür hale getirin ve yerlerini **kontrol edin**.
7. Figürler ve tablolar kenar boşluklarına taşmamalıdır.
8. Şablonda yorum olarak eklenen uyarılar dikkatle okunmalı ve uygulanmalıdır.
9. Tez yazdırılmadan önce PDF olarak kaydedilmelidir. Şablonda yorum olarak eklenen uyarılar PDF dokümanında yer almamalıdır.
10. Tez taslaklarının kontrol işlemleri tamamlandığında, bu durum öğrencilere METU uzantılı öğrenci e-posta adresleri aracılığıyla duyurulacaktır.
11. Tez yazım süreci ile ilgili herhangi bir sıkıntı yaşarsanız, Sıkça Sorulan Sorular (SSS) sayfamızı ziyaret ederek yaşadığınız sıkıntıyla ilgili bir çözüm bulabilirsiniz.

1. Do not change the spacing and placement in the template.
2. Write **defense date** to the related places given on Title page, Approval page, Abstract and Öz.
3. Write the titles of the examining committee members correctly on Approval Page. **Blue ink** must be used for all signatures.
4. For faculty members working in **interdisciplinary programs**, the name of the department that they work full-time should be written on the Approval page. For example, if a faculty member staffs in the biotechnology program and works full-time in the biology department, the department of biology should be written on the approval page. Exceptionally, for the interdisciplinary program chair and your thesis supervisor, the interdisciplinary program name should be written.
5. Write **the page number of the last page** in the related places given on Abstract and Öz pages.
6. All chapters, references, appendices and CV must be started on the right page. **Section Breaks** were used for this. **Change in the placement** of section breaks can result in extra blank pages. In such cases, make the section breaks visible by clicking paragraph (¶) mark and **check their position.**
7. All figures and tables must be given inside the page. Nothing must appear in the margins.
8. All the warnings given on the comments section through the thesis template must be read and applied.
9. Save your thesis as pdf and Disable all the comments before taking the printout.
10. This will be announced to the students via their METU students e-mail addresses when the control of the thesis drafts has been completed.
11. If you have any problems with the thesis writing process, you may visit our Frequently Asked Questions (FAQ) page and find a solution to your problem.

☒ **Yukarıda bulunan tüm maddeleri okudum, anladım ve kabul ediyorum. / I have read, understand and accept all of the items above.**

**Name :**
**Surname :**
**E-Mail :**
**Date :**
**Signature :** _____

# EXPLOSIVE PERCOLATION BASED ACTIVE SLAM EXPLORATION VIA LIDAR IN UNSTRUCTURED MAP

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DOĞAN YILDIZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONIC ENGINEERING

AUGUST 2023

Approval of the thesis:

**EXPLOSIVE PERCOLATION BASED ACTIVE SLAM EXPLORATION VIA LIDAR IN UNSTRUCTURED MAP**

submitted by **DOĞAN YILDIZ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** i**n Electrical and Electronic Engineering, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of the Department, **Electrical and Electronics Engineering** _____

Prof. Dr. Aydan Müşerref Erkmen
Supervisor, **Electrical and Electronics Engineering, METU** _____

**Examining Committee Members:**

Assoc.Prof. Dr. Mustafa Mert Ankaralı
Electrical and Electronics Engineering, METU _____

Prof. Dr. Aydan Müşerref Erkmen
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Kemalettin Erbatur
Mechatronics Engineering, Sabancı University _____

Assist. Prof. Dr. David M. Rosen
Electrical and Computer Engineering, Northeastern University _____

Assoc. Prof. Dr.Emre Özkan
Electrical and Electronics Engineering, METU _____

Date: 10.08.2023

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name Last name : Doğan Yıldız

Signature :

# ABSTRACT

## EXPLOSIVE PERCOLATION BASED ACTIVE SLAM EXPLORATION VIA LIDAR IN UNSTRUCTURED MAP

Yıldız, Doğan
Doctor of Philosophy, Electrical and Electronic Engineering
Supervisor : Prof. Dr. Aydan Müşerref Erkmen

August 2023, 119 pages

This thesis proposes a novel exploration technique for simultaneous localization and mapping (SLAM) in highly unstructured disaster regions for search and rescue (SAR) operations. As is known to all, disaster regions are very dangerous environments for humans to navigate and search for victims. For this reason, robotics applications in SAR operations gain considerable attention for the last decade. In a disaster region searching for survivors with robots needs finding a safe continuous path to reach the victim's location, and extracting the map of the unstructured area to be shared with SAR teams. However, the highly complex structure of rubbles, the GPS-denied region, and the unpredictable nature of the environment make the process difficult for such applications. Also, the unstable characteristic of debris causes the collapse of certain areas of the disaster region, and this brings unwanted dead-end occurrences making the search operation even harder than before. Unfortunately, even if the distinct progress of SLAM applications in the robotic field, these problems still preserve their effectiveness and they are waiting to be answered. In our proposed method we address these problems and offer an efficient and robust solution with explosive percolation-based exploration technique combined with the active SLAM approach for unstructured environments.

The active SLAM approaches mainly focus on the minimization of the total entropy of localization and mapping of the environment. However, in our method, we need to consider not only the entropy values but our main goal of finding the survivors' location for active exploration. Moreover, during the search finding a continuous path by mapping the disaster environment is affected severely by successive dead-ends. To tackle this problem, we offer a fluid behavior-based approach *Explosive Percolation (EP).* With the power of this method, the robot can navigate within a complex unstructured environment without being trapped by dead-ends.

The main contributions of the thesis to literature can be listed as; finding a novel path-search algorithm based on *Explosive Percolation* and combining this method with the SLAM approach to obtain a continuous path in a disaster area without being interrupted by dead-ends.

Keywords: Explosive Percolation, SLAM, Localization, Mapping, Search and Rescue

# ÖZ

## AYRILMIŞ PERKÜLASYON TEORİSİ TABANLI AKTİF ANLIK KONUMLAMA VE HARİTALAMA YOLUYLA IŞIK TESPİTİ VE UZAKLIK TAYİNİ SENSÖRÜ KULLANILARAK YAPISAL OLMAYAN ALANLARIN KEŞFİ

Yıldız, Doğan
Doktora, Elektrik ve Elektronik Mühendisliği
Tez Yöneticisi: Prof. Dr. Aydan Müşerref Erkmen

Ağustos 2023, 119 sayfa

Bu tez, arama ve kurtarma operasyonları için oldukça yapılandırılmamış afet bölgelerinde anlık konumlama ve haritalama için yeni bir keşif tekniği önermektedir. Bilindiği üzere afet bölgeleri, insanların gezinmesi ve kurban araması için oldukça tehlikeli ortamlardır. Bu nedenle, SAR operasyonlarında robotik uygulamalar son on yılda büyük ilgi görmektedir. Bir afet bölgesinde hayatta kalanları robotlarla aramak, kurbanın konumuna ulaşmak için güvenli ve kesintisiz bir yol bulmayı ve arama ve kurtarma ekipleriyle paylaşılacak şekilde yapılandırılmamış alanın haritasını çıkarmayı gerektirir. Bununla birlikte, molozların oldukça karmaşık yapısı, global konumlamanın kullanılamaz olduğu bölge ve ortamın öngörülemez doğası, bu tür uygulamalar için süreci zorlaştırmaktadır. Ayrıca enkazın dengesiz olması, afet bölgesinin belirli bölgelerinin çökmesine neden olmakta, bu da istenmeyen çıkmazları beraberinde getirmekte ve arama çalışmalarını eskisinden daha da zorlaştırmaktadır. Ne yazık ki robotik alanda anlık konumlama ve haritalama uygulamalarının belirgin bir şekilde ilerlemesine rağmen bu problemler hala etkinliğini korumakta ve cevaplanmayı beklemektedir. Önerilen yöntemimizde, bu

sorunları ele alıyoruz ve yapılandırılmamış ortamlar için aktif anlık konumlama ve haritalama yaklaşımıyla birleştirilmiş patlayıcı süzülmeye dayalı keşif tekniği ile verimli ve sağlam bir çözüm sunuyoruz.

Aktif anlık konumlama ve haritalama yaklaşımları temel olarak, yerelleştirme ve çevrenin haritalanmasının toplam entropisinin en aza indirilmesine odaklanır. Bununla birlikte, yöntemimizde, yalnızca entropi değerlerini değil, aynı zamanda hayatta kalanların aktif keşif için konumlarını bulma ana hedefimizi de dikkate almamız gerekiyor. Ayrıca, arama sırasında afet ortamını haritalandırarak sürekli bir yol bulma, birbirini izleyen çıkmazlardan ciddi şekilde etkilenir. Bu sorunun üstesinden gelmek için, akıcı davranışa dayalı bir yaklaşım ayrılmış perkülasyon sunuyoruz. Bu yöntemin gücü ile robot, çıkmaz yollara hapsolmadan karmaşık, yapılandırılmamış bir ortamda gezinebilir.

Tezin literatüre katkıları başlıca şu şekilde sıralanabilir; ayrılmış perkülasyon dayalı yeni bir yol arama algoritması bulmak ve bu yöntemi anlık konumlama ve haritalama yoluyla yaklaşımıyla birleştirerek bir afet bölgesinde kesintiye uğramadan sürekli bir yol elde etmek.

Anahtar Kelimeler: Ayrık Perkülasyon, Anlık Konumlama ve Haritalama, Konumlama, Haritalama, Arama ve Kurtarma

*Dedicated to my family, beloved wife, and daughter*

## ACKNOWLEDGMENTS

Firstly, I would like to thank my supervisor Professor Aydan Müşerref Erkmen for her motivation, guidance, patience, and encouragement which helped me enhance my research. It was a great honor and experience to work with her for the last six years and our cooperation influenced my academic perspective highly. This thesis would not have been completed without her belief and endless and gracious support. I would also like to thank Assoc. Prof. Mustafa Mert Ankaralı, Assoc. Prof. Oğuzhan Çifdalöz and Assoc. Prof. Kemalettin Batur for their guidance in the thesis monitoring committee meetings which help me very much in editing this thesis. In addition, I would like to thank my thesis jury members Assist Prof. David M. Rosen and Assoc. Prof. Emre Özkan for their comments, suggestions, and guidance.

I am very grateful to have my sister, Yaren Yıldız, and my parents, Songül and Efendi Yıldız. I always feel their endless support, love, and patience to me.

Last but not least, I would like to thank my wife, Nefise Yıldız. I am thankful for her eternal love, unending patience during the preparation of this thesis, and continuous encouragement in my life and, I would like to thank my future daughter who accompanied me in the womb of her mother in the final stages of my thesis. Without their support and giving strength, I could not finish this thesis.

## TABLE OF CONTENTS

## LIST OF TABLES

TABLES

## LIST OF FIGURES

FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 Motivation

The inclusion of robotic applications in our daily activities increases day by day with the development of complex mechanical manipulation mechanisms, powerful sensor types, control allocation methods, and artificial intelligence algorithms. The noticeable application of robots can be observed in search and rescue (SAR) operations in disaster regions. Disaster regions can be a result of natural (e.g. earthquakes, landslides, hurricanes, avalanches, and floods) or manmade disasters (bombing, nuclear disasters, fires). The common points of disaster regions are including complex, sharp, and unpredictable rubbles, very small voids to navigate, and hazardous environments for humans such as possible poisonous gases. In those areas, highly hostile to humans, SAR operations executed by robots should be fast enough to reach victims in time and carefully handled to avoid any risk of causing additional damage or collapse in the disaster area. Under these circumstances, the usage of autonomy in robots for those areas is gaining more importance than before. People in SAR operations can be assisted by robots to reach deeper regions of debris with small confined areas and generate a map of the environment to detect critical parts of the debris to handle the situation carefully without endangering themselves and the survivors.

In the last decade, the advancements in SAR robotics have increased with the development of hardware and software technologies. Different types of unmanned ground vehicles (UGV) and unmanned air vehicles (UAV) are utilized for the vast majority of tasks such as; exploration, surveillance, reconnaissance, and inspection in SAR operations [2-11]. Despite the advances in the area of applications, there are still considerable challenges to overcome for further developments. Especially

in highly unstructured environments, robots need to navigate in harsh and challenging fields without global knowledge of the terrain and complex structure of obstacles, and dead-end occurrences with possible collapsing of some part of the debris. Also finding a safe and continuous path with the adaptation of map changing, uncertainty in measurement in harsh environments, uncertainty and fault case scenarios in the robot actuator mechanism can be added to the previously listed challenges.

The utilization of robots in SAR operations still needs meticulous attention to cope with these problems and to present more robust and trustworthy solutions for future practices. By considering the listed challenges, we propose the Explosive Percolation-based SLAM approach for SAR operations in challenging environments. Today robotics SAR solutions consider large and relatively structured environments to be searched such as caves, mines, etc. However, when we considered highly unstructured and hard-to-reach environments like the debris in an earthquake, there are limited research on that area and the existing ones hardly answer the needs within this perspective.

## 1.2 Problem Statement

The main structure of the problem is built on navigation in a highly unstructured field of a disaster region. The voids among the rubbles can be very narrow and the rubbles consist of irregular shapes, which makes environmental recognition hard for mapping and localization. Besides these problems, the structure of the debris is fragile and the structure can change under collapsing debris. This causes the occurrences of successive trapped regions and dead-ends. It is expected from a SAR robot to find a continuous and safe path to the victim's location and map the shape-changing environment adaptively.

The map of the inner structure of the environment cannot be known prior. Therefore the initial position of the robot is not known globally by the robot. This

brings an initial uncertainty for the localization of the robot and the mapping of the region for exploration purposes. The uncertainty of the sensors and the robot's inner actuation system should be considered as well during a search. These uncertainties will highly likely affect the performance of the robot’s search in SAR operations.

The main purpose of exploration is to find the victim’s vital signal in a limited time. Thus, the selection of a target direction that brings in more knowledge to find the location of a victim gains importance for SAR operations. Thus, another important problem during exploration is the selection of the proper search direction within the debris for a guided exploration toward possible victims.

## 1.3 Objectives and Goals

The main objective of this thesis is to enhance SLAM methods with explosive percolation in order to reach the unknown and unstructured regions under the rubbles of the disaster area toward the victim’s vital signs by deciding a possible path utilizing rubble voids within them.

To be able to reach this aim, we can list our goals as,

1) The localization of the robot needs to be sustained without the initial information about the highly unstructured territory. The robot should be capable of initiating the SAR process with unknown initial position states.

2) We select our target points during searching to lead us to the unknown territory of the disaster area. This target point selection should serve to find the vital signal location by exploring the farthest distance in the area. We need to cover disaster regions as far as possible to increase the chance of detecting vital signs.

3) A safe path without obstacles avoiding dead-ends needs to be determined in reaching the victim's location. This path needs to be optimally modified if the environment changes during collapse while SAR operations are undergoing. Since

the disaster regions especially after earthquakes include fragile structures that can collapse and change the internal structure in debris, the SAR robot should be capable of avoiding dead-ends and traps and adapting itself the shape-changing environments.

4) The map of the environment needs to be extracted alongside the victim search to assist SAR teams and also visualize safe paths under debris. The SAR team can initialize the rescue mission by analyzing this safe path and reach the victim's location without endangering themselves and the victim.

## 1.4 Methodology

The search and rescue operations consist of two phases as it is understood by name. The *search* refers to finding the victim’s location and the *rescue* is about the activities related to extracting the victim safely. In our thesis, we focus on the search part and annunciation of the victim's location to SAR teams with a safe path and mapping of the exploration part of the debris.

In localization and mapping algorithms, there are different Bayesian approaches such as Kalman Filter, Extended Kalman Filter, Unscented Kalman Filter, etc. However, to overcome the linearization error and adapt to the nonlinear effects of the robot motion and sensor measurements, and multi-modal noise characteristics apart from the Gaussian Filters, we choose the Particle Filter (PF) method because of its ease of implementation and applicability in real-time problems. In this sense, the FastSLAM 2.0 algorithm is one of the methods that include a particle filter approach. Therefore, in our approach, the localization and mapping of the debris by the SAR robot are sustained with FastSLAM 2.0 algorithm

The features in debris can be very complex shapes to express them in mapping. Therefore, the representation of obstacles and indentations is achieved with the occupancy grid map. The reasons to choose the occupancy grid map are in two ways. The first reason is that the representation of complex shapes can be achieved

with grid cells' occupancy value. In this way, a large and complex area map can be scaled down to a simple scale according to grid size to increase computational performance. The second reason is that the Explosive Percolation method needs to occupancy probability of the obstacles in the penetration region. Similarly, each cell in the occupancy grid map includes a probability value representing the occupancy probability. In this sense, the occupancy grid map method is ideally suited for this approach. The occupancy probabilities can be assigned with different measurement sensors such as cameras, sonar range sensors, or LIDAR. If we consider the search region with low light conditions it will be not suitable for the camera. Also, the sonar sensors have limited sensor range and can give faulty readings in case of absorption of sound waves. Thus, the LIDAR sensor can sustain our needs in such a harsh environment to measure the occupancy values of grids within the extracted map.

The main contribution of this thesis is utilizing *Explosive Percolation (EP)* to obtain a novel continuous path finding within rubbles and avoid successive dead-ends encountered during the search. Fundamentally, the percolation theory deals with the connection of components in large clusters in statistical mechanics. For example, consider a square lattice and try to occupy this lattice with random cells with occupation probability $p$. When this occupation probability is below the threshold probability $p_c$, there will be no large cluster to walk from one side of the lattice to the other side. However, when this value is greater than $p_c$, we can find a path that connects two ends. By utilizing this principle, to extract our obstacle-free path into unknown regions in our occupancy grid map, the explosive percolation method is used to obtain a percolation path that connects the robot location to the unknown target location. The details of EP can be found in Chapter 3.2.

After we obtain a percolation path within the voids in debris, the path should be revised to be followed by the robot. With path pruning and smoothing algorithms, it is aimed to obtain a more suitable path for the robot. For the path control algorithm, we take the advantage of Serret-Frenet frame on the path curve.

## 1.5 Main Contribution of Thesis

The main contribution of the thesis can be listed as follows:

- A novel path-search algorithm is developed for the navigation of robots in a highly unstructured, complex environment. Although our approach can be combined with any SLAM algorithm, we bring together the FastSLAM and Explosive Percolation to find a continuous path into an unknown region of the debris. Therefore, this study is the first implementation endeavor to reveal the power of EP in the robotic field.
- In literature, avoiding successive dead-ends and continuing the search within rubbles still preserve its importance. EP method sustains avoiding obstacles and dead-ends by treating the robot like a fluid particle finding its way among the porous media. An obstacle-free path can be achieved by utilizing the voids of the debris without extra effort.
- Most of the active SLAM approaches in the literature focus on decreasing the entropy in localization and mapping information. To procure this objective, the existing studies consider the coverage of the maximum area. However, in addition to minimization of localization and mapping entropy, in SAR operations the crucial point is the exploration of the debris to find the victim's location. In our approach, this criteria comes first instead of covering the maximum area objective.

## 1.6 Outline of Thesis

The remainder of this thesis is organized as follows.

In Chapter 2, we provide the related work about SAR robot applications in literature, SLAM methods for exploration and exploitation, and percolation theory application areas in literature.

In Chapter 3, The mathematical background of FastSLAM 2.0 is presented in detail. After that, percolation theory basics and explosive percolation are shared.

In Chapter 4, our proposed explosive percolation-based SLAM approach for SAR operations is given in detail with the enhancement of the FastSLAM 2.0 algorithm, frontier target selections, explosive percolation path generation algorithms, path pruning, and smoothing algorithms, and path following control structure.

In Chapter 5, we present the experiments and discussion of the proposed method with different scenarios. For the experiments, we prepare a simulation environment. At the final of this chapter, we share the sensitivity analysis of our approach for different simulation parameters and comparison results of proposed method with the widely used entropy-based SLAM method and mostly used pathfinding algortihms.

In Chapter 6, the conclusion and future work are given with a summary and planned future adaptation of the algorithm.

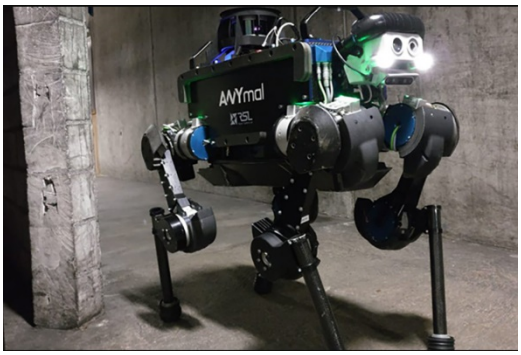
# CHAPTER 2

# LITERATURE REVIEW

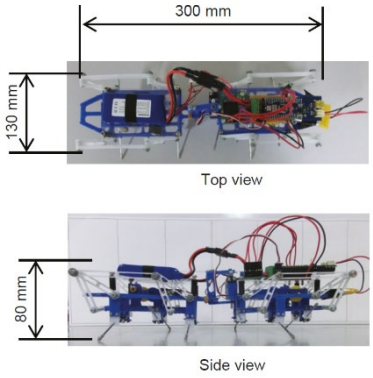
Historically, SAR robot utilization was proposed with the tragic loss of serious events. In 1995, early publications about the use of robots in the Hanshin-Awaji earthquake in Kobe were presented. After 2005, wide usage of SAR robots was observed in a terrorist attack on the World Trade Center, and the natural disasters of Katrina, Rita, and Wilma [1]. As of today, the abilities of SAR robots have been extended to diverse platforms. Ground robots have been observed for underground operations [2], aerial platforms for surveillance and searching [3], and water-based platforms for search and rescue on/under the sea [4].

In the modern world, with the developing sensor types and actuation mechanisms, the expected tasks to be accomplished by robots can be; surveillance, reconnaissance, an inspection of the structure, removal of rubbles, mapping, searching for victims, or a combination of them. Another critical aspect of the robotic SAR missions is the size and shape of the robot. This feature directly affects the success of the mission. There are various sizes and shapes of robots existing in the literature. For example, bio-inspired snake-like robots [5], shape-change drones [6], tracked and/or wheel-type robots with/without a robotic arm to grasp [7], walking robots [8], and micro-scale robots [9]. In Figure 2.1, some of these robot types are shared to form an estimate.

(a)




(b)




(c)

Figure 2.1. (a) Carnegie Mellon University snake robot used for Mexico quake survivors[10]. (b) Darpa SubT challenges the Cerberus team robot [8]. (c) Micro rescue robot for searching disaster victims [9].

Recent studies propose the collaboration of a multi-robot system to eliminate faults and errors during search and rescue. Perception of disaster regions with heterogeneous multi-robot systems shows significant improvements in SAR operations[11].

In this chapter, we present the developments in SAR robotic fields and compare them with our approach to indicate the gaps in literature and how we address them. Firstly, we take a look at the SAR robotic publications in the literature. Later on, SLAM approaches that have been used in SAR are investigated. Finally, to give insight into the percolation theory, the application areas and studies on percolation are shared.

## 2.1 SAR Robots in Literature

Early designs of SAR robots reach out to the COMETS project [12]. In this project, multi-UAV systems were utilized collaboratively with heterogeneous UAVs. The robot team used this architecture for fire detection and monitoring with terrain mapping. ICARUS is a European project to search for victims in a crisis scenario [7]. With two main UGVs, it was aimed to enter small enclosures, detecting human survivors. The high-level instructions come from the base station, therefore, this project is semi-autonomous in that manner. As part of TRADR projects, the researchers focus on the human-robot team effort in disaster response [13], [14]. As in these early research, the studies do not deal with the SAR in a highly complex and unstructured environment. They find their search path based on relatively large disaster regions, not a confined irregular small-scale debris. Also, they are semi-autonomous and they need a person to supervise the robot search.

In recent years, a major competition in the field of search and rescue robotics has been presented under Darpa Subterranean (SubT) Challenge. This competition especially covers the usage of underground UGVs for ground operations [15]. Team CERBERUS is one of the competitors who won the DARPA challenge [8]. The main approach of the CERBERUS is multi-modal and multi-robot mapping which is centralized with a server. They used LIDAR, IMU, vision, and encoder sensors. The local map of the competition area is extracted from each walking, flying, and roving robot. Then, the submaps are integrated to obtain a globally consistent map of the area. The major performance problem of the CERBERUS is the fine-tuning of SLAM and sensor parameters. They need to be hand-tuned and differ among the robot platforms. Another competitor is Team CoSTAR. They used different sensor types LIDAR, visual-inertial, and encoders in their multi-sensor front-end and back-end structure. The system consists of three interfaces: 1) Single-robot front-end interface for local robot trajectory and perception. 2) Multi-robot front-end interface for the base station to receive each robot's trajectory and map knowledge. 3) Multi-robot back-end interface for calculating optimal

trajectory to continue mapping [16]. One of the significant problems of the team is finding a suitable set of parameters for front and back-end sections. The other teams in this competition are Team CSIRO, Team CTU-CRAS-Norlab, and Team MARBLE. Detailed information about the projects can be found in reference [15]. The SAR robots used in this project mainly focus on autonomous mapping in a mine. They compare their mapping and localization performances to cover search regions. Again the mapping environment in these projects is large indoor areas. They do not address finding victims in highly unstructured confined places and a safe path to reach the victim's location. When we compare the pathfinding and mapping in a large and structured environment versus in an irregular complex space with a shape-changing environment because of collapsing debris, we can observe that the studied projects cover the SAR in robotics only at a certain level in a specified environment.

In below Table 2.1, important SAR projects with their description can be found in time sequence. The detailed version can be found in reference [11].

Table 2.1 International SAR projects with their description and usage area.

| **Projects** | **Year** | **Description** | **Usage Area** |
|---|---|---|---|
| COMETS | 2002-2005 | Real-time control of multiple UAVs. | Forest Fire |
| PeLoTe | 2002-2005 | Human-robot team for SAR. | Firefighting |
| MEXT.DDT | 2002-2007 | Rubble robots for earthquakes | Earthquake |
| Guardians | 2006-2010 | Swarm robots for urban ground. | Firefighting |
| NIFTi | 2010-2013 | Human-robot coop. in dynamic envr. for SAR. | Urban disaster |
| Darius | 2012-2015 | Unmanned systems for SAR. | Forest, urban, maritime |

Table 2.1 (Continued)

| | | | |
|---|---|---|---|
| ICARUS | 2012-2016 | Assist for human SAR operations. | SAR integration |
| TRADR | 2013-2017 | Long-term human-robot teaming | Industrial envr. |
| Centauro | 2015-2018 | SAR with telepresence | Harsh envr. |
| AutoSOS | 2020-2022 | Multi-UAV for maritime SAR | Maritime |

In summary, although the success of the given studies, none of them addresses the navigation within rubbles considered highly unstructured and irregular shapes. The search areas for the given studies consider mines, underground tunnels, forest fires, and marine-type search and rescue missions which can be considered relatively well-structured environments. In our approach, we aim to fill that gap in the literature with a novel path-finding approach by utilizing explosive percolation.

## 2.2 SLAM Methods in Literature

Simultaneous localization and mapping (SLAM) include the estimation of robot states with the construction of the environment model with the help of onboard sensors. The states of the robot can be counted as the position and orientation parameters (localization) and the mapping procedure consists of the representation of landmarks and obstacles within the operating environment. Even if the published resourceful research in the SLAM field area, there are still open problems waiting to be answered in terms of the robustness, and resilience of the algorithms for a variety of scenarios in a real-world application.

In the survey of Durrant et al. The foundation of the SLAM problem was laid in 1986 at the IEEE Robotics and Automation Conference [17]. Thrun et al. [18] achieved the usage of the Kalman Filter in SLAM and approach the problem with

probabilistic localization and mapping methods. In Thrun's book, we encounter different probabilistic estimation techniques. Firstly Kalman Filter is used for the estimation of robot states and landmark position estimation based on linear system dynamics. The Gaussian noise assumption is also accepted in robot motion and measurements. However, in the real world, we can encounter non-linearities and multi-modal errors which does not show Gaussian characteristics. Therefore, Extended Kalman Filter (EKF) is a widely excepted method in SLAM problems to handle nonlinearities by linearizing them. In this way, we can continue to use the Gaussian noise assumption. Also, EKF-SLAM is considered an online-SLAM approach, in other words, the posterior estimations are calculated by using momentary pose values. Another SLAM method is GraphSLAM. GarphSLAM from a sparse graph including nonlinear constraints of measurement and motion model. By optimizing these constraints, the maximum likelihoods of robot and landmark states are obtained. The drawback of this method is solving offline-SLAM problems. It needs the history of poses of robots and feature states on the map. In return, GraphSLAM produces more accurate maps than EKF-SLAM.

Apart from the Gaussian assumption, nonparametric filters are encountered commonly in estimation problems. Instead of using a functional form, nonparametric filters utilize random samples selected from posterior functions. Thus, they have a finite size value. The famous one of these filters is the *Particle filter*. The application of particle filters into SLAM problems appears as FastSLAM. The advantages of FastSLAM are maintaining the posterior estimation over each particle not most likely one as in previous ones, solving non-linear robot motion models instead of using linear functions, and implementing an online SLAM algorithm [19].

In the last decade, by taking the previous SLAM methods as a reference, with advancing sensor types fusion we have witnessed the emergence of various 2D/3D SLAM types. These types can be divided into two parts: 1) LIDAR SLAM. 2) Visual SLAM. In Table 2.2, we present some of the lidar and visual-based methods [20],[21].

Table 2.2 LIDAR and Vision-based SLAM algorithms in the literature.

| **LIDAR SLAM** | **VISUAL SLAM** |
|---|---|
| LIO-SLAM (3D LIDAR) | SVO-SLAM (Monocular) |
| HDL Graph SLAM (3D Lidar) | ORB-SLAM (Monocular/Stereo) |
| LOAM (3D Lidar) | PTAM (Monocular) |
| LEGO-LOAM (3D Lidar) | LSD-SLAM (Monocular) |
| HECTOR-SLAM (2D Lidar) | DSO-SLAM (Monocular) |
| Gmapping (2D Lidar) | RTAB (Stereo) |

The examples of some research on mentioned SLAM approaches can be shared as follows: Chen et al. [22] use the LOAM method for indoor mapping and they compare the results with LEGO-LOAM and LIO-SLAM. They present the results by using ROS and Gazebo in a simulation environment with a turtle bot. Lim et al. [23] suggest a new visual SLAM method with a monocular camera and they evaluate the results with the KITTI database. As for the search and rescue operations, Tardioli et al. [24] purposes the usage of semantic feature recognition with EKF to navigate within tunnels for underground operations. Petrlík et al. [25] show a 2D occupancy grid Hector SLAM approach in the aerial platform for SAR operations within a constrained workspace.

So far we present the SLAM algorithms as a passive estimation problem of robot and landmarks states. To improve the mapping and localization results, active SLAM is used to control robot motion. Two concepts come forward in this discussion; exploitation, and exploration. By exploration, we aim to discover unknown areas in a searching area and we use exploitation to diminish the uncertainty in localization error by revisiting the previously discovered areas. Therefore, active SLAM searches for a balance between exploration and exploitation [26]. Entropy reduction for both map and pose estimation error is a proposed method for autonomous SLAM in research [27]. Blanco et al. [28] propose an expected map (EM) and mean information approach [MI] to optimize

pose and map uncertainties. Carlone et al. [29] consider the Kullback-Liebler divergence for the posterior approximation. This metric allows the robot to make decisions between exploration and exploitation.

In our study, we choose the FastSLAM algorithm because of its ease of implementation, and real-time applicability. Furthermore, besides the consideration of map (exploration) and localization uncertainty value (exploitation), we take into account the search for a survival location. Therefore, survivor location is another parameter for the exploration and exploitation strategies of the robot to head in the direction of the victim's location prioritizing vital signs in exploration.

## 2.3 Percolation Theory Applications in Literature

Percolation theory has been a popular studied area in physics to explain the onset of large connectivity within networks, clusters, and porous media. To understand the process of percolation institutively, let us give an example of an infinite-size square grid of points. Consider we connect the neighbor points with a line in random order. We represent this randomness value with $p$ which is occupation probability. As we increase the $p$ value more connections occur in the lattice and after a certain value of $p \geq p_c$ where $p_c$ is percolation threshold, the largest cluster in the lattice will be obtained that connects two ends of the lattice [30]. The illustration is given in Figure 2.2.

Percolation theory has found a broad application area for various problems: networks [31], magnetic models [32], conducting materials [33], and forest fire [34], epidemic disease spread [35], etc.

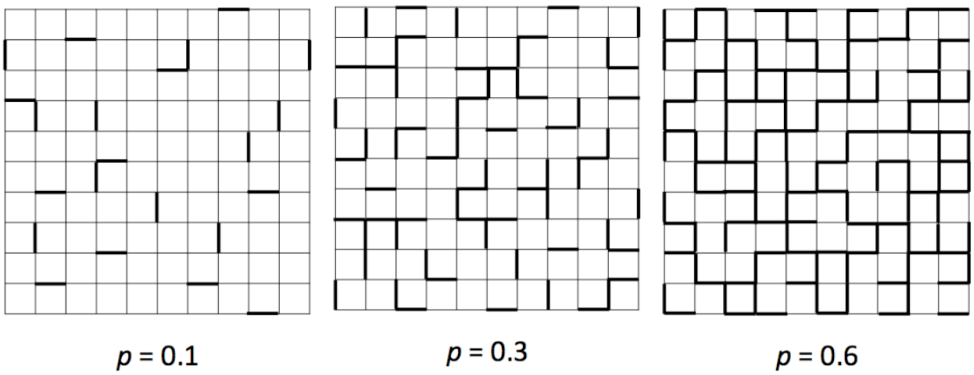



Figure 2.2. Percolation model in the square lattice with occupation probabilities $p = 0.1, p = 0.3$, and $p = 0.6$. For this type square lattice $p_c = 0.5$ [36].

