Complete Coverage Path Planning by an Autonomous Vehicle using Neural Networks

Phase-1 Report submitted in partial fulfilment of the requirement for the degree of

Master of Technology

Submitted by

Mallavaram Sai Harini

Roll No. 214102508

Supervisor

Dr. Indrani Kar

Department of Electronics and Electrical Engineering

Indian Institute of Technology Guwahati

Assam - 781039, India

Nov 11, 2022

ACKNOWLEDGEMENT

On the path of learning, the presence of an experienced guide is indispensable. I would like to thank my supervisor, Dr. Indrani Kar, for her invaluable help, constant support, and persistent guidance throughout the course of this project. I express my sincere gratitude to the Indian Institute of Technology Guwahati for providing me with the opportunity and resources to accomplish my project for phase 1.

ABSTRACT

Much of the focus of research efforts in path plans for mobile robots focus on finding a path from the start point to the destination minimizing one or more parameters such as energy consumption, length of path or travel time. Complete coverage path is the planned path the robot sweeps over all areas of a particular space of two dimensional (2D) environment avoiding obstacles in a systematic and efficient way with the consideration of minimum cost criteria. Deep learning based complete coverage path planning using Graph Neural Network model is studied. Data is to be sent in the form of graph at the input layer of graph neural network. For the static environment algorithms are developed by taking reference from Branch and Bound Algorithm, Distance transform and Complete Coverage D* (CCD*) algorithms. In Branch and Bound technique complete path coverage problem is considered as travelling salesmen problem. Data set is created using these algorithms and the results are verified.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction to Complete Coverage Path Planning

Mobile Robot Complete Coverage Path Planning (CCPP) has wide range of applications and are of good practical value. CCPP has received much attention recently in fields of Robotics and Artificial Intelligence. In 2D environment, CCPP is a special type of trajectory creation that necessitates the robot path traverse the entire workspace avoiding obstacles minimizing the cost. The practical applications include Vacuum robots,Lawn Mowers, Cleaning Robots, Land mine detectors, Automated harvesters,Autonomous under water vehicles,Wall climbing robots etc..

Different algorithms are developed for CCPP considering minimum cost i.e., minimum length ,minimum turn rate, minimum time and minimum energy consumed by a robot. These algorithms are created by modifying Branch and Bound, Distance Transform, Path Transform and Complete Coverage D* (CCD*) algorithms. 2D environment is divided into grid of cells where dimension of each cell is equal to the dimension of robot. Data set of different 2D grids with different positions of obstacles are created and the complete path coverage is found by using these algorithms. Grid is converted in the form of a graph represented using adjacency list and this adjacent list is sent as input to the input layer of Graph Neural Network and it is trained from the output of the algorithms. Neural network model is to be created and used to predict the complete path of any static environment with less computational time compared to other algorithms.

1.2 Literature Survey

Finding a trajectory for an autonomous vehicle that allows it to travel over all of the search area's points while avoiding obstacles is the goal of coverage path planning (CPP). Deep learning based Coverage Path Planning Network(CPPNet) is created. Input of a Convolutional Neural Network is a graph based representation and its output is a edge probability heat graph, where each edge's value represents the probability that it will be a part of the ideal TSP tour. CPP Net is trained using the output from the 2 opt algorithm.The final optimised tour is then chosen via a greedy search. CPP Net provides near optimal path taking significantly less computational time in partially unknown and dynamic environments as described in [2].

Complete coverage path planning algorithm developed based on the D* algorithm by Stentz (1994),and the PT algorithm by Zelinsky et al. (1993), and these two algorithms are briefly surveyed in this section.In this paper, a novel total coverage path plan, the complete coverage D* (CCD*) algorithm. CCD* algorithm use Path transform and D* search based on the high resolution occupancy grid algorithm map to quickly replan in a changing environments and accounting for the robot's size, including a focus on path reduction and motion safety [3].

The challenge of complete coverage will be addressed in this research using an expansion of the distance transform path planning methodology. Simulated experimental data and an implementation on the Yamabico mobile robot are used to demonstrate the findings [4].

The complete coverage path planning of mobile robot in globally unknown environments for mobile robot with obstacle avoidance is developed using biologically inspired neural networks, rolling path planning and heuristic searching approach. Neural network model named shunting model is used to model the environment and calculate the environment information locally each time considering a local planning window while rolling path planning and heuristic searching algorithms are used for path planning.Simulations indicate that the proposed method is effective in an uncertain environment with reduced visibility burden calculation suggesting excellent efficacy and applicability [1].

