

# Path Planning and Navigation of Robot with 2D Lidar Feedback



Author

MUHAMMAD AWAIS KHAN NIAZI

Regn Number

00000276308

Supervisor:

DR. KUNWAR FARAZ AHMED KHAN

DEPARTMENT OF MECHATRONICS ENGINEERING  
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING  
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,  
ISLAMABAD

July, 2022

# Path Planning and Navigation of Robot with 2D Lidar Feedback

Author

MUHAMMAD AWAIS KHAN NIAZI

Regn Number

00000276308

A thesis submitted in partial fulfillment of the requirements for the degree of

MS Mechatronics Engineering

Thesis Supervisor:

DR. KUNWAR FARAZ AHMED KHAN

Thesis Supervisor's Signature: \_\_\_\_\_

DEPARTMENT OF MECHATRONICS ENGINEERING  
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING  
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,  
ISLAMABAD

July, 2022

## **Declaration**

I certify that this research work titled “*Path Planning and Navigation of Robot with 2D Lidar Feedback*” is my own work. The work has not been presented elsewhere for assessment. I have properly acknowledged / referred the material that has been used from other sources.

Signature of Student

MUHAMMAD AWAIS KHAN NIAZI

00000276308

## **Language Correctness Certificate**

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Signature of Student

MUHAMMAD AWAIS KHAN NIAZI

00000276308

Signature of Supervisor

## **Copyright Statement**

Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the author and lodged in the Library of NUST College of E&ME. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) may not be made without the permission (in writing) of the author.

The ownership of any intellectual property rights which may be described in this thesis is vested in NUST College of E&ME, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the College of E&ME, which will prescribe the terms and conditions of any such agreement.

Further information on the conditions under which disclosures and exploitation may take place is available from the Library of NUST College of E&ME, Rawalpindi.

## Acknowledgments

I am thankful to **ALLAH ALMIGHTY** for showing me the right direction throughout this thesis. I also thank NUST, for financial support for my MS degree.

I would also like to express special thanks to my supervisor *Dr. Kunwar Faraz Ahmed Khan* for his help throughout my thesis. It was a wonderful experience to work under his guidance. His absolute grip on the subject helped me a lot during the research.

I would also like to pay special thanks to my GEC members *Dr. Umar Shahbaz Khan* and *Dr. Uzair Khaleeq uz Zaman* for their tremendous support and cooperation. I appreciate their patience and guidance throughout the whole thesis.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

*I dedicate this work to my adored parents for nursing me throughout my life with affections and love, my beloved siblings who always inspired me to perform better, and my wife and my kids Manahil and Usman for showing immense patience as it would not have been possible to do this research work without their support.*

## Abstract

Mobile robots are becoming increasingly popular in modern-day robotics. Mobile robots are being used in many fields like exploration, defence, agriculture, cleaning, lawn mowing, and warehouse management to name a few. The operation of a mobile robot requires accurate data for mapping, path planning, and navigation. Each subsequent operation is dependent on the previous operation so the margin for error is very little. In this work, we develop a complete framework for mapping, complete coverage path planning (CCPP), and navigation along with local planning for handling the complexities of dynamic environments. Some improvements in already existing literature have been proposed like the most popular Boustrophedon Cellular Decomposition (BCD) to decrease cost and increase efficiency in terms of time and energy of the system. This work also proposes a memory-efficient approach for solving local planning problems through the velocity obstacle (VO) method.

Experimental work was carried out in simulated environment of Robot Operating System (ROS). Global path planning results showed significant improvement in terms of minimizing overlapping issues when compared with the original BCD algorithm. Navigation results showed that all waypoints generated by global planner were visited with 94% accuracy. The velocity obstacle approach was implemented as local path planner to handle moving obstacles. To further improve the performance, a two-level approach was used to handle moving obstacles.

**Keywords:** Complete Coverage Path Planning, Boustrophedon Cellular Decomposition, Velocity Obstacles, Local Path Planning, Mobile Robot



# Table of Contents

<b>Declaration</b>	<b>i</b>
<b>Language Correctness Certificate</b>	<b>ii</b>
<b>Copyright Statement</b>	<b>iii</b>
<b>Acknowledgments</b>	<b>iv</b>
<b>Abstract</b>	<b>vi</b>
<b>Table of Contents</b>	<b>vii</b>
<b>List of Figure</b>	<b>ix</b>
<b>List of Table</b>	<b>x</b>
<b>Acronyms</b>	<b>xi</b>
<b>CHAPTER 1: INTRODUCTION</b>	<b>1</b>
<b>1.1 Mapping</b> .....	<b>1</b>
<b>1.2 Global Path Planning</b> .....	<b>1</b>
<b>1.3 Navigation</b> .....	<b>1</b>
<b>1.4 Local Path Planning</b> .....	<b>2</b>
<b>1.5 Robot Operating System (ROS)</b> .....	<b>2</b>
<b>1.6 Problem Statement</b> .....	<b>2</b>
<b>1.7 Objectives</b> .....	<b>3</b>
<b>CHAPTER 2: LITERATURE REVIEW</b>	<b>4</b>
<b>2.1 Heuristic and randomized approaches</b> .....	<b>5</b>
<b>2.2 Cellular decompositions</b> .....	<b>6</b>
2.2.1 Approximate cellular decomposition (ACD)	7
2.2.2 Semi-Approximate Cellular Decomposition (SCD)	8
2.2.3 Exact cellular decomposition	8
<b>2.3 Other approaches</b> .....	<b>11</b>
2.3.1 Artificial potential fields	11
2.3.2 Neural networks and fuzzy logic	11
<b>2.4 Summary</b> .....	<b>12</b>
<b>CHAPTER 3: METHODOLOGY AND EXPERIMENTAL SETUP</b>	<b>17</b>
<b>3.1 Navigation of Robot for Unmapped Environment</b> .....	<b>17</b>
<b>3.2 Mapping of the Environment</b> .....	<b>18</b>
<b>3.3 Global Path Planning</b> .....	<b>18</b>
<b>3.4 Navigation of Robot for Mapped Environment</b> .....	<b>19</b>
<b>3.5 Local Path Planning</b> .....	<b>19</b>
<b>CHAPTER 4: MAPPING AND GLOBAL PATH PLANNING</b>	<b>20</b>
<b>4.1 Mapping and Image Processing</b> .....	<b>20</b>
<b>4.2 Global Path Planning</b> .....	<b>22</b>
<b>4.3 Removal of redundant points from Route array</b> .....	<b>23</b>

<b>4.4</b>	<b>Modified BCD Approach for Global Planner .....</b>	<b>25</b>
<b>CHAPTER 5: NAVIGATION AND LOCAL PATH PLANNING</b>		<b>28</b>
<b>5.1</b>	<b>Navigation .....</b>	<b>28</b>
<b>5.2</b>	<b>Local Path Planning .....</b>	<b>29</b>
5.2.1	Intersection of Lines Approach (Level 1)	30
5.2.2	Velocity Obstacle Approach (Level 2)	34
<b>5.3</b>	<b>Integrating Local Path Planning with Navigation and Global Path .....</b>	<b>40</b>
<b>CHAPTER 6: RESULTS AND DISCUSSION</b>		<b>42</b>
<b>6.1</b>	<b>Simulation Environment .....</b>	<b>42</b>
<b>6.2</b>	<b>Environment Mapping Results .....</b>	<b>43</b>
<b>6.3</b>	<b>Global Path Planning Results .....</b>	<b>44</b>
<b>6.4</b>	<b>Local Path Planning and Navigation .....</b>	<b>44</b>
<b>CHAPTER 7: CONCLUSION AND FUTURE WORK</b>		<b>46</b>
<b>References</b>		<b>47</b>

## List of Figure

<b>Figure 2.1:</b> Randomized approach to coverage path planning.....	6
<b>Figure 2.2:</b> (a) Free space with obstacles, (b) ACD of free space defined in (a), (c) SCD and (d) Exact cellular decomposition .....	7
<b>Figure 2.3:</b> Given polygonal work area. ....	9
<b>Figure 2.4:</b> Trapezoidal decomposition for the work area above. ....	9
<b>Figure 2.5:</b> Adjacency graph for the work area in <b>Figure 2.3</b> .....	10
<b>Figure 2.5:</b> With fewer cells shorter paths are obtained. ....	10
<b>Figure 2.6:</b> A boustrophedon path is composed of back-and-forth motions. ....	10
<b>Figure 2.7:</b> Trapezoidal decomposition (left) and boustrophedon decomposition (right) for the same space. Each cell can be covered with simple back-and-forth motions in both cases. Path is shorter in BCD. [18].....	11
<b>Figure 3.1:</b> Experimental Setup Process .....	17
<b>Figure 3.2:</b> Mapped environment in ROS RViz .....	18
<b>Figure 4.1:</b> Map of environment generated by the SLAM Gmapping package .....	20
<b>Figure 4.2:</b> Cropped image after processing the useful map area.....	21
<b>Figure 4.3:</b> Enhanced boundaries of given map after applying morphological operators.....	22
<b>Figure 4.4:</b> Decomposed cells after applying BCD algorithm.....	22
<b>Figure 4.5:</b> Route list containing index of each grid in ascending order .....	23
<b>Figure 4.6:</b> Route list after removing unnecessary members.....	24
<b>Figure 4.7:</b> Flow Chart Diagram of global path planner.....	26
<b>Figure 4.8:</b> Cost of Modified BCD is lesser as compared to normal BCD.....	27
<b>Figure 5.1:</b> Navigation with all waypoints (a, b, c) and reduced waypoints (d).....	29
<b>Figure 5.2:</b> Positional and velocity vectors of Robot (R) and moving object (O).....	30
<b>Figure 5.3:</b> Calculation of intersection point of robot and moving obstacle .....	31
<b>Figure 5.4:</b> Robot A and Obstacle B moving with velocities $\mathbf{v}_A$ and $\mathbf{v}_B$ respectively [46].....	34
<b>Figure 5.5:</b> Relative Velocity $v_{AB}$ and collision cone $CCAB$ [46].....	35
<b>Figure 5.6:</b> $CCAB$ is translated by $\mathbf{v}_B$ to obtain $VOB$ [46] .....	36
<b>Figure 5.7:</b> Velocity obstacles $VOB_1$ and $VOB_2$ .....	37
<b>Figure 5.8:</b> Velocity obstacle $VOB$ for short time horizon .....	37
<b>Figure 5.9:</b> Trajectory calculation for robot to reach goal avoiding moving obstacle.....	38
<b>Figure 5.10:</b> Trajectory calculated for single robot velocity obstacle .....	39
<b>Figure 5.11:</b> Velocity obstacle in multi obstacle scenario .....	40
<b>Figure 5.12:</b> Flowchart of Local Path Planner .....	41
<b>Figure 6.1:</b> Simulation environment in ROS Gazebo .....	42
<b>Figure 6.2:</b> Total distance covered reduced by modified BCD approach.....	44
<b>Figure 6.3:</b> Performance comparison between two-level and single level VO approach.....	45

## List of Table

<b>Table 2.1:</b> Summary of the analyzed CCPP methods with Approximate cellular decomposition (ACD) and SCD.....	13
<b>Table 2.2:</b> Summary of the analyzed CCPP methods with exact cellular decomposition (ECD).....	13
<b>Table 2.3:</b> Summary of the analyzed CCPP methods which don't use cellular decomposition, NN or Fuzzy Logic.....	14
<b>Table 2.4:</b> Summary of the analyzed CCPP methods which use neural networks and fuzzy logic approaches.....	15
<b>Table 2.5:</b> Summary of the analyzed CCPP methods which use miscellaneous approaches.....	16
<b>Table 6.1:</b> Comparison of Mapping techniques.....	43

## Acronyms

<b>ACD</b>	Approximate Cellular Decomposition
<b>APF</b>	Artificial Potential Fields
<b>BCD</b>	Boustrophedon Cellular Decomposition
<b>CC</b>	Collision Cone
<b>CCPP</b>	Complete Coverage Path Planning
<b>ECD</b>	Exact Cellular Decomposition
<b>GA</b>	Genetic algorithm
<b>GPP</b>	Global Path Planning
<b>GPS</b>	Global Positioning System
<b>IMU</b>	Inertial Measurement Unit
<b>LiDAR</b>	Light Detection and Ranging
<b>LPP</b>	Local Path Planning
<b>MCD</b>	Morse cell decomposition
<b>NN</b>	Neural network
<b>N.A</b>	Not Available
<b>PC</b>	Partially Complete
<b>PRA</b>	Predictive Recursive Algorithm
<b>ROS</b>	Robot Operating System
<b>SCD</b>	Semi-Approximate Cellular Decomposition
<b>SLAM</b>	Simultaneous localization and mapping
<b>SMA</b>	Split-and-Merge algorithm
<b>SSA</b>	Seed Spreader Algorithm
<b>TBM</b>	Template-Based Models.
<b>TCD</b>	Trapezoidal Cellular Decomposition
<b>VO</b>	Velocity Obstacles



# CHAPTER 1: INTRODUCTION

Path planning is generally categorized into two sub-domains – point to point path planning and coverage path planning. Main aim in former is to move robot from point A to point B irrespective of the area it passes through – as long as it's optimal. However, in some cases like agriculture, moving from one place to the other may not be the only requirement. Such applications require the robot to cover complete available space in order to perform certain task.

In this research we will primarily focus on planning a coverage path for a given map (global plan) and navigate the robot accordingly. In order to avoid moving objects we introduce a two-level approach in order to simplify computational complexity while guaranteeing system efficiency.

## 1.1 Mapping

First part of this research is to use LiDAR sensor data and odometry to map the environment. This step is very important as it effects our next steps of local and global path planning. Mapped environment is represented in a binary graphical form where objects are represented as “ones” and open spaces as “zeros”.

## 1.2 Global Path Planning

The data from map is used to plan global path – meaning environment is considered as static till this point. Different approaches were considered to determine the global path which will be discussed in later sections.

## 1.3 Navigation

Once global path is determined, the next step is to navigate the robot according to the points as identified by the global path. The global path is a set of coordinates – each corresponding to a point that the robot must visit in order to cover the entire available area. The robot makes use of the LiDAR sensor and odometry to determine its current location. Once the LiDAR sensor encounters an “un-reported” object, it calls the local path planning function.

This part is very important for dynamic environments. Such situations can arise due to moving objects or static objects that were not in place at the time of mapping. Conditions like these must be handled in a systematic way in order for a mobile robot to work properly.

#### **1.4 Local Path Planning**

A local path planner is a function which handles the dynamic perspective of a mapped environment. The global path planner generates a set of points that the robot can visit as per Euclidean path. However, due to the dynamic nature of the environment, the path between two points or the point itself can be occupied. Such situations are handled using an intelligent algorithm that lets the robot decide the course of action on-the-run. The major part of this research is focused on the dynamic element of the environment and will be discussed in later sections.