Recent research reveals that another important phenomenon in percolation could be its order. In other words, the emergence of large clusters in percolation models shows continuous transitions which are also called second-order transitions. Achlioptas et al. [37] show that by changing the random selection rule in the classic Erdös-Rényi (ER) model, the percolation show discontinuous transition which is known as first-order characteristics. This abrupt transition in the occurrence of a large cluster is called *Explosive Percolation (EP).* In real-world examples, we encounter the EP process in network structure, and disordered media (i.e. flow in porous media, thermal conductivity, polymerization) [38]. Rozenfeld et al. [39] present the evolutionary human protein network process which is initialized with disconnected proteins and added sequentially afterward to lead large isolated components of connected nodes between them. Cho et al. [40] consider the Brownian motion of clusters in diffusion-limited cluster aggregation work. They show that the largest cluster Brownian motion prevents growth and leads to the emergence of giant clusters discontinuously.

Percolation theory is rather new to robotic applications in SLAM, and there are only a limited number of studies that demonstrate the applicability of basic percolation methods. Topal et al. [41] propose a percolation-enhanced multi-robot exploration of the unstructured environment. Actually, this study does not directly apply percolation theory to guide the robot but uses a method inspired by

percolation theory and the demonstration is realized in a relatively structured environment. Karahan et al. [42] show firstly invasion percolation theory and entropy-based SLAM algorithm for robot exploration. Invasion percolation is the main exploration guidance module for predicting upcoming voids from the frontiers of explored parts of the structure. The one problem in this study is the assumption of obstacle continuity which is not always the situation. Moreover, finding a path within voids based on invasion percolation uses three matrices with the size of the occupancy grid map. This brings a considerable computational burden to the SLAM approach although presented in a simple structured environment.

As we mentioned in the introduction, we propose here an explosive percolation-based FastSLAM algorithm to eliminate the previous research gaps in finding a safe path and mapping in highly unstructured disaster regions such as earthquakes and collapsing of structures. And we present a solution by testing it with a different SAR algorithm and at different maps and initial state conditions without giving priory information about the search region to guide a robot within a complex shape-changing environment due to collapsing of debris for SAR operation.

# CHAPTER 3

# MATHEMATICAL BACKGROUND

In this chapter, we provide the mathematical background used in our explosive percolation-based SLAM approach. Firstly we introduce the FastSLAM 2.0 algorithm basics and its inner algorithms which are motion and perception models, occupancy grid map, and particle filter. Secondly, we present the percolation theory mathematical overview and explosive percolation formulation that we used in our approach.

## 3.1 FastSLAM 2.0

FastSLAM divides the SLAM problem into two parts. The first is robot localization and the second one is landmark estimation conditioned on the robot's states. This approach comes from the conditional independence of the SLAM problem [43]. This observation makes valid the use of Rao-Blackwellized particle filters in FastSLAM. Therefore, FastSLAM uses low-dimensional EKFs to estimate map features that are conditionally independent.

The advantages of the FastSLAM algorithm [18]:

- FastSLAM algorithm is computationally advantageous over only EKF-based SLAM.
- FastSLAM maintains the posterior estimation for each particle not for the most likely one. Therefore, this makes FastSLAM most robust compared to other algorithms.
- Non-linear motion models can be adapted to particle filters. In that sense, FastSLAM does not need linear approximation for non-linear models.

- FastSLAM solves the online-SLAM problem and uses particle filters to estimate one pose at a time.

In the following subsections, we analyze the important aspects of the FastSLAM 2.0 algorithm.

### 3.1.1 Motion and Perception Model

In probabilistic estimation problems, the motion model consists of the state transition probability $p(x_t|\ u_t, x_{t-1})$. This model sustains us in the prediction of the next step in motion. Another important feature is the measurement (perception) model $p(z_t|\ x_t)$ which is useful in the calculation of the posterior section (i.e. measurement update step). We will begin firstly with motion models, then describe the measurement model that we used in our approach.

We describe the motion of the robot in 2D space with its 3-pose variable $(x, y, \theta)$. The illustration of the robot's states is given in Figure 3.1.

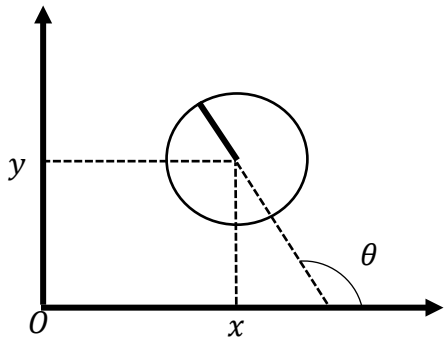



Figure 3.1. The schematic of kinematic motion parameters of the robot in 2D.

The probabilistic kinematics of the motion model can be expressed with different approaches. These are velocity and odometry models. Moreover, these methods can be subdivided into closed-form calculation and sampling calculation. We mention here velocity models with sampling calculation which is used in our approach. The details about these methods can be found in reference [18].

The velocity motion model uses the two velocities as a control parameter for robot states. These are translational and rotational velocity values $v_t, \omega_t$. The positive directions for the inputs: counterclockwise for rotation is positive and forward motion is positive for translation.

**Algorithm 1:** Sample Velocity Motion Model $(u_t, x_{t-1})$

1: $\hat{v} = v + sample(\alpha_1 v^2 + \alpha_2 \omega^2)$

2: $\hat{\omega} = \omega + sample(\alpha_3 v^2 + \alpha_4 \omega^2)$

3: $\hat{\gamma} = sample(\alpha_5 v^2 + \alpha_6 \omega^2)$

4: $x' = x - \frac{\hat{v}}{\hat{\omega}} \sin(\theta) + \frac{\hat{v}}{\hat{\omega}} \sin(\theta + \hat{\omega} \Delta t)$

5: $y' = y + \frac{\hat{v}}{\hat{\omega}} \cos(\theta) - \frac{\hat{v}}{\hat{\omega}} \cos(\theta + \hat{\omega} \Delta t)$

6: $\theta' = \theta + \hat{\omega} \Delta t + \hat{\gamma} \Delta t$

7: $Return\ x_t = (x', y', \theta')$

**Algorithm 2: S**ample Normal Distribution$(b^2)$

1: $Return\ \frac{1}{2} \sum_{i=1}^{12} rand(-b, b)$

In Algorithm 1, instead of using the conditional probability function of motion model $p(x_t |\ u_t, x_{t-1})$, we sample from the motion model to generate the model of density function for fixed control input $u_t$ and $x_{t-1}$. The variables $\alpha_1$ to $\alpha_6$ are the noise parameters to perturb the input parameters. To generate the samples we utilize $sample(b^2)$ given in Algorithm 2 with zero mean and variance $b^2$. Function *rand* represents the pseudo-random generator with uniform distribution. Note that we need to perturb the final orientation with $\hat{\gamma}$. The reason is to eliminate the degeneracy in the calculation of the posterior distribution which has pose states represented in three-dimensional

Next to the motion model, the measurement model covers the description of the environment with sensor measurements in probabilistic robotics. Different types of

sensors (e.g. sonar, LIDAR, visual) are used nowadays in robotics to generate the physical world model. In robot perception, probabilistically we represent the measurements with noise parameters with a conditional density function $p(z_t|x_t, m)$ where $z_t$ is the measurement at time $t$ $m$ is the map of the environment. A map can be thought of as a list of objects in the environment $m = \{m_1, m_2, \dots, m_N\}$. Maps are represented in two ways; featured-based and location-based. In our case, we choose to represent map features with the feature-based method based on the LIDAR beam model.

The feature-based method calculates the feature properties extracted from measurements $z_t$. An advantage of this model is the reduction in computational complexity. By using feature extraction algorithms, we can detect landmarks as map features from sensor measurements. As a note, in our study, these sensor measurements come from the LIDAR beam model. Map features can be edges, corners, and object-distinct properties calculated based on measurements. Let us assume we have a function $f(z_t)$ which takes measurements as input and gives landmark coordinates and signature in the form of $(r_t^n, \phi_t^n, s_t^n)$ in the robot's local coordinate frame. $r_t^n$ represents the range of landmark $n$, $\phi_t^n$ represents bearing of landmark $n$, and $s_t^n$ is the signature of the landmark which can be a numerical value, RGB value, height and color values, or a vector symbolizing that landmark.

The feature-based sensor model can be represented as follows: let's assume the map feature location in the global coordinate frame is represented as $m_{i,x}$ and $m_{i,y}$. We need to add noise values to our sensor model, so we choose zero-mean Gaussian noise on feature range, bearing, and signature parameters which are symbolized as $\epsilon_{\sigma_r^2}, \epsilon_{\sigma_\phi^2}, \epsilon_{\sigma_s^2}$ respectively. See Equation 3.1 for the details.

$$\begin{bmatrix} r_t^i \\ \phi_t^i \\ s_t^i \end{bmatrix} = \begin{bmatrix} \sqrt{(m_{i,x} - x)^2 + (m_{i,y} - y)^2} \\ atan2(m_{i,y} - y, m_{i,x} - x) - \theta \\ s_i \end{bmatrix} + \begin{bmatrix} \epsilon_{\sigma_r^2} \\ \epsilon_{\sigma_\phi^2} \\ \epsilon_{\sigma_s^2} \end{bmatrix} \qquad 3.1$$

### 3.1.2 Occupancy Grid Map

The occupancy grid maps include fine-grained grid cells that constitute a 2D representation of 3D space. To illustrate the occupancy grid cells, let's choose $\boldsymbol{m}_i$ to demonstrate the grid cell. Then the map can be expressed as $m = \{\boldsymbol{m}_i\}$.

Each $\boldsymbol{m}_i$ includes a binary value that shows whether the cell is occupied or not. For occupied cells $\boldsymbol{m}_i = 1$ and for empty cells this value equals to $\boldsymbol{m}_i = 0$. If we don't have the information for an initial value of the cell, we choose $\boldsymbol{m}_i = 0.5$ for that cell. This implies that the information about that cell is minimum and the entropy value is at its maximum value. With this representation, we aim to estimate $p(\boldsymbol{m}_i |\ z_{1:t},\ x_{1:t})$ for all grid cell $\boldsymbol{m}_i$. Therefore, the estimation of the occupancy grid map turns into a binary Bayesian filter Algorithm [18]. The Algorithm can be seen in Algorithm 3.

**Algorithm 3:** Occupancy Grid mapping $(l_{t-1,i}, x_t, z_t)$

```
for all cells m_i
    if m_i is in the field of z_t
        l_{t,i} = l_{t-1,i} + Inverse Sensor Model(m_i, x_t, z_t) − l_0
    else
        l_{t,i} = l_{t-1,i}
    endif
endfor
Return {l_{t,i}}
```

Because we don't have information about each cell's initial state, the initial value of the grid cell $l_0$ can be taken as $0.5$. As for the measurement model, we use the LIDAR beam model. The *inverse Sensor Model* function uses this basis to assign occupied, free, or out-of-range decisions to cells. The *inverse Sensor Model* algorithm is given in Algorithm 4.

**Algorithm 4:** Inverse Sensor Model($\boldsymbol{m}_i, x_t, z_t$)

1: $r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$

2: $\phi = atan2(y_i - y, x_i - x) - \theta$

3: $k = argmin_j\left(\phi - \theta_{j,sens}\right)$

4: $if\ r > \min\left(z_{max}, z_t^k + \frac{\alpha}{2}\right) \left|\phi - \ \theta_{k,sens}\right| > \frac{\beta}{2}$

5: $\quad Return\ l_0$

6: $if\ z_t^k < z_{max}\ and\ \left|r - z_t^k\right| < \frac{\alpha}{2}$

7: $\quad Return\ l_{occ}$

8 $if\ r < z_t^k$

9 $\quad Return\ l_{free}$

10 $endif$

In this algorithm, the robot state is $(x, y, \theta)$, $x_i$ and $y_i$ is the center of gravity coordinate of the cell $\boldsymbol{m}_i$. $k$ index is the closest beam index value to the cell $\boldsymbol{m}_i$. $\beta$ is the width of the sensor beam and $z_{max}$ the measured range of the sensor. If the cell $\boldsymbol{m}_i$ is outside the range of the cell, in line 5 of the algorithm the cell value returns as it is. If it is in the measurement range of the sensor beam $z_t^k$ and between $\frac{\alpha}{2}$ range of the $r$, then $l_{occ}$ is returned. Otherwise, the function returns $l_{free}$.

### 3.1.3 Particle Filter

The particle filter is a nonparametric version of the Bayes Filter. In this approach, the posterior function is represented with a finite number of samples. The posterior function is denoted with $bel(x\_t) = p(x_t|z_{1:t}, u_{1:t}))$ in which the posterior probability is conditioned on past measurements and controls up to time $t$.The advantage of the particle filter is the modeling of nonlinear functions of random variables. The representation of the particle filter is

$$\chi_t = x_t^{[1]}, x_t^{[2]}, \dots, x_t^{[M]}. \quad 3.2$$

A particle represents the one instantiation of state $x$ at time t. $M$ is the total number of the particle set. The particle filter algorithm is shared in Algorithm 5 [18].

**Algorithm 5**: Particle Filter $(\chi_{t-1}, u_t, z_t)$

1: $\bar{\chi}_t = \chi_t = \emptyset$
2: $for\ m = 1\ to\ M$
3: $\quad sample\ x_t^{[m]} \sim p(x_t|u_t, x_{t-1}^{[m]})$
4: $\quad w_t^{[m]} = p(z_t|\ x_t^{[m]})$
5: $\quad \bar{\chi}_t = \bar{\chi}_t + \left\langle x_t^{[m]}, w_t^{[m]} \right\rangle$
6: $endfor$
7: $for\ m = 1\ to\ M$
8 $\quad draw\ i\ with\ probability\ \propto w_t^{[i]}$
9 $\quad add\ x_t^{[i]}\ to\ \chi_t$
10 $endfor$
11 $Return\ \chi_t$

In algorithm $w_t^{[m]}$ is called the importance factor. The importance factor is used to weigh the state prediction $x_t^{[m]}$ with the measurement. In other words, it is the weight of the particle $m$. Another important aspect of the particle filter is resampling or importance sampling between lines 7 and 10. However, the resampling process in particle filters comes with variance problems. The sampling variance increases with the repititive sampling process. In other words, the diversity will be diminished by sampling over and over again with a finite size of $M$ particles. To avoid this problem one of the proposed methods is *low-variance sampling*. This algorithm selects the particles with a random number, but still, preserves the importance weight factor. The algorithm is given in Algorithm 6. The

advantage of a low variance sampler is offering a more systematic approach concerning random sampling.

**Algorithm 6:** Low Variance Sampling $(\chi_t, w_t)$

$\widetilde{\chi}_t = \emptyset$
$r = rand(0, M^{-1})$
$c = w_t^{[1]}$
$i = 1$
$for\ m = 1\ to\ M$
  $U = r + (m-1)M^{-1}$
  $while\ U > c$
    $i = i + 1$
    $c = c + w_t^{[i]}$
  $endwhile$
  $add\ x_t^{[i]}\ to\ \widetilde{\chi}_t$
$endfor$
$Return\ \widetilde{\chi}_t$

### 3.1.4 FastSLAM 2.0 Algorithm

The power of the FastSLAM algorithm comes from the factorization of the full posterior algorithm $p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$ which includes robot states and map features [18].

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) = p(x_{1:t} \mid z_{1:t}, u_{1:t}) \prod_{n=1}^{N} p(m_n | x_{1:t}, z_{1:t})$$

$N$ is the total number of landmarks until time $t$. In each particle, we have $N + 1$ probabilities including the robot state. The particles and their composition in factorization form are given in below Table 3.1. Each particle consists of robot

pose estimation and mean $\mu$ and covariance Σ of each particle in map $m$ resulting from the Kalman filter.

Table 3.1 Particle in FastSLAM with robot states and feature estimation

| | **Robot States** | **Feature 1** | **Feature 2** | **...** | **Feature *N*** |
|---|---|---|---|---|---|
| Particle $k = 1$ | $\{x, y, \theta\}_{1:t}^{[1]}$ | $\{\mu_1, \Sigma_1\}^{[1]}$ | $\{\mu_2, \Sigma_2\}^{[1]}$ | ... | $\{\mu_N, \Sigma_N\}^{[1]}$ |
| Particle $k = 2$ | $\{x, y, \theta\}_{1:t}^{[2]}$ | $\{\mu_1, \Sigma_1\}^{[2]}$ | $\{\mu_2, \Sigma_2\}^{[2]}$ | ... | $\{\mu_N, \Sigma_N\}^{[2]}$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋱ | ⋮ |
| Particle $k = M$ | $\{x, y, \theta\}_{1:t}^{[M]}$ | $\{\mu_1, \Sigma_1\}^{[M]}$ | $\{\mu_2, \Sigma_2\}^{[M]}$ | ... | $\{\mu_N, \Sigma_N\}^{[M]}$ |

The implementation of the FastSLAM 2.0 algorithm is given in Algorithm. Note that, the algorithm given here considers only a single measurement at a time. However, in real life, a robot with SLAM can observe multiple measurements at a time. This issue is handled in our algorithm and the details are given in Chapter 4. The particle in the tuple form is given in Equation 3.3. In equation $x_t^{[k]}$ is robot pose, $N_t^{[k]}$ the number of features for particle $k$ at time $t$, $\tau_n^{[k]}$ is the probabilistic existence of the feature, $\mu_{n,t}^{[k]}, \Sigma_{n,t}^{[k]}$ are the mean and covariance of the feature in the particle.

$$Y_t^{[k]} = \left\langle x_t^{[k]}, N_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, \tau_1^{[k]} \right\rangle, \dots, \left\langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, \tau_{N_t^{[k]}}^{[k]} \right\rangle \right\rangle \tag{3.3}$$

**Algorithm 7:** FastSLAM 2.0 $(z_t, u_t, Y_{t-1})$

1: $for\ k = 1\ to\ M$

2: $take \left\langle x_{t-1}^{[k]}, N_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]}, i_1^{[k]} \right\rangle, \dots, \left\langle \mu_{N_{t-1}^{[k]},t-1}^{[k]}, \Sigma_{N_{t-1}^{[k]},t-1}^{[k]}, i_{N_{t-1}^{[k]}}^{[k]} \right\rangle \right\rangle$

3: $for\ j = 1\ to\ N_{t-1}^{[k]}$

4: $\hat{x}_{j,t} = g\left(x_{t-1}^{[k]}, u_t\right)$ $(Prediction\ of\ pose)$

5: $\bar{z}_j = h\left(\mu_{j,t-1}^{[k]}, \hat{x}_{j,t}\right)$ $(Predict\ measurement)$

6: $H_{x,j} = \nabla_{x_t} h\left(\mu_{j,t-1}^{[k]}, \hat{x}_{j,t}\right)$ $(Jacobian\ wrt\ pose\ of\ robot)$

7: $H_{m,j} = \nabla_{m_j} h\left(\mu_{j,t-1}^{[k]}, \hat{x}_{j,t}\right)$ $(Jacobian\ wrt\ map\ feature)$

8 $Q_j = Q_t + H_{m,j}\Sigma_{j,t-1}^{[k]} H_{m,j}^T$ $(Measurement\ information)$

9 $\Sigma_{x,j} = \left[H_{x,j}^T Q_j^{-1} H_{x,j} + R_t^{-1}\right]^{-1}$ $(Cov.of\ proposal\ distribution)$

10 $\mu_{x_t,j} = \Sigma_{x,j} H_{x,j}^T Q_j^{-1}\left(z_t - \bar{z}_j\right) + \hat{x}_{j,t}$ $(Mean\ of\ proposal\ dist.)$

11 $x_{t,j}^{[k]} \sim \mathcal{N}\left(\mu_{x_t,j}, \Sigma_{x_t,j}\right)(Sample\ pose)$

12 $\hat{z}_j = h\left(\mu_{j,t-1}^{[k]}, x_t^{[k]}\right)$ $(Measurement\ prediction)$

13 $\pi_j = \left|2\pi Q_j\right|^{-0.5} \exp\left(-0.5\left(z_t - \hat{z}_j\right)^T Q_j^{-1}\left(z_t - \hat{z}_j\right)\right)$

$(Correspondence\ likelihood)$

14 $endfor$

15 $\pi_{1+N_{t-1}^{[k]}} = p_0$ $(Likelihood\ of\ new\ feature)$

16 $\hat{c} = argmax\left(\pi_j\right)$ $(Maximum\ Likelihood\ correspondence)$

17 $N_t^{[k]} = \max\left(N_{t-1}^{[k]}, \hat{c}\right)$ $(New\ number\ of\ feature)$

18 $for\ j = 1\ to\ N_t^{[k]}$ $(Update\ Kalman\ Filters)$

19 $if\ j = \hat{c} = 1 + N_{t-1}^{[k]}$ $(For\ new\ feature)$

20 $x\_t\hat{}[k] \sim p(x\_t \dashv \mid x\_(t-1)\hat{}[k], u\_t)$ $(Sample\ pose)$

21 $\mu_{j,t}^{[k]} = h^{-1}\left(z_t, x_t^{[k]}\right)(initialize\ mean)$

22 $H_{m,j} = \nabla_{m,j} h\left(\mu_{j,t}^{[k]}, x_t^{[k]}\right)(Jacobian\ wrt\ map\ feature)$

23 $\Sigma_{j,t}^{[k]} = \left(H_{m,j}^{-1}\right)^T Q_t H_{m,j}^{-1}$ $(Initialize\ Covariance)$

24 $i_{j,t}^{[k]} = 1$ $(Initialize\ counter)$

25 $w^{[k]} = p_0$ $(Importance\ weight)$

26 $else\ if\ j = \hat{c} < N_{t-1}^{[k]}$ $(For\ observed\ feature)$

$x_t^{[k]} = x_{t,j}^{[k]}$

$K = \Sigma_{j,t-1}^{[k]} H_{m,j}^T Q_j^{-1}$ $(Kalman\ gain)$

$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K\left(z_t - \hat{z}_j\right)$ $(Update\ mean)$

$\Sigma_{j,t}^{[k]} = \left(I - K\ H_{m,j}\right)\Sigma_{j,t-1}^{[k]} (Update\ covariance)$

$i_{j,t}^{[k]} = i_{j,t-1}^{[k]} + 1$ $(Increment\ counter)$

$L = H_{x,j} R_t H_{x,j}^T + H_{m,j}\Sigma_{j,t-1}^{[k]} H_{m,j}^T + Q_t$

$w^{[k]} = |2\pi L|^{-0.5} \exp\left(-0.5\left(z_t - \hat{z}_j\right)^T L^{-1}\left(z_t - \hat{z}_j\right)\right)$

$(Importance\ weight)$

$else\ (all\ other\ features)$

$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]}$ $(Preserve\ old\ mean)$

$\Sigma_{j,t}^{[k]} = \Sigma_{j,t-1}^{[k]} (Preserve\ old\ covariance)$

$if\ \mu_{j,t-1}^{[k]}\ outside\ the\ sensor\ range\ of\ robot$

$i_{j,t}^{[k]} = i_{j,t-1}^{[k]}$ $(Not\ change\ counter)$

$else$

$i_{j,t}^{[k]} = i_{j,t-1}^{[k]} - 1$ $(Decrease\ counter)$

$if\ i_{j,t-1}^{[k]} < 0$

$discard\ feature\ j$

$endif$

$endif$

$endif$

$endfor$

$add\ \left\langle x_t^{[k]}, N_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, \tau_1^{[k]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, \tau_{N_t^{[k]}}^{[k]} \right\rangle \right\rangle\ to\ Y_{aux}$

$endfor$

$Y_t = \emptyset$

$do\ M\ times$

51 $draw\ random\ particle\ k\ with\ proportional\ to\ w^{[k]}$

52 $add\ \left\langle x_t^{[k]}, N_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, \tau_1^{[k]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, \tau_{N_t^{[k]}}^{[k]} \right\rangle \right\rangle\ to\ Y_t$

53 $enddo$

54 $Return\ Y_t$

In the algorithm, firstly we take the previous time step particles $Y_{t-1}$. Then we compare the single measurement with previous ones by calculating pose and measurement prediction between lines 4 and 13. The $g\left(x_{t-1}^{[k]}, u_t\right)$ and $h\left(\mu_{j,t-1}^{[k]}, \hat{x}_{j,t}\right)$ functions represent the robot motion function and feature-based measurement functions respectively. At the end of this cycle, we obtain correspondence likelihood $\pi_j$ of measurement compared to other features in particles. In line 16, we calculate the $argmax(\pi_j)$ to identify whether the measurement is a new feature or not. In the rest of the algorithm, we update the mean and covariance values of each feature in particle $k$. If the feature is new, we attain this feature an initial mean, covariance, and existence values. However, if this feature has been observed before, then we update its mean, covariance, and existence values. For the last condition, if the feature is not in the sensor measurement range, we preserve the related parameter. Otherwise, the feature that should be observed in the range is not detected and this means the feature is not reliable and we should diminish its existence value. When this value is less than zero, we remove the feature from the particle set.

### 3.2 Percolation Theory and Explosive Percolation Method

In physics, we have encountered the phase transition of substances and determined the properties of the system based on which phase it is in. In percolation theory one of the important aspects is the phase transition phenomena and its order. In general, the phase transition is called first-order if the system shows discontinuous

characteristics during phase change. Otherwise, it is called second-order. To give insight into phase order the resemblance comes from thermodynamics; for example in thermodynamics, the first-order system shows temperature-fixed phase change either energy is absorbed or emitted. In the second-order system, the transition occurs in a continuous fashion. The entropy and energy of the system change gradually.

In percolation theory, the order of the system is the ratio of the largest cluster to the size of the system. If $N$ is the total nodes in the lattice and $C$ is the largest cluster with $|C|$ nodes, the order $m$ is given by [45]

$$m = \frac{|C|}{N}. \tag{3.4}$$

In literature, there are various percolation types; continuum percolation, site and bond percolation, invasion percolation, etc. In classical percolation theory, all these percolation types exhibit second-order transition [44]. However, in recent research, Achlioptas et [37] al describe a new method during the selection of occupation sites in a network that produce the first-order transition and this percolation type is called *Explosive Percolation* (EP).

In the following subsections, the details and critical exponents of percolation theory and explosive percolation theory are shared.

### 3.2.1 Percolation Theory Basics

In classical percolation, the randomly occupying sites (site percolation) and bonds (bond percolation) are two basic versions of percolation theory on a lattice. This randomness is sustained with the occupation probability $p$. As $p$ increases more and more clusters emerge and unite. When $p$ reaches the critical value $p_c$ a giant cluster emerges in a way that connects two ends of the cluster. The cluster can be an infinite size or a finite size. Currently, we focus on site percolation, however,

the same principles can be applied to bond percolation. In Figure 3.2, we depict the site percolation process with the critical percolation threshold.

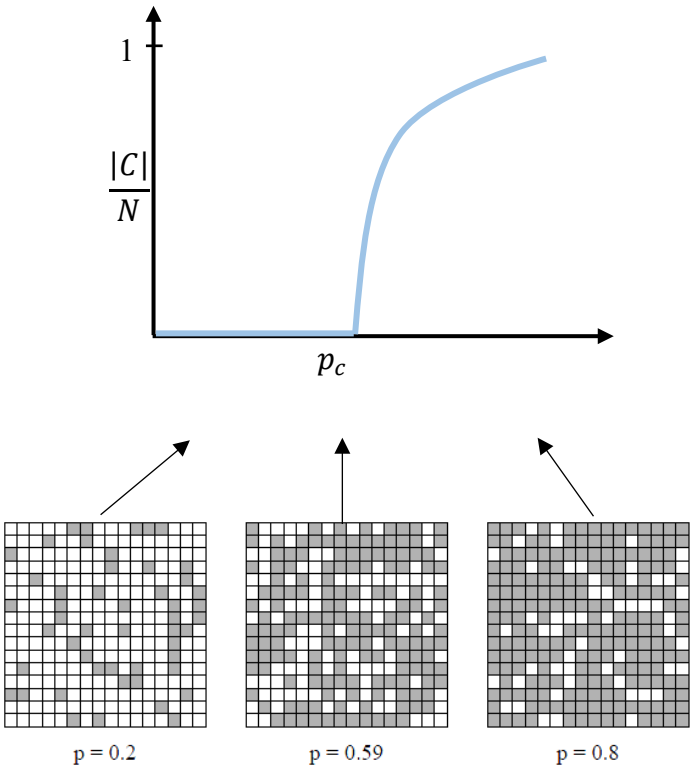



Figure 3.2. Site percolation process with the critical percolation threshold $p_c$

Near the critical threshold value $p_c$, there exists a set of numbers called critical exponents that show the behavior of percolation [45]. The first one is the $\beta$ exponent which takes its role to show the relation between order number and occupation probability.

$$m(p) \sim (p - p_c)^{\beta} \tag{3.5}$$

The average size cluster $\langle s \rangle$ is defined with $\gamma$ which is shown in Equation 3.6.

$$\langle s \rangle(p) \sim (p - p_c)^{-\gamma} \tag{3.6}$$

The correlation length which is defined as the mean distance between two sites is described with $\nu$.

$$\xi(p) \sim |p - p_c|^{-\nu} \tag{3.7}$$

These parameters have a special property that they are only dependent on dimension $d$ of the lattice. They are free from the configuration of the lattice.

In a site percolation with a lattice of dimension $d$, let's take the side length with $L$. Then we will have $L^d$ sites in the lattice. The occupancy probability of each site is symbolized with $p$. We define $X_i$ to represent if a site is occupied or not. If it is occupied $X_i = 1$, otherwise $X_i = 0$. The total number of occupied cells is then $\sum_i X_i$. From occupation probability, this should be expected to equal $L^d p$. The cluster with size $s$ can be defined as having $s$ occupied cells in the lattice. If the total number of clusters with size $s$ is represented with $N_s$, we can define the probability of any given site as a part of a cluster with size $s$:

$$n_s = \frac{N_S}{L^d} \tag{3.8}$$

$$\sum_s s n_s \tag{3.9}$$

$$\frac{s n_s}{\sum_{s'} s' n'_s} \tag{3.10}$$

Also, the average cluster size can be computed with the following equation:

$$\langle s \rangle = \sum_s s \frac{s n_s}{\sum_{s'} s' n'_s} \tag{3.11}$$

### 3.2.2 Explosive Percolation

In the year 2000, Dimitris Achlioptas proposed a method for the evaluation of a large cluster in percolation. The aim is to control cluster size $|C|$ on the onset of percolation in a way that it is delayed until the emergence. This process called as Achlioptas process. The first application and test of the Achlioptas process were conducted by Tom Bohman and Alan Frieze. This study shows the delay of the giant cluster on the Bohman-Frieze model. The algorithm is based on the selection of edges that connects two isolated nodes in a cluster. In this way, the percolation

threshold delaying was proved. In 2009, Dimitris Achlioptas, Raissa M. D'Souza, and Joel Spencer proposed the *Product Rule* (PR) during the evaluation of the selection of edges in a network. The process can be summarized as follow (Figure 3.3 (a)):

- Let edge $e_t$ connects two clusters $C(e_t^1)$ and $C(e_t^2)$ at time $t$
- Let edge $e_t'$ connects another two clusters $C(e_t'^1)$ and $C(e_t'^2)$ at time $t$
- Then $e_t$ is the selected edge if $| C(e_t^1)||C(e_t^2)| < | C(e_t^1)||C(e_t^2)|$. Otherwise $e_t'$ is accepted.

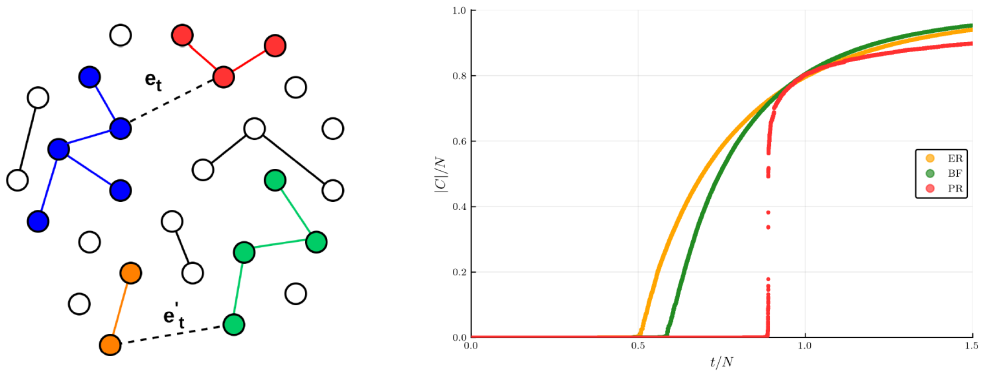
**Algorithm 8:** Product Rule($T$)

$A = \emptyset$
$t = 1$
$While\ t \leq T$
  $if\ \ | C(e_t^1)||C(e_t^2)| < | C(e_t^1)||C(e_t^2)|.$
    $A = A\ \cup \{e_t\}$
    $t = t + 1$
  $else$
    $A = A\ \cup \{e_t'\}$
    $t = t + 1$
  $endif$
$endwhile$

The comparison of the product rule (PR), Bohman-Frieze (BF) model, and Erdos-Rényi (ER) model can be shown in Figure 3.3 (b). Erdos-Rényi (ER) model is the random edge selection on a network in classical percolation. The order parameter is 0 until $p_c = 0.5$ for ER the model. In the BF model, it is delayed for a certain value $p_c$. In the product rule, we can observe this value is delayed until $p_c = 0.88$.

Moreover, after the critical point, the order parameter reaches discontinuously larger order values.

The debate about the discontinuous behavior of the Achlioptas process is still ongoing [45], however, it is obvious that the product selection rule can be used to control the emergence of a large cluster in a network. At the critical point, clusters are waiting to merge, therefore, after the critical point, the connection of edges results in an abrupt global connection. The rules can be varied in selection. In basics, there are two rules used in selection: product rule and sum rule.