A method for dealing with the Coverage Path Planning (CPP) problem for a fleet of AI-driven UAVs while accounting for congestion, collision avoidance, and path efficiency. The technique makes use of both decentralised Artificial Neural Networks (ANN) and the A* pathfinder. The neural network establishes a link between a UAV's present state and the optimum action to perform at each time step. For the single-UAV situation, the exploration approach is recorded in a labelled database; the neural network learns and duplicates it in the multi-UAV case, being able to generalise the gained abilities to new maps not present in the database [5]. Coverage Path Planning for Decomposition Reconfigurable Grid-Maps Using Deep Reinforcement Learning Based Travelling Salesman Problem.TSP is solved by utilising Reinforcement Learning to train a Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) layers (RL) [6]. An enhanced biologically inspired neural network approach for wall climbing robot path planning has been suggested. When the wall climbing robot executes coverage tasks for bridge towers and vertical walls, it consumes varying amounts of energy while ascending and descending. The energy consumption model of a wall climbing robot is constructed, and then a new next step selection strategy is proposed, which can ensure complete coverage planning while also planning the best path by considering energy consumption and the Heuristic factor Hj, which is defined from the perspective of global energy consumption [7]. Planning and correction of the (Autonomous Under Water Vehicles) AUV coverage path in real time - The algorithm is based on the idea that the optimal coverage of any polygon must be composed of the line segments that run parallel to its edges [8].

Incorporating heuristic information from the problem domain into a formal mathematical theory of graph searching and illustrating an optimality property of a class of search techniques. It is computationally efficient, ensures total coverage, and does not suffer from the problem of local extrema. Its performance is proven by: i)high-fidelity simulations on Player/Stage and ii) actual testing on autonomous cars in a laboratory setting [9]. A shunting equation developed from Hodgkin and Huxley's (1952) membrane equation characterizes the dynamics of each neuron in the topologically structured neural network. Local lateral connections exist between neurons. The robot path is produced independently using the neural network's dynamic activity landscape and the prior robot position[10]. The CCNN algorithm enhanced the movement directions of the path in grid maps, reducing turning-angles and making the path smoother. In addition, an improved A* algorithm that may effectively reduce path turns is described in order to avoid the deadlock[10].

Cooperative Autonomy for Resilience and Efficiency, a distributed algorithm that not only offers resilience to the robot team against individual robot failures, but also increases overall operational efficiency through event-driven replanning [11].Using multi-AUVs cooperative full coverage path planning approaches based on the GBNN algorithm. All AUVs in the GBNN and discrete programming AUV cooperative full coverage algorithm exchange environment information, and each AUV considers other AUVs in the vicinity as moving obstacles, ensuring AUV detection task coverage in the allotted water space without collision [12]

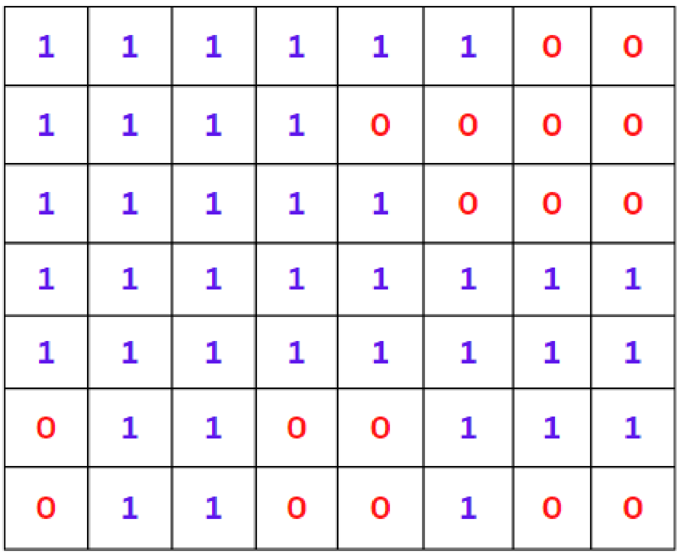
An enhanced CCPP technique based on bioinspired neural networks and pedestrian position prediction with "active obstacle avoidance strategy" The suggested method begins by predicting the moving pedestrian's trajectory using the Kalman filter technique. The projected location is then fed into the overall neural network, with the external inputs for the activity landscape changed. Furthermore, an optimum avoidance method for mobile robots is devised [13].