#### **1.5 Robot Operating System (ROS)**

The experimental work of this thesis was carried out in Robot Operating System (ROS). ROS is a set of software and libraries and tools that helps in building robotic applications. It comes with powerful developer tools to be used as robot core operating system. It also comes with powerful simulation tools with realistic physical conditions to mimic the real-world scenarios. The programming language used to carry out the experimental work of this thesis is Python. Some external libraries based on C++ were also used.

#### **1.6 Problem Statement**

During robot navigation we often encounter situations where an object moving in certain direction with certain velocity is likely to collide with the robot. Such collisions can be avoided by early prediction of positions based on positional and velocity vectors. This research is aimed to suggest optimized path based on early prediction of the moving objects using combination of conventional and velocity obstacle approach. Furthermore, we suggest improvements in already existing popular cell decomposition method for global path planning.



## 1.7 Objectives

The objectives of this research work are as below:

- i. To map an unknown environment.
- ii. Generate a collision-free path that covers the whole environment with minimum possible overlaps.
- iii. Navigate the robot in the environment according to the generated path.
- iv. Generate local path in case any obstacle (moving or static) is encountered that was not present in the global map – and navigate the robot accordingly.

## CHAPTER 2: LITERATURE REVIEW

In this chapter, we discuss coverage path planning studies found in the literature. We first focus on different kind of approaches used in past studies for coverage planning problem. In the next section we will take a look at methods used in previous studies to avoid moving objects. Last, we discuss the surveyed methods in Section 2.4.

Coverage path planning is an integral part of several robotic applications, such as vacuum cleaning robots [1], agricultural robots [2], painter robots [3], demining robots [4], lawn mowers [5], disinfecting robots [6], and window cleaners [7]. The research interest in mobile robotics (indoors and outdoors) has clearly motivated the research of coverage path planning. According to Zuo Llang Cao et al. [8] a mobile robot should fulfil following criteria for complete coverage operation:

1. Robot needs to cover complete given area.
2. Robot must not overlap previously visited paths and should fill the region in a smooth manner.
3. Sequential operation is required which should not be repetitive.
4. Obstacles must be avoided.
5. Complicated motions should be disintegrated into simpler sub-paths.
6. An “optimal” path is desired under available conditions.

It was further noted that it is not possible to fulfil all mentioned criteria for an environment that is complicated in nature.

Coverage path planning problem is related to the covering salesman problem, which itself is a variant of the traveling salesman problem. An agent is supposed to visit a neighborhood of each city instead of visiting each city. This minimizes the travelling distance for the agent [9]. With increasing dimension, the complexity of the problem increases resulting in drastic increase of computation time.

Depending upon the completeness of coverage, algorithms can be classified as complete or heuristic. Furthermore, these algorithms can be classified into Offline or Online coverage algorithms. An offline algorithm makes use of prior environment knowledge like basic map and stationary objects. In real world, an offline algorithm

works well only until everything goes according to the prior information. However, small changes in the environment and moving objects require to have a more intelligent algorithm which is termed as an “Online Algorithm”. Online algorithms make use of sensors in real-time. Therefore, such algorithms can also be called sensor-based coverage algorithms.

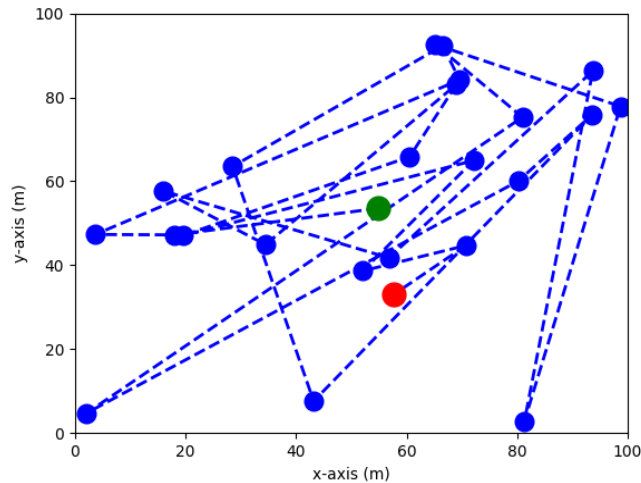
Many studies on coverage path planning can be found in the literature. Also, some authors have surveyed those studies in the past. In his survey, Choset [10] concluded that most complete coverage algorithms used an exact cellular decomposition, either explicitly or implicitly, to achieve coverage. Thus, he organized the coverage algorithms into four categories: heuristic and approximate, partial-approximate and exact cellular decompositions.

C. S. Tan et al. [11] discussed many approaches used until recent years. These cover classical approaches like Random Walk, Artificial Potential Field, Sampling based and search-based methods like A star, D star and other such variants. The review article also discusses modern methods like Genetic Evolution, Swarm Intelligence, Neural Networks and Reinforcement Learning and Deep Learning.

Closely reviewing different methods used in the past, the analyzed methods classified as: heuristic and randomized approaches (section 2.1), cellular decompositions (2.2), other approaches (2.3). Finally, we focus on online path planning techniques for avoiding obstacles (2.4) as primary focus of this thesis is on novel approach based on proportional navigation-based method for avoidance of moving objects.

## **2.1 Heuristic and randomized approaches**

In order to solve coverage problem, randomized approaches are among the few approaches that provide almost complete coverage. Though such approaches are poor in optimization but they are largely used in applications like vacuum cleaning. Figure 2.1 shows a sample random coverage path.



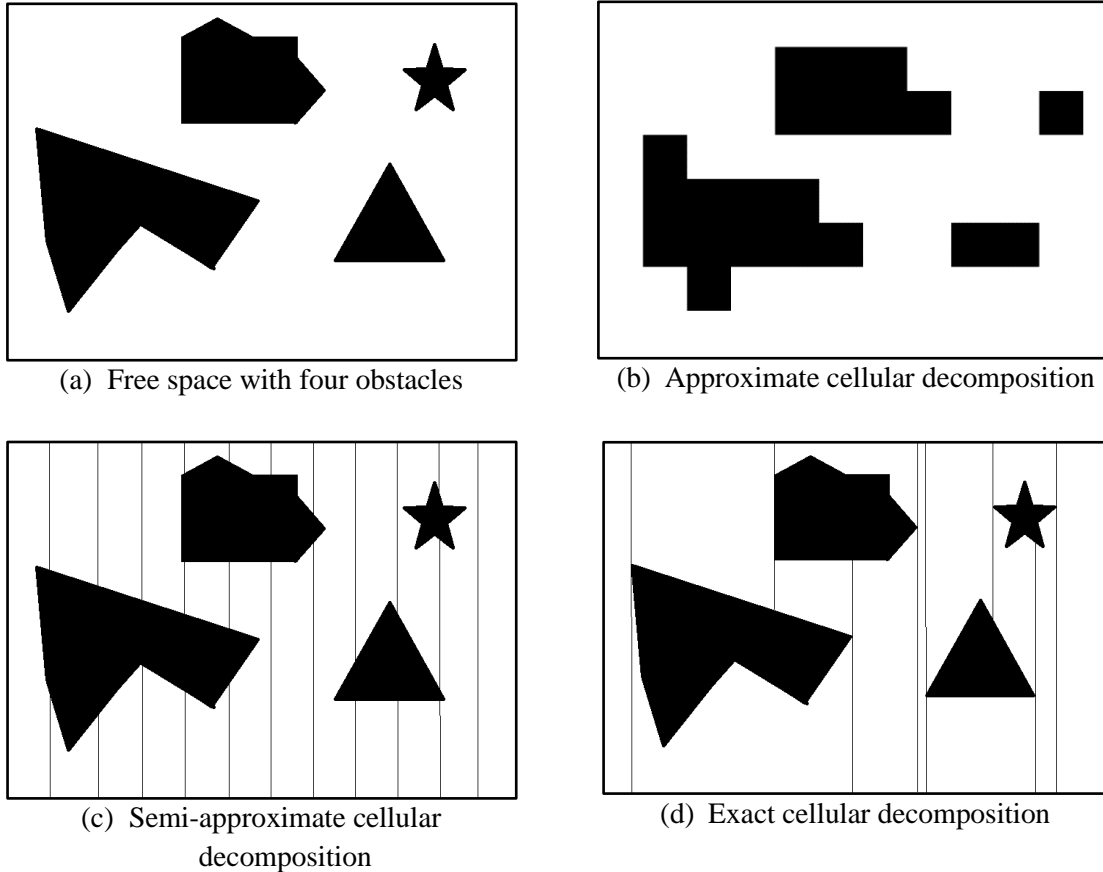
**Figure 2.1:** Randomized approach to coverage path planning.

Randomized approach has some advantages [10]. Such approach does not require expensive sensors like cameras RGB-D or others. Only sensor required is to detect edges in order to avoid obstacles. Additionally, random approach is based on a very basic algorithm that does not require computational complexity. However, randomized approach is very much in-efficient in terms of energy and time consumption. Moreover, when coverage area is large enough, time required to solve complete coverage problem tends to infinity making system practically useless.

## 2.2 Cellular decompositions

In cellular decomposition, the area to be covered is ‘decomposed’ into smaller and simpler sub-divisions that can be covered using simple motions like back-and-forth or spiral. This guarantees complete coverage as the robot covers each sub-divided region one after the other. A common boundary between two cells means they are adjacent. When cells are represented in a hierarchical form in terms of adjacency, it is called as adjacency graph.

Cellular decomposition algorithms have been classified into: approximate, semi-approximate, and exact cellular decompositions [12]. See figure 2.2.



**Figure 2.2:** (a) Free space with obstacles, (b) ACD of free space defined in (a), (c) SCD and (d) Exact cellular decomposition

### 2.2.1 Approximate cellular decomposition (ACD)

In approximate cellular decomposition (ACD), the map is sub-divided into same-sized approximated occupancy grids. Algorithm is then applied to each grid. Since the cells are of same size and shape, and each grid is an approximated depiction of the actual space the resultant is prone to errors [13].

The first ACD model was proposed by Hans P. Moravec and Alberto Elfes [14]. In this model cells were decomposed into grids of same size and shape.

Zielinsky [15] came up with a different approach where a specific number was assigned to each grid of the space using a distance transform algorithm. Gradient descent rule was then applied to achieve complete coverage of the path.

Gabriely and Rimon [16] proposed the Spiral-STC algorithm. In Spiral-STC algorithm, workspace is subdivided into cells. The robot starts moving in an arbitrary direction

and checks for unvisited free cells in 4-neighborhood cells. The preference is set in anticlockwise direction. However, this approach did not cater for the partially occupied cells. A similar approach was used by E. Gonzalez et al. [17] in BSA algorithm. This approach adds a ‘back-tracking’ mechanism to the previously proposed Spiral-STC algorithm. Backtracking occurs when all other 3 cells in the neighborhood are either visited or occupied. The algorithm chooses previously visited cell as least preferred cell. The algorithm continues to append path until start position is reached – guaranteeing complete coverage.

### **2.2.2 Semi-Approximate Cellular Decomposition (SCD)**

In semi-approximate cellular decomposition (SCD), the cells are partially discretized with fixed width but they can have different shapes from top and bottom. Figure 2.2 (c) shows how cells discretization generally works in SCD model.

Each discretized cell is covered using regular zigzag movements. As cell discretization is a randomized approach, in a way, it does not guarantee cell coverage in a single pass. Therefore, in order to visit regions that are left unvisited, the robot may require to visit some parts of the cell twice or even more.

### **2.2.3 Exact cellular decomposition**

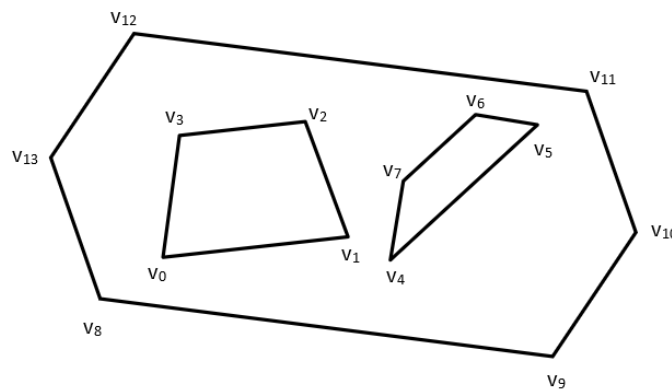
The exact cellular decomposition method is the most advanced method among other decomposition techniques. This method guarantees complete coverage with lesser revisits to the same points. In this method the target space is divided into a set of cells which are unique in size and shape. These cells are non-intersecting, and therefore, their union accounts for the environment that needs to be visited by the robot. Since the cells do not contain obstacles within, navigation controller model for the robot is much simpler in comparison to the methods discussed above.

Exact cellular decomposition can be achieved through different techniques, some of which have been discussed below:

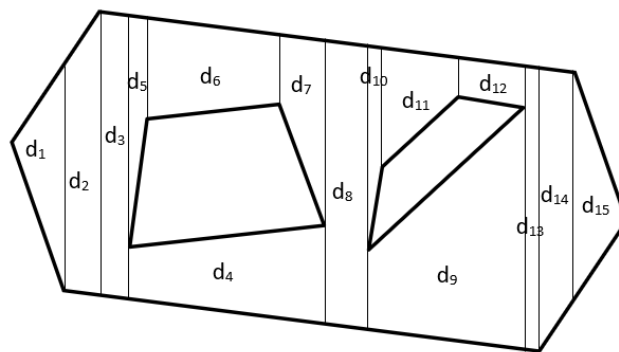
### 2.2.3.1 Trapezoidal decomposition

Trapezoidal decomposition is a popular exact cellular decomposition technique, which can guarantee complete coverage [12]. The robot's free space is decomposed into trapezoidal cells. Coverage can easily be achieved with back-and-forth motions due to trapezoidal shape of the decomposed cells. Since this technique generates trapezoidal cells, the obstacles are required to be polygonal. Additionally, this technique requires prior knowledge of the environment so it can only be applied offline.

Consider a polygonal work area as shown in Figure 2.4. As per trapezoidal decomposition algorithm, the work area and its adjacency graph are shown in figure 2.5. Two cells are said to be adjacent if they share the same boundary. Each decomposed space ( $d_1$  to  $d_{15}$ ) in adjacency graph can be covered using back-and-forth motion to achieve complete coverage. However, it is worth noticing that for each parent node with two (or more) child nodes, a decision needs to be made in order to prioritize the visiting order. This order can be manipulated in order to optimize the coverage problem.