(a) (b)

Figure 3.3. (a) The node selection in a network. (b)The comparison of the order parameter of ER, PR, and BF methods.

The explosive percolation proposed by Achlioptas was investigated on a network. This situation can be extended to site and bond percolation. In our case, we focus on the explosive site percolation due to the correspondence with the occupancy grid map. The details of the selection on an occupancy grid map can be found in Chapter 4.

# CHAPTER 4

# EXPLOSIVE PERCOLATION-BASED SLAM

## 4.1 General

In our proposed method, a novel active SLAM approach based on explosive percolation is presented towards navigation within highly complex unstructured environments for SAR operations. In a disaster region, the SAR robot has no initial information about the environmental structure. It needs to navigate within debris and extract a safe and robust path without being trapped within rubbles and a map of the environment to be utilized by SAR teams until reaching the victim's location. During search, the SAR robot should avoid dead-ends and obstacles and adapt to the dynamic environment by updating its local map in case of debris collapses. These are the crucial challenges that robot encounters in SAR mission. With the help of the explosive percolation-based active SLAM approach, the SAR robot overcomes the problems to reach the victim's location safely.

The general structure of the proposed approach is given in Figure 4.1. Our approach consists of the successive stages to perform the SAR operation in a disaster region.

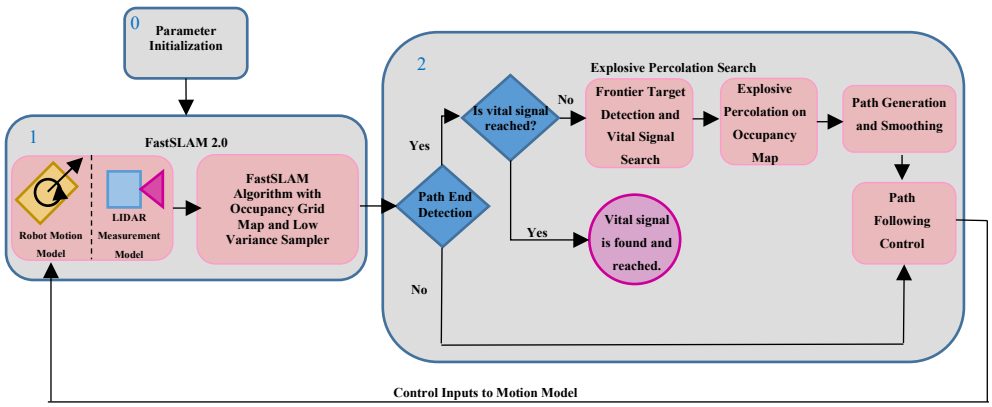



Figure 4.1. An Overview of explosive percolation enhanced FastSLAM.

The stages can be considered under two main parts: The first block belongs to the FastSLAM 2.0 algorithm which performs iteration of robot states using a motion model, measurements of the environment by LIDAR at each robot state, and the creation of an occupancy grid map. This block accepts the parameter initialization values as input and gives the occupancy grid map and particles of robot and obstacle states as output. Also, the particle filter variance control is realized in this block to increase the diversity of particles. In the second block, the explosive percolation search is performed according to the vital signal found and path-end detection criteria. Firstly this block accepts the occupancy grid with particle filter states. If the path-end is not detected, the path following control algorithm feedbacks motion control input to the robot motion model, and the loop is closed. If the path end is detected, the location is checked whether the vital signal is found or not. If the vital signal is found the process is succeeded. If it is not, then the frontier detection algorithm is utilized on the occupancy map to detect the target point, then the explosive percolation path detection is used to generate a percolation path to the target point on the occupancy grid map. After path generation and smoothing algorithms, the path following control algorithm is run again to sustain control input to robot motion mode to close the loop.

In this section, we present the details of our approach and how we implement the explosive percolation to SLAM with the given algorithms in Figure 4.1. The basics of FastSLAM 2.0 algorithms with particle filter and percolation are given in Chapter 3 separately. Therefore, in this part, we focus on the enhancement of the FastSLAM 2.0 algorithm to handle multiple landmarks measurement and the implementation of the explosive percolation method to the enhanced version of FastSLAM 2.0 algorithm to realize our main goal of exploiting the unexplored voids to percolate into the unknown region of the map and to reach the victim as soon as possible. In essence, the implementation of explosive percolation comes with decision points at the searching stage in block 2. To understand the process, let's imagine highly unstructured and complex debris. If a robot does not reach a pre-determined target point, it continues to follow the path with the path following

control algorithm until reaches that point. However, if the robot succeeds reaching in the target point, it checks whether the selected target point is a victim's location or not. If this target point is the victim's location, then the procedure is stopped, and the safe path among rubbles with the map of debris is transmitted to the SAR team. Otherwise, the robot selects a new target point within the field of view of LIDAR. The unexplored regions with high entropy values are selected as target points to explore the region further. In this way, we can reach the outermost regions in the searching area and increase the probability of finding a vital signal. After frontier target detection on the occupancy grid map, the explosive percolation is utilized to obtain a safe and obstacle-free path within the voids of rubbles. Because of that, our main objective is to reach the victim as fast as possible, we exploit the unexplored voids to percolate into the unknown region of the map. However, the extracted path on the occupancy map is highly intended to be followed by the robot with the path following control algorithm. Thus, we perform path pruning and smoothing processes on the found path. After that, path control feeds the control parameters to robot motion to continue the process until the victim's vital signal is detected.

In the rest of this chapter, the mentioned algorithms are discussed in detail with example scenarios.

## 4.2 Enhancement of FastSLAM 2.0

The FastSLAM 2.0 algorithm, given in Algorithm 7 in Chapter 3, has limitations to be applied for multiple detections of landmark locations. As indicated in Chapter 3, this algorithm uses single measurement at a time. However, when LIDAR sweeps a certain area within its field of view, multiple landmark points are extracted from LIDAR point cloud data. To handle this situation, we have to modify the given FastSLAM 2.0 algorithm of [18] towards a methodology that handles multiple measurements at a time and updates all particles in the particle filter with each landmark measurement.

Before going into the enhancement of FastSLAM 2.0 algorithm, as we know from Chapter 3, the FastSLAM algorithm uses a particle filter to update robot and landmark states. Therefore, each particle should include the robot and landmarks states. In that perspective, the particle structure in the SLAM algorithm is constructed as in Equation 4.1.

$$p^k = \left[ \begin{bmatrix} x & N \\ y & 0 \\ \theta & 0 \end{bmatrix} \begin{bmatrix} \mu_x & \Sigma_{11} & \Sigma_{12} & \Sigma_{13} & i \\ \mu_y & \Sigma_{21} & \Sigma_{22} & \Sigma_{23} & 0 \\ \mu_s & \Sigma_{31} & \Sigma_{32} & \Sigma_{33} & 0 \end{bmatrix}_1 \cdots \begin{bmatrix} \mu_x & \Sigma_{11} & \Sigma_{12} & \Sigma_{13} & i \\ \mu_y & \Sigma_{21} & \Sigma_{22} & \Sigma_{23} & 0 \\ \mu_s & \Sigma_{31} & \Sigma_{32} & \Sigma_{33} & 0 \end{bmatrix}_N \right]_k, \quad 4\text{-}1$$

where

- $p_k$ symbolizes the k-th particle.
- $x, y, \theta$ stand for estimated position and orientation of robot in 2D space at the k-th particle.
- $\mu_x, \mu_y, \mu_s$ are estimated position and identification of a landmark. The identification means that a numerical value symbolizes the distinctive properties of the landmark such as order number, RGB value, and multi-dimensional vector characterizing the landmark.
- $\Sigma_{ij}$ stands for the covariance matrix elements for a landmark. i is the total encountered number of the specific landmark. If the encountered number diminishes to 0, then this observed landmark is considered a noisy parameter or dynamic object. Therefore, a such landmark that has low reliability is removed from the particle set.
- $N$ is the total landmark number in the k-th particle.

The FastSLAM 2.0 introduced in Chapter 3 is improved according to the given pseudo algorithm below. To be able to process the observed multiple landmarks, we have to iterate the Kalman Filter update stage of Algorithm 7 in Section 3.1.4 for each landmark. As shown in Algorithm 9, in lines 4 and 5 we calculate the correspondence likelihood and new feature detection as in Algorithm 7. However, different from Algorithm 7, notice that in line 3, we iterate the measurement update

for each measurement landmark which does not exist in Algorithm 7. Also, between lines 11-19, we add else criteria to check the conditions of the landmark. These conditions can be such that the landmark is observed before and not in the field of view and at that moment the landmark is measured and updated and it should not be updated twice. In this way, we avoid repetitive updates of the Kalman Filter for the same landmark. If the landmark which is supposed to be observed is not detected within the LIDAR cone, the landmark counter is diminished in line 16. If this counter is smaller than 0, this means that the landmark is not a solid one and should be discarded. The whole process is repeated for each landmark to update its Kalman Filter parameters.

**Algorithm 9:** Enhanced FastSLAM 2.0 Algorithm$(z_t, u_t, Y_{t-1})$

```
1:  Initialization of Particles (Equation 4 − 1)
2:  for each particle k = 1: K
3:      for each measurement landmark m = 1: M
4:          Correspondence likelihood Estimation (Algorithm 7 Lines 3 − 14)
5:          Detection of New Feature (Algorithm 7 Lines 15 − 17)
6:          If new feature detected
7:              Create new landmark in particle k (Algorithm 7 Lines 20 − 25)
8           else if observed feature
9               Update Kalman filter of the feature in particle k (Algorithm 7 Lines 27 − 33)
10          else
11              if observed Landmark and not in the field of view of sensor
12                  Landmark counter i stay same
13              else if Landmark within measurements and updated
14                  Landmark counter i stay same
15              else
16                  Landmark counter i = i − 1
17                  if i < 0
18                      discard Landmark
```

19 | | | | | *end*
20 | | | | *end*
21 | | | *end*
22 | | *end*
23 *end*
24 *Run low variance sample* (*Algorithm* 6)

To better understand the process, on a simple map the particle structure propagation and their update sequence are shared below. In Figure 4.2, the robot's location on the simple map and the robot's local occupancy grid map with landmark locations is depicted. The landmark locations are indicated with green circles. The observed landmarks are determined according to the corner points of the obstacles. The details about the determination of landmarks are given in subsection 4.3. In this scenario, we observe three distinct landmark locations and in Table 4.1 their estimated position values at each particle are given for the initial time $t = 0$. For computational power reasons we select 10 particles constructed as in Equation 4-1.

At the beginning of Algorithm 9, each particle structure (Equation 4-1) is initialized with 3x2 zero matrix (Line 1). The first column represents the position states of the robot and the second column represents the total landmark $N$ which is 0 initially.

With the initial measurement of LIDAR, we detect 3 distinct landmark locations given in Figure 4.2. In line 2, the $M$ value is equal to 3.

For the initial landmark point in local coordinates $L_1^{Local} = [43{,}67 - 22.04]$, the correspondence likelihood estimation $\pi_j$ (Algorithm 9 Line 4) is equal to 0, because there is no previous measurement that resembles the current landmark. Therefore the first landmark is identified as a new landmark (Algorithm 9 Line 5).

Line 7 in Algorithm 9 is activated and the new landmark is created for the first particle $p^1$ (where $k = 1$).

$$p^1 = \left[\begin{bmatrix} 0.0047 & 1 \\ 0.0222 & 0 \\ -0.0047 & 0 \end{bmatrix} \begin{bmatrix} 43.6723 & 0.0001 & 0.0019 & 0 & 1 \\ -22.0473 & 0.0019 & 0.0506 & 0 & 0 \\ 0.9995 & 0 & 0 & 0 & 0 \end{bmatrix}\right]_1$$

Because there is no previous observed feature in Line 8, and there is no condition to satisfy the criteria as in Lines 11,13 and 15 in else condition in Algorithm 9, these are passed in the algorithm for this case. The covariance estimation for $L_1^{Local}$ is calculated according to Algorithm 7 between Lines 20-25 in Section 3.1.4. After the first landmark detection, $N$ is equal to 1, and for iteration, $m$ is equal to 1 and $k$ is equal to 1.

In a similar way, the second landmark $L_2^{Local} = [14.04 \ -1.13]$ is processed between Lines 3-5. The correspondence likelihood estimation $\pi_j$ (Algorithm 9 Line 4) is equal to 0 again, because the $L_2^{Local}$ is not the same landmark with $L_1^{Local}$ and the correspondence likelihood estimation with respect to other observed landmarks will give 0 value. If it is observed before, the likelihood estimation will give a value close to 1. With the same process as in $L_1^{Local}$, $L_2^{Local}$ is added to $p^1$ as a new landmark.

$$p^1 = \left[\begin{bmatrix} 0.0047 & 2 \\ 0.0222 & 0 \\ -0.0047 & 0 \end{bmatrix} \begin{bmatrix} 43.6723 & 0.0001 & 0.0019 & 0 & 1 \\ -22.0473 & 0.0019 & 0.0506 & 0 & 0 \\ 0.9995 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 14.0444 & 0.0001 & 0.0002 & 0 & 1 \\ 1.1354 & 0.0002 & 0.0007 & 0 & 0 \\ 1.0009 & 0 & 0 & 0 & 0 \end{bmatrix}\right]_1$$

After the second landmark detection, $N$ is equal to 2, and for iteration, $m$ is equal to 2 and $k$ is equal to 1. Finally, the third landmark $L_3^{Local} = [29.21 \ 22.72]$ is processed between Lines 3-5 and detected a new landmark according to $\pi_j = 0$. The final structure of $p^1$ at time $t = 0$ is then,

$$p^1 = \left[\begin{bmatrix} 0.0047 & 3 \\ 0.0222 & 0 \\ -0.0047 & 0 \end{bmatrix} \begin{bmatrix} 43.6723 & 0.0001 & 0.0019 & 0 & 1 \\ -22.0473 & 0.0019 & 0.0506 & 0 & 0 \\ 0.9995 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 14.0444 & 0.0001 & 0.0002 & 0 & 1 \\ 1.1354 & 0.0002 & 0.0007 & 0 & 0 \\ 1.0009 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 29.2167 & 0.0001 & -0.0018 & 0 & 1 \\ 22.7258 & -0.0018 & 0.0524 & 0 & 0 \\ 0.9997 & 0 & 0 & 0 & 0 \end{bmatrix}\right]_1$$

After the third landmark detection, $N$ is equal to 3, and for iteration, $m$ is equal to 3 and $k$ is equal to 1. This process is continued until all particle $k = 1..10$ is processed. After the all particles are calculated according to Algorithm 9, we will

obtain a structure like in Table 4.1. The first column consists of the robot's estimated initial position states. The other columns include the landmarks' estimated positions and covariance matrices with landmark counter. Each particle estimates multiple landmark points according to given Algorithm 9 at the initial time.

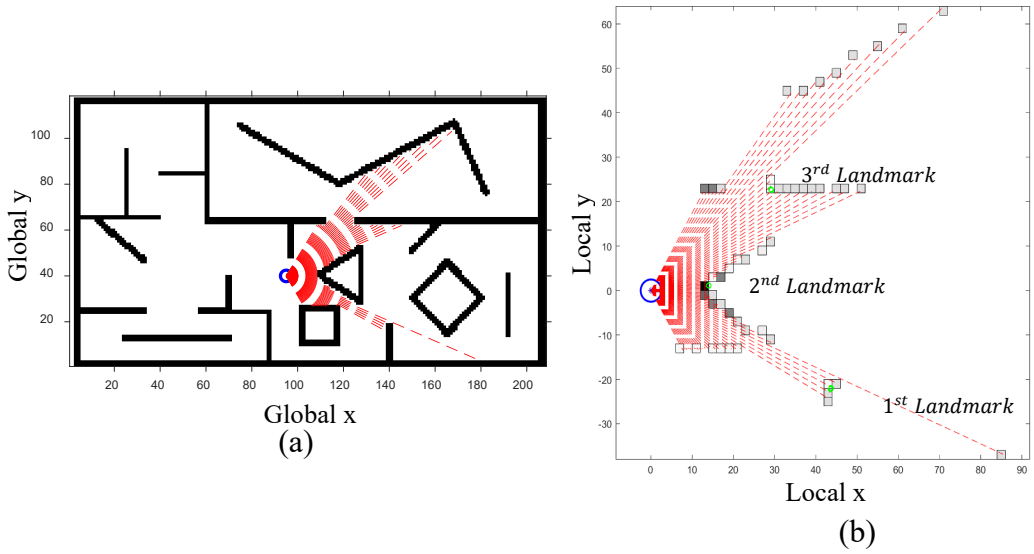



Figure 4.2. (a) Robot in global map with LIDAR raycasting. (b) Robot local map with landmark indicated with green dots.

Table 4.1 Particle filter propagation with the measurement at initial time $t = 0$.

| | *Robot States* | $1^{st}$*Landmark* | $2^{nd}$*Landmark* | $3^{rd}$*Landmark* |
|---|---|---|---|---|
| $p^1$ | $\begin{bmatrix}\begin{bmatrix}0.0047 & 3\\ 0.0222 & 0\\ -0.0047 & 0\end{bmatrix}\end{bmatrix}$ | $\begin{bmatrix}43.6723 & 0.0001 & 0.0019 & 0 & 1\\ -22.0473 & 0.0019 & 0.0506 & 0 & 0\\ 0.9995 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}14.0444 & 0.0001 & 0.0002 & 0 & 1\\ 1.1354 & 0.0002 & 0.0007 & 0 & 0\\ 1.0009 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}29.2167 & 0.0001 & -0.0018 & 0 & 1\\ 22.7258 & -0.0018 & 0.0524 & 0 & 0\\ 0.9997 & 0 & 0 & 0 & 0\end{bmatrix}_1$ |
| $p^2$ | $\begin{bmatrix}\begin{bmatrix}0.0289 & 3\\ -0.0390 & 0\\ 0.2567 & 0\end{bmatrix}\end{bmatrix}$ | $\begin{bmatrix}43.7968 & 0.0001 & 0.0019 & 0 & 1\\ -21.9091 & 0.0019 & 0.0497 & 0 & 0\\ 0.9995 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}14.0358 & 0.0001 & 0.0002 & 0 & 1\\ 1.0988 & 0.0002 & 0.0007 & 0 & 0\\ 1.0009 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}29.1371 & 0.0001 & -0.0018 & 0 & 1\\ 22.7976 & -0.0018 & 0.0530 & 0 & 0\\ 0.9997 & 0 & 0 & 0 & 0\end{bmatrix}_2$ |
| $p^3$ | $\begin{bmatrix}\begin{bmatrix}0.0427 & 3\\ -0.0267 & 0\\ 0.0484 & 0\end{bmatrix}\end{bmatrix}$ | $\begin{bmatrix}43.7308 & 0.0001 & 0.0019 & 0 & 1\\ -22.0557 & 0.0019 & 0.0504 & 0 & 0\\ 0.9995 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}14.0439 & 0.0001 & 0.0002 & 0 & 1\\ 1.1756 & 0.0002 & 0.0007 & 0 & 0\\ 1.0009 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}29.2338 & 0.0001 & -0.0018 & 0 & 1\\ 22.7039 & -0.0018 & 0.0525 & 0 & 0\\ 0.9997 & 0 & 0 & 0 & 0\end{bmatrix}_3$ |
| $p^4$ | $\begin{bmatrix}\begin{bmatrix}0.0438 & 3\\ 0.0151 & 0\\ -0.1734 & 0\end{bmatrix}\end{bmatrix}$ | $\begin{bmatrix}43.6463 & 0.0001 & 0.0020 & 0 & 1\\ -22.1829 & 0.0020 & 0.0512 & 0 & 0\\ 0.9995 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}13.9964 & 0.0001 & 0.0002 & 0 & 1\\ 1.0669 & 0.0002 & 0.0008 & 0 & 0\\ 1.0009 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}29.3227 & 0.0001 & -0.0018 & 0 & 1\\ 22.6325 & -0.0018 & 0.0520 & 0 & 0\\ 0.9997 & 0 & 0 & 0 & 0\end{bmatrix}_4$ |
| $p^5$ | $\begin{bmatrix}\begin{bmatrix}0.0166 & 3\\ -0.0440 & 0\\ -0.1643 & 0\end{bmatrix}\end{bmatrix}$ | $\begin{bmatrix}43.6226 & 0.0001 & 0.0020 & 0 & 1\\ -22.2351 & 0.0020 & 0.0511 & 0 & 0\\ 0.9995 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}14.0132 & 0.0001 & 0.0002 & 0 & 1\\ 1.0748 & 0.0002 & 0.0008 & 0 & 0\\ 1.0009 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}29.2918 & 0.0001 & -0.0018 & 0 & 1\\ 22.5781 & -0.0018 & 0.0521 & 0 & 0\\ 0.9997 & 0 & 0 & 0 & 0\end{bmatrix}_5$ |
| $p^6$ | $\begin{bmatrix}\begin{bmatrix}0.0446 & 3\\ 0.0351 & 0\\ 0.2785 & 0\end{bmatrix}\end{bmatrix}$ | $\begin{bmatrix}43.8208 & 0.0001 & 0.0019 & 0 & 1\\ -21.8184 & 0.0019 & 0.0497 & 0 & 0\\ 0.9995 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}13.9904 & 0.0001 & 0.0002 & 0 & 1\\ 1.1928 & 0.0002 & 0.0007 & 0 & 0\\ 1.0009 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}29.1441 & 0.0001 & -0.0018 & 0 & 1\\ 22.8827 & -0.0018 & 0.0531 & 0 & 0\\ 0.9997 & 0 & 0 & 0 & 0\end{bmatrix}_6$ |
| $p^7$ | $\begin{bmatrix}\begin{bmatrix}-0.0401 & 3\\ 0.0077 & 0\\ 0.1898 & 0\end{bmatrix}\end{bmatrix}$ | $\begin{bmatrix}43.7021 & 0.0001 & 0.0019 & 0 & 1\\ -21.9135 & 0.0019 & 0.0499 & 0 & 0\\ 0.9995 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}14.03363 & 0.0001 & 0.0002 & 0 & 1\\ 1.0817 & 0.0002 & 0.0007 & 0 & 0\\ 1.0009 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}29.0947 & 0.0001 & -0.0018 & 0 & 1\\ 22.8103 & -0.0018 & 0.0529 & 0 & 0\\ 0.9997 & 0 & 0 & 0 & 0\end{bmatrix}_7$ |
| $p^8$ | $\begin{bmatrix}\begin{bmatrix}-0.0335 & 3\\ 0.0257 & 0\\ 0.1510 & 0\end{bmatrix}\end{bmatrix}$ | $\begin{bmatrix}43.6940 & 0.0001 & 0.0019 & 0 & 1\\ -21.9251 & 0.0019 & 0.0501 & 0 & 0\\ 0.9995 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}14.0834 & 0.0001 & 0.0002 & 0 & 1\\ 1.0383 & 0.0002 & 0.0007 & 0 & 0\\ 1.0009 & 0 & 0 & 0 & 0\end{bmatrix}$ | $\begin{bmatrix}29.1168 & 0.0001 & -0.0018 & 0 & 1\\ 22.8086 & -0.0018 & 0.0528 & 0 & 0\\ 0.9997 & 0 & 0 & 0 & 0\end{bmatrix}_8$ |

Table 4.1 (continued)

| | |
|---|---|
| $p^9$ | $\left[\begin{bmatrix} 0.0334 & 3 \\ 0.0244 & 0 \\ -0.2070 & 0 \end{bmatrix} \begin{bmatrix} 43.6228 & 0.0001 & 0.0020 & 0 & 1 \\ -22.1992 & 0.0020 & 0.0513 & 0 & 0 \\ 0.9995 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 13.9960 & 0.0001 & 0.0002 & 0 & 1 \\ 1.1289 & 0.0002 & 0.0008 & 0 & 0 \\ 1.0009 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 29.3255 & 0.0001 & -0.0018 & 0 & 1 \\ 22.6247 & -0.0018 & 0.0520 & 0 & 0 \\ 0.9997 & 0 & 0 & 0 & 0 \end{bmatrix}\right]_9$ |
| $p^{10}$ | $\left[\begin{bmatrix} -0.0034 & 3 \\ 0.0210 & 0 \\ 0.0803 & 0 \end{bmatrix} \begin{bmatrix} 43.6970 & 0.0001 & 0.0019 & 0 & 1 \\ -21.9836 & 0.0019 & 0.0503 & 0 & 0 \\ 0.9995 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 14.0697 & 0.0001 & 0.0002 & 0 & 1 \\ 1.0555 & 0.0002 & 0.0007 & 0 & 0 \\ 1.0009 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 29.1750 & 0.0001 & -0.0018 & 0 & 1 \\ 22.7679 & -0.0018 & 0.0526 & 0 & 0 \\ 0.9997 & 0 & 0 & 0 & 0 \end{bmatrix}\right]_{10}$ |

## 4.3 Landmark Detection

Raw sensor measurements should be converted into meaningful features to be used in the FastSLAM as landmarks. In the case of LIDAR, raw sensor measurements are point cloud data obtained from raycasting. To convert raw sensor measurements into features we apply a feature extraction algorithm.

In the literature, common techniques to extract features are based on identifying lines, corners, or specific objects with patterns or appearance. In our study, we select sharp corner points as features to identify landmarks. The reason is that in SAR operations over disaster regions such as earthquake rubbles, we encounter highly indented surfaces and structures. There are no regular surfaces or lines to be utilized for landmark positions in such a disaster area. Also, we could have used the occupancy grid map points as landmark points during the search, however, because the computational burden of observing multiple occupancy grid sites in searching increases the size of particle filter structure (Equation 4-1) larger than observing the corner points, we prefer using the corner points as landmark points which sustains us a distilled version of landmark extraction. Therefore, we convert raw sensor measurement to corner features with a feature extraction algorithm and then, the found feature locations are fed into $f(z_t)$ given in Equation 3.1 in Chapter 3 (given below) to represent the feature-based sensor model.

$$\begin{bmatrix} r_t^i \\ \phi_t^i \\ s_t^i \end{bmatrix} = \begin{bmatrix} \sqrt{\left(m_{i,x} - x\right)^2 + \left(m_{i,y} - y\right)^2} \\ atan2\left(m_{i,y} - y, m_{i,x} - x\right) - \theta \\ s_i \end{bmatrix} + \begin{bmatrix} \epsilon_{\sigma_r^2} \\ \epsilon_{\sigma_\phi^2} \\ \epsilon_{\sigma_s^2} \end{bmatrix}$$

The feature extraction algorithm detects corner points by using the REE algorithm [46]. REE algorithm name comes from the usage of one rectangle and two ellipses on a point cloud. REE slides on the boundary of the point cloud shape and detects the corner points on the shape. The rectangle $R$ embeds two ellipses $E_1$ and $E_2$ such that $R \supset E_1 \supset E_2$ as in Figure 4.3 (a). In Figure 4.3 (b), the snapshot of an REE structure on an arbitrary point cloud boundary of a shape is depicted.

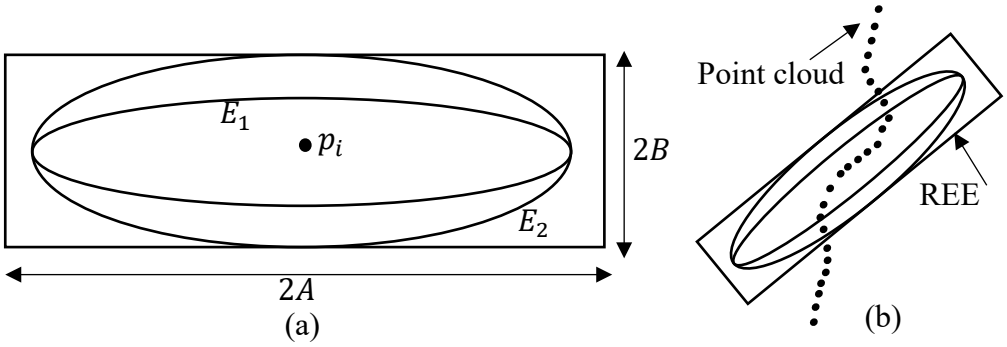



Figure 4.3. (a) Structure of REE algorithm. (b) Snapshot of the REE shape on an irregular boundary.

The relation between the rectangle and ellipses is given in Equation 4.2.

$$R = 2A \times 2B$$
$$E_1 = \pi \times \frac{3A}{4} \times B$$
$$E_2 = \pi \times \frac{3A}{4} \times \frac{B}{2} \qquad \text{4-2}$$
$$\theta = slope\ of\ curve\ at\ point\ p_i$$

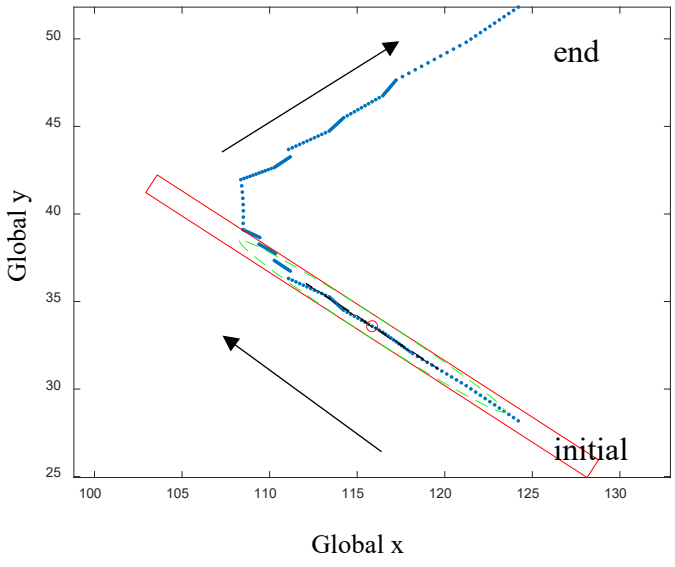
The semi-minor and semi-major axis of the Ellipse $E_1$ is $\frac{3A}{4}$ and $B$. For the ellipse $E_2$, these values are $\frac{3A}{4}$ and $\frac{B}{2}$ respectively. The slope $\theta$ at $p_i$ is found by calculating the fitted line of $n$ points on both sides of $pi$. Let $n_{R_i}, n_{E_{1,i}}, n_{E_{2,i}}$ represent the number of curve points within rectangle and ellipses. For each curve point, we run the following algorithm to detect the corner point.

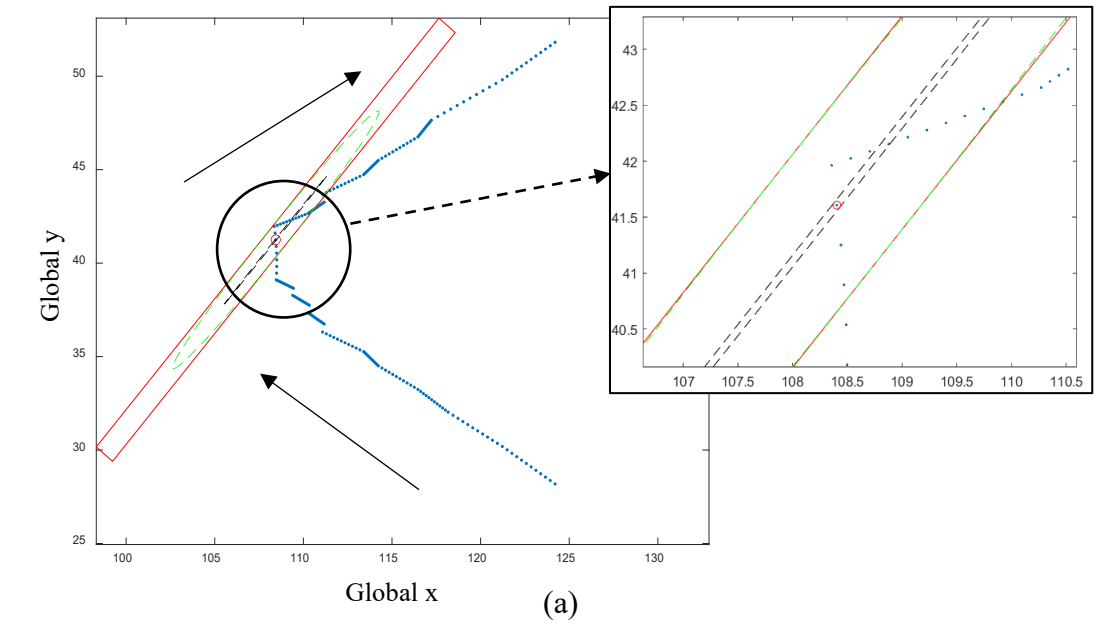
**Algorithm 10:** Corner Feature Detection$(p_i)$

1: $foreach\ counter\ point\ p_i$
2: $\quad Count\ n_{R_i}, n_{E_{1,i}}, n_{E_{2,i}}$
3: $end$
4: $Calculate\ G = \{p_i : n_{R_i} = n_{E_{1,i}}\}$
5: $Create\ groups\ of\ point\ G$
6: $foreach\ group\ G_k$
7: $\quad Corner = \min_{n_{E_{2,i}}} \{G_{k,i} : n_{E_{2,i}} < \eta\}$
8: $end$

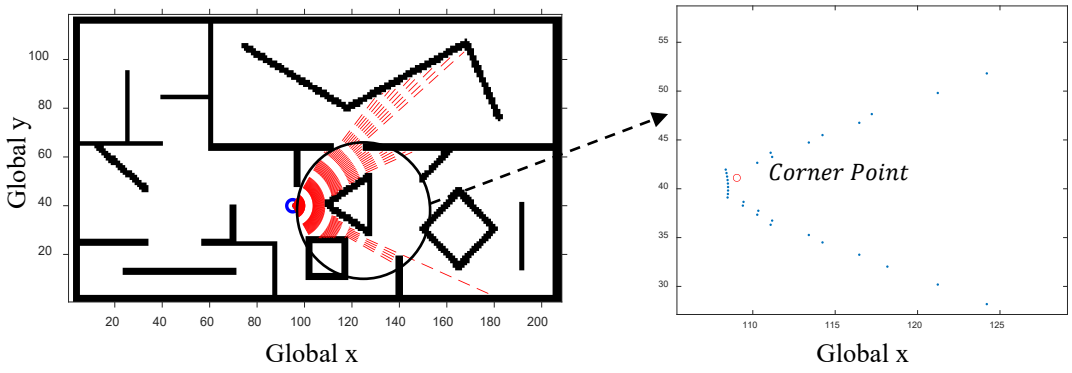
In Algorithm 10, firstly we count the $n_{R_i}, n_{E_{1,i}}, n_{E_{2,i}}$ for each point in the point cloud set (Lines 1-3). Then, we collect the points that satisfy $n_{R_i} = n_{E_{1,i}}$ and add them to set $G$ (Lines 4-5). For each found group $G_k$, they consist of candidate corner points. To obtain the actual corner point, a threshold value $\eta$ is selected. Number of points $n_{E_{2,i}}$ in $G_k$ which is lower than $\eta$ represents the corner point with index $i$. The details can be found in reference [46].