Chapter 2

Technical Background and Preliminaries

2.1 Representation of floor in the form of a grid map

The entire two dimensional floor with obstacles is divided into grids of cells where the dimension of each cell is equal to the dimension of the robot. It is represented as :

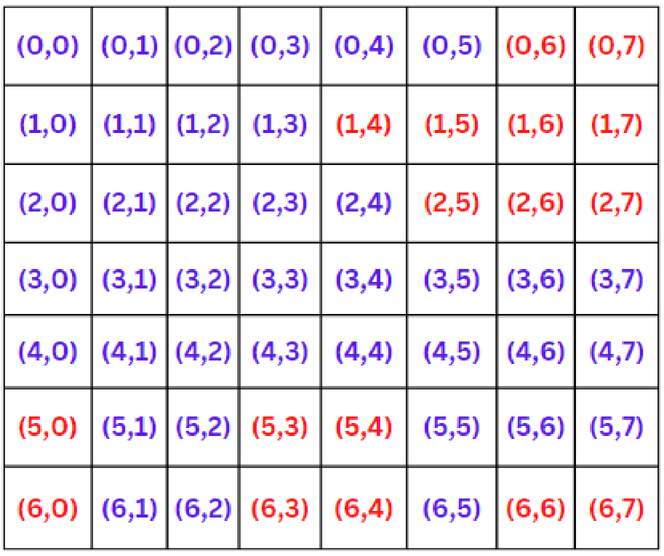


1	1	1	1	1	1	0	0
1	1	1	1	0	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1
0	1	1	0	0	1	0	0

Figure 2.1: Grid map of a floor

Floor is divided into cells where each cell can have a value of 0 or 1. 0 represents the obstacles in the floor and 1 represent the free space.Robot has to traverse the cells with value 1 at least once for complete path coverage.

Grid is divided into rows and columns. Each cell is identified uniquely by its row and column number.



(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
(6,0)	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)

Figure 2.2: Cell representation

Nested list is used to store grid map in python code. It is defined as floor_map=[[1,1,1,1,1,1,0,0],[1,1,1,1,0,0,0,0],[1,1,1,1,1,0,0,0],[1,1,1,1,1,1,1,1], [1,1,1,1,1,1,1,1], [0,1,1,0,0,1,1,1],[0,1,1,0,0,1,0,0]]

2.2 Grid map representation using Graph Data Structure

A Graph is a non-linear data structure consisting of Vertices(V) and Edges(E). Vertices are also referred as nodes. Edges are lines that connect any two vertices in a graph.In our problem each cell is considered as a vertex or a node. Here we consider the distance between two adjacent nodes (node that is left or right or top or bottom of other node) as 1 unit.
In our scenario we are taking two cases of adjacent nodes.
Case 1: Robot can take turn of only 90 degrees.

Here for each node adjacent node is at left,right,top,bottom.

Case 2: Robot can take turn of 90 degrees and 45 degrees.

Here for each node adjacent node can be at left,right,top,bottom,diagonal.

The distance (edge) between diagonal cells is taken as $\sqrt{2}$ units

For example consider cell(1,1) from the above grid map

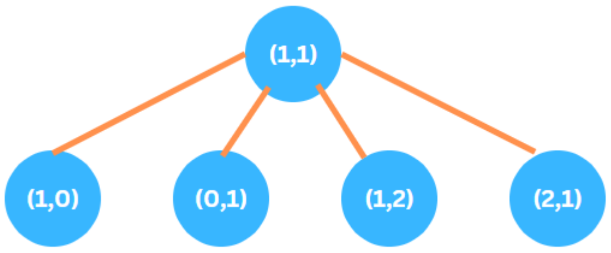


Figure 2.3: Case 1: Adjacent cell representation

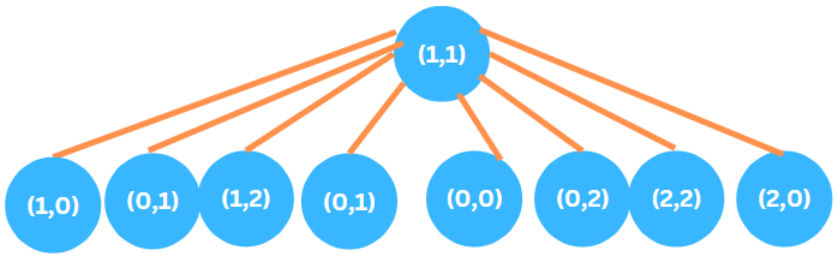


Figure 2.4: Case 2: Adjacent cell representation

In case 1 a cell can have maximum 4 adjacent cells or nodes

In case 2 a cell can have maximum 8 adjacent cells or nodes including diagonals

The following two are the most commonly used representations of a graph.

1. Adjacency Matrix

2. Adjacency List

1. Adjacency Matrix representation:

An adjacency matrix is a two-dimensional array with the dimensions V x V, where V is the number of vertices in a graph.Let the adjacent matrix be adj[][]. When there is no edge between cells then adj[i][j]=0. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w. Adjacency matrix for undirected graph is always symmetric.