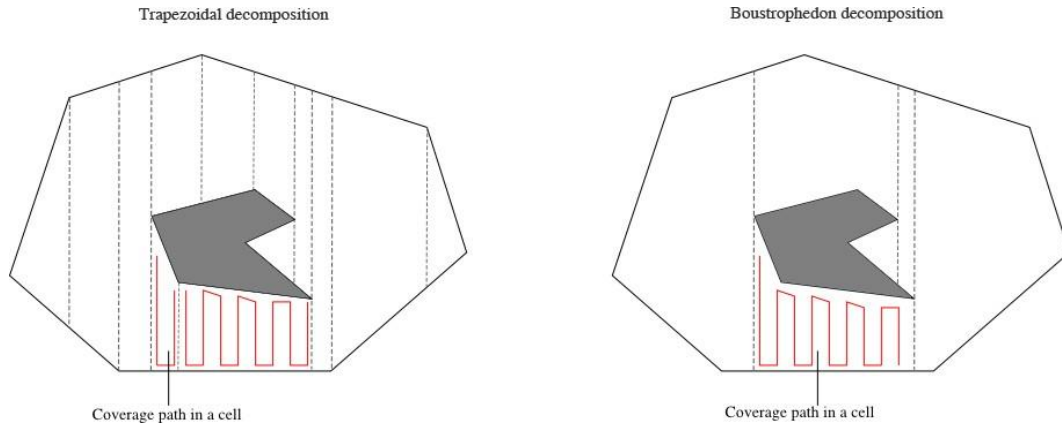
**Figure 2.3:** Given polygonal work area.



**Figure 2.4:** Trapezoidal decomposition for the work area above.







**Figure 2.8:** Trapezoidal decomposition (left) and boustrophedon decomposition (right) for the same space. Each cell can be covered with simple back-and-forth motions in both cases. Path is shorter in BCD. [18]

It is an offline method i.e. it can only be applied to environment with prior knowledge.

### 2.3 Other approaches

Other than decomposition techniques, researchers have proposed many other interesting ideas, some of which are discussed below.

#### 2.3.1 Artificial potential fields

Some approaches to coverage path planning use Artificial Potential Fields (APF). Pirzadeh and Snyder [19] proposed an indirect control strategy to achieve complete coverage using APF. The algorithm discretizes the environment and robot motion. The robot motion is only restricted to left, right, up and down motions. Diagonal neighbors are not considered. Most APF-based approaches encounter local-minima problems so they are not complete.

#### 2.3.2 Neural networks and fuzzy logic

Recently researchers have been working on solutions based on modern approaches like neural networks and Fuzzy Logic. Yasutomi et al. [20] presented a learning based CCP approach. The robot using this approach could operate in an unknown environment to map it and avoid obstacles. However computational complexity of this model makes it un-suitable for complicated environments and it can only be used for structured indoor environments.

Tse et al. [21] proposed a neural network model that generates path through back propagation. The robot memorizes the previously used path during cleaning process. If new map is detected the path memory is cleared.

Lang et al. [22] proposed a Fuzzy Logic based model. However, definition of fuzzy rules for such model was a challenge due to which the system saw problems in path as it was not smooth at turning and traversing. Fu and Lang [23] later came up with the motion error solution but un-structured complex environments remained a challenge even for that model.

## **2.4 Summary**

Several methods discussed above guarantee complete coverage online, that is, they can be used to cover all points on the free space of unknown environments. Furthermore, some of those methods account for kinematic constraints on the vehicle and also efforts have gone in optimization of the coverage path. However, a universal algorithm that guarantees an optimal path has not yet been developed.

Different aspects of studied algorithms and approaches are briefly explained in the tables given below.

**Table 2.1:** Summary of the analyzed CCP methods with Approximate cellular decomposition (ACD) and SCD.

	Article	Algorithm	Completeness	On/Offline	Prior knowledge required	Remarks	Intended application
A C D	[24]	N.A	PC	Online	Yes	kinematic constraints are not accounted for	Mobile robotics
	[25]	N.A	PC	Online	No		Mobile robotics
	[17]	N.A	Full	Online	No	Partially occupied cells are also filled	Mobile robotics
	[26]	N.A	Full	Online	No		Mobile robotics
S C D	[27]	N.A	Not complete	Online	No		Underwater environment

**Table 2.2:** Summary of the analyzed CCP methods with exact cellular decomposition (ECD).

Article	Algorithm	Completeness	On/Offline	Prior knowledge required	Remarks	Intended application
[12]	TCD	Full	Offline	Yes	Probably the most popular ECD.	2D spaces
[18]	BCD	Full	Offline	Yes	Relatively easy to implement	2D spaces
[28]	MCD	Full	Offline	Yes		Generic
[29]	MCD	Full	Online	No	First online (sensor-based) ECD proposed.	Generic
[30]	N.A	Full	Online	No	Alternative critical point detection method to [28]	Generic
[31]	N.A	Full	Online	No	Improves sensor-based Morse decomposition by detecting critical points also on non-convex obstacles	Mobile robotics

[32]	MCD+GVD	Full	Online	No	2-technique combination	Generic
[8]	N.A	Full	Online	Yes	Requires the boundary of obstacles and walls in advance	Lawn mowers
[33]	CC RM	Full	Online	No		Mobile robotics
[34]	N.A	Not complete	Offline	No		Mobile robotics
[35]	N.A	Not complete	Online	No		Mobile robotics

**Table 2.3:** Summary of the analyzed CCPM methods which don't use cellular decomposition, NN or Fuzzy Logic.

	Article	Algorithm/ method	Completeness	On/Offline	Prior knowledge required	Remarks	Intended application
A P F	[19]	N.A	Not complete	Offline	Yes	Local minima issues exist	n/a
T B M	[36]	N.A	Not complete	Offline	Yes	Not complete. Unable to handle environmental changes.	Generic
	[37]	N.A	Full	Offline	Yes		Generic

**Table 2.4:** Summary of the analyzed CCPP methods which use neural networks and fuzzy logic approaches

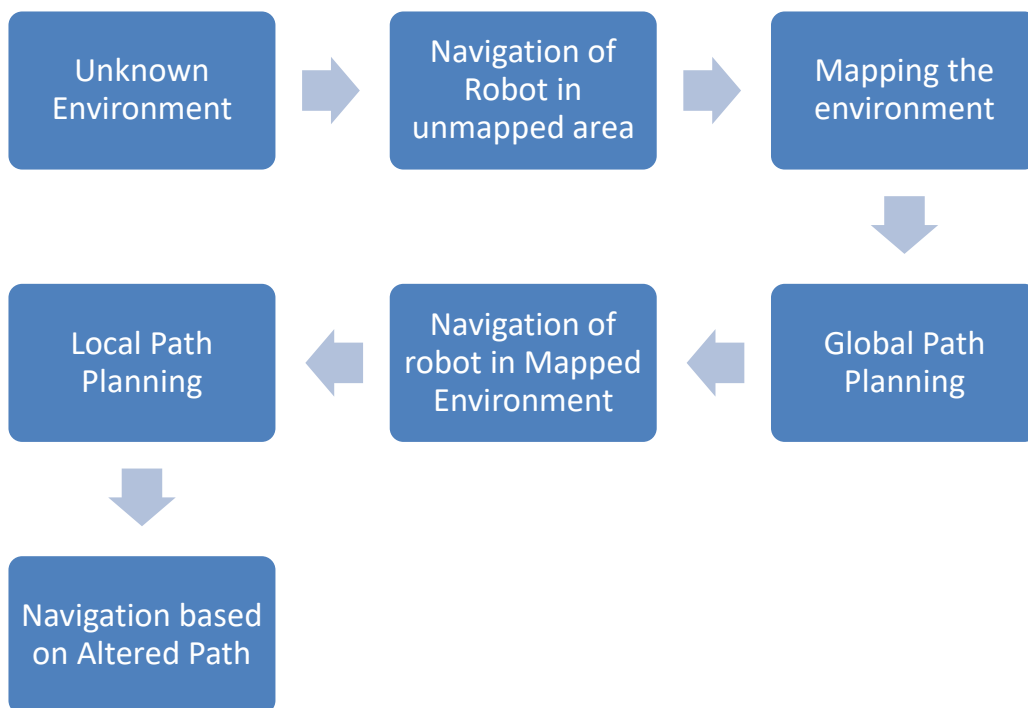
Article	Algorithm/ method	Completeness	On/Offline	Prior knowledge required	Handles non polygonal obstacles	Remarks	Intended application
[20]	NN	Not complete	Online	No	Yes		Mobile robotics
[21]	NN	Not complete	Online	No	Yes		Mobile robotics
[22]	Fuzzy logic	Not complete	Online	No	Yes		Mobile robotics
[23]	Fuzzy logic	Not complete	Online	No	Yes		Mobile robotics
[38]	NN	Full	Online	No	Yes	Parameter setting dependent model	Mobile robotics
[39]	NN	Full	Online	No	Yes	Parameter setting dependent model	Mobile robotics

**Table 2.5:** Summary of the analyzed CCPP methods which use miscellaneous approaches

Article	Algorithm/ method	Completeness	On/Offline	Prior knowledge required	Handles non polygonal obstacles	Remarks	Intended application
[40]	SSA	Full	Online	Yes	Yes	Prior knowledge of the environment required.	Generic
[41]	Heat trail	Full	Online	No	Yes	Requires heater effectors to mark the path and a temperature sensor to determine previously marked areas.	Mobile robotics
[42]	3- component model	Not complete	Online	No	Yes		Cleaning robots
[43]	GA	Full	Offline	Yes	No		Generic
[44]	2-level path planning	Not complete	Online	No	Yes	Interesting 2-level approach	Lawn mowing
[45]	SMA	Full	Offline	Yes	Yes	Agricultural field	Agricultural field
[45]	PRA	Full	Online	Yes	Yes	Agricultural field	Agricultural field

## CHAPTER 3: METHODOLOGY AND EXPERIMENTAL SETUP

The experimental setup consists of a two-wheeled differential drive robot modelled and simulated in Robot Operating System (ROS) in Linux (Ubuntu) environment. The robot also has a caster wheel for balancing purpose. A LiDAR sensor has been installed on the robot chassis that is able to measure distance in all directions (360 degrees). The environment consists of boundary walls and objects. The following flowchart summarizes the steps that are performed in this setup:



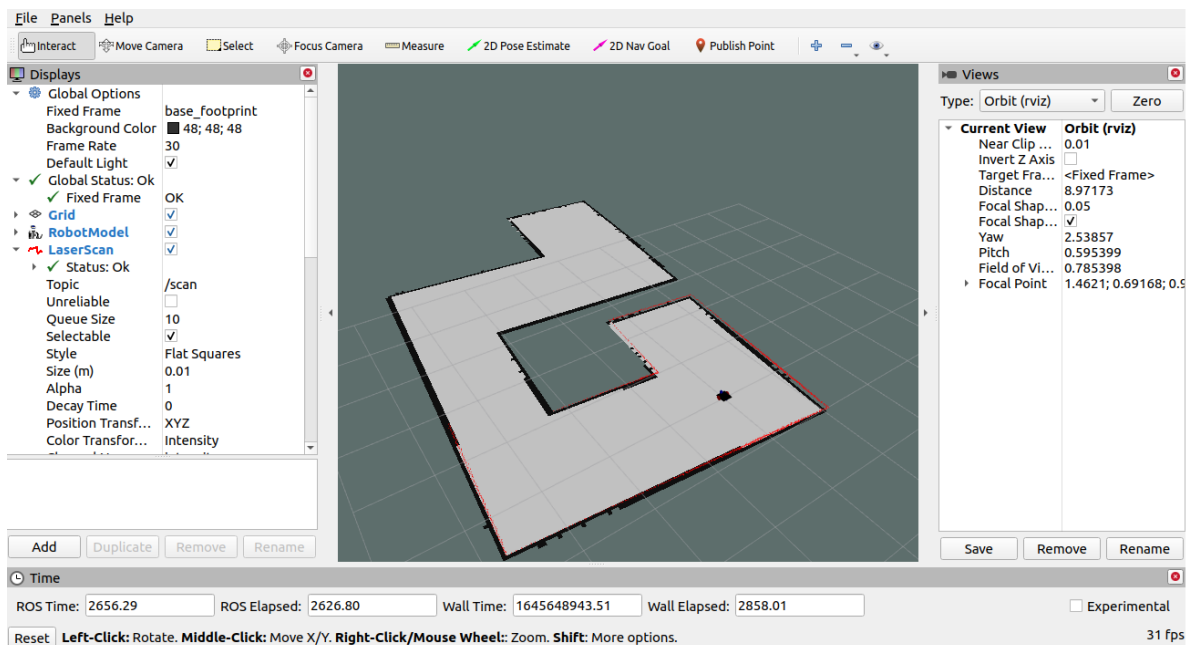
**Figure 3.1:** Experimental Setup Process

### 3.1 Navigation of Robot for Unmapped Environment

When the robot is first introduced in an unknown environment, it is required to map the area in order to plan path. The robot moves in the environment either by manual control using keyboard or automatically through programmed script written in Python that makes use of the LiDAR sensor in order to avoid obstacles. The script is based on random walk technique.

### 3.2 Mapping of the Environment

When the robot is moving in the environment, the values from LiDAR sensor and odometry frame in ROS are translated into a graphical representation that contains information about the environment in terms of occupied and free spaces. This is done by running ROS's Gmapping package in the background while the robot is navigating. Once the whole environment is covered the Gmapping package is closed which generates the map as an image. This image is processed using Python code and converted to a binary image as a representation of occupied and free spaces.



**Figure 3.2:** Mapped environment in ROS RViz

### 3.3 Global Path Planning

Once the robot completes its lap in the unknown environment, the Gmapping package generates the occupancy map in form of an image. Each pixel in the image represents a certain distance. The occupied spaces are represented in black whereas the free space is represented in white color. However, there are certain distortions in the image and some of the boundaries have gaps which need to be catered before any further operations are done. For this reason, the image is first treated using morphological operations like dilation or erosion. Since our main concern is closing of boundaries – which are represented in black color, we applied erosion



technique – as a result of which the map output came out to be well described. Another advantage of enhancing the boundaries is that the robot can now have a *safe distance* in order to avoid colliding with the walls and objects due to different navigational errors.

The global planner function generates a set of points that need to be visited in order to complete coverage problem. The detailed explanation of global planner function is given in the next chapter.

### **3.4 Navigation of Robot for Mapped Environment**

As mentioned earlier, the global planner function outputs a set of points in actual order. The navigation function receives the input in form of a point array. The robot continues to move in the direction of the immediate next unvisited point until it reaches its final goal. In other words, each visited point in the array acts as local start point for a local goal point. Once visited, the previous goal point becomes start point for the next goal point. Between start and goal point, the function keeps looking for any obstacles that may disturb the normal trajectory of the sequence. If obstacles are detected, the trajectory that needs to be followed is determined by the local path planning function.