An example scenario in our methodology is given in Figure 4.4. In the previous section, landmark points on the local map are shown on particles of the FastSLAM.






(a)




(b)

Figure 4.4. (a) REE algorithm process on a point cloud data obtained from LIDAR scan. (b) In the global map, the corner point detection of the triangular shape after we apply REE algorithm.

We select the landmark point on the tip of the triangular shape to show the progress. As shown in Figure 4.4 (a) the REE shape slides on each boundary point of the triangular shape from the initial point to the endpoint in the direction of the arrow. The rectangle is shown with red, inner ellipses $E_1$ and $E_2$ is shown with green and black dash lines respectively. Beginning from the initial point, for each

point Line 2 is applied and we count $n_{R_i}, n_{E_{1,i}}, n_{E_{2,i}}$ at each point. After that at line 4, we calculate the points $n_{R_i} = n_{E_{1,i}}$ and add them to set $G$. As in Figure 4.4 (a), within rectangle and ellipse 1 $n_{R_i} = n_{E_{1,i}} = 13$. Finally between lines 6-8, the point which belongs to minimum $n_{E_{2,i}}$ is selected as the corner point. In Figure 4.4 (b), as a result of REE process, the corner point is detected at the coordinates [109 41.2] on the global map. After this point is obtained as a landmark, we apply Equation 3.1 to represent this point in the feature-based measurement model.

## 4.4 Frontier Target Selection and Vital Signal Search

In the SAR exploration strategy, the idea is to build a map of the unknown territory and unveil the search world as much as possible in order to detect vital signals. To accomplish this task, we need to drive the robot in a certain direction on the map. However, the main question here is: *how do we determine this direction?* The direction can be determined towards the boundary between explored and unexplored world. This boundary is where new information about the environment can begin to be collected. Thus, we select our target points on the boundary of the occupancy map where the explored region is separated from the unexplored part of the world. In order to detect such target points on the boundary, we use the frontier-based exploration strategy [47].

To detect the frontier cells on the local occupancy map during the search, we need to detect the occupancy of each cell. The initial prior probability of each cell before discovery is uniformly distributed and equal to 0.5. This value represents maximum entropy and minimum knowledge about the occupancy of the cell. The occupied cell probability is $p(m_i) > 0.5$ where $m_i$ represents the cell $i$ in occupancy map $M$. With the increasing value of $p(m_i)$ greater than 0.5, we have more confidence in the cell being occupied. If a cell is an open cell or empty one, it is represented with $p(m_i) < 0.5$. In each measurement, if the LIDAR raycasting does not detect an obstacle in the cell, $p(m_i)$ gets a lower value than 0.5 and finally

takes 0 value which indicates that the cell is empty. We separate the frontier cell by selecting open cells adjacent to the unknown cell. To understand the process better, we will now analyze Figure 4.5.

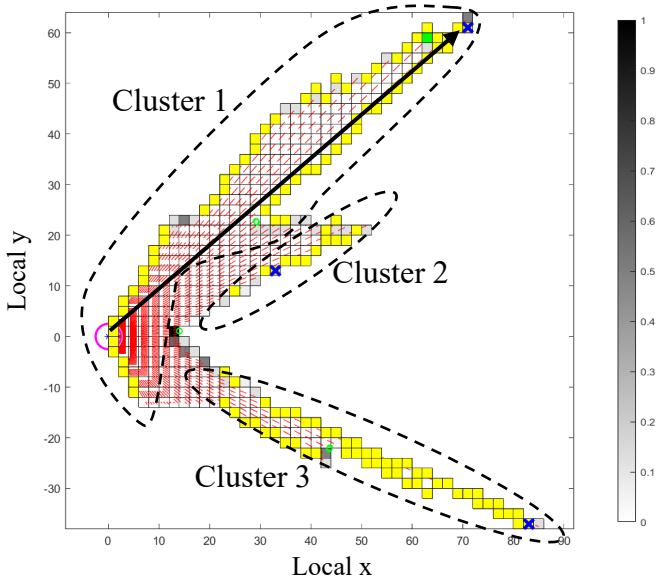



Figure 4.5. Frontier-based search method on a occupancy grid map.

The magenta circle shows us the location of the robot. The occupied cells are represented with gray ($0.5 < p(m_i) < 1$) or black color ($p(m_i) = 1$) cells according to their occupancy probability. The yellow cells are the frontier cells of the occupancy map at that instance within the field of the LIDAR sensor. To detect target points within the frontier cells, we create clusters by separating them according to the location and distance of the occupied cell. In other words, to create a cluster we select the nearest neighbor empty cells and add them to the cluster. If the distance between two empty cells is greater than $\eta_{cluster}$, we announce the created cell group as a separate cluster. The lower threshold value causes finer clusters, however, if we select a larger threshold, this causes gross-size clusters. It must be carefully selected. After we create clusters with the selected threshold value, the target points are detected by measuring the largest distance cell to robot location among each cluster to obtain maximum information. Three blue cross

locations in Figure 4.5 indicate these target locations. Because we need to select one of them to continue searching, we select the one that has the largest distance to the robot location. Under these conditions, we select the target point at the location of [69 63] indicated with the arrow on the occupancy map. The distances of other target points are presented in Table 4.2.

Table 4.2 Target points distance values

| Target points | *Distance to Robot location* |
|---|---|
| Target Point 1 (Cluster 1) | 93.43 |
| Target Point 2 (Cluster 2) | 37.34 |
| Target Point 3 (Cluster 3) | 90.07 |

What if we have a vital signal near our search area and we detect this vital signal at a distance from the robot location, how will this affect our target selection? To answer this question, we need to mention how we represent the vital signal on the occupancy map. In our research, we assume that a SAR robot has a vital signal detection circle with a determined radius and the vital signal of a victim has a certain radius of emission to be detected by the SAR robot. Figure 4.6 depicts the situation on the global map.

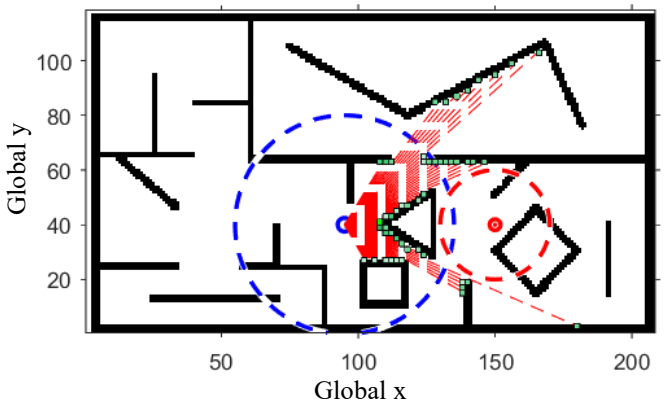



Figure 4.6. The blue circle represents the robot searching radius. The red circle represents the vital signal of the victim emission range.

The detection of the vital signal by the robot directly affects the target point selection criteria. If the vital signal is in the direct line of sight of the robot, the target point is selected as the location of the victim. However, if the robot's vital detection circle and the vital signal emission circle of the victim are intersected, the closer target point to the vital signal direction is selected as the target point. For example, if the intersection of the circles is given as in Figure 4.6, the target point will be selected as in Figure 4.7.

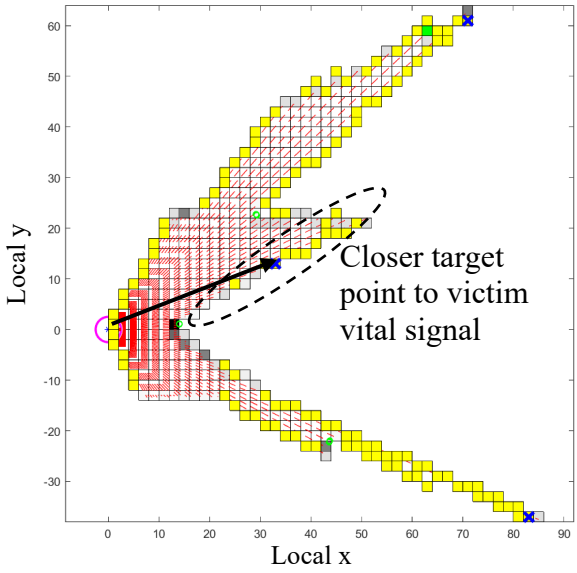



Figure 4.7. Target point selection in case of vital signal detection by the robot.

## 4.5 Exploration Path Generation with Explosive Percolation

The idea behind using percolation theory on SLAM is based on the applicability of percolation theory to irregular and highly complex environments. However, the application of percolation theory to SLAM needs prior occupational probability of each cell within the environment, and this probability value should be higher than the percolation probability $p_c$ in order to enable penetration of the selected region. In the SLAM approach, this situation is not possible, because we obtain the occupancy probability of each cell during the search. In other words, the

occupation probabilities of cells are determined during the process. Explosive percolation theory eliminates this problem for the two reasons that follow. Explosive percolation theory provides two major advantages in penetrating unknown highly irregular regions in a disaster area.

1) We do not need any prior occupancy value of cells within the search region. As we know that during the SLAM process, we determine the occupation probability of a cell. There will be no prior information about the occupation of a cell until observed by LIDAR. Therefore we can adopt this approach to the unknown territory through active search. Explosive percolation does not need any prior information about the occupational probability of a cell.
2) We can control the percolation threshold value to create a cluster among the voids of the occupancy grid map to acquire a path that connects two ends of the searching region. The threshold control sustains us on how fast or slow we connect two target points in a map. The details on how to control the threshold are given below.

The general description and algorithm of the explosive percolation are given in Section 3.2.2. In explosive percolation, the creation of large clusters is tried to be delayed at the emergence of percolation clusters by changing the selection rule of cells in a lattice (or nodes if it is a network). The delay of the process is selected in explosive percolation research to observe how smaller clusters connect to create a giant cluster abruptly. These selection rules can be summation or product rules according to selection. In our situation, we apply the procedure reversely compared to Algorithm 8, since we aim to attain the percolation cluster as fast as possible by creating larger clusters within voids of the occupancy grid map. For the selection rule, we utilize the summation rule, however, the product rule can also be selected.

Consider $C$ to be the cluster and $|C|$ is the size of it. Let's select different sizes of clusters as $C_a, C_b, C_c, C_d$ with their sizes $|C_a|, |C_b|, |C_c|, |C_d|$. Assume that $s_1$ connects the cluster $|C_a|, |C_b|$ and $s_2$ connects the clusters $|C_c|, |C_d|$. If $|C_a| +$

$|C_b| \geq |C_c| + |C_d|$ then we select $s_1$ to create a larger cluster, otherwise, we select $s_2$.

$$s_{selected} = \begin{cases} s_1, & |C_a| + |C_b| > |C_c| + |C_d| \\ s_2, & otherwise \end{cases} \quad 4\text{-}3$$

**Algorithm 11:** Explosive Percolation-Based Exploration in SLAM $(m_k, P_R, P_T)$

```
foundpath ← 0
while (foundpath = 0)
    Select random cells s1, s2 ϵ mk
    if s1, s2 are selected before
        Select new cells s1, s2 ϵ mk
    end if
    insert s1 into the map mk and compute |C1|
    insert s2 into the map mk and compute |C2|
    if |C1| ≥ |C2|
        Select s1
    else
        Select s2
    end if
    Determine the closest cluster to PR and PT
    if clusters are the same identity
        foundpath ← 1
    endif
endwhile
```

The algorithm of explosive percolation on the occupancy map is given in Algorithm 11. In the algorithm, $m_k$ represents the occupancy grid map at that instant, $P_R$ is the robot's location on the local map, $P_T$ is the target location on the local map. When the same cluster identity is caught around the robot and target

location, we can say that we found a percolation cluster that connects these two points on the occupancy grid map.

Let's assume that we select our target point as indicated in Figure 4.5. If we run the explosive percolation algorithm on this occupancy grid map of the robot, we will observe the progress as in Figure 4.8.

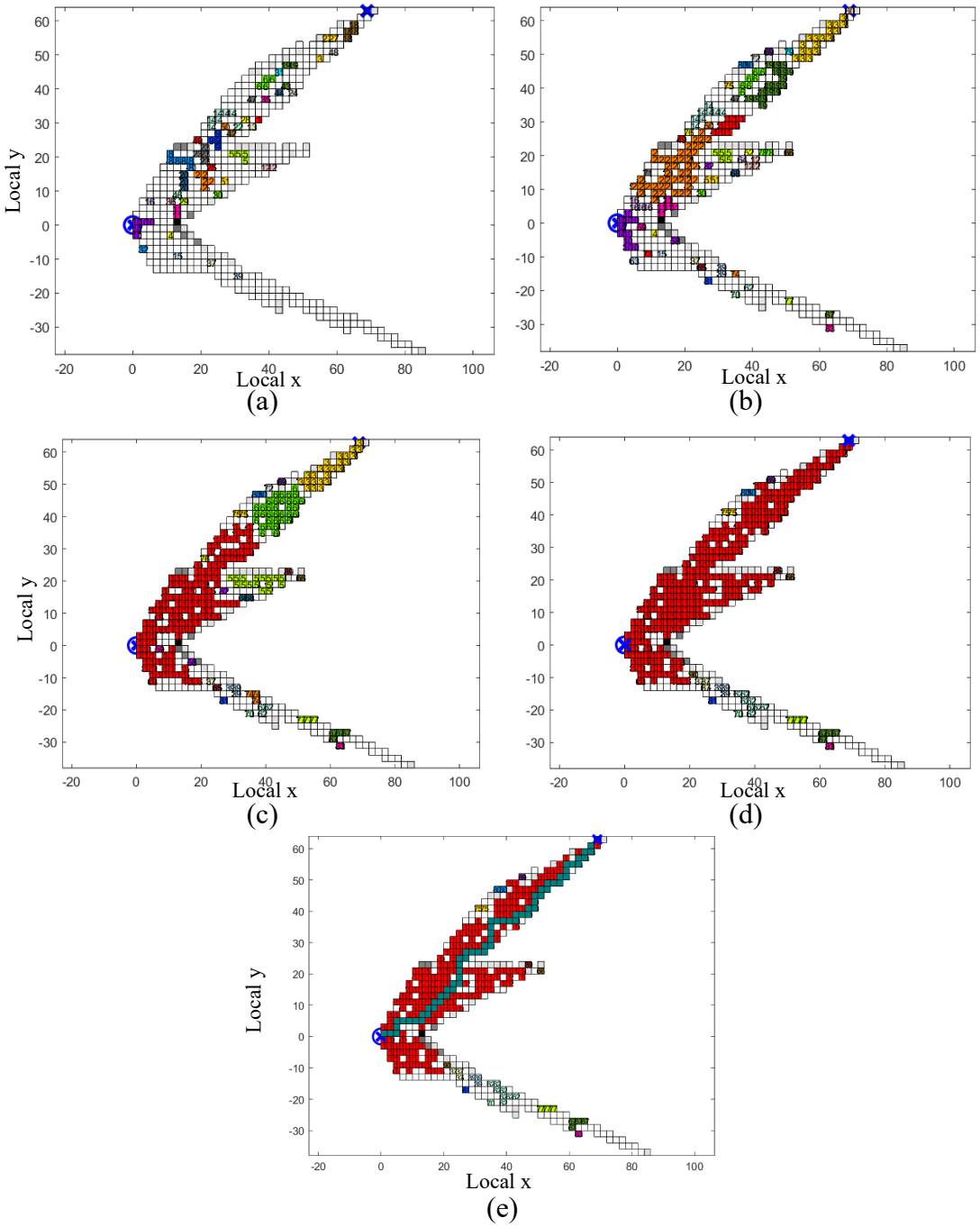


Figure 4.8. (a) Explosive percolation initial cluster emergence. (b) Clusters begin to merge with respect to the neighbor cell cluster index. (c) Cluster approximate to the emergence of percolation cluster. (d) Percolation cluster connecting the robot and target locations is found. Here the red cluster is our percolation cluster. (e) The percolation path is found in the percolation cluster.

Let's compare Algorithm 11 with Figure 4.8 step by step. Initially, all map is free from clusters. Therefore the initial run of Algorithm 11 with a random selection of cells only creates single-cell clusters. By iterating the algorithms, we will obtain small-size clusters as in Figure 4.8 (a). As we notice, we obtain larger clusters having 2 or more cells in this stage. Lines between 3-6 sustain the selection of 2 new random cells that are not selected before. After the selection of random cells, firstly at line 7, we insert $s_1$ cell into to map and compute the $C_1$ cluster size $|C_1|$ where $s_1$ belongs to. We repeat a similar process for $s_2$ at line 8. Then, between lines 9-13, we compare the cluster sizes and select the greater 1. This selection causes the creation of larger clusters as in Figure 4.8 (b) and (c). Finally, between lines 14-17, we check the condition of whether the same cluster cells connect the target points or not. If they are connected, we make $foundpath$ value 1 and end the algorithm. Figure 4.8 (d) indicates the final case in which the red cluster connects two target points.

We observe the emergence of smaller cells at the beginning of the process. By selecting the voids in the occupancy map, we percolate further and create larger clusters. In Figure 4.8 (b) the clusters are merged within the neighbor cell according to the summation rule. The cells are numbered and colored if they share the same cluster. The merging is sustained with the 4-neighborhood of a cell. When we detect a percolation cluster that connects the robot and target locations, we stop the process.

At this stage, we have a candidate cluster to find the percolation path. Without any cluster, it is expected that the voids within the map include all paths that connect target points. The percolation cluster is the certain part of the map that connects

these two points. Therefore, the cluster should be valid and represent the current observed map by LIDAR to find the percolation path. This validity of the percolation cluster can be checked by measuring the coverage of the percolation cluster. The best way to measure coverage is the calculation of the dimension of the cluster and comparing it with the map dimension. Because the map and cluster do not have a regular shape, the fractional order dimension method can be utilized to overcome this problem. The validity of the percolation cluster can be shown with its convergence of fractional dimensions to occupancy grid lattice fractional dimension. In Figure 4.9, we can deduce that the cluster fractional dimension converges the dimension of the lattice as expected.

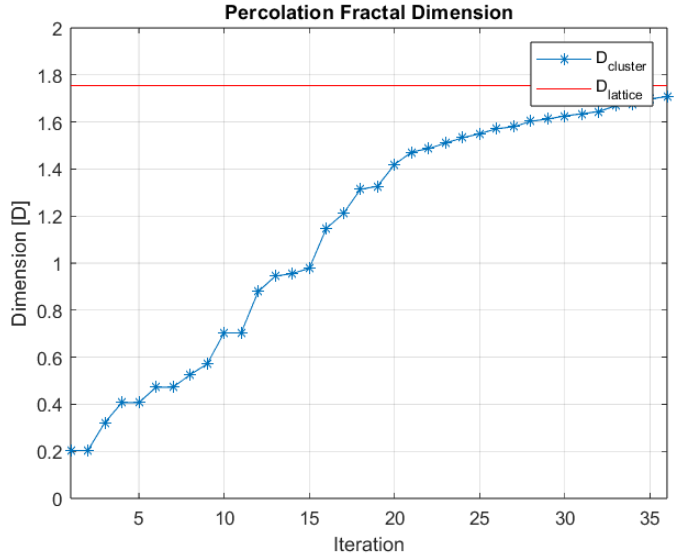



Figure 4.9. Fractional dimension convergence of the largest cluster to lattice dimension during explosive percolation process.

The fractal dimension is calculated according to the radius of the gyration $R_s$ of the largest cluster until the emergence of the percolation cluster. It is assumed that the size of a cluster is proportional to the fractal dimension with $N \propto R_s^D$ where $N$ is the size of a cluster and $D$ is the fractal dimension [48]. Therefore the dimension $D$ will be equal to $D = \frac{\log(N)}{\log(R_s)}$.

Percolation path connects the robot and target point location on map. After finding the percolation cluster, we can now search for the percolation path. The important point is that instead of searching the percolation path on the occupancy grid map, we search it in the percolation cluster which has fewer void cells. To obtain the

$$J_k = \min_k \sqrt{(x_k - x_T)^2 + ((y_k - y_T)^2)} \qquad k = 1,2,3,4 \tag{4-4}$$

optimum path, we should utilize a cost function. The cost function is obtained by selecting a cell that has a minimum Euler distance to the target location. We begin the selection within the 4-neighborhood of robot location. Then, we select the cell $k$ that has a minimum $J_k$ for the next iteration until we reach the target location. The percolation path at the end of the process is shown in Figure 4.8 (e). The green cells indicate the percolation path to reach the target location from the robot position. The crucial point here is that the explosive percolation path is found within the voids of the occupancy map. Therefore the obstacles and other objects that are represented with occupied cells in the occupancy map are naturally avoided by our approach and we can obtain a safe and obstacle-free passage toward the target location.

## 4.6 Path Pruning and Smoothing Algorithms

The path found in explosive percolation is comprised of grid cells that have too many indentations and sharp turns, making the path hard to follow for the SAR robot. Because of such inconvenience and hard-to-follow path patterns, we should find a much simpler and easy-to-follow path scheme. To achieve this path pruning and smoothing algorithms have been utilized.

Firstly, the pruning algorithm runs on the percolation path. If there is no blockage between connected ordered cells in the percolation path, those points are symbolized with a line with initial and end points only. Otherwise, the inflection point is found if there exists any occupied region that blocks the connected cells. In Figure 4.10, this situation is shown in an example percolation map. The star points

symbolize inflection points. If the inflection point is not inserted, the line connects the successive path cells intersects with the gray occupied cells. This will cause the situation in which the robot encounters an obstacle during the path following. We notice that connected lines do not intersect with occupied cells shown by gray.

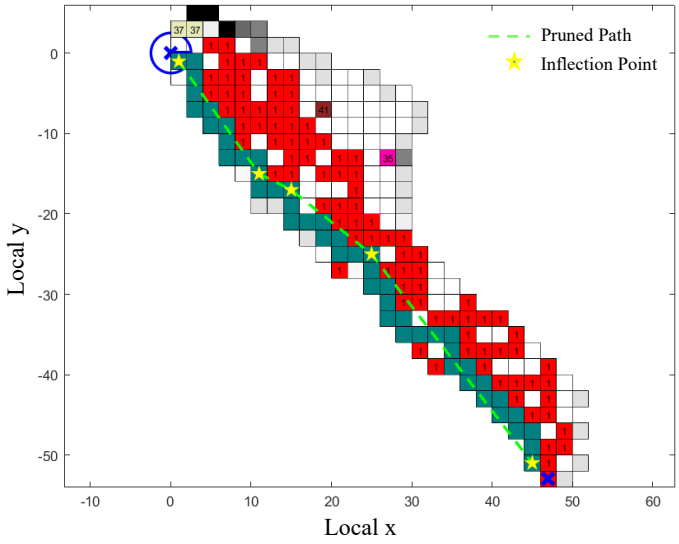



Figure 4.10. Path pruning strategy. Points with yellow stars symbolize inflection points.

Secondly, inflection points found in the pruning stage should be handled to have smooth turns for the robot to make the turn-in easy and with less effort. The Bezier Curve algorithm has been used to smooth the inflection points [49]. Generally, $n$ degree Bezier curve is defined as:

$$P_{[t_0,t_1]}(t) = \sum_{i=0}^{n} B_i^n(t) P_i, \quad t \in [0,1]. \tag{4-5}$$

Where $t$ is the positional parameter and $P_i$ is control points with $P_{t_o} = P_0$ and $P(t_1) = P_n$ and $B_i^n(t)$ is Bernstein polynomial given by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{m-i}, \quad i = 0,1,2,\dots,n \tag{4-6}$$

To apply the Bezier curve method to a pruning path, we extract each corner point. Then as in Figure 4.11, we apply the Bezier curve within $L$ distance of an inflection point $P_m$ on the path. We select an increment $d$ value to divide $L$ distance equally on each side of the inflection point. Then found control points $\{P_{m-i}, \dots, P_{m-1}, P_m, P_{m+1}, \dots, P_{m+i}\}$ are processed by utilizing Equation 4.5.

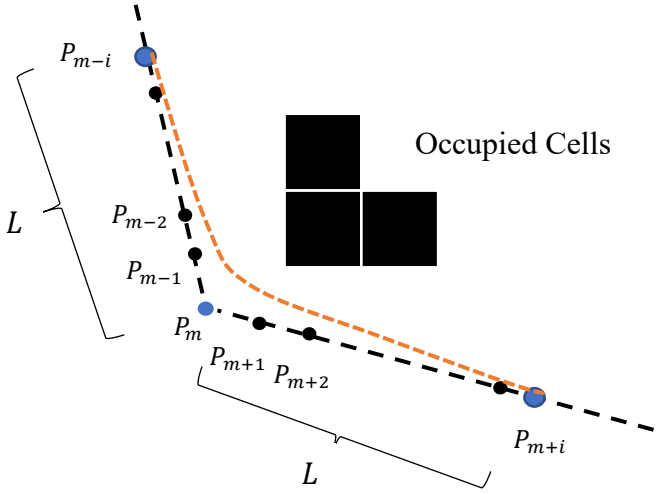



Figure 4.11. Smoothing of a corner point with Bezier Curve.

If we apply the Bezier curve method to inflection points in Figure 4.10, we obtain the following smooth path to be followed by the robot. In Figure 4.10, the dash green line indicates the pruned percolation path and yellow stars are the inflection points. By applying Bezier curve method, we obtain the dash dot cyan line as indicated in zoomed figure in Figure 4.10. As we noticed the inflection points are smoothed with the proposed method. In this way, the robot can make its manuever without deviation from following path at inflection points.

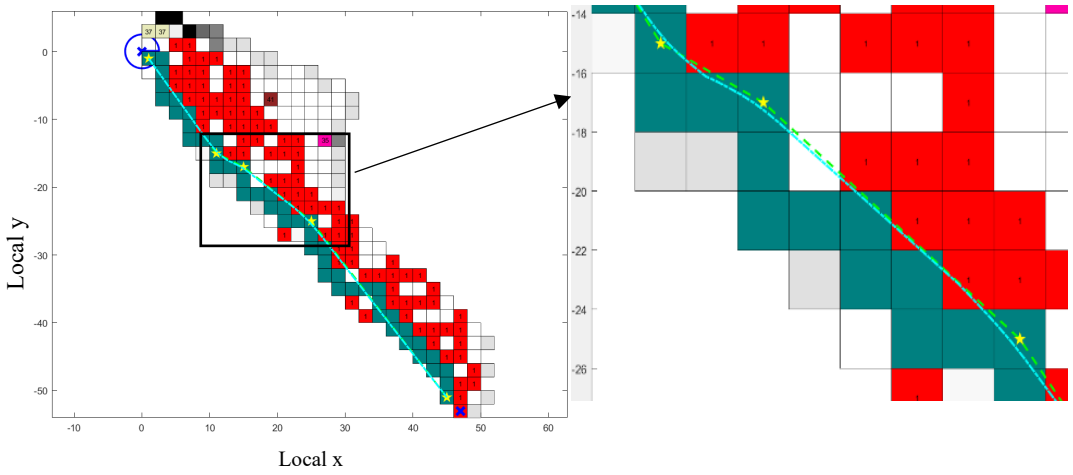



Figure 4.12. Smooth path creation with Bezier Curve.

Let's apply the pruning and smoothing strategy to Figure 4.8 (e). Because there is no detected intersection with the occupied cell in the pruning method as in Figure 4.10, we obtain a direct line that connects the robot location to the target location on the percolation path.

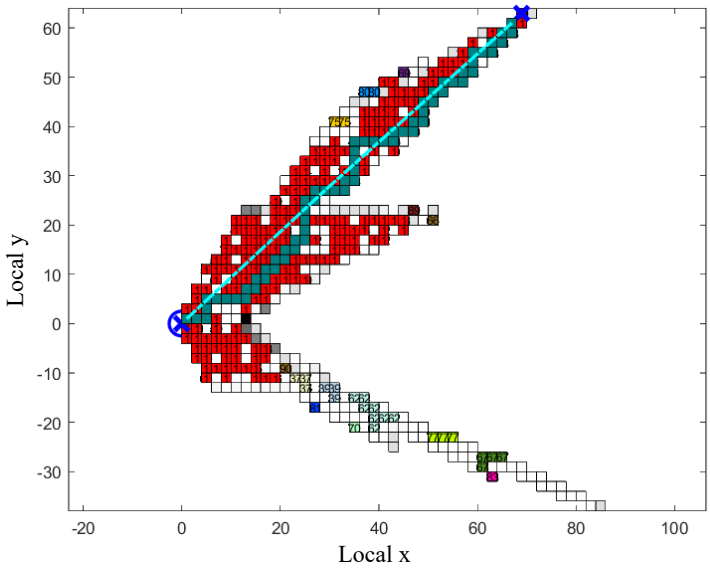



Figure 4.13. Pruning and smoothing strategy on example scenario in Figure 4.8 (e). Because there is no inflection point, the target and robot location are connected with a line on the percolation path.

## 4.7 Path Following Control Algorithm

After path generation and smoothing, the remaining question is how does the SAR robot follow the path successfully? In our strategy, we construct our path control algorithm based on Serret-Frenet formulas [50]. In this method, the robot follows a virtual target which is the orthogonal projection of the robot image on the found path.

The process is detailed in Figure 4.14. The path $P$ is defined with $\kappa(s)$ where $s$ is the curvilinear distance from the beginning of the path and $\kappa(s)$ is the curvature of the path. The desired orientation on the path is represented with $\theta_r = \pm\kappa(s)s$ (Positive for the counter-clockwise direction of Serret-Frenet frame).