Adjacency matrix consumes more space of $\{O(V^2)\}$.The amount of space consumed by adjacency matrix is same even if the graph is sparse.

2. Adjacency List representation:

An adjacency list is a collection of lists used to represent a finite graph. Each unordered list within an adjacency list describes the set of neighbors of a that particular vertex in the graph. In our problem Adjacency list is implemented as nested list where the adjacent list for a particular cell contains the neighbouring cells as mentioned in case 1 and case 2. Consider the adjacent list of grid map as adj_list then for a cell [i,j] the adjacent nodes are found from adj_list[i][j] element in an adjacency list matrix.Obstacle cells are considered as adjacent cells to any cell.

As the floor map is sparse, for the effective implementation Adjacency list representation is preferred compared to Adjacency matrix representation.

2.3 Tree Data Structure

A tree is a non-linear and hierarchical data structure composed of a collection of nodes, each of which holds a value as well as a list of pointers to other nodes (the "children").It is made up of a central node, structural nodes, and sub-nodes that are linked together via edges. A tree data structure comprises roots, branches, and leaves that are all related to one another.

The starting position of a robot i.e., the start cell is considered as a root node.The path is traversed until all the nodes are covered. For example consider a simple 4*4 grid.

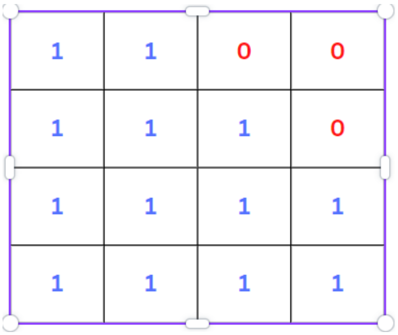


Figure 2.5: 4*4 Grid

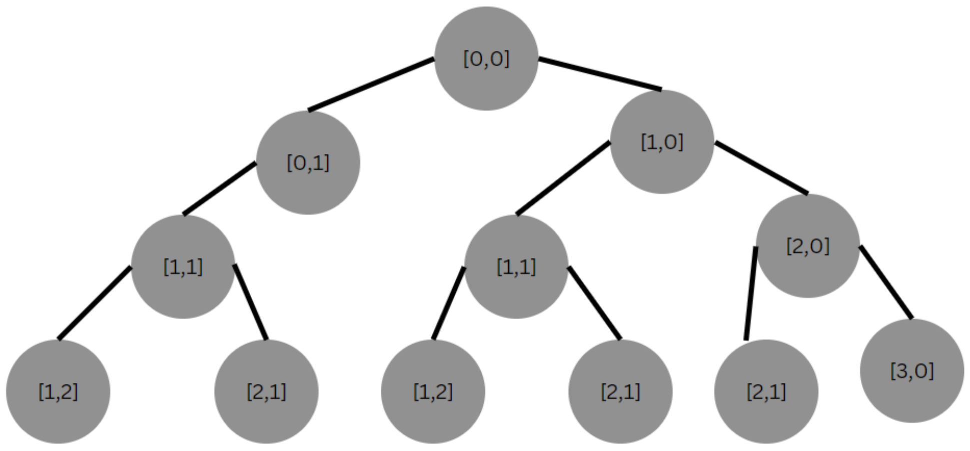


Figure 2.6: Tree structure

2.4 Travelling Salesman Problem using Branch and Bound Algorithm

Travelling Salesman Problem: The travelling salesman problem (TSP) is an algorithmic problem in which the goal is to discover the shortest path between a collection of nodes or vertices and all the vertices that must be visited. The cities that a salesman could visit are represented by the nodes or vertices in the problem statement. The salesman's objective is to limit both travel cost and distance travelled to a minimum.
Branch and Bound Algorithm: Branch and Bound approach computes a bound on the best potential answer that we can receive if we down this node for the present node in the tree. If the bound on the best potential solution is worse than the existing best (best computed thus far), we disregard the subtree rooted with the node.
It should be noted that the cost of passing via a node comprises two expenses.
1) The cost of travelling from the root to the node (When we reach a node, we have this cost computed)
2) The cost of getting a response from the present node to a leaf (compute a bound on this cost to decide whether to ignore subtree with this node or not).