### **3.5 Local Path Planning**

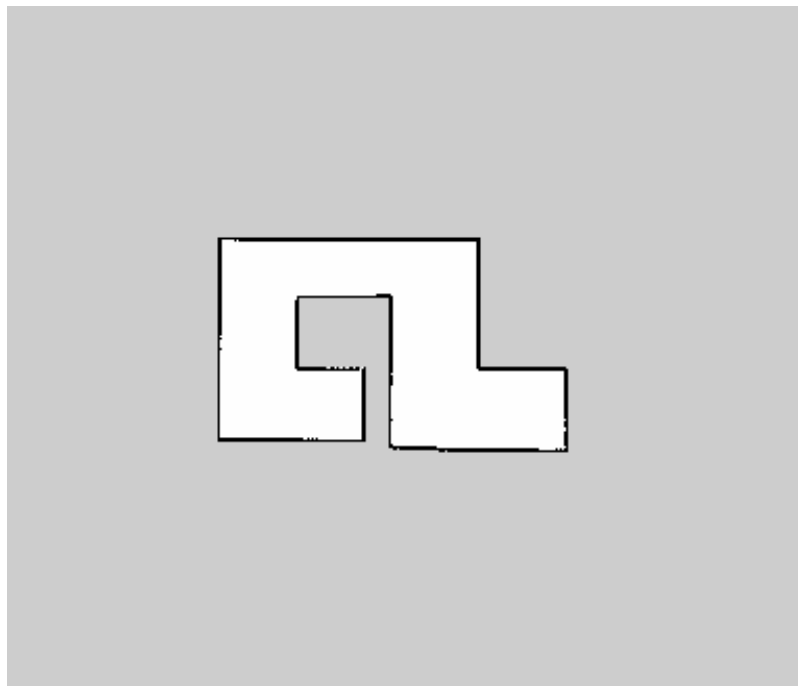
The global path planner generates points as waypoints. This ensures that the robot visits each point in order to cover the complete map. However, in certain cases the shortest distance between two waypoints may be blocked by obstacle or it can be “unsafe” due to dynamic factors like moving obstacles. For this reason, we need a local planner that takes into account, the dynamic factors, and generates waypoints between two global points. We propose a two-level approach for generating local path that will be discussed in detail in Chapter 5.

## CHAPTER 4: MAPPING AND GLOBAL PATH PLANNING

In order to plan path, we need to have a map initially. Whether the map is of complete environment or a small portion within the coverage environment, it is necessary in order to have some knowledge for decision. However, if the robot explores new map areas while navigating, it is said to be online path planning. Any path planned due to prior information is known as offline path planning as already discussed in Chapter 2.

### 4.1 Mapping and Image Processing

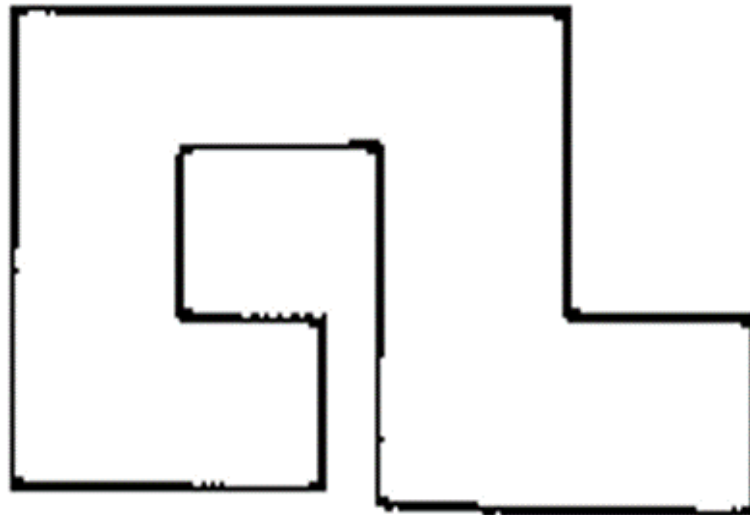
Our model is based on priorly mapped environment. But in order to compare results for an actual or simulated environment we need to have an environment with traceable and known features. Therefore, we create a simulated environment in ROS Gazebo and use built-in *SLAM Gmapping* package to map the environment. The robot is navigated in the simulated environment by user input or through random walk process using on-board LiDAR sensor. During robot motion, readings from LiDAR sensor and wheel odometry data are used to build map of the environment. Once the environment is fully mapped the map is saved as image in local disk.



**Figure 4.1:** Map of environment generated by the SLAM Gmapping package

The map generated by SLAM Gmapping package generally contains three colors. Gray color shows unexplored areas, black means the area is occupied by objects and white shows explored free spaces. The SLAM Gmapping package generates another file containing some important data that explains the pixel density and the map origin information. In order to perform path planning we first need the image to be cropped according to our needs such that unrequired areas are discarded. This also reduces processing power required for the next step.

In order to crop image, the first thing we need to do is to convert the image from grayscale to binary image. The threshold was set such that the gray area was also treated as occupied space. In the next step we used line sweep approach to ‘scan’ the image from one side to the other in order to detect occupied cells. This simple approach was enough to detect farthest boundary pixels the locations of these cells helped us crop the useful part of our map.



**Figure 4.2:** Cropped image after processing the useful map area

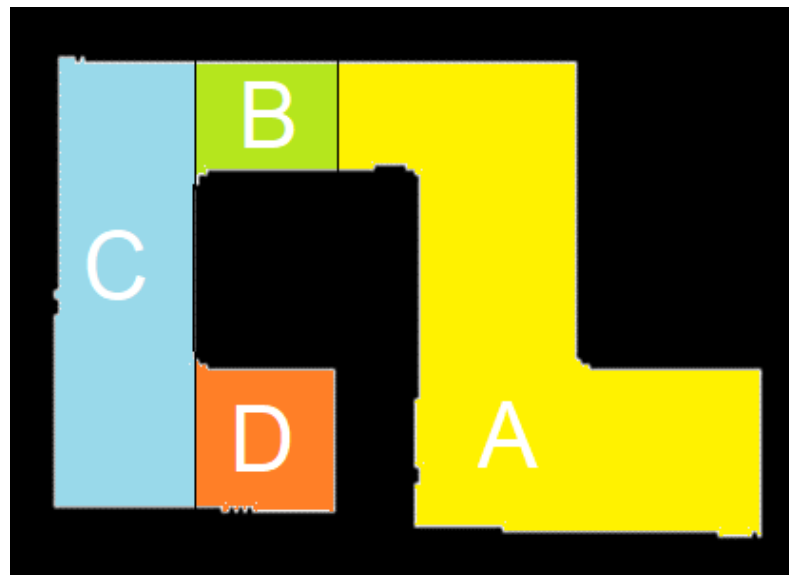
Since we will require the image to be scaled smaller in next steps the boundaries needed to be enhanced such that we do not get boundary details missed. Another advantage of thicker boundaries would be the safety factor for the robot. Boundaries were enhanced using morphological operators of dilation and erosion using OpenCV library. First erosion operator was applied to remove noise near boundaries and then dilation for having smooth, uniform boundaries.



**Figure 4.3:** Enhanced boundaries of given map after applying morphological operators

#### 4.2 Global Path Planning

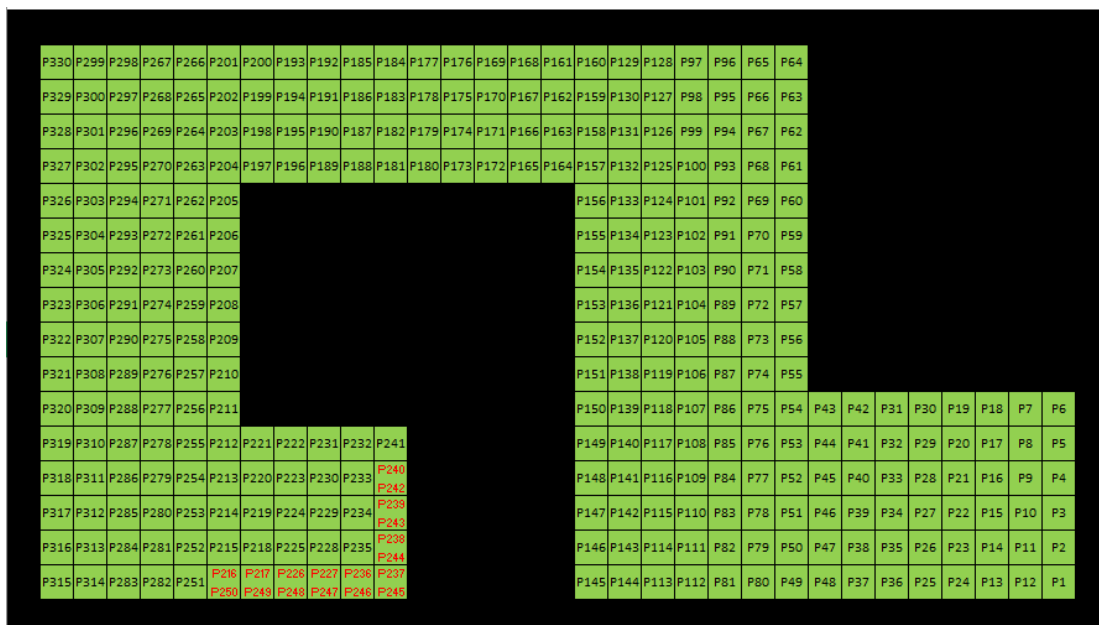
The map was first decomposed into sub-parts using Boustrophedon Cellular Decomposition (BCD) technique [18]. For illustration purposes, consider **Figure 4.4** where each decomposed cell has been shaded and named differently for referencing later.



**Figure 4.4:** Decomposed cells after applying BCD algorithm

The robot starts from cell A so the normal order, according to BCD algorithm, by which these cells should be visited comes out as A-B-C-D. We continue with this convention for generating the global path, however within the path planning function, we define a cost function which will decide whether or not this sequence is to be followed exactly or partially.

The map is then divided into square sized grids with size of grid matching to that of the robot. This size can be altered according to the size of the robot or if any overlapping or gap is desired. The global path planner function starts from start point of cell A. From there the planner function checks state of four neighbors (left, right, front, back). The states are stored in a  $n \times m$  sized array where  $n$  is the number of column grids and  $m$  is the number of row grids. There are 3 possible states: free, occupied and visited. Unvisited states that are free are also categorized as free. At every fixed obstacle the path must be such that it makes a u-turn. The path is generated in form of a 2d point list (**Figure 4.5**). Therefore, each item in list is actually the location of grid in the map.

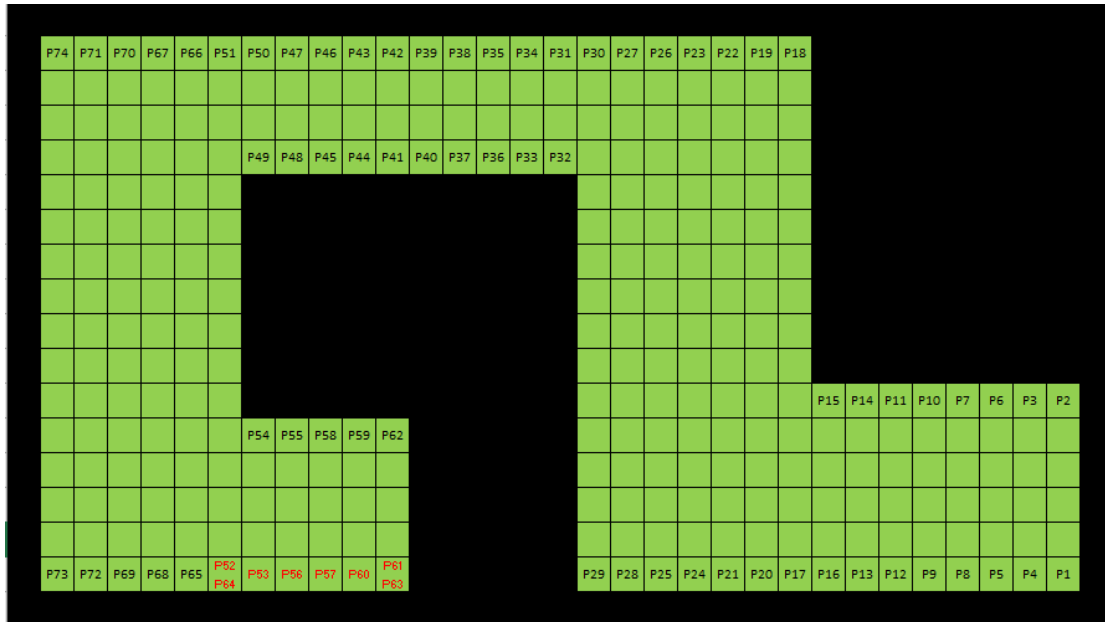


**Figure 4.5:** Route list containing index of each grid in ascending order

### 4.3 Removal of redundant points from Route array

**Figure 4.5** shows the path generated by global planner function. However, strictly following a route that consists of so many coordinates can greatly affect the navigation function of robot. Moreover, a navigation function that keeps record of visited and unvisited nodes of

this many closely spaced grids may consume a lot of memory. For this reason, critical points were identified that would best define the path. **Figure 4.6** shows the updated list of members of route array for global path.



**Figure 4.6:** Route list after removing unnecessary members

Two options were considered for removing redundant points. The first used method was from the equation of line. As equation of line is given by:

$$(y - y_1) = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

Considering scenario where the robot is at P2 (**Figure 4.5**),  $y_1$  is the y-coordinate of P1,  $x_1$  is the x-coordinate of P1,  $y_2$  is the y-coordinate of P3,  $x_2$  is the x-coordinate of P3, and  $x$  and  $y$  are the x and y coordinates of P2. If the equation above satisfies, it means P2 lies within line P1 – P3 therefore we can remove P2.

Similarly, we iterate through all the sequence to remove unnecessary points in our path. As a result, route list is updated as seen in **Figure 4.6**.

Another approach that we used for reducing the number of waypoints was simpler. Considering the same example we used previously, if the point is consideration is P2, it is said to be in the same line as P1 – P3 if it follows this condition:

$$dist(P1P2) + dist(P2P3) = dist(P1P3)$$

The above condition is only true if P2 lies in the Euclidean path of P1 and P2. Otherwise, it will form a triangular shape in which sum of two sides is always greater than the third side. Therefore, the condition to retain P2 is as follows:

$$dist(P1P2) + dist(P2P3) > dist(P1P3) \quad \dots (1)$$

In our experimental work we used both methods. It was seen that the first method took longer processing time than the latter, therefore, we used the distance method instead of point on line.

It is also important to mention here that due to some assumptions and possible errors in image processing techniques used – it is often not possible for such conditions to be perfectly met. Therefore, a certain safety factor must be used. Therefore equation (1) can be written as:

$$dist(P1P2) + dist(P2P3) > dist(P1P3) + \mu$$

Where  $\mu$  is the factor by which a certain grid may appear deviated even though it may be practically be considered to be in line with the adjacent grids.

#### 4.4 Modified BCD Approach for Global Planner

The flowchart diagram of the global planner function is given in **Figure 4.7**. The global path planner function generates path in similar way as explained previously. However, when the path planner function is generating paths at the boundary of two different (decomposed) cells the function checks the cost for end points of both cells from current location. This cost function is simply the Manhattan distance between current point to the end point. In example quoted above, the cost function for global path planner function at boundary CD while the robot is covering cell C, is calculated as:

$$\begin{aligned} Cost_{CD} &= x_D + y_D - x_{curr} - y_{curr} \\ Cost_{CC} &= x_D + y_D - x_{curr} - y_{curr} \\ \sigma &= Cost_{CD} - Cost_{CC} \end{aligned}$$

If  $\sigma \geq 0$  the path planner function will continue to cover cell C before moving on to cell D. Otherwise the path planner will jump to cell D before covering remaining part of cell C. After covering cell D completely, the path planner function will continue covering cell C from the point where it left.