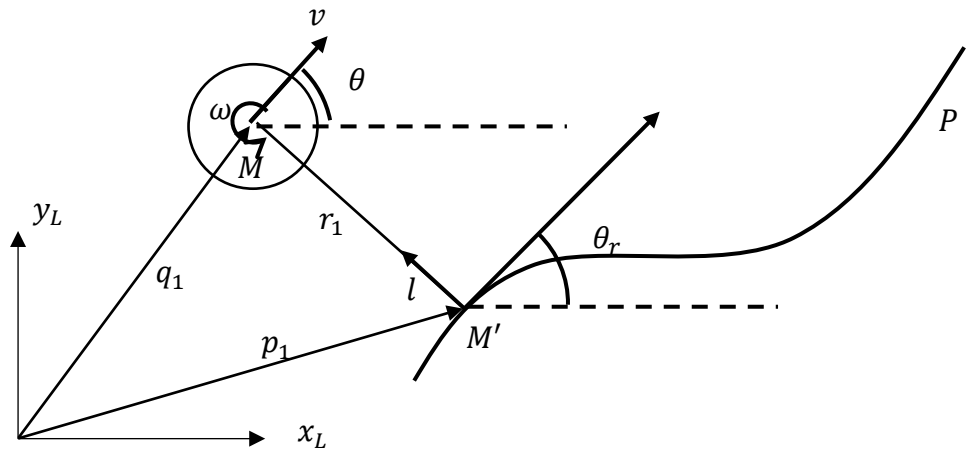



Figure 4.14. The Serret-Frenet frame with the orthogonal projection of the robot $M$ on the path $P$

The point $M'$ is the orthogonal projection of the robot $M$ on the path. $l$ is the orthogonal distance between $M$ and $M'$. The relationship between the translational distances $q_1, r_1$ and $p_1$ can be calculated with the following transformation:

$$\boldsymbol{q_1} = \boldsymbol{R_{\theta_r} r_1} + \boldsymbol{p_1} \tag{4-7}$$

$$\boldsymbol{R_{\theta_r}} = \begin{bmatrix} \cos(\theta_r) & -\sin(\theta_r) & 0 \\ \sin(\theta_r) & \cos(\theta_r) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-8}$$

Differentiating Equation 4-7, we obtain the following relation,

$$\dot{\boldsymbol{r}}_{\boldsymbol{1}} = \boldsymbol{R}_{\boldsymbol{\theta}_r}\dot{\boldsymbol{q}}_{\boldsymbol{1}} - \boldsymbol{R}_{\boldsymbol{\theta}_r}^{\boldsymbol{T}}\dot{\boldsymbol{R}}_{\boldsymbol{\theta}_r}\boldsymbol{r}_{\boldsymbol{1}} - \boldsymbol{R}_{\boldsymbol{\theta}_r}^{\boldsymbol{T}}\dot{\boldsymbol{p}}_{\boldsymbol{1}} \tag{4-9}$$

Where we use the relationships,

$$\boldsymbol{r}_{\boldsymbol{1}} = [0 \quad l \quad 0]^T \tag{4-10}$$

$$\boldsymbol{q}_{\boldsymbol{1}} = [x \quad y \quad 0]^T \tag{4-11}$$

$$\boldsymbol{v}_{\boldsymbol{B}} = \boldsymbol{R}_{\boldsymbol{\theta}_r}{}^{\boldsymbol{T}}\dot{\boldsymbol{p}}_{\boldsymbol{1}} = [\dot{s} \quad 0 \quad 0]^T \tag{4-12}$$

$$\dot{\boldsymbol{R}}_{\boldsymbol{\theta}_r} = \boldsymbol{R}_{\boldsymbol{\theta}_r}\, cpm([0 \quad 0 \quad \dot{\theta}_r]^T)$$

$$cpm([0 \quad 0 \quad \dot{\theta}_r]^T) = \begin{bmatrix} 0 & -\dot{\theta}_r & 0 \\ \dot{\theta}_r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} (Cross\ Product\ Matrix) \tag{4-13}$$

Now if we rewrite Equation 4.9, we obtain the following relation,

$$\begin{bmatrix} 0 \\ \dot{l} \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(\theta_r) & \sin(\theta_r) & 0 \\ -\sin(\theta_r) & \cos(\theta_r) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & -\dot{\theta}_r & 0 \\ \dot{\theta}_r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ l \\ 0 \end{bmatrix} - \begin{bmatrix} \dot{s} \\ 0 \\ 0 \end{bmatrix} \tag{4-14}$$

Finally, we can find the relation $\dot{l}$ and $\dot{s}$,

$$\dot{l} = [-\sin(\theta_r) \quad \cos(\theta_r)] \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{4-15}$$

$$\dot{s} = \frac{[\cos(\theta_r) \quad \sin(\theta_r)]}{1 \mp \kappa(s) l} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{4-16}$$

$$\tilde{\theta} = \theta - \theta_r \overset{\frac{d}{dt}}{\rightarrow} \dot{\tilde{\theta}} = \dot{\theta} \mp \kappa(s)\dot{s} \tag{4-17}$$

We can also use the relations $\dot{x} = v\cos(\theta)$, $\dot{y} = v\sin(\theta)$, $\dot{\theta} = \omega$ and collect the system of the equation as

$$\left.\begin{aligned} \dot{l} &= v\sin(\tilde{\theta}) \\ \dot{s} &= \frac{v\cos(\tilde{\theta})}{1 \mp \kappa(s) l} \\ \dot{\tilde{\theta}} &= \omega \mp \frac{\kappa(s) v\cos(\tilde{\theta})}{1 \mp \kappa(s) l} \end{aligned}\right\} \tag{4-18}$$

We aim to follow a desired path and construct a control algorithm such that path-following errors converge to zero. Let's assume that the direction of a movement along the desired curve is counter-clockwise. Therefore, by taking the "-" sign in the counter-clockwise direction, we can write Equation 4.18 as $\dot{l} = v\sin(\tilde{\theta})$, and $\dot{\tilde{\theta}} = u = \omega - \frac{\kappa(s)v\cos(\tilde{\theta})}{1-\kappa(s)l}$.

Samson control algorithm suggests that to minimize path errors control inputs can be considered as follows,

$$\left.\begin{array}{c} v = constant \\ u = -k_2\, l\, v\dfrac{\sin(\tilde{\theta})}{\tilde{\theta}} - k_3\tilde{\theta} \end{array}\right\} \qquad 4\text{-}19$$

Where $k_2, k_3 > 0$. As we notice the closed loop function is proportional to orthogonal distance $l$ with $k_2$ gain, and orientation angle error $\tilde{\theta}$ with $k_3$ gain. These terms feedback to robot motion until the orthogonal direction and orientation errors converge to zero. The closed-loop system with the gain $k_2$ and $k_3$ is asymptotically stable according to the following Lyapunov function.

$$V(l,\tilde{\theta}) = k_2\frac{l^2}{2} + \frac{\tilde{\theta}^2}{2} \qquad 4\text{-}20$$

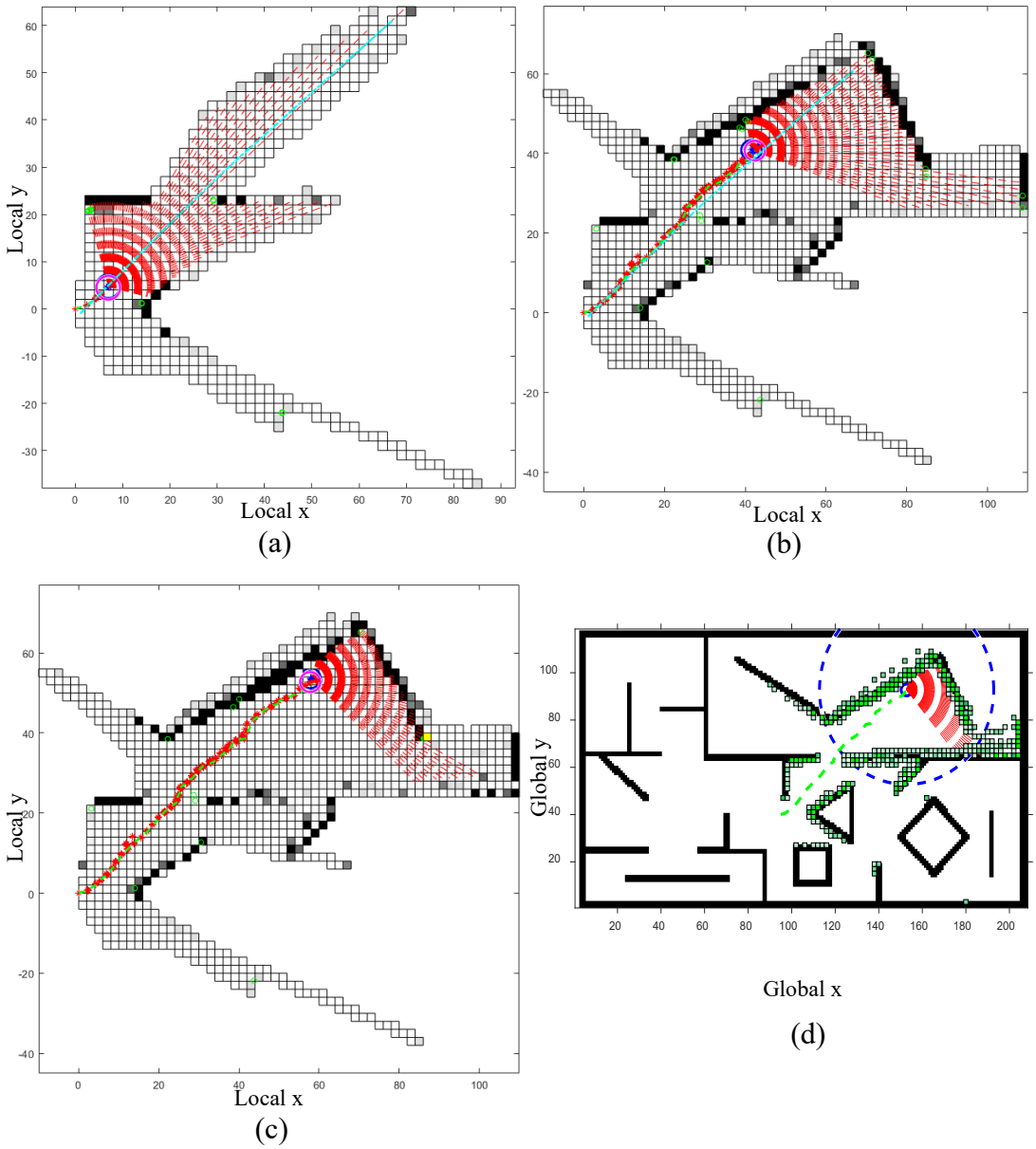
**Proof:** Consider the Lyapunov candidate function $V(l,\tilde{\theta})$. The function $V(l,\tilde{\theta})$ is positive define fiunction for $l$ and $\tilde{\theta}$ . By taking the time derivative of this function,

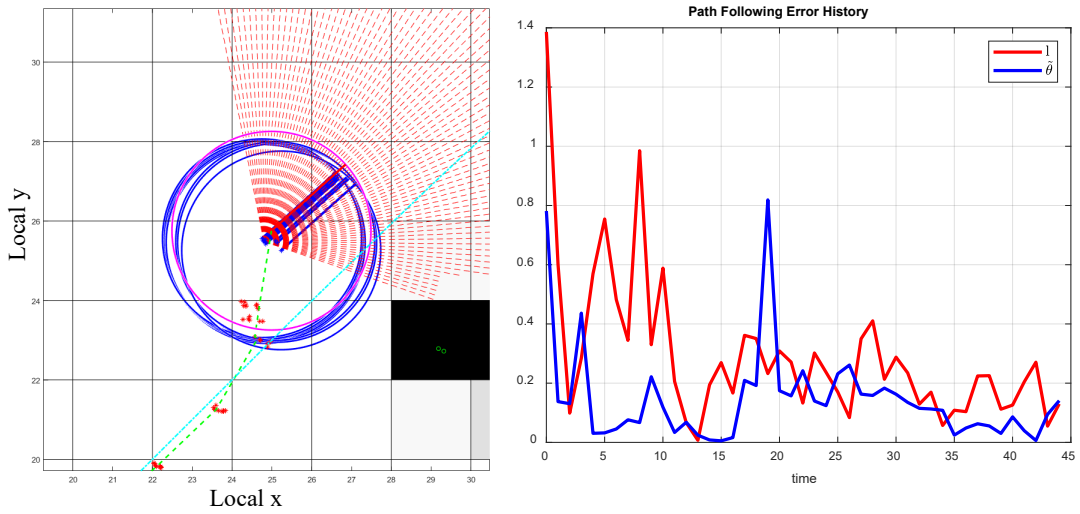
$$\begin{aligned} \dot{V} &= k_2 l\dot{l} + \tilde{\theta}\dot{\tilde{\theta}} \\ &= k_2 l sin(\tilde{\theta})v + \tilde{\theta}u \\ &= -k_3\tilde{\theta}^2 \leq 0 \end{aligned}$$

Notice that $\dot{V}$ is negative semi-definite and $k_3 > 0$. According to LaSalle's Theorem, the only condition that satisfies the path following case is $l = 0$ and $\tilde{\theta} = 0$. Therefore, the origin will be asymptotically stable.

If we apply the control structure given in Equation 4-18 to our explosive percolation path, the robot will follow the orthogonal projection of itself on the percolation path

with the minimization of path errors $l$ and $\tilde{\theta}$. The example scenario can be observed in Figure 4.15.






(e) (f)

Figure 4.15 (a), (b),(c). Path following process with Samson control algorithm in the occupancy grid map. The blue line represents the path to be followed. (d)The path of the robot is shown on the global map with the green line. (e) Probabilistic distribution of robot states. (f) The following error parameters history.

In Figure 4.15 (a) the path is indicated with a blue line. The robot position estimated with the particle filter is shown with red dots. Because the robot position estimation is based on the landmark detection and estimation in the FastSLAM algorithm, in Figure 4.15 (b) we can observe deviations from the path due to estimation errors in FastSLAM. However, as we observe more landmarks and the same landmark locations during the exploration process, these errors are diminished within the FastSLAM algorithm. The error history is given in Figure 4.15 (f). The red line represents orthogonal distance history with time and the blue line represents the orientation angle error history with time. As we can see, the errors are in a trend that converges to zero. Because of the errors in the particle filter, the robot location states are calculated within a probabilistic range. Therefore, deviations can occur as shown in Figure 4.15 (f). This probabilistic distribution of the robot state is indicated in Figure 4.15 (e) The blue circles represent the probabilistic distribution of the robot's position and the magenta circle represents the most probable one as a result of the importance factor assignment of particles in FastSLAM. In Figure 4.15 (d), we show the path of the robot on the global map. The green line is the path followed by the

robot during exploration. As a result of this simple scenario, we can deduce that the robot successfully follows the percolation path within the error values shown in Figure 4.15 (f) until the selected target point.

# CHAPTER 5

# EXPERIMENTS

## 5.1 Simulation Environment

The simulation environmet (Figure 5.1) is created in 2D as a monochrome bitmap. Each black cells represent the occupied regions in unstructured forms.

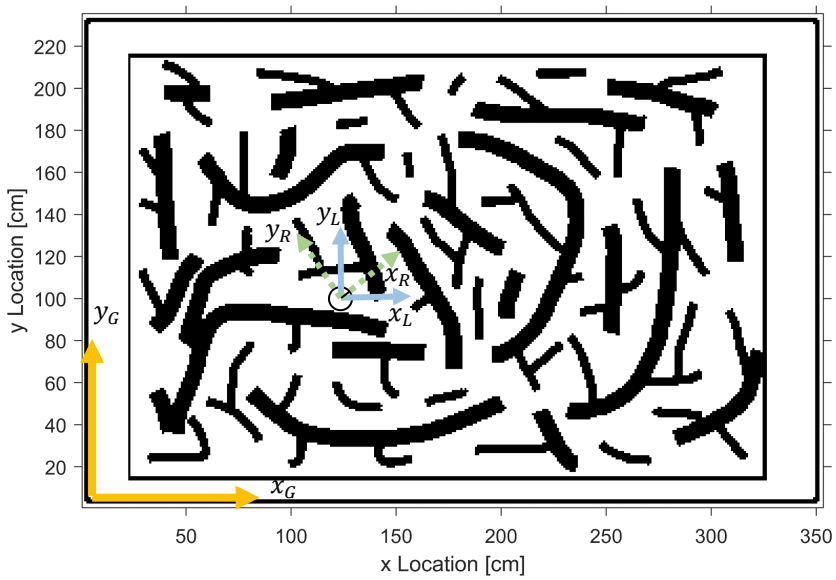



Figure 5.1. Global map used in the simulation. The drawings are selected randomly to represent debris

In Figure 5.1, a planar SAR robot is represented with a circle. We assume that a LIDAR sensor with a 120° field of view coincides with the robot frame $(x_R, y_R)$ indicated with dashed green arrows. The local frame $(x_L, y_L)$ identifying the basis of the occupancy grid map has its origin fixed where the robot begins its exploration. The global map $(x_G, y_G)$ is used to construct simulation terrain and indicates us the progress within the view of global coordinates. With the initialization of the robot motion, all map features, target detection, and path generation processes take place on the local frame. Lidar sensor readings are

represented first in the robot frame and then converted into the local frame to detect landmarks and occupied cells in the map. The conversion between the frame is sustained with the following transformation matrices and the transformation scheme is outlined in Figure 5.2.

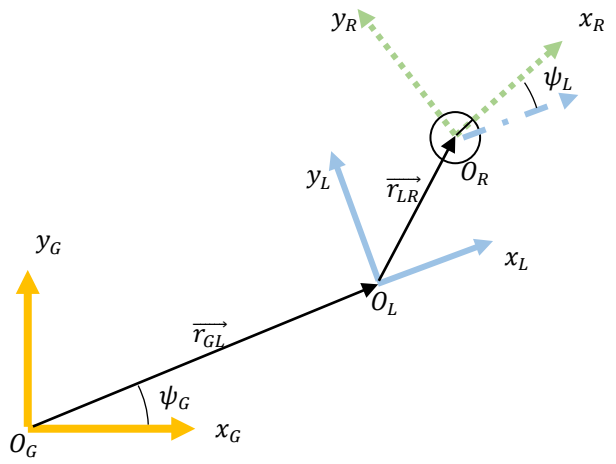



Figure 5.2. Axis transformation between global, local, and robot frames

$\overrightarrow{r_{GL}}$ represents the position vector of the local frame with respect to the global frame. $\overrightarrow{r_{LR}}$ represents the position vector of the robot frame with respect to the local frame. The transformation matrix $C^{GL}(\psi_G)$ is the transformation matrix from local to global frame, given as a rotation matrix,

$$C^{GL}(\psi_G) = \begin{bmatrix} \cos(\psi_G) & -\sin(\psi_G) \\ \sin(\psi_G) & \cos(\psi_G) \end{bmatrix} \quad 5\text{-}1$$

The transformation matrix $C^{LR}(\psi_L)$ is the rotation matrix from local to global frame and it is given as,

$$C^{LR}(\psi_L) = \begin{bmatrix} \cos(\psi_L) & -\sin(\psi_L) \\ \sin(\psi_L) & \cos(\psi_L) \end{bmatrix} \quad 5\text{-}2$$

The parameters used in the simulation are shared in Table 5.1. They are selected according to the size of the simulation environment which is confined to a 2m x 3m rectangular 2D space and the robot size is selected relative to this space to move

freely within the debris. By considering the global map scale the lidar range and resolution are selected as in Table I. Moreover, in the sensitivity analysis provided in section 5.5, we focus on how the method we propose is affected by changes in the selected parameter set.

Table 5.1 Simulation Parameter Settings.

| **Simulation Parameters** | | |
|---|---|---|
| **Parameter** | **Description** | **Value** |
| $r_{robot}$ | Radius of robot | 2.5cm |
| $\Delta t$ | Time resolution | 0.1sec |
| $\Delta d_{grid}$ | Grid resolution | 2cm |
| $\Delta \beta_{Lidar}$ | Lidar angular resolution | 2.4° |
| $L_{Lidar}$ | Lidar Range | 100cm |

In this section, on the other hand, we provide the simulation results of the proposed explosive percolation-enhanced active SLAM algorithm in highly complex, fractional-order unstructured maps.

Firstly we present the simulation environment and our assumptions and parameters during these experiments. In Chapter 4, we demonstrate the applicability of our algorithms on a simple map. However, in Chapter 5, we handled a more complex simulation environment in order to examine the effectiveness and success rate in the performance analysis of our novel approach.

We illustrate the ability of our methodology based on three different scenarios. The first scenario begins the process with a random initial point on the global map and tests the performance of the SAR robot with our algorithm against detecting the victim's vital sign successfully. The robot doesn’t know the initial position on the global map, therefore by constructing the local map, it is expected from the robot to extract a safe path to the victim's location. In the second scenario, the same process is repeated for a different random initial point on the global map to reveal that the methodology shows its success under different locations on the map and it is not

based on a specific location on the map. In the final scenario, the success of the robot is examined in case of a debris collapse and dead-end occurrence during a search. Due to map changes with the collapse, it is expected that the SAR robot performs the detection of successive dead-end occurrences and traps, adapts the local map to changing environments, and continues the search by detecting different target points on the map until the victim location is found.

In the final part of this chapter, we investigate the sensitivity analysis of our algorithm by changing the error parameters of the motion module, frontier target selection threshold, dead-end detection threshold, and the selection rule preference in explosive percolation and comparison of performance on finding a vital signal of entropy-based active SLAM versus explosive percolation based active SLAM approach.

## 5.2 Explosive Percolation Exploration with Random Initial Position 1

The first scenario is about finding the location of the victim within debris by utilizing the voids among rubbles and extracting a safe obstacle-free path for reaching the victim. The robot's initial position is selected as $\bar{p}_R^G = [104\ 103]^T$ in global coordinates and the victim's location is selected as $\bar{p}_V^G = [153\ 47]^T$. (Figure 5.3)

With the simulation initialization, at time $t = 0$, the SAR robot perceives the initial peripheral of the environment with the help of LIDAR. By using frontier search, the initial target point is determined and the exploration process begins via the explosive percolation to extract the obstacle-free path to the target location. In this way, the robot penetrates unknown regions of the debris to locate the victim.

The robot's explosive path to the victim's location on the global map can be seen with the green dashed line in Figure 5.3. This path represents our obstacle-free safe path to reach victims during search and rescue missions. The green dashes are cells obtained from the projection of the local map of the robot onto the global map as

zoomed in Figure 5.4. The local map in Figure 5.4 includes the occupied and void cells according to their occupancy probability. The green circles are the landmark locations and the red points are the possible position states of the robot, outcomes of the particle filter in FastSLAM 2.0. The red dashed lines are the raycasting of the LIDAR sensor.

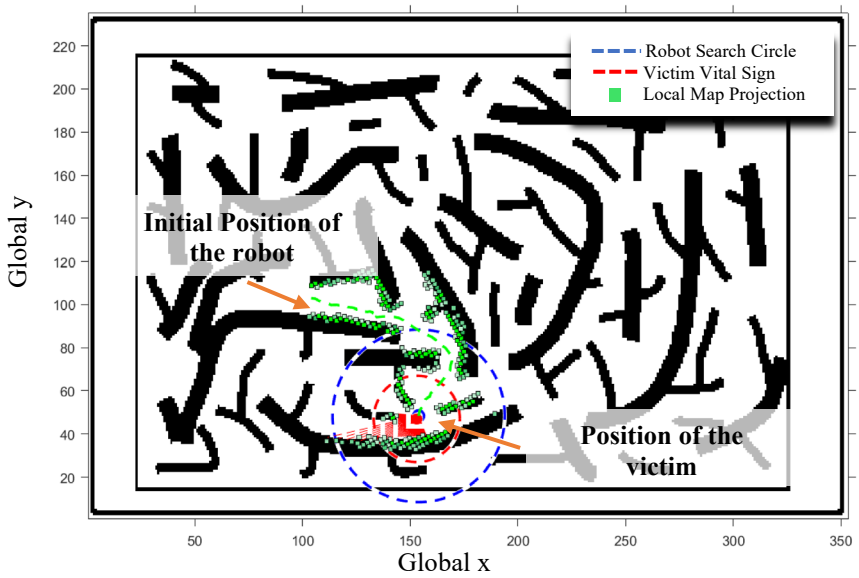



Figure 5.3. The explosive path is shown with a green line on the global map and the projection of the local map onto the global map is expressed with green cells.

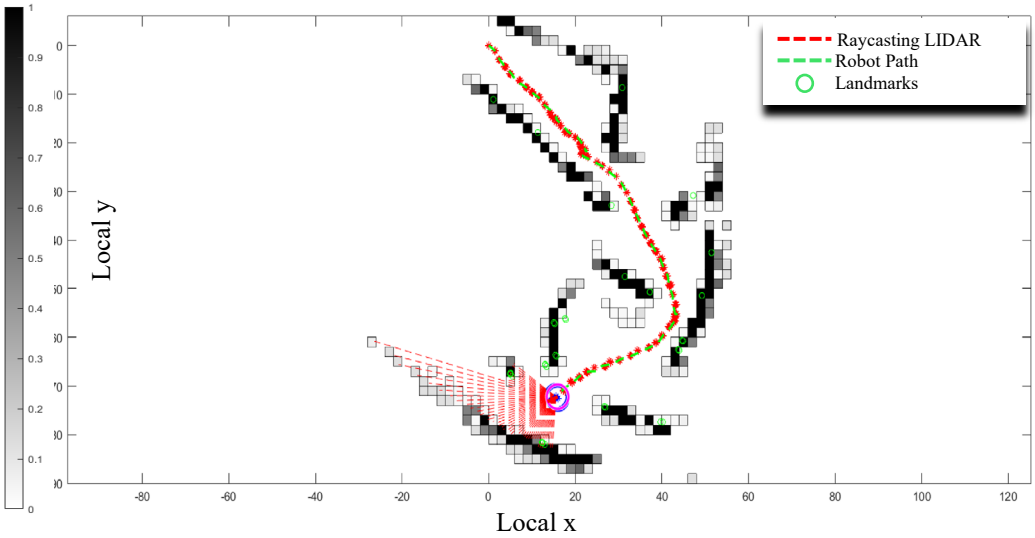
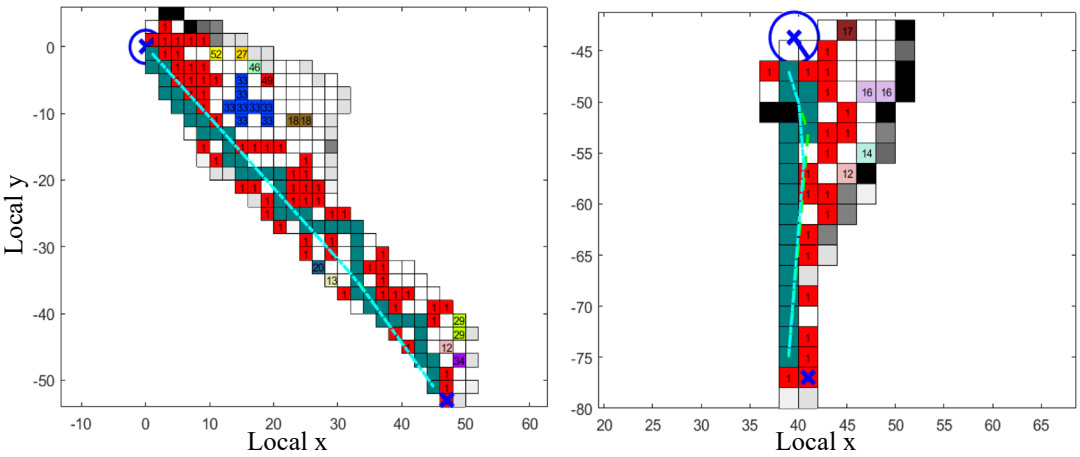
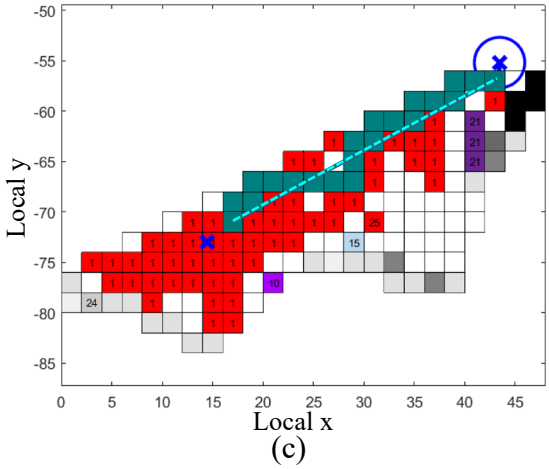



Figure 5.4. The local map of the SAR robot. The occupied cells are represented with gray and black according to their occupancy probability.

The explosive percolation cluster emergence in each target location and the found percolation path are given in Figure 5.5. The red cells are the final percolation cluster according to the summation rule of explosive percolation theory. Other clusters are shown with different colors and they can not join or create percolation cluster during the process because they did not evolve sufficiently during selection to create a large cluster to find percolation path. In Figure 5.5 (a), we observe the percolation path to the first target point resulting from the frontier target selection. The green cells in the figure are the percolation path and by pruning and smoothing the percolation path we obtain the desired path shown with cyan color.After reaching the first target, the second target is determined with frontier target selection algortihm again. The same process is followed as in target 1 and the result is given in Figure 5.5 (b). In Figure 5.5 (c), because the victim is within the LIDAR line of sight, the SAR robot directly aims at the victim's location as a target and percolates in that direction.




(a) (b)



(c)

Figure 5.5. Explosive percolation cluster occurrence and path post-process in each target detection. (a) First target percolation path. (b) Second target percolation path. (c) Target is the victim's location here. The robot directly heads into this location after detection.

The results of the first scenario are evaluated in two parts. Firstly, the success rate of the SAR robot in locating the victim utilizing the voids between rubbles is evaluated from an arbitrary location on the global map which is not given a priori to the robot. Secondly, the explosive percolation theory extracts obstacle-free safe paths to target locations and also generates a guide for the robot to explore the unknown regions of the map Moreover, instead of searching all voids cells, the percolation cluster suffices to obtain a safe path to the target location. This is a natural and critical bottom line of performance efficiency of percolation in obstacle-free path generation. If we investigate the fractional order of the percolation cluster (Figure 5.5 (a)), the fractional dimension at the emergence of the percolation cluster convergences to the fractional dimension of the lattice 1.8 that attests to the suitable compactness in the size of the cluster while finding automatically a path connecting the robot to the target within the fractional cluster.

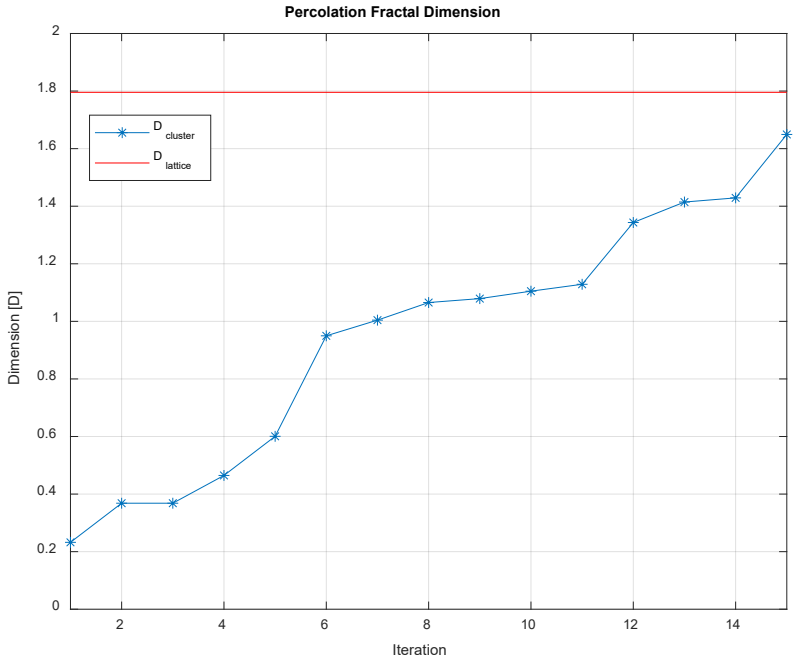



Figure 5.6. Clusters fractional dimension propagation until the emergence of percolation cluster for Figure 5.5 (a).

## 5.3 Explosive Percolation Exploration with Random Initial Position 2

The same process in the previous section is repeated in this part with a different initial position of the robot. The robot's initial position is selected as $\bar{p}_R^G = [252\ 149]^T$ in global coordinates and the victim's location is selected as $\bar{p}_V^G = [240\ 88]^T$ (Figure 5.7). The reason for selecting a different location on the map is showing our methodology does not depend on a specific location on the map. It performs its intended function in the different locations on map with different structure shape.

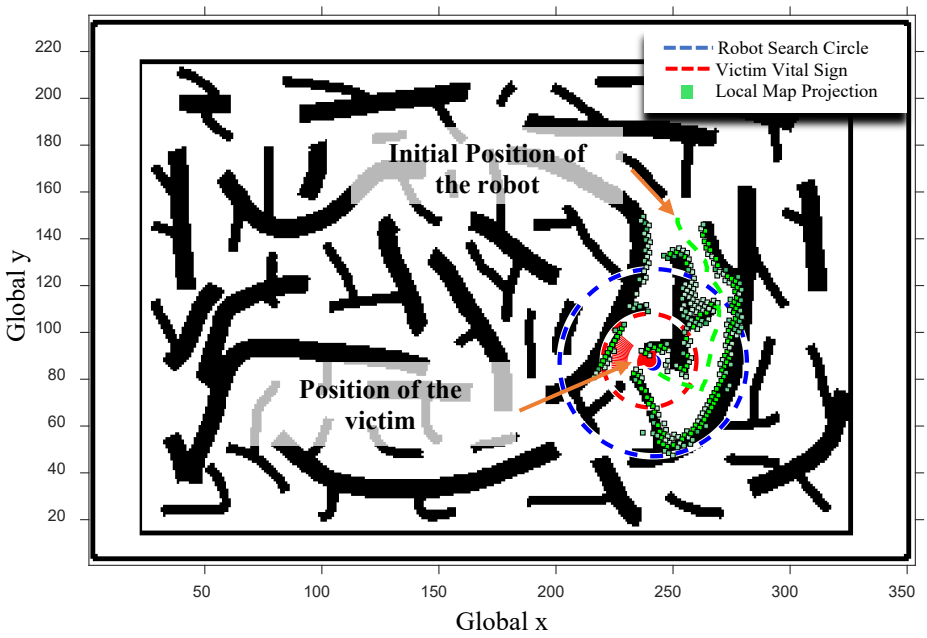



Figure 5.7. The explosive path in experiment 2 is shown with a green line on the global map and the projection of the local map onto the global map is expressed with green cells.

In this experiment, we tested the success of our algorithm with different unstructured parts of the map different from the first scenario. As expected from the algorithm, by utilizing the voids within obstacles, the SAR robot searches the area until the detection of the victim's location. The local map of the robot is shared in Figure 5.8. The SAR robot follows the percolation path result of explosive percolation. The green line indicates this percolation path. As we can observe from the figure, the robot extracts the obstacle-free path among occupied cells. At the moment the victim is detected, the robot selects this location as the target point and turns its heading toward the victim's location.