Similar to backtracking but also uses state space tree or a graph represen-

tation for solving a problem.It is used for solving optimization problem.For exploring the solutions of a problem we go for Breadth First Search first and the select a particular node whose cost is minimum and explore the adjacent nodes of the minimum cost node and the process continues whereas the back-tracking does Depth First Search. Implementation: To find the shortest tour that is going through all the vertices once.
Consider the following example:

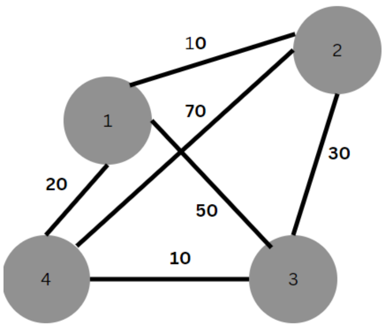


Figure 2.7: TSP Graph

As can be seen in the graph below, there are 4 vertices in the graph. We must identify the shortest path that traverses all of the vertices once and returns to the beginning vertex. We assume the initial vertex to be 1.
Explanation using state space tree: Root vertex is 1.It can traverse through any of its adjacent vertex.Cost is noted for each edge level wise (breadth wise).Note the node which has minimum cost.Now traverse through all its adjacent nodes which are not covered till now and find the minimum node among them and this process continues until all the nodes are covered.

Any node is represented by i/cost where i is the vertex and cost cost traversed from root to this node. Initially all adjacent cells of root and their cost is found. Then it checks all the leaf nodes with minimum cost and it is the next cell in the path. Then all its adjacent nodes are found and the process repeats until all the nodes are traversed once.After finding one path its cost is compared all the current leaf nodes and the paths containing cost greater than the calculated path are removed from the tree.This process is repeated until there is only one path.