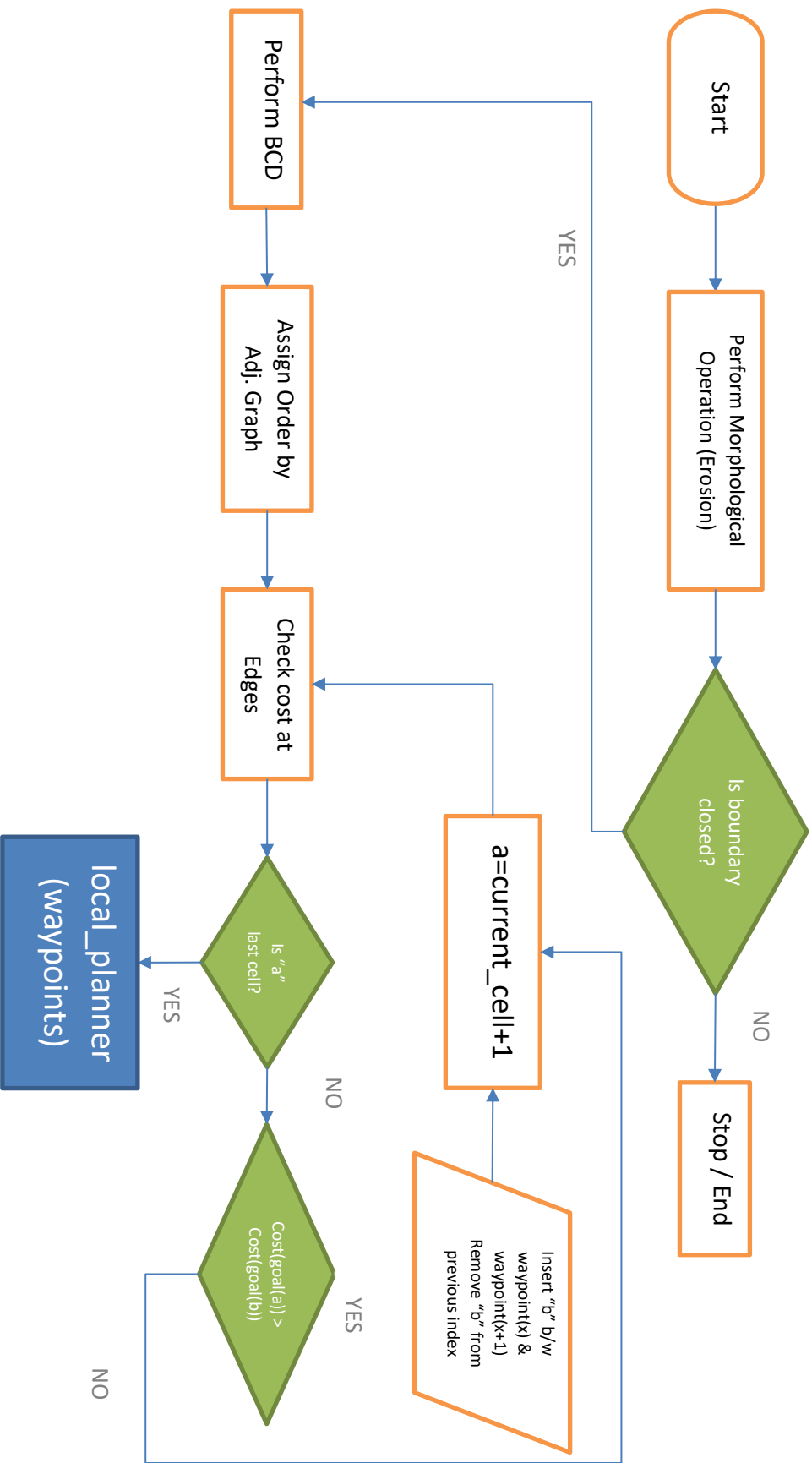
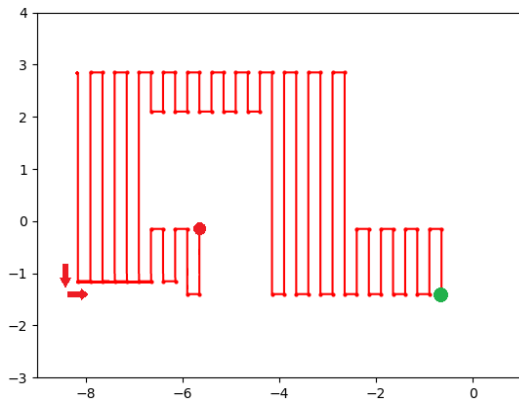
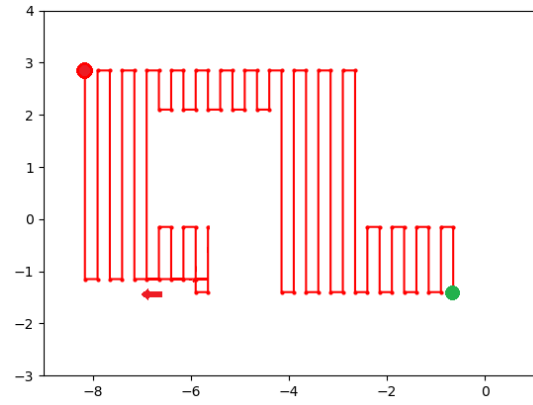


Figure 4.7: Flow Chart Diagram of global path planner





a. Normal BCD Approach



b. Modified BCD Approach

**Figure 4.8:** Cost of Modified BCD is lesser as compared to normal BCD

**Figure 4.8** shows how modified Boustrophedon Cellular Decomposition approach reduces the distance that needs to be covered by the robot in order to cover the same area that normal BCD approach covers in travelling greater distance. This result is further discussed in Chapter 6.

## CHAPTER 5: NAVIGATION AND LOCAL PATH PLANNING

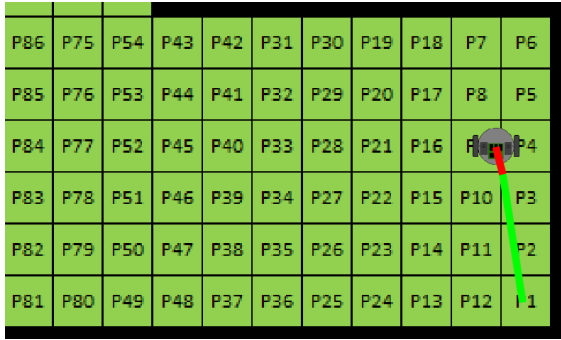
As discussed in previous chapter, route generated by global path planner is an array containing points which need to be visited by the robot in a proper sequence. The navigation function receives this array as an input. In Section 4.3 we discussed how we reduced the number of points in route array. In the next section we will discuss how reducing the number of waypoints improve our navigation while also improving efficiency in terms of computational complexity.

### 5.1 Navigation

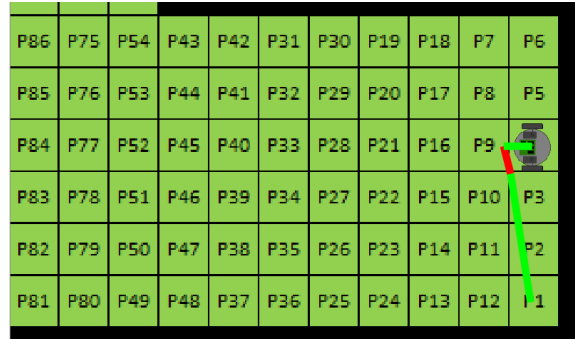
The navigation function navigates the robot to the waypoints defined in route array. This handles the robot poses and velocity by which the robot must move in order to achieve its goal. Each recently visited point acts as the start point for the next-in-sequence point referred to as *local goal*. In theory, large number of waypoints should not affect the robot motion. However, in real world scenario, over-defined path can cause many problems for the navigation function. When a large number of waypoints are defined, it causes the navigation function to strictly follow each waypoint. This causes robot to satisfy conditions for a very big number of local goals. Small deviations from local goal may result in robot *missing* the target. In order to fulfil requirement of visiting the local goal, the robot may need to make big turns - unnecessarily wasting energy and time.

We used the route array with reduced waypoints as discussed in Section 4.3 for the purpose of navigation. The first element in route array is the start point of our planned path. When any point is passed to the navigation function, it instructs the robot to position itself towards that point. Once the pose conditions are met, the navigation function then instructs the robot to start moving in the front direction. Any moving obstacles or static obstacles that were not part of the global map are handled by local planner and will be discussed in next section.

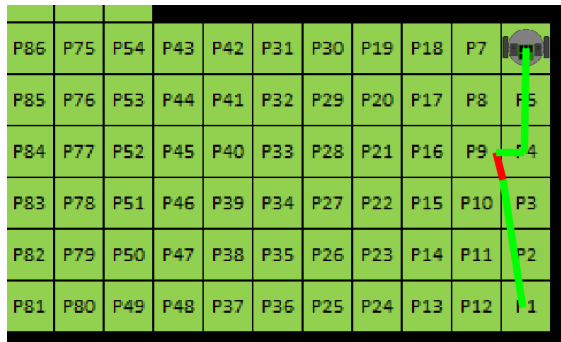
Once the robot reaches its local goal, the said waypoint is marked as visited. This point then acts as start point for the next-in-sequence waypoint which is the new local goal. The previously explained process of pose setting for local goal is repeated for this point and the robot continues navigation until the last point in global path is visited. **Figure 5.1** shows how reduced waypoint approach saves unnecessary moves, time and energy.



a. Robot deviates from local goal



b. Robot sets pose to rectify error



c. Robot follows remaining route



d. Smooth navigation with reduced waypoints

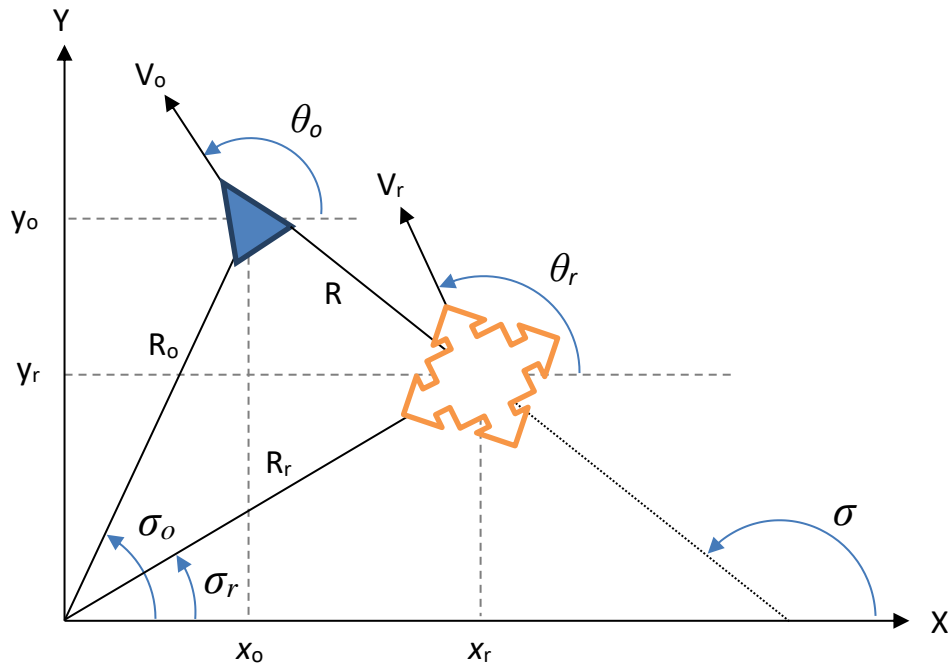
**Figure 5.1:** Navigation with all waypoints (a, b, c) and reduced waypoints (d)

## 5.2 Local Path Planning

While robot is passing through the waypoints, it may encounter obstacles that were not defined in global map. This could be due to change in configuration of the obstacles or addition of moving or static obstacles in the environment. Local path planner is a function that runs along the navigation function and keeps check on the obstacles. If an obstacle is encountered within a set radius, this function directs the robot to change its path in order to avoid collision. It is pertinent to mention here that static obstacle is a special case of moving obstacle where velocity of obstacle is zero.

A two-level approach was used for avoiding obstacles. Level 1 check ensures whether or not the obstacle may hinder the *regular* path of the robot. Basic reason for using two-level approach is the computational simplicity of level 1. The second level is computationally expensive velocity obstacle approach and is only triggered if level 1 signals possible collision. Moreover, level 1 only triggers if single obstacle is detected. In case of multiple obstacles, the

local planner directly calls for the velocity obstacle approach bypassing level 1. **Figure 5.2** shows a typical scenario where an obstacle is detected within a set radius around the robot.

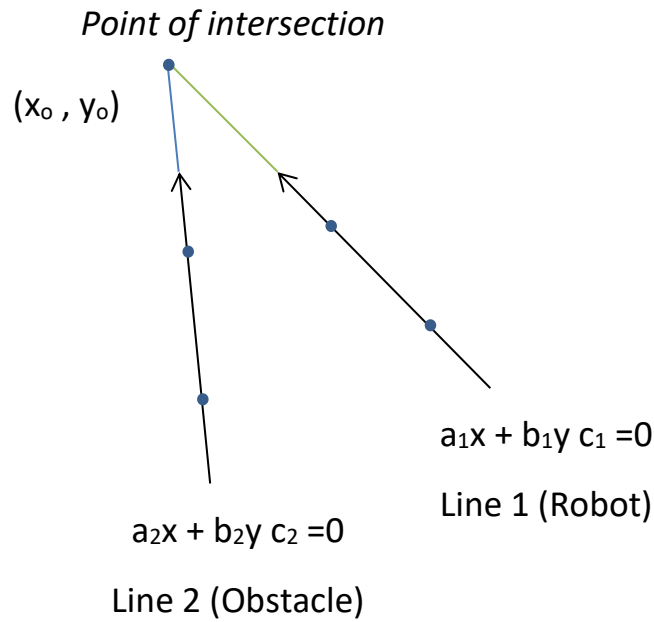


**Figure 5.2:** Positional and velocity vectors of Robot (R) and moving object (O)

### 5.2.1 Intersection of Lines Approach (Level 1)

The first level approach is based on the principles of intersection of lines. Positions of robot and obstacle are obtained at two time intervals  $t_1$  and  $t_2$ . With the help of these four points, we can calculate the intersection point of the robot and the moving obstacle. **Figure 5.3** explains the simple approach using two-point form of equation of line.

Since all non-parallel lines intersect at some point, this data is not enough to assume whether or not the robot and the obstacle will collide. In order to be able to come to the conclusion, we found the time required for the robot and the obstacle to reach the point of intersection. If the time difference is in safe range, we continue with the normal motion of robot, otherwise we generate local path using Velocity Obstacle (VO) approach herewith referenced to as level 2 approach.