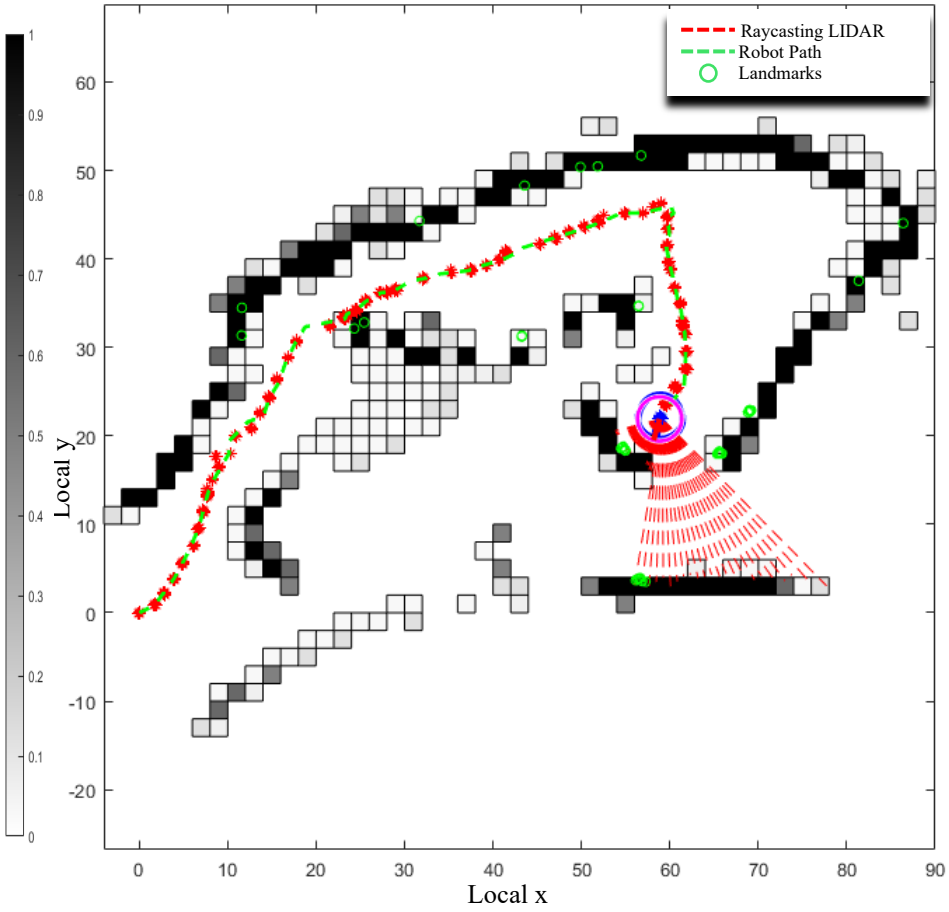
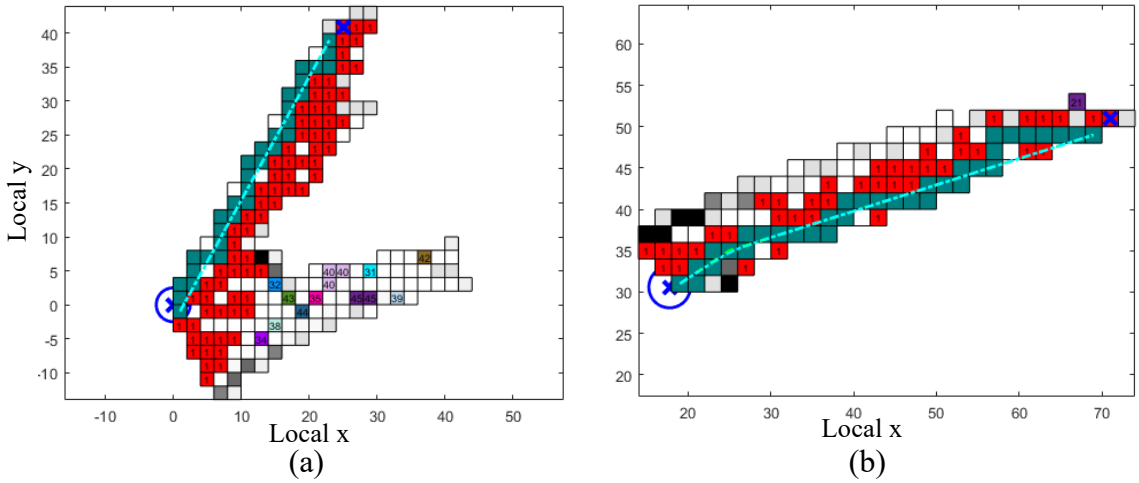



Figure 5.8. The local map of the SAR robot in experiment 2. The occupied cells are represented with gray and black according to their occupancy probability.



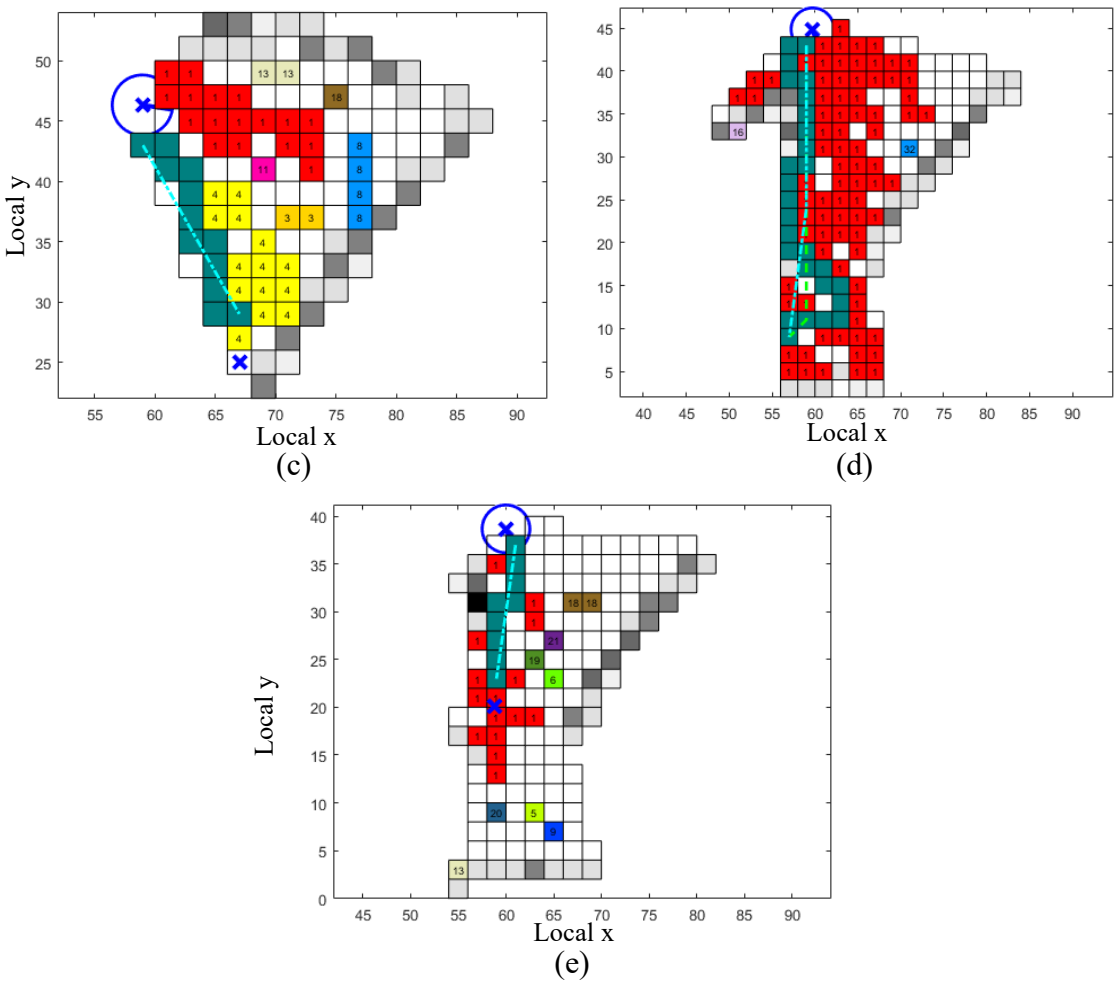



Figure 5.9. Explosive percolation cluster occurrence and path-finding process in each target detection. (a) First target percolation path. (b) Second target percolation path. (c) Third target percolation path. (d) Fourth target percolation path (e)Target is the victim's location here. The robot directly heads into this location after detection.

The percolation process after the detection of each target point based on the frontier target selection algorithm can be observed in Figure 5.9. After the detection of the first target point, the robot draws a path to this location with explosive percolation. When the percolation path with green cells is found as in Figure 5.9 (a), we need to prune and smooth the path to be followed by the robot. The percolation cluster is shown with red cells and it is numbered as 1. After reaching the first target point, the second target point is detected with a frontier-based algorithm and the

percolation path is extracted as in Figure 5.9 (b). The robot follows the path by using the path following the control algorithm. This process is repeated for each target point in Figure 5.9 (c) and Figure 5.9 (d) until the victim's location is detected. In Figure 5.9 (e), we see that the victim is within the LIDAR sensor range, and the SAR robot percolates into this location as the target point.

During search, we select the target point that has the furthest distance from the robot and gives us the maximum information about the environment. When the robot's vital signal detection (blue) circle and the victim's vital signal emission (red) circle intersect, the SAR robot then selects the target point which is closer to the vital signal location. In this way, the search path is tried to be minimized. The described case can be seen in Figure 5.10. In the upper left figure, the intersection of two circles is given. At the bottom of the left side, the percolation path is indicated. In the right figure, the target points are indicated with a blue cross. The selected target point is the closest one among the others with respect to vital signal location.

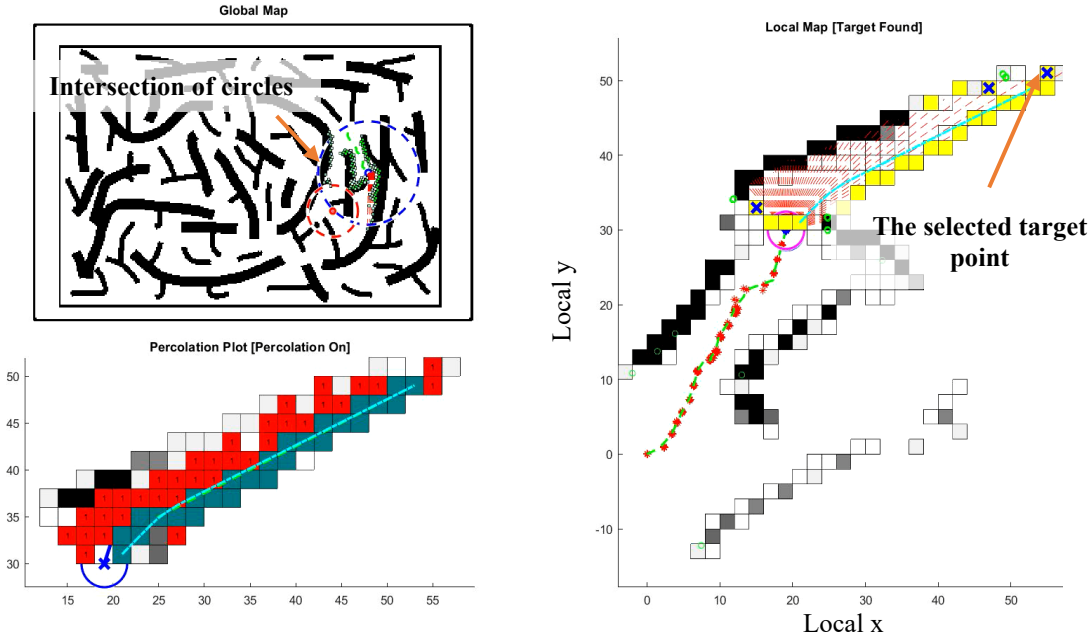



Figure 5.10. Selection of target point as the robot closes to the victim location.

Again we analyze the fractional order of the percolation cluster within the occupancy map lattice. For example, in Figure 5.9 (a), if we compare the fractional order of the occupancy lattice and clusters until the emergence of the percolation cluster, we obtain a plot given in Figure 5.11. This shows us the percolation cluster dimension convergences to the dimension of the lattice. In this way, we can use the percolation cluster as a resemblance of the occupancy map lattice to find a percolation path between the robot and target locations.

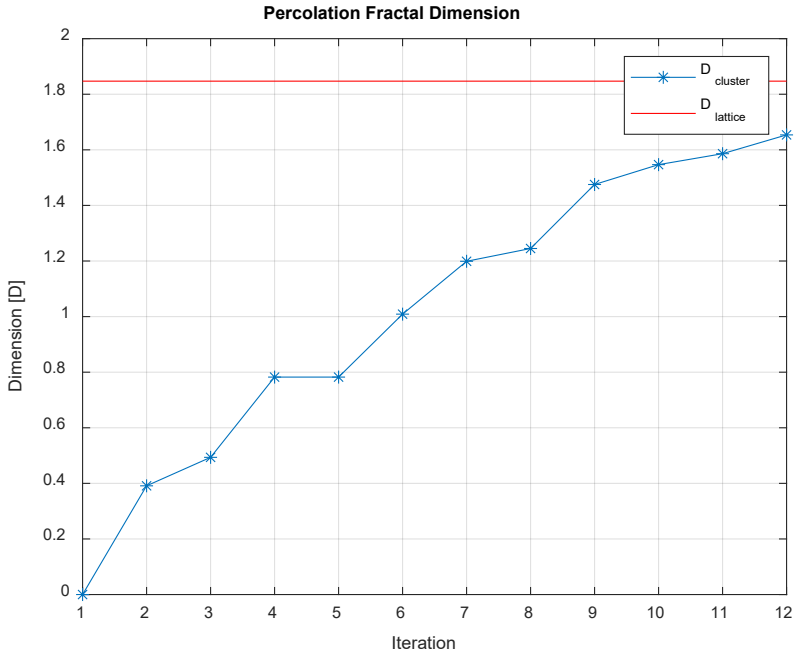



Figure 5.11. . Clusters fractional dimension propagation until the emergence of percolation cluster for Figure 5.9 (a).

## 5.4 Map Changing and Encounter Dead-End with Collapsing of Debris

The last scenario tests the proposed method's capability and performance in the case of collapsing debris within the search area. The structures in a certain part of the debris can be unstable and may result in the collapsing of rubbles. In that case, the environment and the shape of voids among rubbles change. In the worst case, the search path can be blocked, and dead-end forms during the exploration of the

robot. Then, it is expected from the robot to continue the search and find alternative target points in the unknown regions of the disaster area.

In order to analyze our method under changing environment features, we have arranged a simulation scenario. In this scenario, at $t = 5$ in simulation time, we deliberately blocked the percolation path as can be seen in Figure 5.13. Let us look at the simulation environment before the collapse, in Figure 5.12, the initial condition for robot and victim locations are given in global coordinates. The robot's location is $\bar{p}_R^G = [120\ 100]^T$ and the victim's location is $\bar{p}_V^G = [70\ 135]^T$. The blue circle is the vital signal sense region of the SAR robot and the red circle is the vital signal emission range of the victim. At simulation time $t = 5$, the collapsing debris is inserted into the simulation (Figure 5.13).

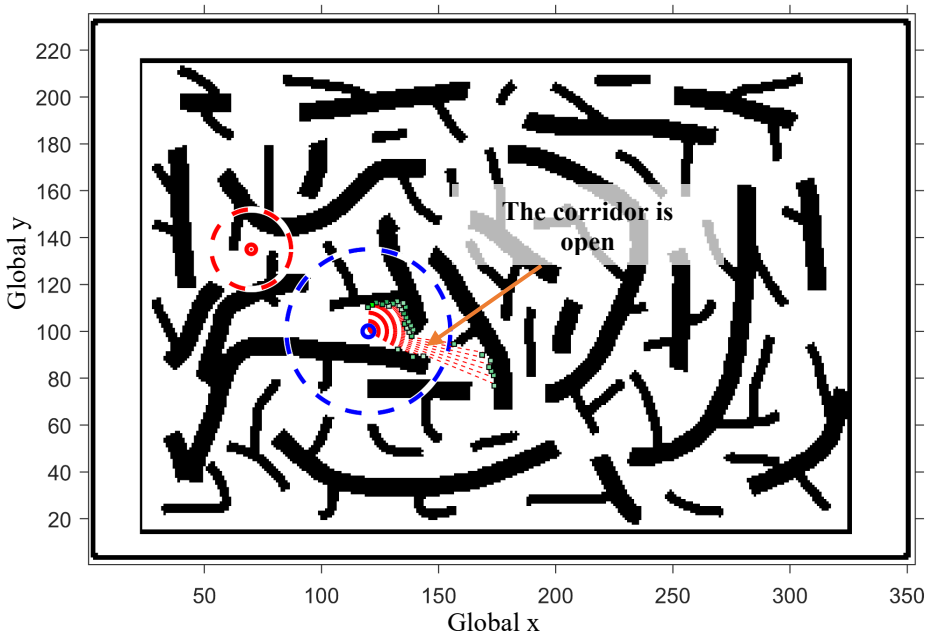



Figure 5.12. The initial condition of robot and victim locations on the global map.

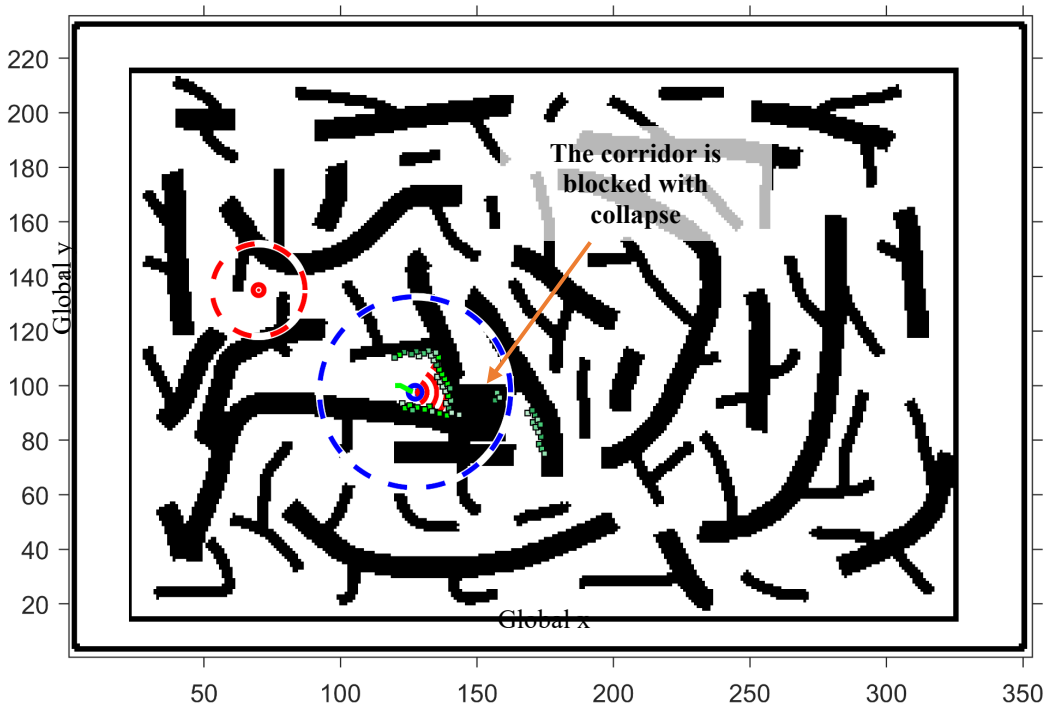
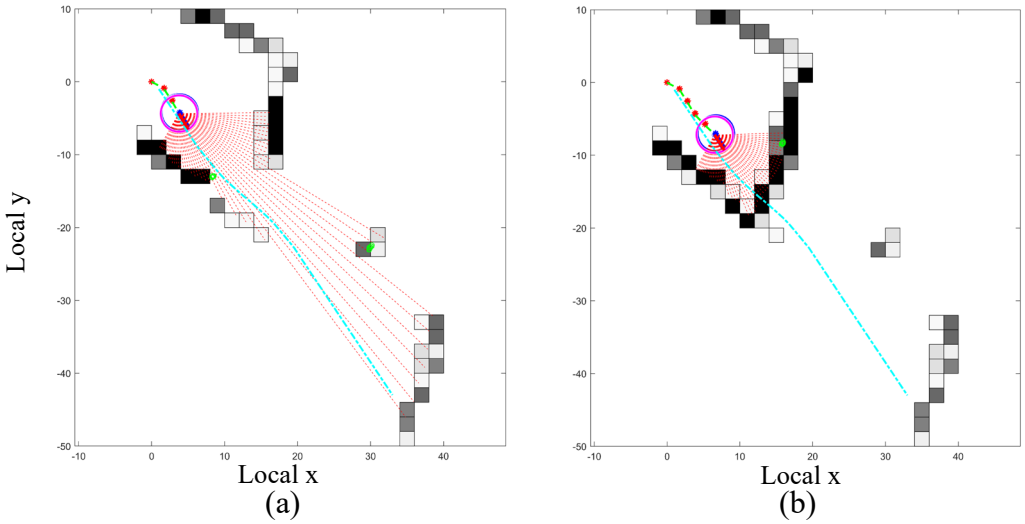



Figure 5.13. The collapsing of debris at $t = 5$ on global map.



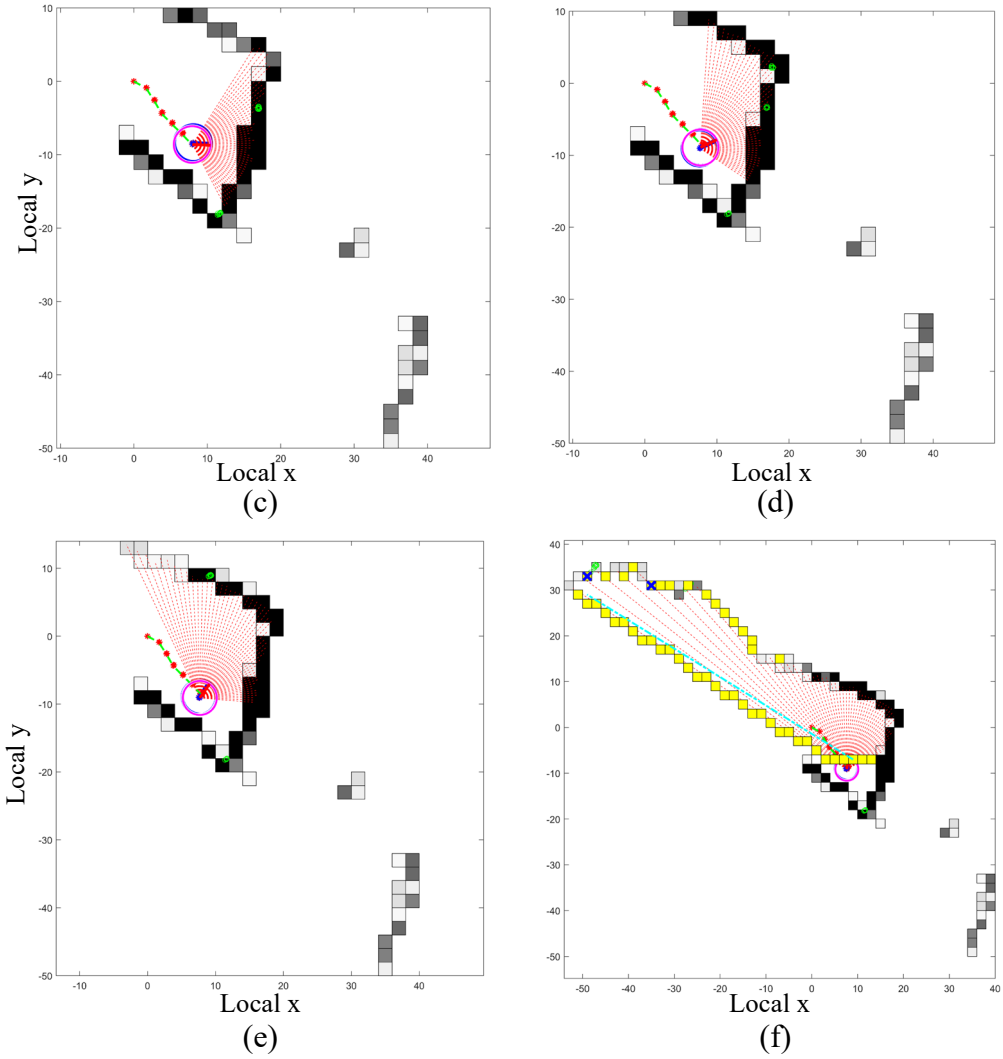



Figure 5.14. Dead-end detection process in the local map.

The detection of a dead-end can be observed in detail in Figure 5.14. In Figure 5.14 (a), the corridor is open for the robot to reach the selected target. However, when the collapse of debris occurs, the map is changed and the reflection of this change in the local map can be seen in Figure 5.14 (b). The path shown with cyan is blocked by the occupied cells. In such a case robot calculates the intersection of the path with the occupied cells in the occupancy map based on LIDAR measurements. To verify a dead-end situation, we count this intersection number over time. We can see the situation in Figure 5.15. When these intersections with occupied cells (blue dashed line) are greater than the value 10 (green dashed line), the dead-end

signal (red line) increases. In final, when this signal hits the threshold value 3, the dead end is confirmed. After this point, the SAR robot searches for a new target to continue its exploration. In Figure 5.14 (c),(d), and (e) we can see this process. In Figure 5.14 (c) the robot begins the search for a new exploration region. Because the measurements by LIDAR observe the previously discovered occupied regions, the search continues as in Figure 5.14 (d) and (e). In Figure 5.14 (f), the robot detects new target points within unexplored area through new voids and continues to search and rescue mission.

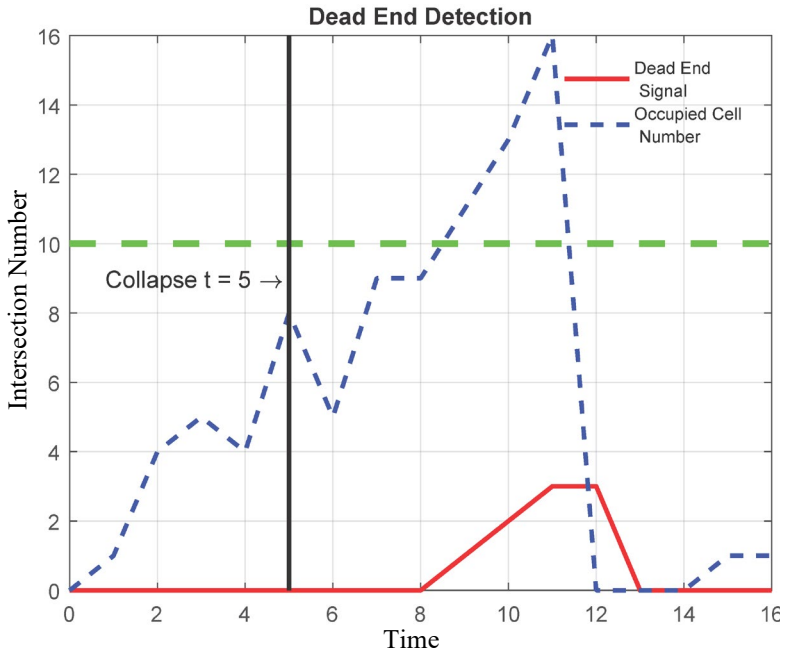



Figure 5.15. Dead-end detection and occupied cell number intersected with the path and their time propagation.

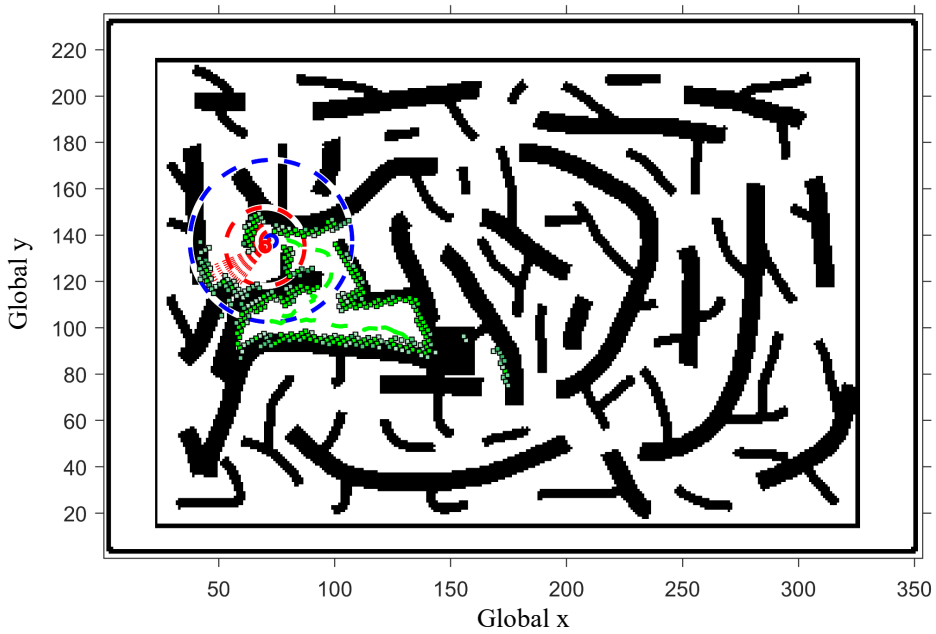



Figure 5.16. The explosive path in experiment 3 is shown with a green line on the global map and the projection of the local map onto the global map is expressed with green cells.

The whole process is shared in Figure 5.16 until the victim's vital signal is found. The green dashed line is our safe and obstacle-free explosive percolation path to reach the victim. The green squares are the projection of the occupancy map onto the global map. As we can notice in the figure, the two circles overlap at the victim's location and the robot identifies the victim's location within the debris. The process from the perspective of the robot on the local map can be examined in Figure 5.17. In this map, the green dashed line represents our path on the local map. The red points are the distribution of robot positions as a result of particle filter estimation in FastSLAM 2.0. The gray and black cells are the occupied cells according to their occupancy probabilities in the occupancy grid map.

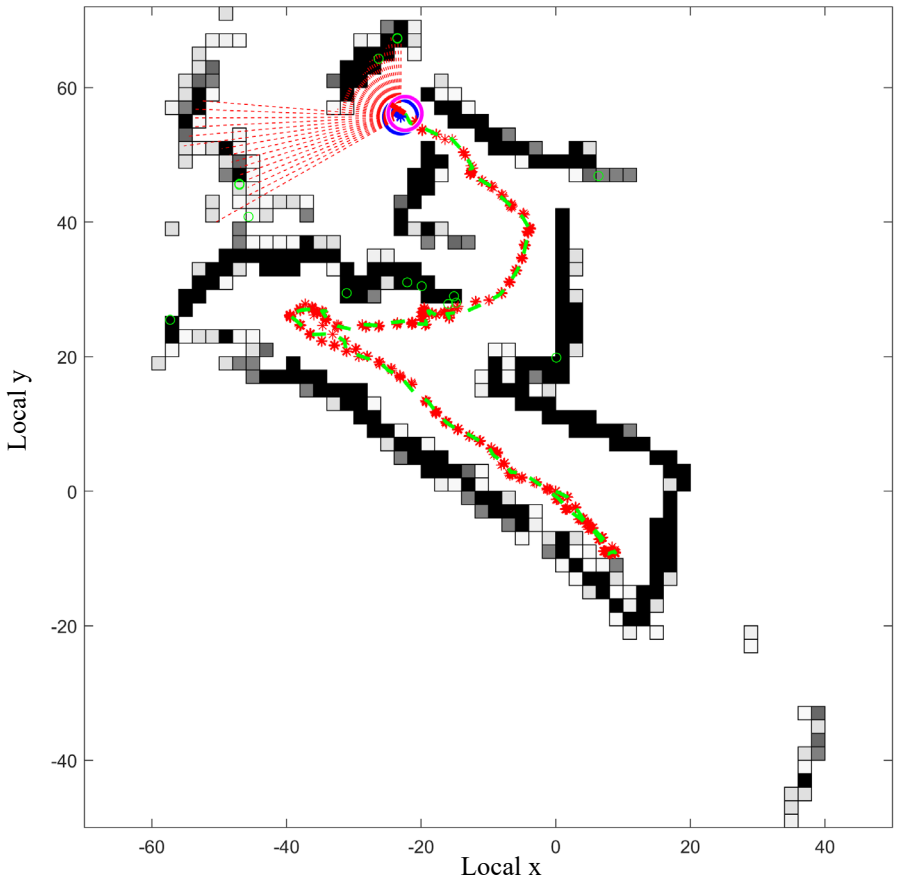
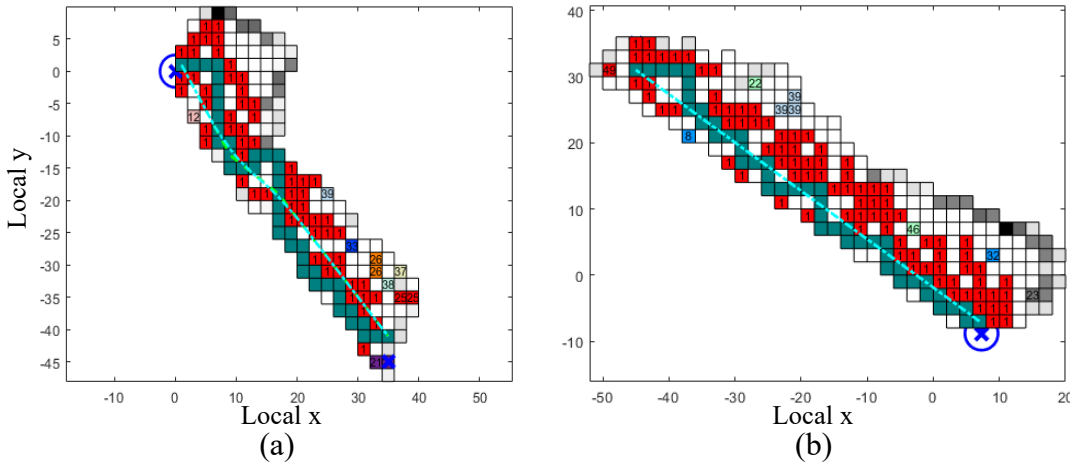



Figure 5.17. The local map of the SAR robot in experiment 3 until the detection of the victim's location.