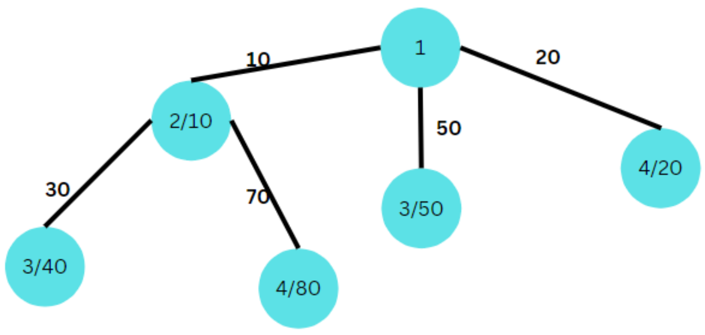


Figure 2.8: Branch and Bound Implementation using State Tree Diagram

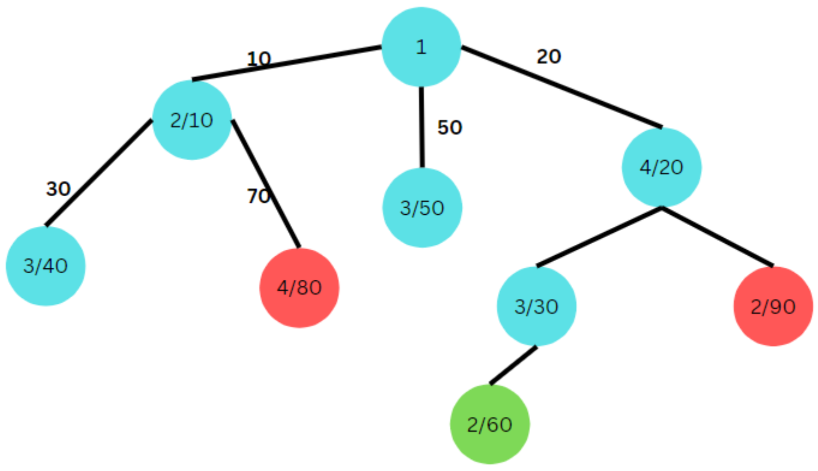


Figure 2.9: State Tree Diagram 2

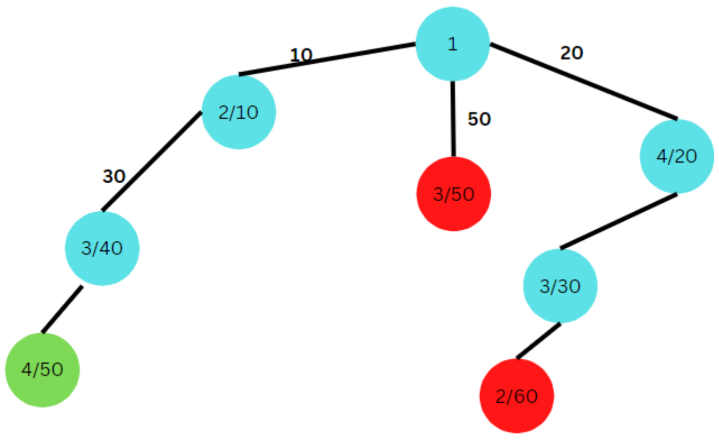


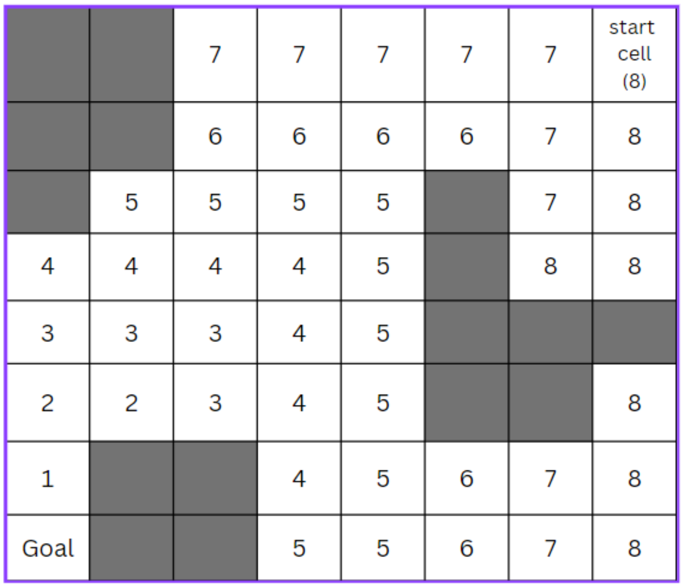
Figure 2.10: State Tree Diagram 3

The final path traversing all nodes with minimum cost is found as 1-2-3-4

2.5 Distance Transform Algorithm

This approach considered the task of path planning to finding paths from the goal location back to the start location.A distance wave front is propagated by the path planner. through all of the environment's open space grid cells from the destination cell. The distance wave front sweeps around all the obstacles, and finally through all of the available open space in the environment. For every place of departure inside the environment showing the mobile robot's beginning position, The fastest route to the target is found by going downhill through the most difficult downhill path if there is no downward path available, and if the first cell reaches a plateau, it may be argued that no path exists from the start cell to the goal cell i.e. the goal is unreachable.
Consider the following grid map. Distance transform algorithm is as follows: Initially all the obstacles are marked in the map. We can select start and Goal cell as any cell. They can vary. The distance wave front starts at the goal cell, The value of this cell can be marked as 0.Then it checks all its adjacent cells including diagonals and they are marked as 1.And all the cells adjacent to cells with value 1 are marked as 2. The distance wave front then travels through all the cells in the grid map and the values are marked without considering obstacle cells.

The complete coverage path can be planned as: Path starts at the start cell and checks for the cells with greater value among all adjacent cells and traverse through that path. From that cell it checks its adjacent cells and the path continues until it reaches the goal cell. It is ensured that by following this algorithm all the cells in the grid are travelled and complete coverage path planning is achieved.
One significant advantage that distance transform path planning has over other path planning methods is that it can easily be induced to exhibit different types of robot navigation behaviours. distance transform could be modified to produce "conservative", "adventurous" path planning behaviours in addition to the "optimum" i.e. shortest path behaviour.
This algorithm considers the distance between adjacent cells as 1 unit including



▨	▨	7	7	7	7	7	start cell (8)
▨	▨	6	6	6	6	7	8
▨	5	5	5	5	▨	7	8
4	4	4	4	5	▨	8	8
3	3	3	4	5	▨	▨	▨
2	2	3	4	5	▨	▨	8
1	▨	▨	4	5	6	7	8
Goal	▨	▨	5	5	6	7	8

Figure 2.11: Distance Transform

the diagonal cells. There is no additional cost given for taking turns.

Distance Transform Algorithm :

```
Set Start Cell to Current Cell Set all Cells to Not Visited
    Loop
    Find unvisited Adjacent cell with highest DT
    If No Adjacent Cell found then
    Mark as Visited and Stop at Goal
    If Neighbouring Cell DT less than or equal to Current Cell DT then Mark
    as Visited and Stop at Goal
    Set Current cell to Neigbouring cell
    Loop End
```

2.6 Complete Coverage D* Algorithm for Path Planning of a Floor-Cleaning Mobile Robot

Inspired by Zelinsky et al Path transform (PT) technique, an unique complete coverage approach complete coverage D* (CCD*) is created, which is based on the D* search of the environment's two-dimensional occupancy grid map.