**Figure 5.3:** Calculation of intersection point of robot and moving obstacle

### 5.2.1.1 Calculating Point of Intersection

The calculations done in order to find point of intersection are explained below:

Suppose L1 and L2 represent the lines of motion of the robot and moving obstacle respectively.

$$L1: \quad a_1x + b_1y + c_1 = 0 \quad \dots (1)$$

$$L2: \quad a_2x + b_2y + c_2 = 0 \quad \dots (2)$$

Suppose lines L1 and L2 intersect at  $x_0$  and  $y_0$ . Equation (1) and (2) can be written as:

$$L1: \quad a_1x_0 + b_1y_0 + c_1 = 0$$

$$L2: \quad a_2x_0 + b_2y_0 + c_2 = 0$$

By Cramer's Rule:

$$\frac{x_0}{b_1c_2 - b_2c_1} = \frac{-y_0}{a_1c_2 - a_2c_1} = \frac{1}{a_1b_2 - b_1a_2}$$

Where,

$$L1: a_1 = y_1 - y_2, b_1 = x_2 - x_1, c_1 = y_1x_2 - y_2x_1$$

$$L2: a_2 = y_1 - y_2, b_2 = x_2 - x_1, c_2 = y_1x_2 - y_2x_1$$

$$D = a_1b_2 - a_2b_1$$

$$D_x = c_1b_2 - c_2b_1$$

$$D_y = a_1c_2 - a_2c_1$$

Lines L1 and L2 will only intersect if  $D \neq 0$

$$x_0 = D_x/D$$

$$y_0 = D_y/D$$

If L1 and L2 are parallel  $D=0$ , resulting in (inf, inf) value for point of intersection which suggest that parallel lines can not intersect.

### 5.2.1.2 Predicting collision between Robot and Obstacle

Till this point we have obtained the line of intersection of the robot and the obstacle. The next step is to use the velocities and current locations of robot and the obstacle to obtain the time required to get to the point of intersection for each the robot and the moving obstacle. This is done by dividing distance from current location to the point of intersection by the magnitude of velocity.

$$t_{obs} = \frac{dist(P_{obs} P_{int})}{|v_{obs}|} \dots (3)$$

$$t_{rob} = \frac{dist(P_{rob} P_{int})}{|v_{rob}|} \dots (4)$$

$$\alpha = \frac{1}{|t_{rob} - t_{obs}|} \dots (5)$$

Where  $t_{rob}$  and  $t_{obs}$  is the time taken by the robot and obstacle, respectively, to reach the point of intersection.,  $P_{obs}$  is the location of obstacle at sampling instant,  $P_{rob}$  and  $P_{int}$  are the positions of robot and point of intersection.  $v_{rob}$  and  $v_{obs}$  are the velocities of robot and the moving obstacle.

Based on the time difference we calculate the *collision factor* ( $\alpha$ ). For safety purpose the value of collision factor must be less than  $\frac{1}{4}$  which essentially means that there must be a difference of at least 4 seconds between  $t_{rob}$  and  $t_{obs}$ . Values greater than  $\frac{1}{4}$  mean there is a possible chance of collision, and therefore, the said case is referred to the second level approach – the velocity obstacle approach, which will be discussed in the next section.

It must also be noted that the *collision factor* of  $\frac{1}{4}$  is derived mathematically and experimentally for the experimental setup used in this study where robot radius is 16 cm and the velocities of the robot and obstacles are between 20 – 30 cm/s. Special cases where angle between the two objects is less than 15 degrees are also dealt with smaller values of *collision factor* because in such scenario the obstacle and robot may collide even before reaching the point of intersection (collision).

### 5.2.2 Velocity Obstacle Approach (Level 2)

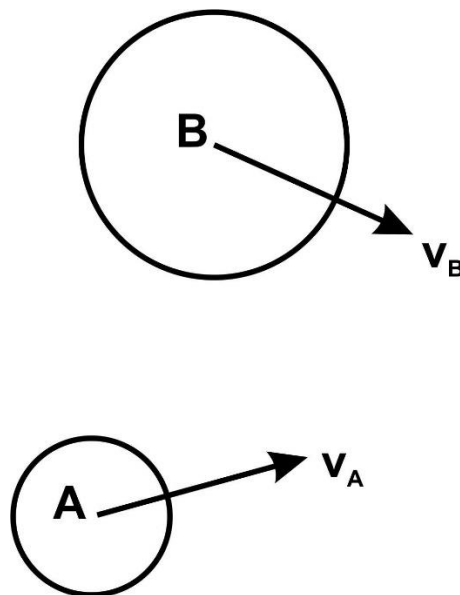
The intersection of lines approach, discussed previously, acts as a filter for referring complicated scenarios to level 2 approach known as the Velocity Obstacle Approach [46]. The velocity obstacle (VO) approach addresses the problem of path and motion planning in changing environments. Path planning becomes very difficult in dynamic environments, because it requires simultaneous solving of path while the robot is navigating.

The VO approach restricts analysis to circular obstacles and robots. However, this essentially cannot be termed as limitation because all polygons can be represented by a number of circles [47].

Consider robot A and obstacle B, shown in **Figure 5.4** at time  $t_0$  moving with velocities  $\mathbf{v}_A$  and  $\mathbf{v}_B$  respectively. In order to proceed further, and to make visualizations simpler, let us consider our robot as a point mass. We do it by squeezing the robot A to a point and growing all other obstacles (in this case obstacle B) by the size of robot diameter. New radii of A is  $\hat{A}$  and that of B is  $\hat{B}$ . We define the set of colliding relative velocities between  $\hat{A}$  and  $\hat{B}$  as *collision cone* ( $CC_{AB}$ ).

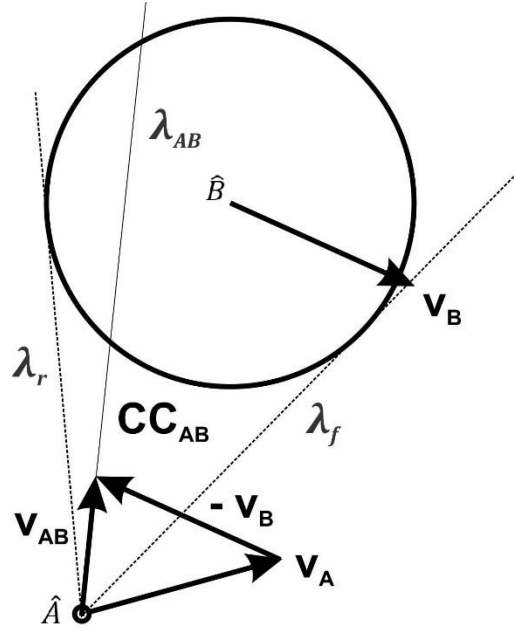
$$CC_{AB} = \{\mathbf{v}_{AB} \mid \lambda_{AB} \cap \hat{B} \neq \emptyset\} \dots (1)$$

Where  $\mathbf{v}_{AB}$  is the relative velocity of robot  $\hat{A}$  with respect to  $\hat{B}$  ( $\mathbf{v}_{AB} = \mathbf{v}_A - \mathbf{v}_B$ ), and  $\lambda_{AB}$  is the line in direction of  $\mathbf{v}_{AB}$ .



**Figure 5.4:** Robot A and Obstacle B moving with velocities  $\mathbf{v}_A$  and  $\mathbf{v}_B$  respectively [46]





**Figure 5.5:** Relative Velocity  $v_{AB}$  and collision cone  $CC_{AB}$  [46]

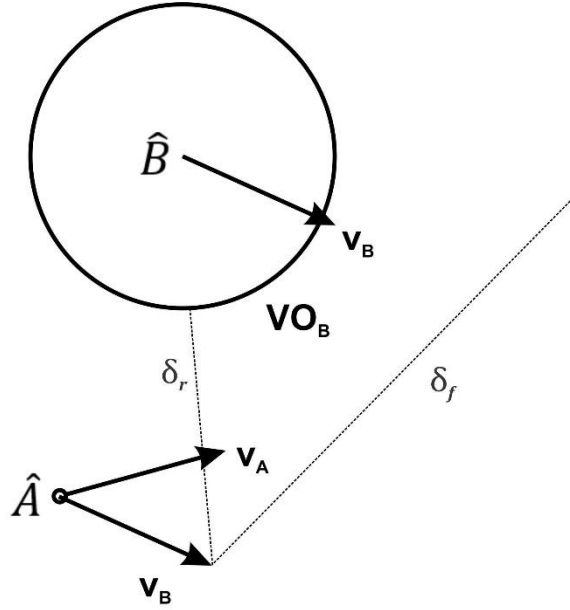
The collision cone is the planer area bounded by two tangents to  $\hat{B}$  from  $\hat{A}$  as shown in **Figure 5.5**. Any relative velocity lying within the collision cone i.e  $\lambda_r$  and  $\lambda_f$  will result in collision between the robot A and the obstacle B. If the relative velocity lies outside the velocity cone it is guaranteed to avoid the collision provided the robot and obstacle continue to move with the same velocities.

### 5.2.2.1 Handling multiple obstacles

The collision cone approach only handles single obstacle. In order to handle multiple obstacles, we need to develop relationship based on absolute velocities of robot A. This is done by adding velocity of obstacle B ( $v_B$ ) to each element of the collision cone  $CC_{AB}$ . In other words, we translate the collision cone by  $v_B$  (**Figure 5.6**).

$$VO_B = CC_{AB} \oplus v_B \dots (2)$$

Where  $\oplus$  is the Minkowski vector sum operator as the velocity cannot be directly added to a group of elements. Here Minkowski operator means that we need to add the velocity  $v_B$  to each element of the collision cone  $CC_{AB}$  individually.



**Figure 5.6:**  $CC_{AB}$  is translated by  $\mathbf{v}_B$  to obtain  $VO_B$  [46]

The velocity of the robot  $\mathbf{v}_A$  should be selected such that it is outside the velocity obstacle. This guarantees that the robot will not collide with the moving object B.

$$A(t) \cap B(t) = \emptyset \text{ if } \mathbf{v}_A(t) \notin VO(t) \dots (3)$$

As explained earlier, static object is a special case of moving object where velocity is zero. Therefore, in this case, if the obstacle is static,  $\mathbf{v}_B = 0$  returns velocity obstacle identical to that of the collision cone. For multiple moving objects we take union of all the individual velocity obstacles:

$$VO = \cup_{i=1}^n VO_{B_i} \dots (4)$$

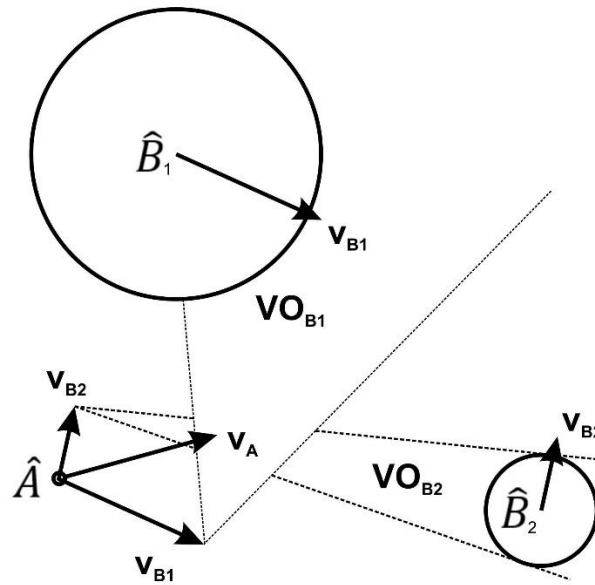
Where  $n$  is the number of obstacles and  $VO_{B_i}$  denotes each individual velocity obstacle. The velocities that make the robot avoid collision then consists of the set of velocities that lies outside the velocity obstacles as given in **Figure 5.7**.

In case of many obstacles, it is better to take prioritized entities into account for a certain time based on their distance from the robot and velocities. Velocity Obstacle calculates obstacle's trajectory based on linear approximation, therefore using it to predict remote collisions may be inaccurate if the obstacle does not move along a straight line. That is also why approximation of farther obstacles is discouraged. Therefore, we separate *possible* collisions that take place before time  $t$  and after time  $t$ . This time  $t$  is defined as suitable time

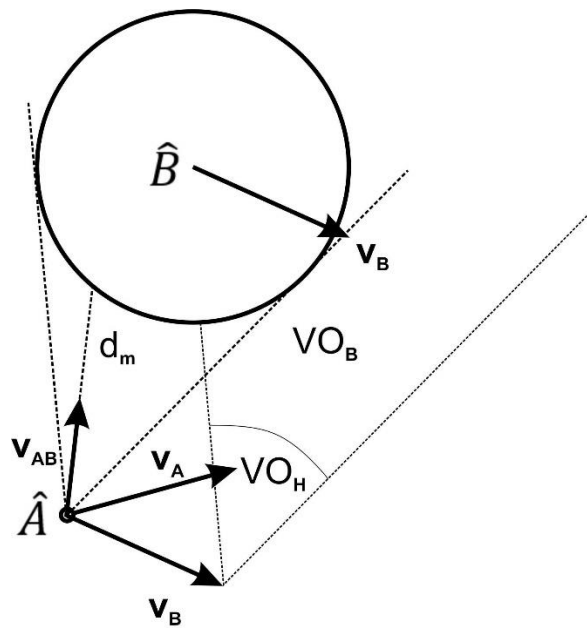
horizon  $T_h$ . All possible collisions that occur in time less than  $T_h$  are termed as *imminent* collisions. To account for imminent collisions, we modify set  $VO$  such that set  $VO_H$  is subtracted from it.

$$VO_H = \left\{ \mathbf{v}_A \mid \mathbf{v}_A \in VO \mid \|\mathbf{v}_{AB}\| \leq \frac{d_m}{T_h} \right\} \dots (5)$$

Where  $d_m$  is the shortest relative distance between A and B.



**Figure 5.7:** Velocity obstacles  $VO_{B_1}$  and  $VO_{B_2}$

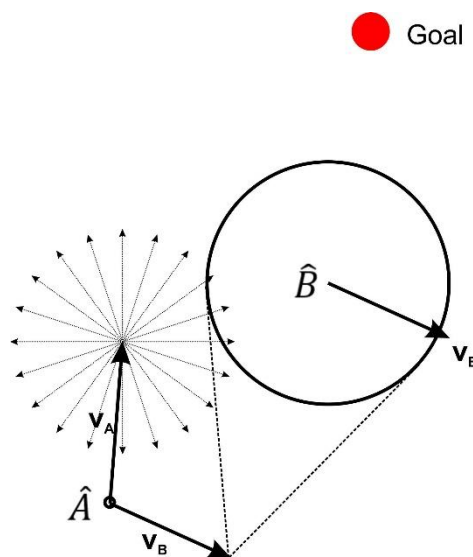


**Figure 5.8:** Velocity obstacle  $VO_B$  for short time horizon

The set  $VO_H$  represents velocities resulting in possible collision after the time horizon. **Figure 5.8** shows  $VO_H$  removed for specific time interval in order to only consider velocity obstacles representing possible collisions within time horizon.