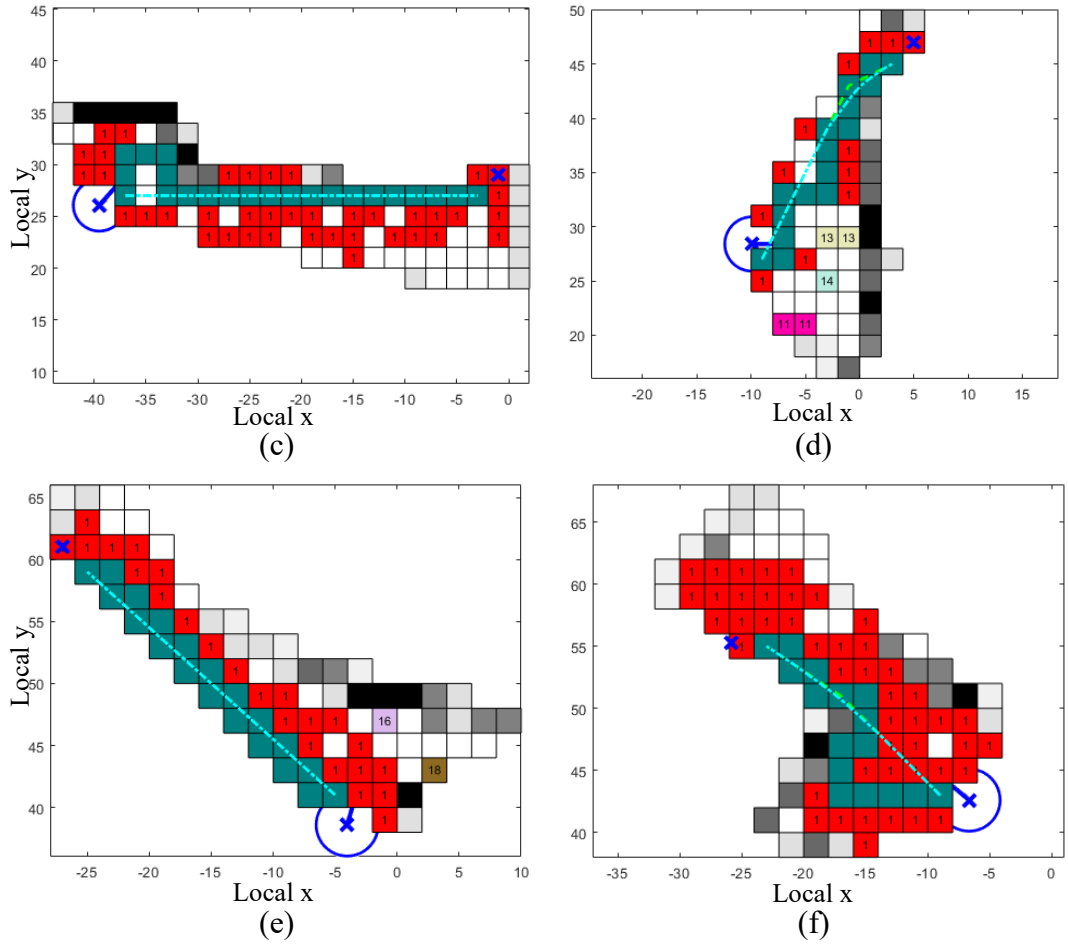



Figure 5.18. Explosive percolation cluster occurrence and path-finding process in each target detection.

The explosive percolation cluster emergence and path-finding process for each selected target point are shared in Figure 5.18. In Figure 5.18 (a), the percolation cluster to the first target point is given before the occurrence of the dead-end. After the dead-end occurs, the new detected target point can be observed in Figure 5.18 (b). The selection process of a new target point after dead-end is summarized in Figure 5.14. When the robot reaches a target point, the explosive percolation process is started again to keep going the exploration until the location of a victim is found. As we can see from Figure 5.18 (b), (c), and (e), they connect the robot and target points directly without the intervention of occupied cells, therefore, there is a line connecting those points. However, in Figure 5.18 (a), (d), and (f) the path

is deflected with occupied cells. In this case, the pruning and smoothing algorithms provide us a much smoother path for the robot to follow.

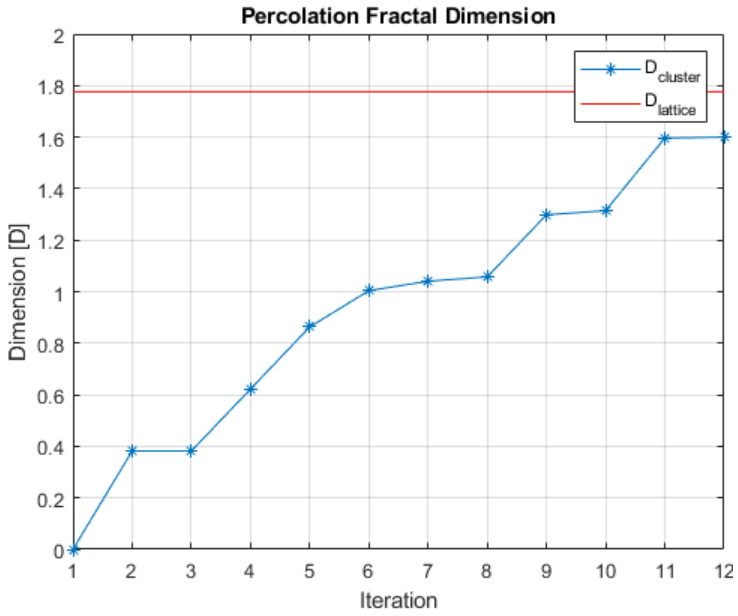



Figure 5.19. Clusters fractional dimension propagation until the emergence of percolation cluster for Figure 5.18 (a).

If we compare the fractional order of the occupancy lattice and clusters until the emergence of the percolation cluster, we obtain a plot given in Figure 5.11 for the percolation cluster in Figure 5.18 (a). This shows us the percolation cluster dimension convergence the dimension of the lattice. In this way, we can use the percolation cluster as a resemblance of the occupancy map lattice to find a percolation path between the robot and target locations. A similar analysis can be applied to each percolation cluster, we only give a single cluster as an example to indicate the cluster validity to find a percolation path.

## 5.5 Sensitivity Analysis

In this part, we investigate the sensitivity analysis of our algorithm by changing the error parameters of the motion module, dead-end detection threshold, and the selection rule preference in explosive percolation.

### 5.5.1 Motion Error Parameter Analysis

In localization and mapping, motion errors affect significantly the performance of SLAM due to harsh terrain conditions. This situation frequently causes errors in the odometry of the robot during movement. We can demonstrate this effect by changing the $R_T$ process covariance matrix in FastSLAM. The susceptibility of explosive percolation-based FastSLAM to motion errors on state transition $p(x_t|x_{t-1}, u_t)$ is investigated in this part.

For the experiment, the global map in Figure 5.20 is selected. The initial condition of the SAR robot is selected as $\bar{p}_R^G = (104,103)$ and the vital signal location is $\bar{p}_V^G = (170, 75)$ in global coordinates. The $R_t$ parameter sets are determined as given in Table 5.2.

Table 5.2 Process Covariance Matrix $R_T$ Parameter Set

| | $R_t = diag\left[\left(\sigma_x^2\ \sigma_y^2\ \sigma_\theta^2\right)\right]$ |
|---|---|
| Set 1 | [0.9 0.9 0.001] |
| Set 2 | [0.9 0.9 0.05] |
| Set 3 | [0.01 0.01 0.05] |
| Set 4 | [0.1 0.1 0.01] |
| Set 5 | [0.05 0.05 0.005] |
| Set 6 | [0.01 0.01 0.001] |

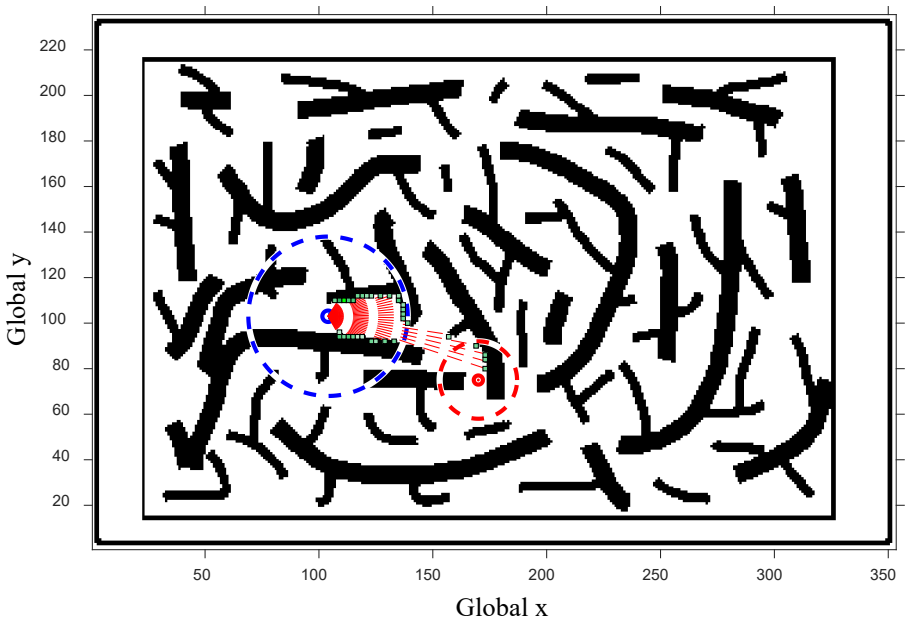
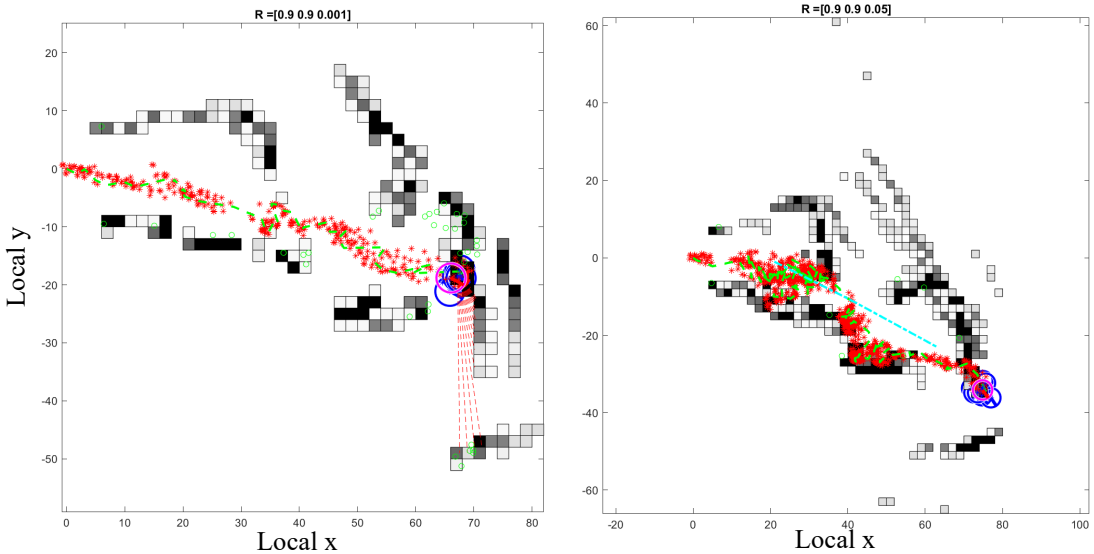



Figure 5.20. Selected global map to test process noise effects on explosive percolation-based SLAM. The robot and vital signal location are given in blue and red circles respectively.



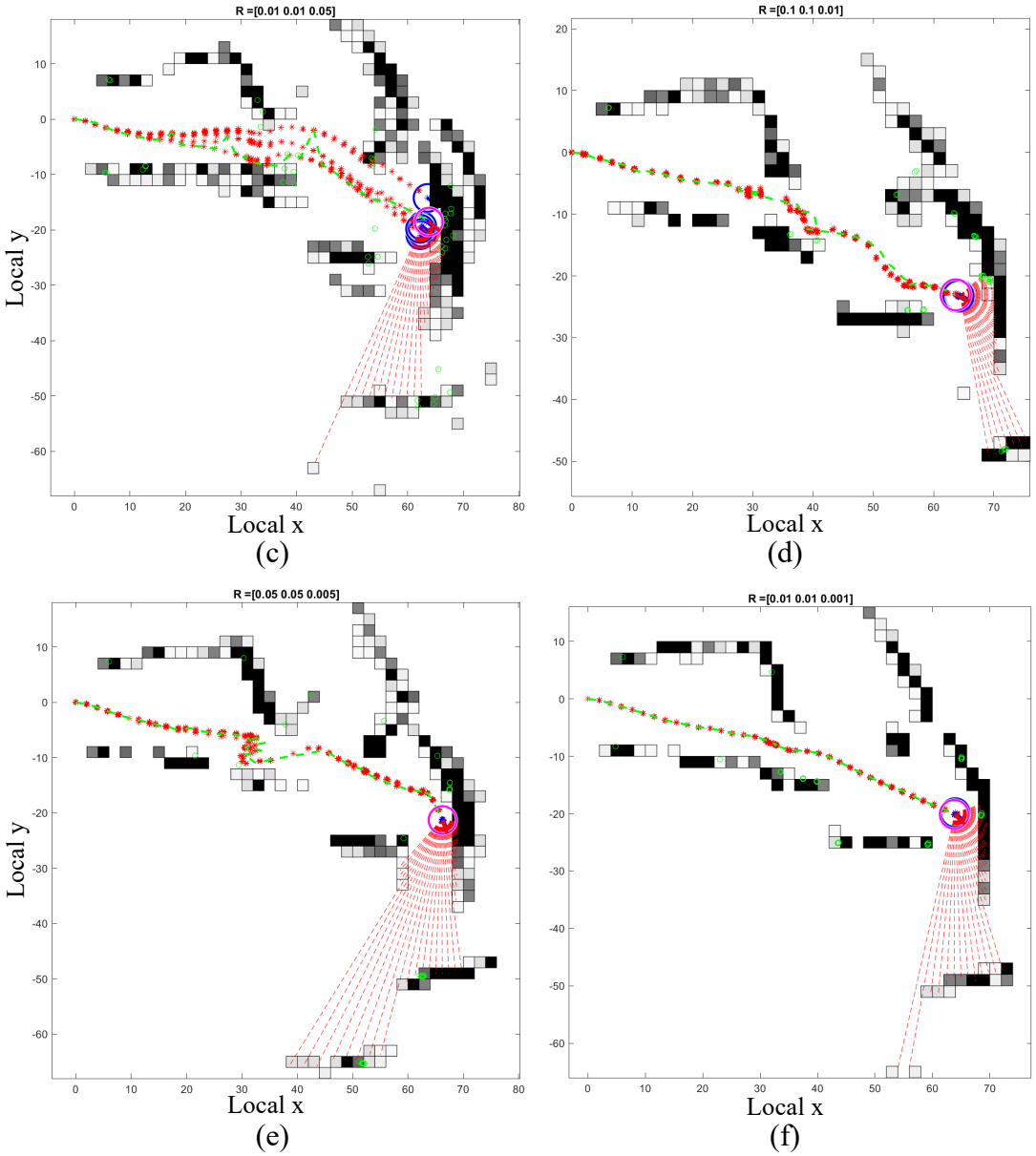



Figure 5.21. Simulation results according to selected process noise values in Table 5.1

When we examine the results in detail, the high noise in position and orientation gives the worst results as in Figure 5.21 (b) corresponding to set 2 parameters in Table 5.2. If we keep the position Gaussian noise level the same and decrease the orientation noise value (set1 parameters), we obtain an improved result in Figure 5.21 (a) and the SAR robot performs a better result in the following path with respect to Figure 5.21 (b)

In reverse, if we keep the noise level the same for orientation and decrease the position noise values (set 3 parameters), the result is obtained as in Figure 5.21 (c). We can deduce that the position noise values cause scattered possible position values in the particle filter. This will cause a more interwoven occupancy grid map which makes it hard to detect obstacles and find a proper path to percolate. In the end, the percolation performance diminishes and the obstacle-free path is not extracted properly.

If we decrease the noise level to the given parameters in set 4 and set 5, the occupancy map results are more clear and the robot can follow to percolation path to reveal further obstacles and unknown regions of the map. The results can be investigated in Figure 5.21 (d) and (e). In Figure 5.21 (f), a low noise parameter is selected to see the results. As expected, the robot follows the path with minimum scattered in the position and orientation in the particle filter as if a deterministic dynamical system. The results give a more stable occupancy grid map and position states of the robot which is ideally not possible in a disaster search environment.

### 5.5.2 Dead-End Detection Threshold Analysis

One of the important aspects during search within rubbles is detecting dead-ends and avoiding them. The collapses during searching can block the percolation path and create dead-end forms. When we consider that search and rescue missions are time-critical operations and finding the victim's location is the main objective, we should not consume search time with dead-ends.

In our study, the details of dead-end avoidance are mentioned in Chapter 5 with a map-changing example scenario. In this part, we analyze the sensitivity of the dead-end detection algorithm to dead-end threshold selection. To examine the threshold selection importance, we select a dead-end threshold set as, $S_{DeadEnd} = \{5,10,15,20\}$. For each selection, we initialize the robot at the global location of

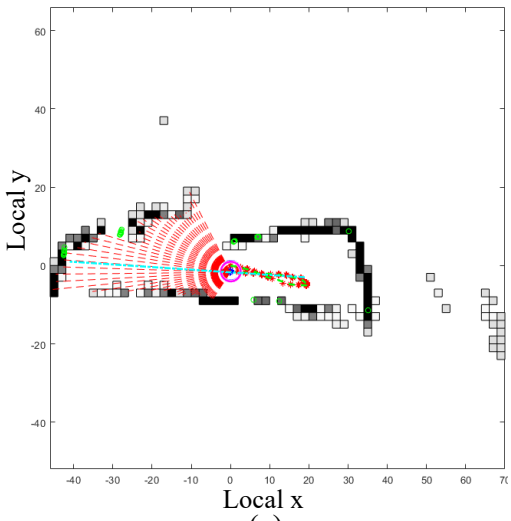
$\bar{p}_R^G = (104, 103)$. When the simulation time is at $t = 5$, we create an artificial dead-end as in Section 5.3.

The results are shared in Figure 5.22. In the first selection of threshold value $T_{DeadEnd} = 5$, the experiment shows us the SAR robot detects the dead-end occurrence first at time $t = 8$ in Figure 5.22 (b). The occupied cell number on the percolation path shown in blue increases after this time and when the increase of dead-end signal is detected which is indicated in red, the dead-end occurrence is verified at $t = 12$. Even if the detection of dead-end occurrence is detected in advance, the drawback of this selection is that the dead-end detection signal is more susceptible to a path crossing with occupied cells. Because of this reason, in the early part of the experiment, we can see some picks that cross the threshold value in Figure 5.22 (b). This can lead to false inferences for the SAR robot in the detection of dead-ends. The robot path after the detection of the dead-end signal is shown in Figure 5.22 (a), the new target point is detected, and the SAR robot follows the path indicated with cyan color.

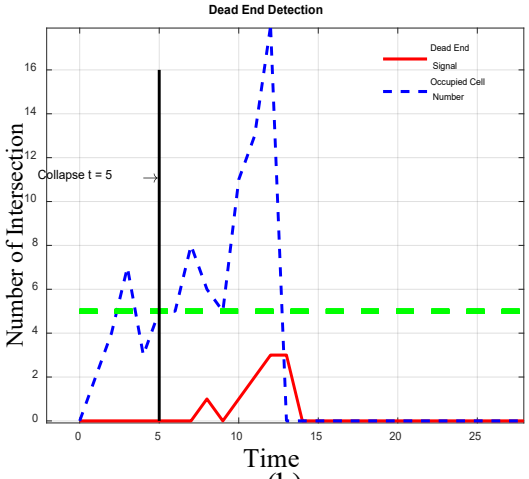
The results show us that the selection of threshold $T_{DeadEnd} = 10$ gives us a better result than the previous one when compared to picking up a false path crossing with the occupied cells. The first crossing of the path with occupied cells at $t = 10$. We detect the dead-end occurrence at $t = 12$ after the first threhold passage in Figure 5.22 (d). After the dead-end occurrence, the path followed by the robot and continuing in search are shown in Figure 5.22 (c). As we can see, we observe the dead-end without being too much close to the dead-end, and false inferences are minimized with the increasing threshold value.

For the next selection of dead-end threshold $T_{DeadEnd} = 15$ and $T_{DeadEnd} = 20$, the dead-end signal is triggered at $t = 16$ and $t = 21$ respectively. As we notice that the detection of a dead-end is substantiated at the very late part of the process with respect to the previous selections. This causes the exploration of the robot even further toward the dead-end location even if it is not necessary to be explored further. Therefore, the robot takes a longer path until the detection of dead-end and

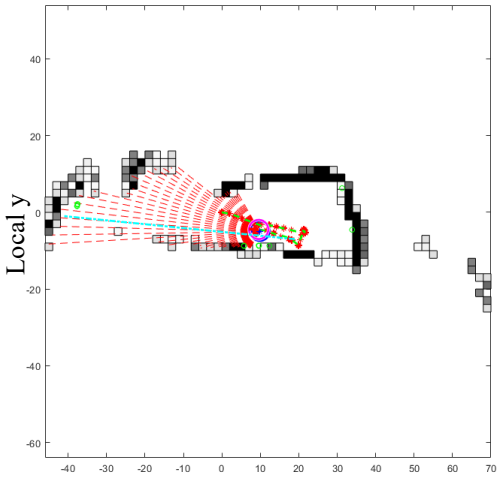
loses valuable time to explore new regions to detect the vital signal of the victim. The results are shared in Figure 5.22 (e) and (f) for $T_{DeadEnd} = 15$ and Figure 5.22 (g) and (h) for $T_{DeadEnd} = 20$.



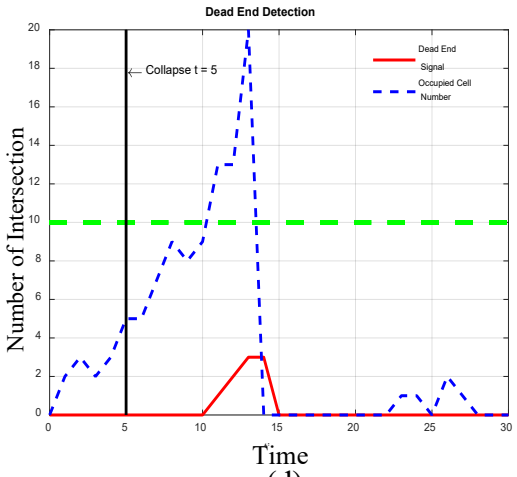

(a)




(b)




(c)




(d)

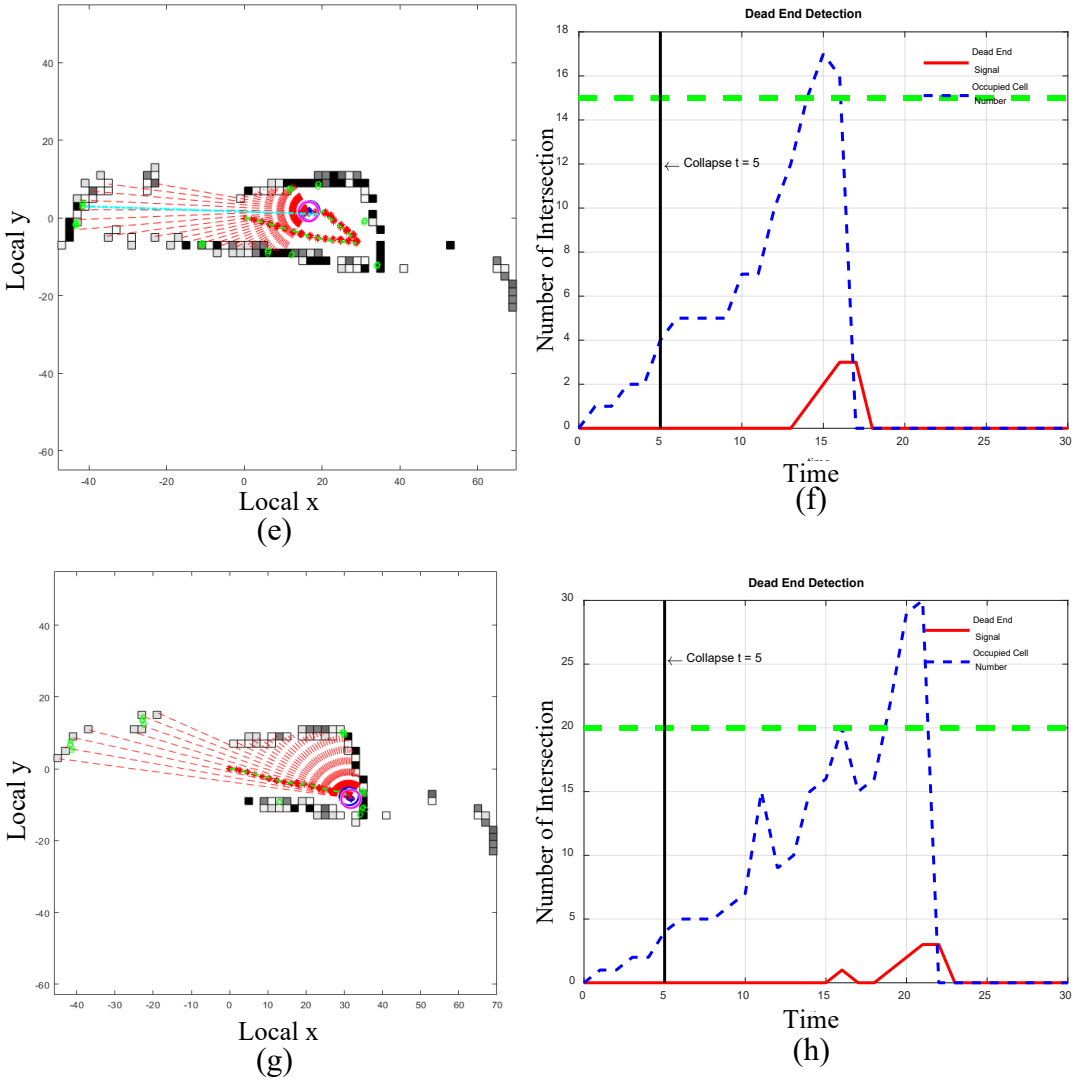



Figure 5.22. Dead-end threshold analysis results with different threshold values.

### 5.5.3 Explosive Percolation Selection Rule Analysis

In order to create clusters in a lattice via explosive percolation, we mainly have two selection rules. These are the summation and production rules. In our study, we select the summation rule to create a percolation cluster between the robot and target locations. However, to see the difference or similarities between the selection rule and how the selection affects the creation of a percolation cluster, we create an example occupancy map lattice within the field of view of the robot as in Figure

5.23. The robot location is indicated with a blue circle and the target is indicated with a blue cross.

For both summation and product rule, the explosive percolation cluster emergence is repeated 20 times in Figure 5.23. The results are analyzed according to the fractal dimension of the percolation cluster in lattice and percolation probability of percolation. In Figure 5.24, we can examine the similarity between product and summation rules with respect to the fractal dimension of the percolation cluster. The red line shows us the fractal dimension of the lattice as 1.8. The dimension of a cluster in each iteration until the creation of the percolation cluster for the summation rule is indicated with orange lines. And the blue one is for the product rule. Both summation and product rule shows similar performance and reaches the almost same fractal dimension at the creation of the percolation cluster. This shows us that the percolation path can be found in a similar dimension of clusters using either of rules. Otherwise, if one of them had a lower dimensional value at the percolation cluster emergence, this would have indicated a much lower dimension cluster that would be a candidate to find a percolation path. Therefore we would have found a percolation path by searching less dense percolation clusters and this would have increased the search performance. However, according to the results, the performances are the same for both methods.

In explosive percolation, the important aspect is advancing or delaying the emergence of percolation clusters by utilizing the selection of the largest cluster in each other or smaller clusters in each other during the percolation process respectively. S we want the creation of the largest percolation cluster as soon as possible to speed up the process, we reversibly applied the selection by connecting the largest cluster. This methodology in our study is expressed in detail in Section 4.5. The creation of an explosive cluster in the early part of the iterations means that we obtain the percolation probability at an earlier time of the process and this increases the speed of finding the percolation cluster. Within this perspective, we examine both methods to reveal which method brings the percolation probability in the earlier phase of the iteration. In the sense of percolation probability, the results

are the same again for both methods. They perform a similar percolation probability performance at 0.8 level. The results are shared in Figure 5.25. As we indicate before, both methods exhibit similar percolation probability by converging the 0.8 value during iteration.

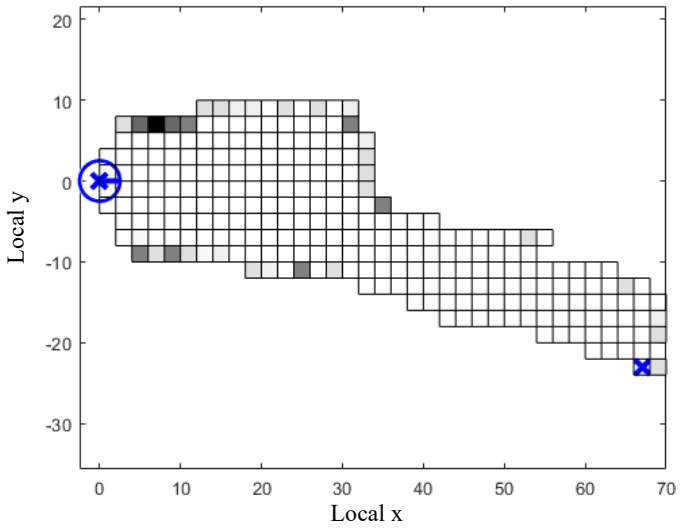



Figure 5.23. Example occupancy grid map lattice to perform summation and product rule selection.

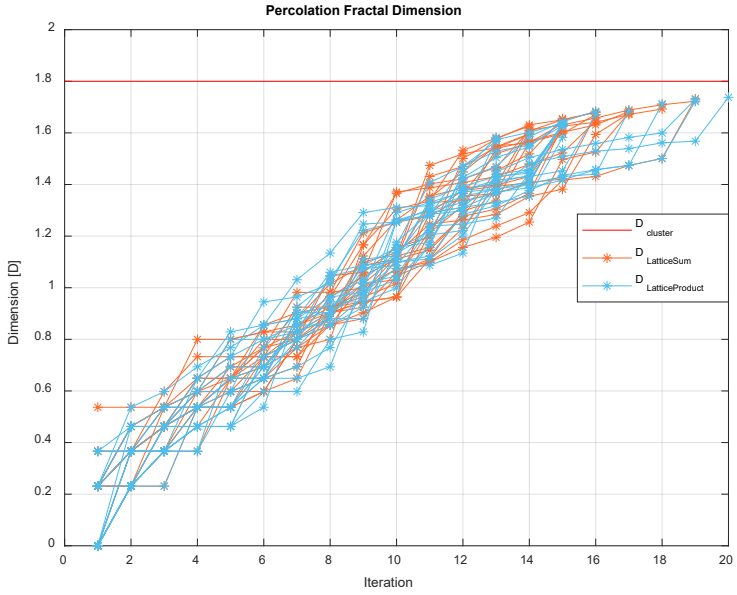



Figure 5.24. Fractal dimension comparison for each selection rule until the emergence of percolation cluster.

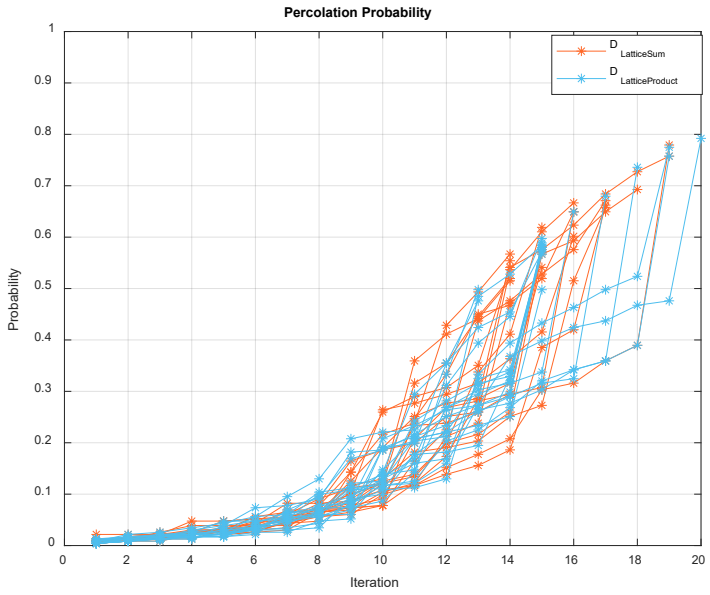



Figure 5.25. Percolation probability comparison for each selection rule until the emergence of the percolation cluster.

### 5.5.4 Robot's Vital Signal Detection Circle Change Analysis

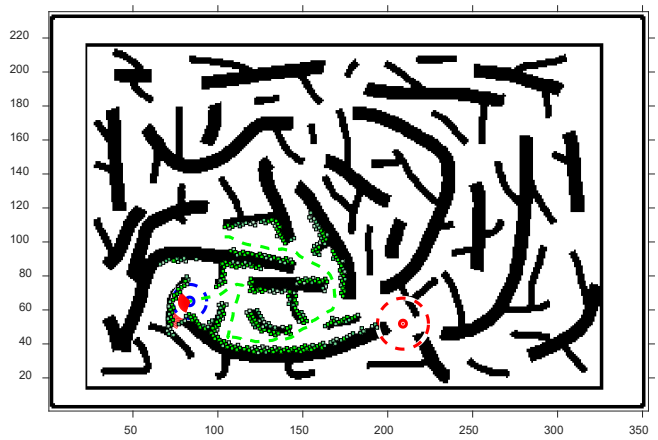
We mention that the detection of the vital signal emission signal is presented with a blue circle for the SAR robot in Section 4.4. The circle radius directly affects the target selection when the robot senses the vital signal of the victim. If this cone is smaller, the robot's target selection near the vital signal can lead to a different target and this may cause getting away from the victim's location. Finally, the process will take much more time than anticipated. On the contrary, if this detection circle is larger with an enhanced sensor system, the target is selected according to which one of the targets is closest to the victim's location. Therefore, the search can be processed with much less effort and without taking too much distance in the search area victim's location can be detected.