Unlike the original PT method, the CCD* algorithm considers the robot's dimensions, with an emphasis on motion safety and path length and search time savings. Furthermore, the suggested CCD* method may generate new complete coverage paths when the environment changes.
Path transform algorithm is similar to Distance Transform Algorithm
The suggested approach extends the PT algorithm to account for any free cells in the grid map that the robot's mask covers with its dimensions. To provide quick replanning capabilities in changing surroundings, an additional enhancement is created by employing D* instead of the wavefront algorithm. CCD* algorithm plans the first complete coverage path while the robot is at rest at the start location, and then how it replans the path when the robot encounters obstacles on its way to the final point of the entire coverage path.The D* method is triggered from the robot's position rather than the specified goal position (the goal node is replaced by the start node S in initial planning or the node R in replanning). As a result, the ultimate position of the entire coverage path varies at each algorithm iteration as the robot moves.

2.7 Complete Coverage Path Planning Method for Mobile Robot in dynamic Environments

Total coverage for mobile robot motion planning with obstacle avoidance, a path planning system based on biologically inspired neural networks, rolling path planning, and heuristic searching is provided. The biologically inspired neural network is used to represent the environment and calculate environment information, while the path planning is done using the rolling planning approach and the heuristic searching algorithm.

Hodgkin and Huxley were the first to suggest an electrical circuit-based model for a patch of membrane in a biological brain system. Grossberg expanded on this model and published the shunting model, which was taken from Hodgkin and Huxley's membrane model and led to a plethora of model modifications and uses.

$$\frac{dx_i}{dt} = \text{-}Ax_i + (B - x_i)\left([I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+\right)(D + x_i)[I_i]^- \tag{2.1}$$

where xi is the neural activity (membrane potential) of the i th neuron. Parameters A, B and D are non-negative constants representing the passive decay rate, the upper and lower bounds of the neural activity respectively, and K is the number of neural connections of the i th neuron to its

$$I_{\,i} = \begin{cases} E, \; \textit{If it is an uncovered area} \\ \text{-}E, \; \textit{If it is an obstacle area} \\ 0, \; \textit{otherwise} \end{cases} \tag{2.2}$$

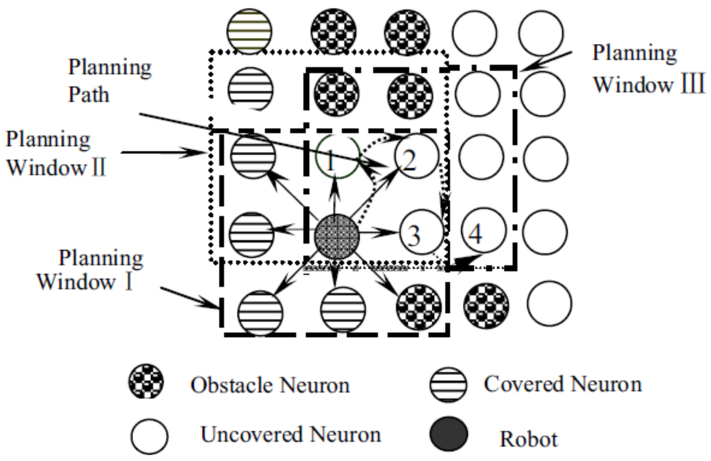


Figure 2.12: Representation of a local window [1]

Using shunting equation neuron values of a local window are calculated and accordingly the path moves to the next neuron

2.8 Introduction to Graph Neural Networks

A GNN is an optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries (permutation invariances). This GNN employs a single multilayer perceptron (MLP) (or your preferred differentiable model) on each graph component; this is referred to as a GNN layer. We apply the MLP to each node vector and obtain back a learnt node-vector. We repeat this procedure for each edge, learning a per-edge embedding, as well as for the global-context vector, learning a single embedding for the whole graph. Adjacency List or Adjacency Matrix is passed at the input layer of the Neural Network.

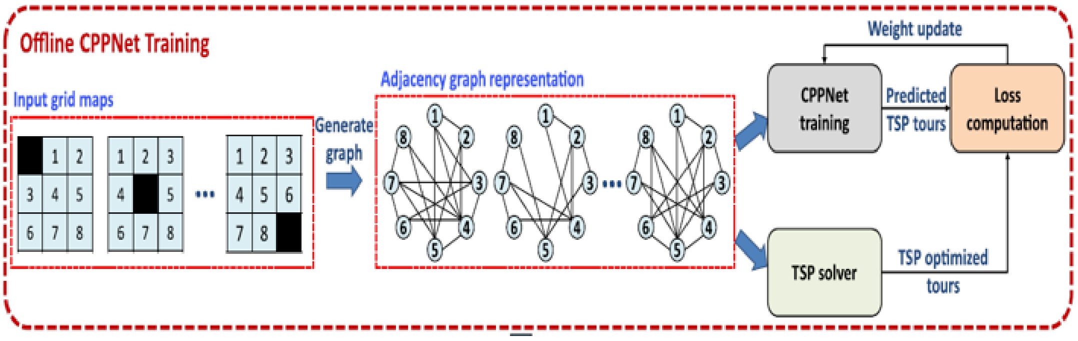


Figure 2.13: GNN Implementation[2]

First each grid map from data set is converted to adjacency list or adjacency matrix . It is passed at the input layer. The NN model is trained using the results in data set.