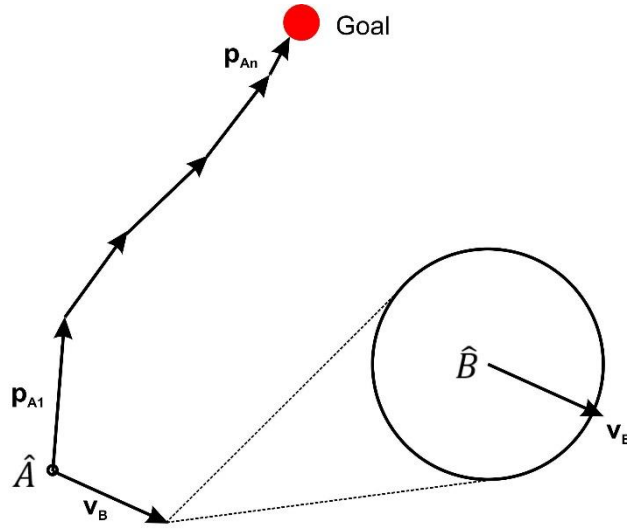
### 5.2.2.2 Generating waypoints for local path

The Velocity Obstacle  $VO$  helps us avoid velocities which result in possible collision. This helps us generate trajectory that does not fall in prohibited region. In order to find trajectory from start to goal position, *random search* method is used. In random search method, we start from the initial position of the robot and test with  $n$  number of positional vectors along  $360^\circ$  search space. The magnitude of positional vector must be such that the robot can reach the end point from start point in time  $\Delta t$  with velocity less than or equal to the maximum velocity the robot may attain. We used 20 number of samples for the search space, each at an angle of  $18^\circ$  from the adjacent vector. Our next maneuver needs to fulfil two conditions: the selected vector must not be projecting inside the  $VO$  and the selected vector must result in position such that it is closest to the goal.



**Figure 5.9:** Trajectory calculation for robot to reach goal avoiding moving obstacle

Once a vector among search space is selected, we continue with the search method by search space method from the last vector until we reach our goal.



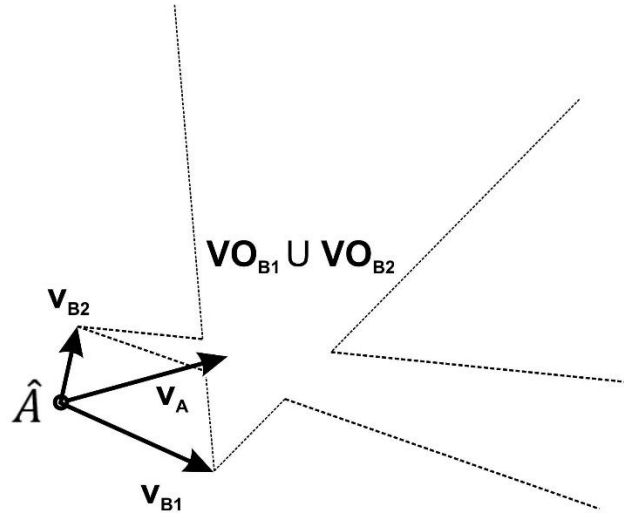
**Figure 5.10:** Trajectory calculated for single robot velocity obstacle

The result seen in **Figure 5.10** is the trajectory as seen in multiple frames considering obstacle velocity into account. The timestamp at each vector interval is calculated using the velocity and distance parameters. It is important to mention here that the velocity obstacle is time varying, therefore we also need to take the obstacle's velocity,  $\mathbf{v}_B$  into account.

$$t_m = \sum_{i=1}^m \Delta t_i$$

$$d_m = \sum_{i=1}^m |p_{A_i}|$$

Where  $t_m$  is the calculated total time taken by robot from start position to  $p_{A_m}$ ,  $\Delta t$  is the time robot will take to cover each positional vector,  $d_m$  is the total distance covered by robot from start position to  $p_{A_m}$ .



**Figure 5.11:** Velocity obstacle in multi obstacle scenario

For multiple obstacle scenario the trajectory is calculated in the same way as for the single obstacle. However, the velocity obstacle in this case is simply the union of all velocity obstacles. The process of trajectory calculation is instantaneous, therefore any errors in previous calculations can be rectified in the next iteration.

### 5.3 Integrating Local Path Planning with Navigation and Global Path

Recalling Section 4.4 and Section 5.1, we identified critical points for our global path. The critical points define our path in the minimum possible waypoints. For  $n$  number of waypoints, the navigation function visits each point one by one, starting from first till  $n$ th. Between two global waypoints, the trajectory is handled by the local path planner. Depending on the obstacles, local path may either be handled by level 1 (Section 5.2.1) or level 2 (Section 5.2.2) approach. The local planner outputs the *next move* waypoint to the navigation function which navigates the robot to that point. This process continues until local goal (next global waypoint) is reached. **Figure 5.12** shows the flowchart diagram of the local path planner function and its linkage with global path planner and navigation function.

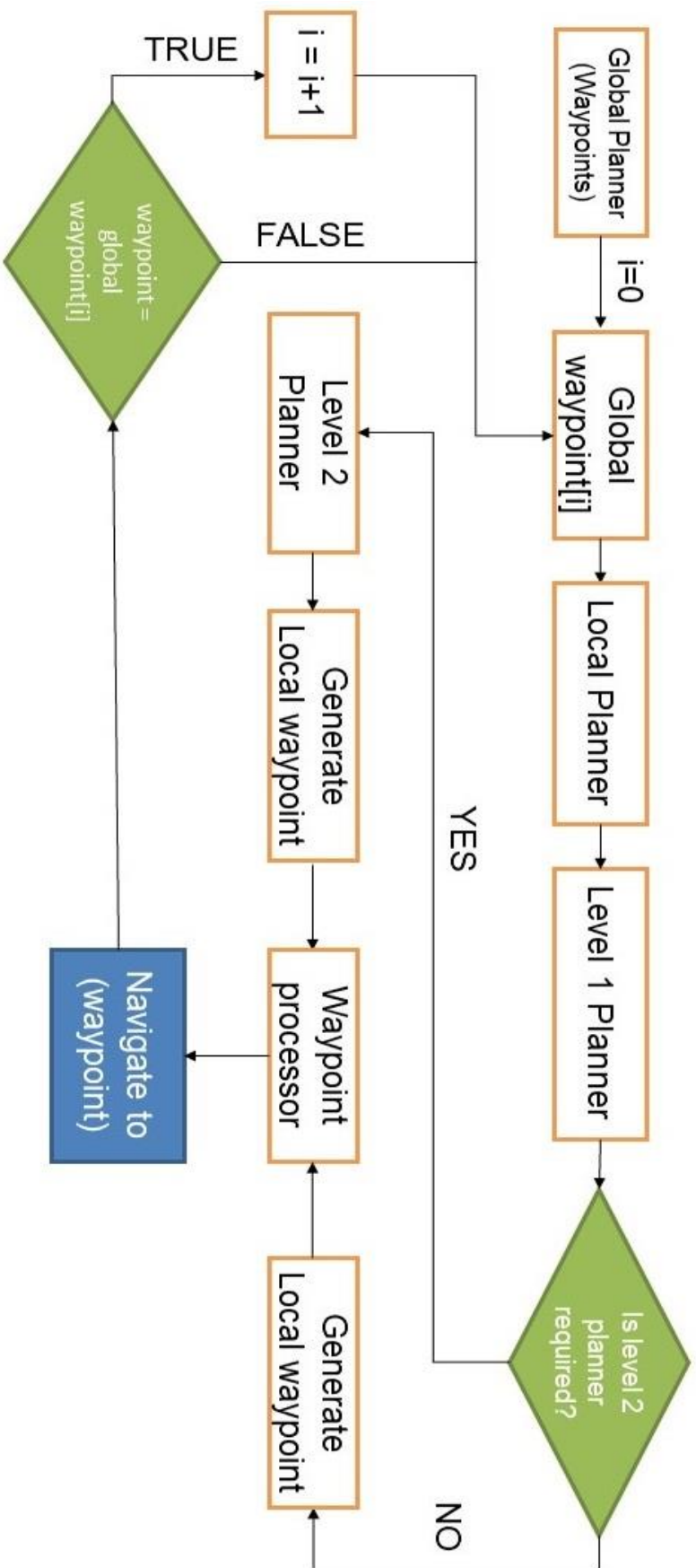


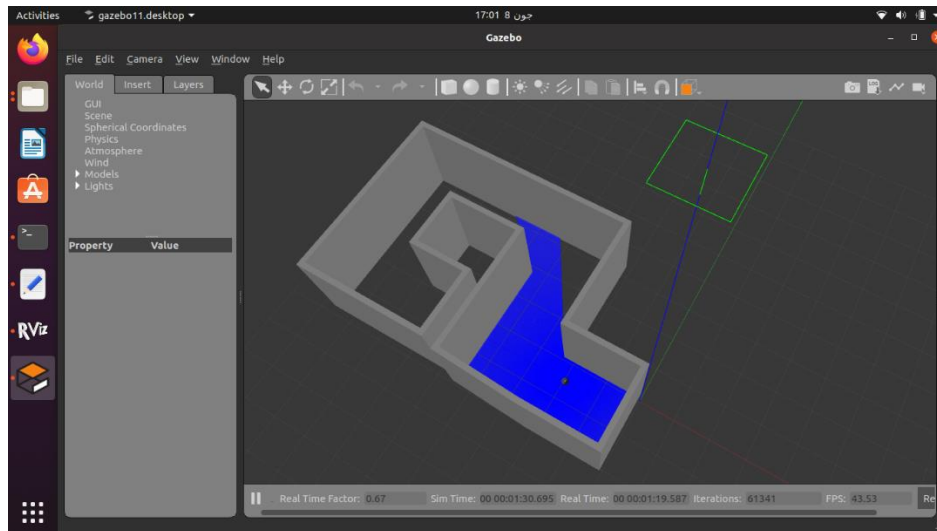
Figure 5.12: Flowchart of Local Path Planner

## CHAPTER 6: RESULTS AND DISCUSSION

The experimental work was carried out in simulated environment of ROS Gazebo. The environment was mapped using onboard 2D LiDAR sensor under ROS RViz. For global planning the map was decomposed using Boustrophedon Cell Decomposition Approach and a set of global waypoints were generated. These waypoints were passed to the local planner where path between each global waypoint was handled by the local path planner. The local planner intelligently detects any obstacles in the environment – moving or static and generates trajectory for the navigation function to drive robot as per dynamic constraints. A two-level approach was used for local path planner in order to minimize computational complexity. The local planner communicates with the navigation function in order to move the robot towards the next waypoint.

### 6.1 Simulation Environment

The experiment was conducted in simulated environment of Robotics Operating System (ROS) Gazebo. The operating system used is Linux Ubuntu 20.04. The real-time simulation environment is shown in **Figure 6.1**. The robot is a two-wheeled differential drive robot with onboard 2D LiDAR sensor. ROS's inbuilt odometry package is used for localization. The environment consists of walls and moving obstacles.



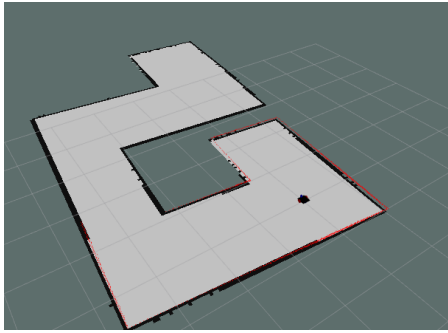
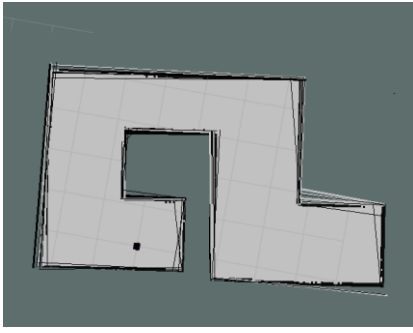
**Figure 6.1:** Simulation environment in ROS Gazebo



## 6.2 Environment Mapping Results

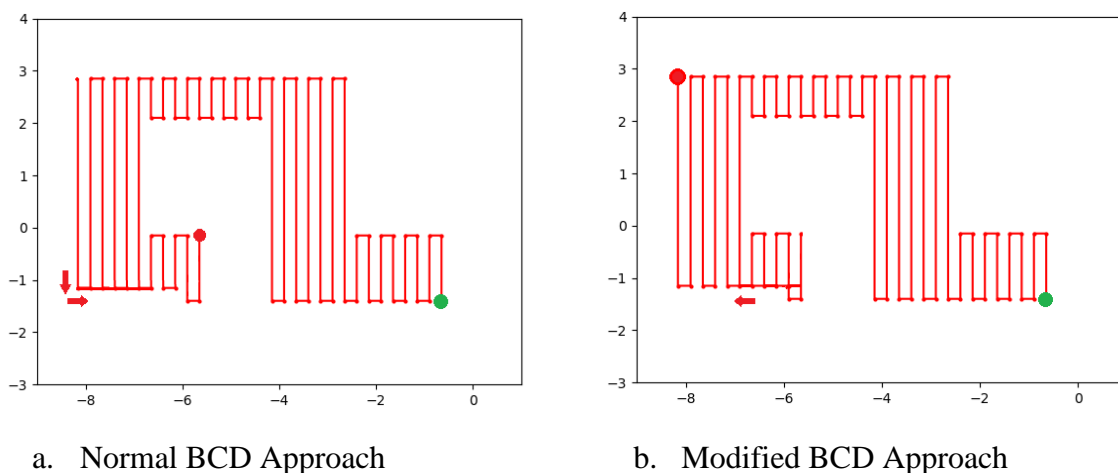
The SLAM gmapping package in ROS provides a good framework to build maps of the environment. It uses LiDAR and pose data from the odometry frame to generate 2D map of the coverage area. Pose data was gathered using two different methods, using inertial measurement unit (IMU) sensor package, and robot odometry frame. Since gmapping package does not support IMU, Hector SLAM package was used to generate map. However, hector SLAM consumes too much memory in comparison to the gmapping package. The main reason for that is the computational complexity involved in IMU. Results showed that gmapping technique generated map with 97% accuracy in comparison to hector SLAM's 85% and with 5 times as much speed as IMU-based hector SLAM. The reason for such high accuracy of gmapping is that it uses data from the odometry frame which is the actual data pose data so any possible errors are only caused due to LiDAR data. Whereas in case of hector SLAM, the robot pose data is acquired from IMU which is not free from errors due to the computational complexity involved.