As for the sensor type to detect victim location, in our study, we utilize LIDAR and symbolize vital signal detection sensor with a circle which is out of the scope of this study. If the victim is in the direct line of sight of the LIDAR cone, the victim's

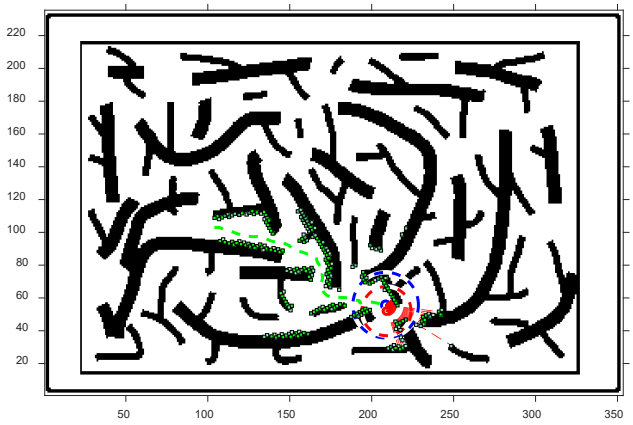
location is attained as the target to penetrate this location. However, which sensor type can we utilize for the robot's detection circle? Apart from the scope of this dissertation, the heat detection sensors can be fused with seismic sensors to detect the movement under rubbles. Also, an infrared camera can be used with LIDAR to locate the victim's location faster. The usage of these sensors can be thought of as enlarging the victim's vital signal detection circle in our study.

To analyze how the size of the detection circle affects percolation path performance, we utilize the different sizes of circles and compare percolation path performance until reaching the location of the victim location. For the circle radius set, we select $\mathcal{R}_{radius} = \{10,20,40\}$. We keep the radius of the vital signal emission fixed with $r_{vital} = 15$. We select our initial conditions as in Experiment 1 in Section 5.2 with one difference which is the victim location selected as $\bar{p}_V^G = [209\ 52]^T$.

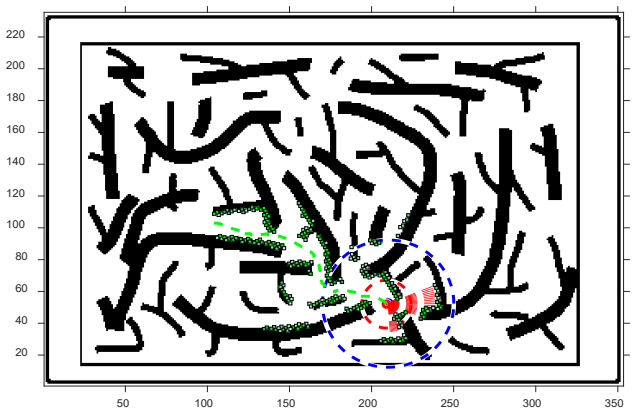
According to the results, in Figure 5.26 (a) $r_{detection}$ is selected as 10. Because the radius is small size, the vital signal can not be properly detected, the two circle does not intersect, and the robot continues to search by getting away from the victim. Eventually, the robot will find the victim's location but this will cause too much time consuming in disaster regions. In Figure 5.26 (b) and Figure 5.26 (c), because the robot senses the victim's signal, it chooses the closest target to the victim's location. Two circles get a chance to intersect because of that $r_{detection}$ is large enough to detect a vital signal in a disaster region.




(a)




(b)




(c)

Figure 5.26. Vital signal detection circle of the robot analysis (a) The $r_{detection} = 10$ is selected in $\mathcal{R}_{radius}$. (b) The $r_{detection} = 20$ is selected in $\mathcal{R}_{radius}$ (c) The $r_{detection} = 40$ is selected in $\mathcal{R}_{radius}$.

## 5.6 Comparison Entropy-Based Active SLAM with Explosive Percolation-Based Active SLAM

We demonstrate the performance of our novel method by comparing it with the entropy-based active SLAM method [51] which is the most cited one. Among the active SLAM algorithms, the entropy-based method is the highly excepted one. The major difference, however, in the literature is that entropy-based SLAM method is utilized to extract the map of the searching area and diminish the map entropy by exploring the unknown regions. In our case, the active SLAM method is aimed to find the target location among the rubbles and extract a safe path to the victim location instead of mapping all regions and decreasing the whole map entropy. To be able to compare the two methods, we modify the entropy-based method in case of finding the victim's location, the process is stopped and announce the location of the victim. The target selection in the entropy-based method is related to finding maximum map entropy going toward to target direction. Therefore, we select the target among frontier cells that has the minimum information in other words maximum map entropy. The utility function to measure map entropy takes the following form:

$$t^* \approx \arg\max_t \left( \underbrace{\mathbb{H}[p(m|x,u,z)]}_{current\ map} - \underbrace{\mathbb{H}[p(m_t|x,u,z)]}_{\substack{predicted\ map \\ at\ target}} \right) \tag{5-3}$$

$$\mathbb{H}[P(m|x,u,z) = \sum_{i,j} -p(m_{ij}) \log\left(p(m_{ij})\right) - \left(1 - p(m_{ij})\right) \log\left(1 - \ p(m_{ij})\right) \tag{5-4}$$

Where $t^*$ is the maximum entropy target location, $m$ is the current occupancy grid map, and $m_t$ is the predicted map at the target location, $m_{ij}$ is the cell occupancy probability in the occupancy map $x$ is the robot states, $u$ is the control input, $z$ is the measurements.

Table 5.3 Initial Conditions for SAR Robot and Target Location on Map.

| | Experiment 1 | Experiment 2 | Experiment 3 | Experiment 4 | Experiment 5 |
|---|---|---|---|---|---|
| $p_R^G$ | $x_R^G = 30$<br>$y_R^G = 201$<br>$\theta_R^G = -60°$ | $x_R^G = 30$<br>$y_R^G = 201$<br>$\theta_R^G = -60°$ | $x_R^G = 30$<br>$y_R^G = 201$<br>$\theta_R^G = -60°$ | $x_R^G = 320$<br>$y_R^G = 210$<br>$\theta_R^G = -120°$ | $x_R^G = 320$<br>$y_R^G = 210$<br>$\theta_R^G = -120°$ |
| $p_T^G$ | $x_T^G = 90$<br>$y_T^G = 136$ | $x_T^G = 145$<br>$y_T^G = 184$ | $x_T^G = 105$<br>$y_T^G = 164$ | $x_T^G = 300$<br>$y_T^G = 70$ | $x_T^G = 250$<br>$y_T^G = 150$ |

To be able to compare the two methods, the map in Chapter 5.1 is utilized and the 5 different locations of the vital signal together with the robot's initial position state is determined as in Table 5.3. For the first three experiments, the robot's initial position is kept the same, and for the last 2 experiments, a different robot's initial position is selected. The search performance is compared based on search path length among rubbles until the location of the victim is found. The results can be seen in Figure 5.27. In this figure, the search path for each experiment is represented with the blue line for the explosive percolation-based SLAM method and the red line for the entropy-based SLAM method. In those experiments, the same FastSLAM algorithm, frontier-based target detection, and path-detection algorithms are used. As a result, we can say that the performance of vital signal detection in explosive percolation search shows a better result in almost all experiment cases with respect to the entropy-based method. The reason for that is even if the finding victim signal condition is implemented entropy-based method, the selection of target criteria is giving priority to diminishing map entropy by selecting a target that has the maximum map entropy. As a result of this action, the robot follows a different path regime until finding the victim's location.

The entropy-based method is susceptible to the same motion error parameter because the same FastSLAM algorithm is used. These motion error values also affect the occupancy map performance directly. Therefore the selection of high

entropy targets is affected by the motion errors significantly by giving priority to diminishing map entropy constantly.

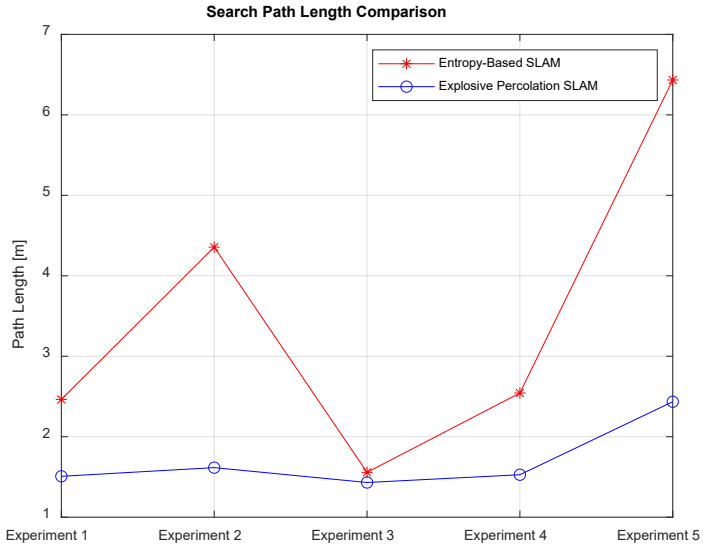
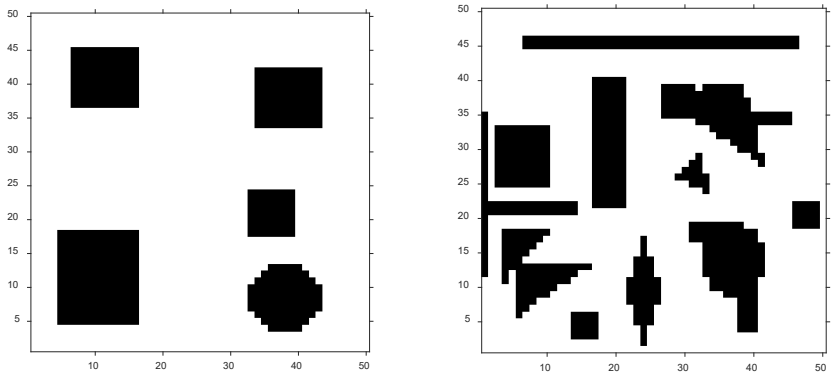



Figure 5.27. Search path length comparison until finding of vital signal location on the global map.

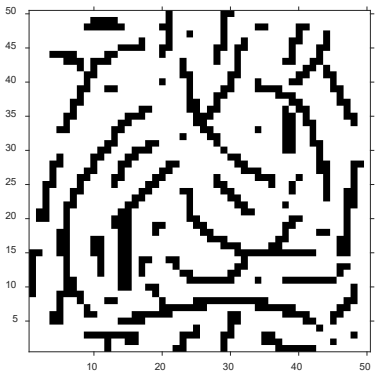
## 5.7 Comparison of Pathfinding Method Performances

Within the nature of the explosive percolation-based SLAM method, we bind the two points in space. In other words, we try to find a safe path between target and source points. In the literature, there are different widely known pathfinding algorithms. These algorithms are compared to each other in terms of their performance characteristics. Therefore, it would be nice to compare our methods with some common widely used ones. We have chosen A* (4-direction), A* (8-direction), rapidly -exploring random trees (RRT) and probabilistic road maps (PRM), and genetic algorithm (GA) [52] to compare with explosive percolation (EP). For A* algorithms Manhattan, Euler Distance, and Chebyshev heuristic functions are selected, however, there was no significant difference observed between these functions, therefore Euler Distance is selected as a heuristic function. The algorithms are compared in terms of expansion cells, and path

lengths. The maps are selected in 50x50 sizes from simple to complex. The maps are given in Figure 5.28.



(1) (2)



(3)

Figure 5.28. Map formats to be used in the comparison of pathfinding algorithms from simple to complex are presented.

The initial point on the maps is selected as $x_i = 1, y_i = 1$ and the target point is selected as $x_t = 50, y_t = 40$. The performance results of the search algorithms are shared in Table 5.4.

Table 5.4 Pathfinding algorithm comparison for each map given in Figure 5.27.

| Maps | Algorithms | *Expansion Cell* | *Path Lengths* | *Time[sec]* |
|---|---|---|---|---|
| Map1 | A* (4-Direction) | 1406 | 88 | 0.26 |
| | A* (8-Direction) | 472 | 70.13 | 0.16 |

Table 5.4 (continued)

| | | | | |
|---|---|---|---|---|
| | RRT | N/A | 82 | 0.01 |
| | PRM | N/A | 71.95 | 0.33 |
| | GA | N/A | 114 | 0.03 |
| | EP | 1165 | 65.83 | 0.7 |
| Map2 | A* (4-Direction) | 1244 | 88 | 0.35 |
| | A* (8-Direction) | 541 | 68.96 | 0.15 |
| | RRT | N/A | 80.02 | 0.02 |
| | PRM | N/A | 75.86 | 0.2 |
| | GA | N/A | 71 | 0.04 |
| | EP | 956 | 59.84 | 0.78 |
| Map3 | A* (4-Direction) | 1456 | 91 | 0.17 |
| | A* (8-Direction) | 1140 | 78.40 | 0.21 |
| | RRT | N/A | 96.92 | 0.07 |
| | PRM | N/A | 105.5 | 0.12 |
| | GA | N/A | 96 | 0.2 |
| | EP | 1206 | 73.92 | 0.86 |

According to Table 5.4, we compare two performance criteria one is expansion cell number and the other one is path length. Because the RRT algorithm counts failed attempts number during search and PRM uses iterative vertex numbers, GA runs with cost function the expansion cell can not be used as performance criteria for those. However, A* algorithm expands the cells in the map to find a path. Therefore expansion cell number is applicable for this method. As we can see from the results, mostly the explosive percolation (EP) method uses fewer cells to find a path with respect to A* (4-direction) method. As for the A* (8-direction), it uses less expansion cell number and comparatively performs better results than the A* (4-direction) method. As we can see from Table 5.4, for map 2 and map 3 EP method and A* (8-direction) algorithms almost show similar path length results. However, A* (8-direction) algorithm uses lower cell expansion than the EP method. Moreover, when

it comes to comparing path length to reach the target point, the EP method shows better results with respect to other algorithms except for A* (8-direction) algorithms. To be able to find the proper solution for PRM, RRT, and GA, we had to optimize the parameter. When it is compared with EP which adapts itself to all complexity, for RRT and PRM we had to increase the node number for PRM and the distance threshold value for RRT and number of population and generation size for GA in a complex map 3. This will cause the update of these parameters in case of facing complex structures during searching with PRM or RRT.

An important deduction from the results is that when we compare the methods according to processing time, the EP method shows slower performance than the other ones. The reason for this result can be related to the implementation of the code. With proper handling of the memory locations and coding structure, the time can be diminished further.

The reason that explosive percolation shows better results in terms of path length is that explosive percolation theory uses lower-dense of clusters to find a percolation path into the target location and explosive percolation is a specialized theory, especially for unstructured and complex environments as in disaster regions where include small size passage between the voids through rubbles, connected irregular shape spaces between debris. Moreover, instead of searching all occupancy grid maps, explosive percolation focuses on creating the largest cluster that connects the robot location to the target location. This sustains the performance of finding the shortest path length among other algorithms in unstructured complex searching areas.

When comparing the performances of other algorithms, another important point that should be mentioned at which conditions algorithm performs better results. As we know from the A* algorithm, it guarantees the shortest path between two points. It is true for coarse graphs and the selection of heuristic functions to foresee the lowest cost function value for the selection of a site. When the map gets complicated, we can see that Explosive percolation gives a lower path length value than the A*

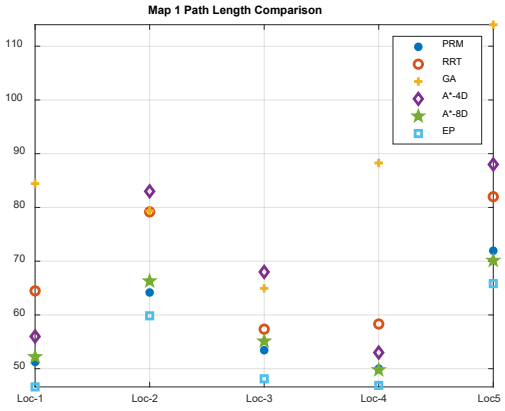
algorithm as in the case in Map 3 in Table 5.4. Moreover, the resolution value selection of the A* algorithm affects finding the shortest path distance. If we use a low-resolution value, it is most likely that we can find a larger distance value instead of finding the shortest path. On the other hand, if we select high resolution, this means we have to expand more cells and this will cause the high computational time to get the shortest distance value. The reader should keep in mind that distinction. For the other algorithms, they used iterative and probabilistic methods, therefore the path length value can be changed from one iteration to another. If one user wants to apply a pathfinding algorithm with shortest path criteria to a highly structured environment EP method and A* method can be used for that purpose, however, if the high-resolution value is set, the performance will be diminished for both algorithms. On the other hand, PRM and RRT methods can provide a non-optimal solution however the computational resources can be used efficiently and the processing time is lower concerning other algorithms. When selecting a method, the performance selection should be determined carefully.

In Table 5.4, we keep the same location for the initial and the target location. To observe more outcomes to compare the methods' performance based on path length, we have created random 5 locations on each map. Here the initial point is preserved with the same location $x_i = 1, y_i = 1$, however, the target location is varied randomly. The random locations are shared in Table 5.5.

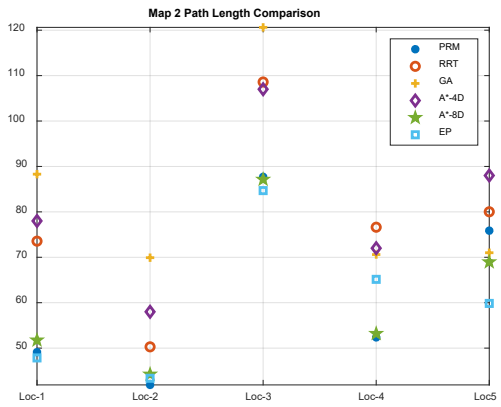
Table 5.5 Target locations for each map in Figure 5.27.

| | **Map 1** | **Map 2** | **Map 3** |
|---|---|---|---|
| | *Target (x,y)* | *Target (x,y)* | *Target (x,y)* |
| Location 1 | (10,48) | (10,40) | (20,42) |
| Location 2 | (40,45) | (30,30) | (25,30) |
| Location 3 | (42,21) | (35,48) | (45,8) |
| Location 4 | (45,10) | (45,13) | (2,45) |
| Location 5 | (50,40) | (50,40) | (50,40) |

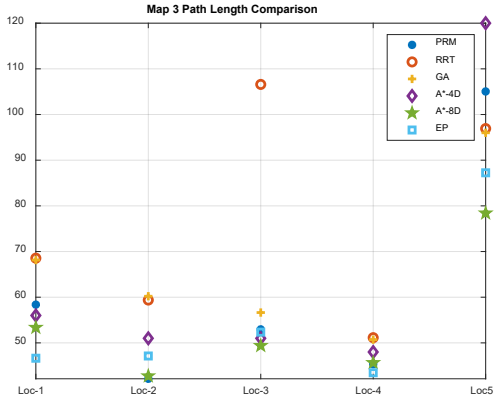
Based on the locations for each map, the selected path algorithms are compared with the explosive percolation method, and the results are shared in Figure 5.29. The selected methods are PRM, RRT, GA, A* (4- Direction), A* (8- Direction) and Explosive Percolation. The blue dot represents PRM, the orange circle represents RRT, the yellow plus represents GA, the purple diamond is for A* (4-Direction), the green star is for A* (8-Direction), and finally blue square is for Explosive percolation. When we analyze the results, we can observe that in almost all cases the Explosive percolation gives better path length results with respect to other methods. Except when it is compared with A* (8-Direction), it gives better performance in terms of path length at some conditions. In Map-2 at Location 4 and Map-3 at Locations 2 and 5, A* (8-Direction) performs a lower path length. For the Location-3 case in Map-2 and Map-3, PRM and EP show close results. For Location 4 in Map-2 and Location 2 in Map 3, PRM exhibits lower values for path length. However, for the other points, the EP method shows better performance, especially if the target location is further from the initial position as in Location -5. These results indicate that the EP method presents relatively better performance results for random target locations as well.




(a)




(b)




(c)

Figure 5.29. Path length comparison of path-finding algorithms PRM, RRT, GA, A*, and Explosive Percolation for each location is given in Table 5.5 for the maps in Figure 5.27 (a) For Map 1 in Figure 5.27 (b) For Map 2 in Figure 5.27 (c) For Map 3 in Figure 5.27

# CHAPTER 6

# CONCLUSION AND FUTURE WORK

In this thesis, a novel explosive percolation-based active SLAM algorithm is developed for the unstructured disaster environment to detect victim location within debris The SAR robot utilizes the FastSLAM algorithm with the occupancy grid map of the environment during searching and detects the target points to be searched with the frontier based method. To reach these target points among unstructured and highly complex environments explosive percolation is used and by following the percolation path which connects the robot's location to the target location unknown regions of the map are explored to detect the vital signal of the victim.

In the presented approach, contrary to active SLAM approaches that aim to map an entire search area in the literature, our main aim is to find the location of the victim among rubbles by using the voids in the disaster area. The proposed explosive percolation search method guides the robot by utilizing the voids among rubbles to obtain a safe and obstacle-free path toward to unknown region of a highly complex fractional-order disaster environment.

The disaster regions have the possibility of collapsing of debris, therefore dead-ends can occur and environments can change during mapping. In the proposed method, with the help of the explosive percolation path, the successive dead-end forms are detected and avoided during searching. Therefore an uninterrupted search and rescue navigation can be sustained.

The proposed method is tested within a simulation environment with different scenarios in an unstructured map. Also, the performance of the proposed method is compared with other active SLAM approaches. Simulation results show us the proposed method is more effective and performs better than compared active SLAM approach.

We can list the main contribution of the thesis as follows,

- A novel path-search algorithm based on Explosive Percolation theory in the FastSLAM algorithm is presented for a highly unstructured environment.
- An active search algorithm to detect victims' location in the disaster area is developed and tested with Explosive Percolation enhanced FastSLAM even under changing environments due to collapses.
- Achievement of handling successive dead-ends and continued search of new areas in a highly complex environment of fractional order by percolating.
- The first implementation and testing of explosive percolation theory in the robotic field literature

An important conclusion of the proposed method is that based on the simulation results with different scenarios and conditions, like the SLAM applications for indoor and outdoor usage the exploration of unstructured complex environments in disaster regions can be out of a problematic case for further SAR missions with the application of our novel method. With the proposed dissertation, we have brought in a new solution for confined harsh environment exploration for robotic SAR missions which have very limited research on it. Moreover, with the application of explosive percolation theory, not only the enhancement of SLAM for harsh conditions is sustained but also a new application area of explosive percolation is achieved for further engineering problems.

In this thesis, our novel proposed method is demonstrated within the simulation environment. However, to observe the real performance, with real robotic equipment and sensors, the algorithm should be tested in a controlled unstructured environment. This is envisaged as part of future work.

In this dissertation, explosive percolation is applied in 2D environments, however, it can be extended also to 3D environments. When the nature of a disaster region is taken into account, the rubbles create 3D irregular volumes, by dividing this volume into voxel grids and applying explosive percolation theory, we can detect our percolation path into an unknown space through the empty volume between debris.

In this way, we can detect our 3D explosive percolation path through holes and cavities that the SAR robot can fit into it.

Another challenging part is the usage of the proposed algorithm with multi-robot architecture. Our methodology is scaleable to multi-robot SLAM and using the multi-robot SLAM approach can unveil the true nature of explosive percolation theory in an unstructured environment.

Lastly, 3D map generation and merging of those maps in a multi-robot SLAM approach is another challenging research area. In our approach we demonstrate the results in a 2D environment, however, a 3D view of the rubbles can sustain more detailed information for the search and rescue team in a disaster area.

## REFERENCES


[1] R. R. Murphy et al., “Search and rescue robotics,” in Springer Handbook of Robotics, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1151–1173.

[2] K. Berns, A. Nezhadfard, M. Tosa, H. Balta, and G. D. Cubber, “Unmanned ground robots for rescue tasks,” in Search and Rescue Robotics - From Theory to Practice, InTech, 2017.

[3] H. Surmann et al., “Integration of UAVs in urban search and rescue missions,” in 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2019.

[4] A. Matos et al., “Unmanned maritime systems for search and rescue,” in Search and Rescue Robotics - From Theory to Practice, InTech, 2017.

[5] J. Gonzalez-Gomez, J. Gonzalez-Quijano, H. Zhang, and M. Abderrahim, “Toward the sense of touch in snake modular robots for search and rescue operations,” Proc. ICRA 2010 Workshop on Modular Robots: State of the Art, vol. 01, 2010.

[6] D. Falanga, K. Kleber, S. Mintchev, D. Floreano, and D. Scaramuzza, “The foldable drone: A morphing quadrotor that can squeeze and fly,” IEEE Robot. Autom. Lett., vol. 4, no. 2, pp. 209–216, 2019.

[7] G. De Cubber et al., “Search and Rescue robots developed by the European ICARUS project,” 10 2013.

[8] M. Tranzatto et al., “CERBERUS in the DARPA Subterranean Challenge,” Sci. Robot., vol. 7, no. 66, p. eabp9742, 2022.

[9] D. Misaki and Y. Murakami, “Development of a multi-leg type micro rescue robot for disaster victim search,” in 2011 IEEE International Conference on Robotics and Biomimetics, 2011.

[10] M. Hutson, “Searching for survivors of the Mexico earthquake—with snake robots,” Science, 10 2017.

[11] J. P. Queralta et al., “Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision,” IEEE Access, vol. 8, pp. 191617–191643, 2020.

[12] A. Ollero et al., “Multiple eyes in the skies - Architecture and perception issues in the comets unmanned air vehicles project,” IEEE Robot. Autom. Mag., vol. 12, no. 2, pp. 46–57, 2005.

[13] M. A. Neerincx, J. van Diggelen, and L. van Breda, “Interaction design patterns for adaptive human-agent-robot teamwork in high-risk domains,” in Engineering Psychology and Cognitive Ergonomics, Cham: Springer International Publishing, 2016, pp. 211–220.

[14] J. de Greeff, T. Mioch, W. van Vught, K. Hindriks, M. A. Neerincx, and I. Kruijff-Korbayová, “Persistent robot-assisted disaster response,” in Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, 2018.

[15] K. Ebadi et al., “Present and future of SLAM in extreme underground environments,” arXiv [cs.RO], 2022.
[16] A. Agha et al., “NeBula: Quest for robotic autonomy in challenging environments; TEAM CoSTAR at the DARPA Subterranean Challenge,” arXiv [cs.RO], 2021.
[17] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part I,” IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99–110, 2006.
[18] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. London, England: MIT Press, 2005.
[19] FastSLAM. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.
[20] M. Filipenko and I. Afanasyev, “Comparison of various SLAM systems for mobile robot in an indoor environment,” in 2018 International Conference on Intelligent Systems (IS), 2018.
[21] B. Garigipati, N. Strokina, and R. Ghabcheloo, “Evaluation and comparison of eight popular Lidar and Visual SLAM algorithms,” arXiv [cs.RO], 2022.
[22] T. H. Chan, H. Hesse, and S. G. Ho, “LiDAR-based 3D SLAM for indoor mapping,” in 2021 7th International Conference on Control, Automation and Robotics (ICCAR), 2021.
[23] H. Lim, J. Lim, and H. J. Kim, “Real-time 6-DOF monocular visual SLAM in a large-scale environment,” in 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014.
[24] D. Tardioli et al., “A robotized dumper for debris removal in tunnels under construction,” in ROBOT 2017: Third Iberian Robotics Conference, Cham: Springer International Publishing, 2018, pp. 126–139.
[25] M. Petrlik, T. Baca, D. Hert, M. Vrba, T. Krajnik, and M. Saska, “A robust UAV system for operations in a constrained environment,” IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 2169–2176, 2020.
[26] C. Cadena et al., “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” IEEE Trans. Robot., vol. 32, no. 6, pp. 1309–1332, 2016.
[27] B. Ristic and J. L. Palmer, “Autonomous exploration and mapping with RFS occupancy-grid SLAM,” Entropy (Basel), vol. 20, no. 6, p. 456, 2018.
[28] J.-L. Blanco, J.-A. Fernandez-Madrigal, and J. Gonzalez, “An entropy-based measurement of certainty in Rao-blackwellized particle filter mapping,” in 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006.
[29] L. Carlone, J. Du, M. Kaouk Ng, B. Bona, and M. Indri, “Active SLAM and exploration with particle filters using Kullback-Leibler divergence,” J. Intell. Robot. Syst., vol. 75, no. 2, pp. 291–311, 2014.
[30] A. Hunt, R. Ewing, and B. Ghanbarian, Percolation theory for flow in porous media. Cham, Switzerland: Springer International Publishing, 2014.

[31] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts, “Network robustness and fragility: Percolation on random graphs,” in The Structure and Dynamics of Networks, Princeton: Princeton University Press, 2011, pp. 510–513.
[32] V. S. Dotsenko, M. Picco, P. Windey, G. Harris, E. Martinec, and E. Marinari, “Self-avoiding surfaces in the 3d Ising model,” Nuclear Physics B, vol. 448, no. 3, pp. 577–620, 1995.
[33] C. Grimaldi and I. Balberg, “Tunneling and non-universality in continuum percolation systems,” arXiv [cond-mat.dis-nn], 2006.
[34] T. Beer and I. G. Enting, “Fire spread and percolation modelling,” Mathematical and Computer Modelling, vol. 13, no. 11, pp. 77–96, 1990.
[35] C. A. Browne, D. B. Amchin, J. Schneider, and S. S. Datta, “Infection percolation: A dynamic network model of disease spreading,” Front. Phys., vol. 9, 2021.
[36] B. Ghanbarian, H. Ebrahimian, A. G. Hunt, and M. T. van Genuchten, “Theoretical bounds for the exponent in the empirical power-law advance-time curve for surface flow,” Agric. Water Manag., vol. 210, pp. 208–216, 2018.
[37] D. Achlioptas, R. M. D’Souza, and J. Spencer, “Explosive percolation in random networks,” Science, vol. 323, no. 5920, pp. 1453–1455, 2009.
[38] R. M. D’Souza and J. Nagler, “Explosive Percolation: Novel critical and supercritical phenomena,” arXiv [cond-mat.dis-nn], 2015.
[39] H. D. Rozenfeld, L. K. Gallos, and H. A. Makse, “Explosive percolation in the human protein homology network,” Eur. Phys. J. B, vol. 75, no. 3, pp. 305–310, 2010.
[40] Y. S. Cho and B. Kahng, “Discontinuous percolation transitions in real physical systems,” Phys. Rev. E Stat. Nonlin. Soft Matter Phys., vol. 84, no. 5 Pt 1, p. 050102, 2011.
[41] S. Topal, I. Erkmen, and A. M. Erkmen, “Percolation enhanced prioritised multi-robot exploration of unstructured environments,” Int. J. Reason.-based Intell. Syst., vol. 2, no. 3/4, p. 217, 2010.
[42] M. Karahan, A. M. Erkmen, and I. Erkmen, “Prioritized mobile robot exploration based on percolation enhanced entropy based fast SLAM,” J. Intell. Robot. Syst., vol. 75, no. 3–4, pp. 541–567, 2014.
[43] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem,” 11 2002.
[44] N. Bastas, K. Kosmidis, and P. Argyrakis, “Explosive site percolation and finite-size hysteresis,” Phys. Rev. E Stat. Nonlin. Soft Matter Phys., vol. 84, no. 6 Pt 2, p. 066112, 2011.
[45] C. Perot, “Phase Transition Continuity Dependence on Edge Evaluation and Acceptance,” Universität Leipzig, 2019.

[46] M. Sarfraz and Z. N. K. Swati, “Mining corner points on the generic shapes,” Open J. Appl. Sci., vol. 03, no. 01, pp. 10–15, 2013.
[47] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA’97. “Towards New Computational Principles for Robotics and Automation,” 2002.
[48] D. Stauffer and A. Aharony, Introduction to percolation theory: Second edition. Philadelphia, PA: Taylor & Francis, 2018.
[49] Z. Duraklı and V. Nabiyev, “A new approach based on Bezier curves to solve path planning problems for mobile robots,” J. Comput. Sci., vol. 58, no. 101540, p. 101540, 2022.
[50] J. Płaskonka, “Different kinematic path following controllers for a wheeled mobile robot of (2,0) type,” J. Intell. Robot. Syst., vol. 77, no. 3–4, pp. 481–498, 2015.
[51] H. Carrillo, P. Dames, V. Kumar, and J. A. Castellanos, “Autonomous robotic exploration using occupancy grid maps and graph SLAM based on Shannon and Rényi Entropy,” in 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015.
[52] R. Kala, “Code for Robot Path Planning using Probabilistic Roadmap,” Indian Institute of Information Technology, Allahabad, Jun. 2014.

# APPENDICES

## CURRICULUM VITAE

Surname, Name: Yıldız, Doğan

### EDUCATION

| Degree | Institution | Year of Graduation |
|---|---|---|
| PhD | METU Electrical and Electronics Engineering | 2023 |
| MS | METU Mechanical Engineering | 2016 |
| BS | METU Mechanical Engineering | 2013 |
| High School | Hasan Ali Yücel Anatolian High School, Ankara | 2008 |

### FOREIGN LANGUAGES

Advanced English, Intermediate German

### PROFESSIONAL EXPERIENCE

| Year | Place | Position |
|---|---|---|
| 2013-2019 | Turkish Aerospace | Flight Mechanics and Control Design Engineer |
| 2019-2020 | Havelsan | Expert Control and Modelling Design Engineer |
| 2020-2023 | TR Airworthiness - TAI | Expert Certification Engineer |
| 2023- | Wingcopter | Senior Flight Control Software Engineer |