**Table 6.1:** Comparison of Mapping techniques

Desc	SLAM Gmapping	Hector SLAM
Result		
Accuracy	97%	85%
Computational Complexity	Normal	Very high

### 6.3 Global Path Planning Results

The global path planning was based on Boustrophedon Cellular Decomposition approach for coverage area. In this work we proposed a *modified* BCD approach which allows the algorithm to jump to adjacent cells before it completes covering current cell if an optimized solution exists. Experimental results showed up to 25% performance improvement in terms of total distance covered by the robot covering same environment. Moreover, unlike other modern approaches, this method is not based on AI or neural networks so it is computationally inexpensive.



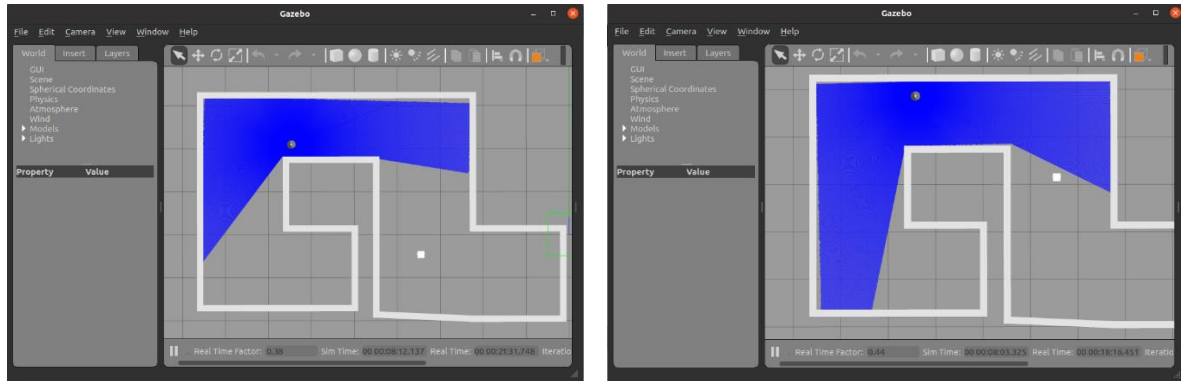
**Figure 6.2:** Total distance covered reduced by modified BCD approach

Example shown in **Figure 6.2** shows reduction of about 3.25 meters of distance covered by the robot in about 6 square meter area. The total distance covered by the robot within these cells using normal BCD approach was 32.25m whereas the robot covered same area by travelling 29m using modified BCD approach.

### 6.4 Local Path Planning and Navigation

In this work we proposed a unique two-level approach in order to generate trajectory to counter moving or *unexpected* obstacles. Velocity obstacle approach is a popular method in mobile robotics to deal with moving obstacles. However, this approach is computationally expensive. We used a simpler *intersection of lines* approach as a filter to refer only complicated cases to the *VO* approach. Using this approach significantly improved the performance. As a performance indicator we ran two instances of the program on same PC - with the same moving obstacles and environment. First, we ran the code with velocity obstacle as the only approach

for local planner for our test map. Next, we completed navigation for the same environment using proposed two-level approach. Results seen in **Figure 6.3** show real-time factor of latter increase by almost 15% as compared to the method where first approach was not used. This proves that our proposed method reduces stress on the processor.



Velocity Obstacle Approach  
Real-time Factor 0.38 (Slower)

Two-level Approach  
Real-time Factor 0.44 (Faster)

**Figure 6.3:** Performance comparison between two-level and single level VO approach

## CHAPTER 7: CONCLUSION AND FUTURE WORK

In this work we discussed and analyzed different methods for coverage problem found in literature. Among those methods, Boustrophedon Cellular Decomposition was found to be the most popular cellular decomposition technique among researchers. Most modern coverage path planning methods that make use of Artificial Intelligence or Neural Networks are also based on BCD approach. However, all these methods suggest improvements in Boustrophedon Cellular Decomposition in their own way. We suggested an effective and computationally inexpensive technique that can reduce the distance that needs to be travelled by the robot in order to cover the environment. The number of waypoints generated by global planner were reduced to only *critical waypoints* in order for the navigation function to act smoothly and reduce the number of local goals.

We also suggested a two-level approach in order to avoid moving obstacles. The first level is a memory efficient algorithm that refers only complicated cases to the level two based on Velocity Obstacles approach. This reduces stress on the processor and improves performance.

Results obtained showed significant improvements in terms of performance and memory. Mapping, global path planning, navigation and local path planning - all modules were successfully tested in simulated environment of ROS. The environment was mapped prior to applying path planning technique.

Some suggestions for future work are as under:

1. Implementation in real world environment.
2. Online path planning with no prior mapped environment.
3. Integration of other sensors like RGB-D or Inertial Measurement Unit (IMU) for localization.
4. Handling dynamic obstacles with non-zero acceleration.

## References

- [1] Iwan Ulrich, Francesco Mondada, and J. D. Nicoud, "Autonomous Vacuum Cleaner," *Robotics and Autonomous Systems*, vol. 19, no. 3, 1997.
- [2] X. T. Yan, A. Bianco, C. Niu, R. Palazzetti, G. Henry, Y. Li, W. Tubby, A. Kisdi, R. Irshad, S. Sanders, R. Scott, "The AgriRover: A Reinvented Mechatronic Platform from Space Robotics for Precision Farming," in *Reinventing Mechatronics: Developing Future Directions for Mechatronics*, Cham, Springer International Publishing, 2020, pp. 55-73.
- [3] Jörg Marvin Gülzow et al., "Recent Developments Regarding Painting Robots for Research in Automatic Painting, Artificial Creativity, and Machine Learning," *Applied Sciences* 10(10):3396, 2020.
- [4] P. Gonzalez de Santos, J.A. Cobano, E. Garcia, J. Estremera, M.A. Armada, "A six-legged robot-based system for humanitarian demining missions," *Mechatronics*, vol. 17, no. 8, pp. 417-430, 2007.
- [5] Ernest L. Hall, S. J. Oh, E. Kattan, Z. L. Cao and Y. Y. Huang, "Experience With a Robot Lawn Mower," *Robots* 10, 1986.
- [6] B. Nasirian, M. Mehrandezh, F. Janabi-Sharifi, "Efficient Coverage Path Planning for Mobile Disinfecting Robots Using Graph-Based Representation of Environment," *Frontiers in Robotics and AI*, vol. 8, 2021.
- [7] M. Farsi, K. Ratcliff, J.P. Johnson, C. Allen, K. Z. Karam, R. Pawson, "Robot control system for window cleaning," *Proceedings of 1994 American Control Conference*, 1994.
- [8] Z. L. Cao, Y. Huang and E. L. Hall, "Region Filling Operations with Random Obstacle Avoidance for Mobile Robots," *Journal of Robotic Systems*, 5 (2), 87-102, 1988.
- [9] Esther M. Arkin and Refael Hassin, "Approximation algorithms for the Geometric Covering Salesman Problem," *Discrete Applied Mathematics*, vol. 55, no. 3, pp. 197-218, 1994.
- [10] H. Choset, "Coverage for robotics – A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, 2001.
- [11] C. S. Tan, R. Mohd-Mokhtar and M. R. Arshad, "A Comprehensive Review of Coverage Path Planning in Robotics Using Classical and Heuristic Algorithms," *IEEE Access*, vol. 9, pp. 119310-119342, 2021.
- [12] J. Barraquand, J. C. Latombe, "Robot Motion Planning: A Distributed Representation Approach," *The International Journal of Robotics Research*, p. 628–649, 1991.
- [13] Shannon V. Spires, Steven Y. Goldsmith, "Exhaustive Geographic Search with Mobile Robots along Space-Filling Curves," *1st International Workshop in Collective Robotics*, 1998.

- [14] Hans P. Moravec, Alberto Elfes, "High resolution maps from wide angle sonar," *IEEE International Conference on Robotics and Automation*, vol. 2, 1985.
- [15] Przemyslaw A. Zielinski, "An Approximation Technique for Stochastic Environmental Modeling," *Water Resources Research*, vol. 29, pp. 3379-3387, 1993.
- [16] Y. Gabriely, E. Rimon, "Spiral-STC: An on-line coverage algorithm of grid environments by a mobile robot," *IEEE International Conference on Robotics and Automation*, vol. 1, 2002.
- [17] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, C. Bustacara, "BSA: A Complete Coverage Algorithm," *IEEE International Conference on Robotics and Automation*, 2005.
- [18] H. Choset and P. Pignon, "Coverage Path Planning: The Boustrophedon Cellular Decomposition," *Zelinsky, A. (eds) Field and Service Robotics. Springer, London*, 1998.
- [19] A. Pirzadeh and W. Snyder, "A unified solution to coverage and search in explored and unexplored terrains using indirect control," *Proc. Conf. IEEE Int Robotics and Automation*, pp. 2113-2119, 1990.
- [20] Fumio Yasutomi, Daizo Takaoka, Makoto Yamada, and Kazuyoshi Tsukamoto, "Cleaning Robot Control," *Proceedings of IEEE International Conference on Robotics and Automation*, 1988.
- [21] P. W. Tse, S. Lang, K. C. Leung and H. C. Sze, "Design of a navigation system for a household mobile robot using neural networks," *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence*, vol. 3, pp. 2151-2156, 1998.
- [22] B. Chee, S. Lang and P. Tse, "Fuzzy Mobile Robot Navigation and Sensor Integration," *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, vol. 1, pp. 7-12, 1996.
- [23] Yili Fu and S. Y. T. Lang, "Fuzzy logic based mobile robot area filling with vision system for indoor environments," *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99*, pp. 326-331, 1999.
- [24] A. Zelinsky, R.A. Jarvis, J. C. Byrne, S. Yuta, "Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot," *In Proceedings of International Conference on Advanced Robotics*, pp. 533-538, 1993.
- [25] Y. Gabriely, E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of Mathematics and Artificial Intelligence*, vol. 31, p. 77-98, 2001.
- [26] Young-Ho Choi, Tae-Kyeong Lee, Sang-Hoon Baek, Se-Young Oh, "Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform," *IEEE International Conference on Intelligent Robots and Systems*, p. 5788-5793, 2009.
- [27] Susan Hert, Sanjay Tiwari, Vladimir Lumelsky, "A terrain-covering algorithm for an AUV," *Autonomous Robots*, vol. 3, pp. 91-119, 1996.

- [28] E.U Acar, H. Choset, "Critical point sensing in unknown environments," *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, 2000.
- [29] E. Acar, H. Choset, "Sensor-based coverage of unknown environments: Incremental construction of morse decompositions," *International Journal of Robotics Research* 21 (4), p. 345–366, 2002.
- [30] P. N. Atkar, H. Choset, A. A. Rizzi, E. U. Acar, "Exact cellular decomposition of closed orientable surfaces embedded in  $R^3$ ," *Proceedings of ICRA Robotics and Automation IEEE International Conference*, vol. 1, p. 699–704, 2001.
- [31] E. Garcia, P. G. de Santos, "Mobile-robot navigation with complete coverage of unstructured environments," *Robotics and Autonomous Systems*, vol. 46, p. 195–204, 2004.
- [32] E. U. Acar, H. Choset and J. Y. Lee, "Sensor-based coverage with extended range detectors," *IEEE Transactions on Robotics*, vol. 22, pp. 189-198, 2006.
- [33] Z. J. Butler, A. A. Rizzi and R. L. Hollis, "Contact sensor-based coverage of rectilinear environments," *Proc. IEEE Int Intelligent Control/Intelligent Systems and Semiotics Symposium*, pp. 266-271, 1999.
- [34] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," *Proc. ICRA Robotics and Automation IEEE Int. Conf*, vol. 1, pp. 27-32, 2001.
- [35] S. C. Wong and B. A. MacDonald, "A topological coverage algorithm for mobile robots," *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, vol. 2, pp. 1685-1690, 2003.
- [36] C. Hofner and G. Schmidt, "Path planning and guidance techniques for an autonomous mobile cleaning robot," *Robotics and Autonomous Systems*, vol. 14, p. 199 – 212, 1995.
- [37] R. N. De Carvalho, H. A. Vidal, P. Vieira and M. I. Ribeiro, "Complete coverage path planning and guidance for cleaning robots," *Proceedings IEEE Int Industrial Electronics*, vol. 2, pp. 677-682, 1997.
- [38] Simon X. Yang and Chaomin Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 718-724, 2004.
- [39] X. Qiu, J. Song, X. Zhang and S. Liu, "A complete coverage path planning method for mobile robot in uncertain environments," *Proc. Sixth World Congress Intelligent Control and Automation WCICA*, vol. 2, p. 8892–8896, 2006.
- [40] V. J. Lumelsky, S. Mukhopadhyay and K. Sun, "Dynamic path planning in sensor-based terrain acquisition," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 4, p. 462–472, 1990.
- [41] R. A. Russel, "Heat trails as short-lived navigational markers for mobile robots," *Proc. Conf. IEEE Int Robotics and Automation*, vol. 4, pp. 3534-3539, 1997.

- [42] J. Y. Park and K. D. Lee, "Study on the cleaning algorithm for autonomous mobile robot under the unknown environment," *Proceedings 6th IEEE International Workshop on Robot and Human Communication. RO-MAN'97*, pp. 70-75, 1997.
- [43] P. A. Jimenez, B. Shirinzadeh, A. Nicholson and G. Alici, "Optimal area covering using genetic algorithms," *IEEE/ASME international conference on advanced intelligent mechatronics*, pp. 1-5, 2007.
- [44] M. Bosse, N. Nourani-Vatani and J. Roberts, "Coverage Algorithms for an Under-actuated Car-Like Vehicle in an Uncertain Environment," *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 698-703, 2007.
- [45] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of Field Robotics*, vol. 26, no. 8, pp. 651-668, 2009.
- [46] P. Fiorini, Z. Shiller, "Motion Planning in Dynamic Environments Using Velocity Obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760-772, 1998.
- [47] N. Badler, J. O'Rourke, H. Toltzis, "A spherical representation of a human body for visualizing movement," *Proceedings of the IEEE*, vol. 67, pp. 1397-1403, 1979.
- [48] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, pp. 995-1001, 2000.
- [49] J. Borenstein, Y. Koren, "Real-Time Obstacle Avoidance for Fast Mobile Robots," *IEEE Transactions on Systems Man and Cybernetics*, pp. 1179 - 1187, 1989.
- [50] Hadi Gaderi, Meysam Yadegar, Nader Meskin, Mohammad Noorizadeh, "A Novel Proportional Navigation based Method for Robotic Interception Planning with Final Velocity Control," *IEEE Access*, vol. 9, pp. 106428-106440, 2021.
- [51] J. Barraquand, J. C. Latombe, "Robot Motion Planning: A Distributed Representation Approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628-649, 1991.



### **Completion Certificate**

It is certified that the thesis titled “**Path Planning and Navigation of Robot with 2D Lidar Feedback**” submitted by CMS ID. 00000276308, NS Muhammad Awais Khan Niazi of MS-2018 Mechatronics Engineering is completed in all respects as per the requirements of Main Office, NUST (Exam branch).

Supervisor: \_\_\_\_\_

Dr. Kunwar Faraz Ahmed Khan

Date: \_\_\_\_ July 2022