Chapter 3

Design and implementation of Algorithms for CCPP

3.1 Algorithm1

First algorithm is designed taking reference from Branch and Bound Algorithm. Branch and Bound follows combination of BFS and DFS Techniques which ensures that all the nodes are traversed only once. For our case in some scenarios in order to cover the complete path the robot might have to traverse back when all the surrounding cells of a particular cell are visited . So in this algorithm backtracking of cells is allowed when a certain criteria is satisfied.

CCPP is an NP Hard problem. It takes exponential time complexity using Greedy or DFS or backtracking techniques.Using Dynamic Programming also the time complexity does not vary much.Compared to these methods Branch and Bound method has somewhat reduced time complexity. But it is also high when implemented in practical scenario. Here turn angle of only 90 degrees is considered and diagonal cells are not considered adjacent.

3.2 Algorithm 2

It is developed taking reference from Distance Transform Algorithm and modified according to the requirement.Both the algorithms are defines as follows:

Algorithm 1 Find complete coverage path

```
1: $floor\_map[]$ : Represent the grid map of a floor in the form of a nested list
2: $start$ : Start cell is assigned as [0,0]
3: $adj\_list[]$ : Nested list - To get the adjacency list of floor_map
4: $total$ : Count of total no of empty cells in a floor_map
5: $TreeNode$ : It is a class following Tree Data Structure. Each cell in the
   path is represented as the object of a TreeNode. The information of each
   cell in path is described by the attributes and methods defined in TreeNode.
       Attributes:
       $data-$ Represents Cell value
       $cost-$ Cost of path traversed from the root node till this node
       $back-$ Back tracking length
       $length-$ Tree level of that node
       $children[]-$Children nodes
       $parent-$ Parent node
       Methods:
       $add\_child$ : This method adds child nodes to a node
       $get\_cost$ : This method calculates the path traversed to a node.
       Calculates the angle and checks if there is a turn. It also takes into
       account of no of turns.For a turn additional cost is added
6: $findpath(start, adj, cost, turn, floor\_map, total)$ : Method to compute
   complete coverage path
       $root$ : Create new TreeNode object with data as start cell
       $lnodes[]$ : List of all Leaf TreeNode objects are stored in it.Initially
       root is appended to lnodes
       $paths$ :List of all paths are stored
       $while(Runs\ until\ lnodes\ list\ becomes\ empty)$ :
           $mincost\_index(lnodes)$ : Method to get the index of min cost node
           $temp$ : Node with minimum cost of all the leaf nodes
           $for(length\ of\ adjacent\ nodes\ of\ temp)$ :
               $check-$ ith Adjacent node
               Check if this node is traversed previously in the path.
```

```
            $if$ : If yes then if all the adjacent cells of this node are traversed
            then the path can continue with this cell creating a new object
            with data as this cell allowing backtracking and back attribute
            of this node is increases by 1 or if this node is not traversed
            previously the path can continue with this cell as new node.
            $else$ : Continue with the next iteration
            New node is added as child to temp.Path length and node level
            is increased by 1
            $if(node.length - node.back == total)$ :If yes this path is added
            to paths list
                This path length is compared to all the leaf nodes and if
                the leaf node length is greater than the path length then leaf
                node path is removed
            $else$ : Continue with the next adjacent cell of temp.
        If all the adjacent nodes are verified then get the new node of
        minimum cost from lnodes list and continue the process until the
        lnodes list is empty
    The path with minimum length can be obtained from paths list.
```

Algorithm 2 : Complete coverage path planning with the modification of Distance Transform Algorithm

```
1: $floor\_map[]$ :Represent the grid map of a floor in the form of a nested list
2: $start$ : Start cell is assigned as [0,0].It can also be given other cell
3: $goal$ : Any destination cell can be given
4: $adj\_list[]$ : Nested list - To get the adjacency list of floor_map Diagonal
   nodes are also considered as adjacent nodes
5: $total$ : Count of total no of empty cells in a floor_map
6: $TreeNode$ : It is a class following Tree Data Structure. Each cell in the
   path is represented as the object of a TreeNode. The information of each
   cell in path is described by the attributes and methods defined in TreeNode.
       Attributes: data,cost,back,length,children
       Methods:
       $get\_cost$ : This method calculates the path traversed to a node.
       Turn rate is not taken into consideration in this algorithm
       $calculate\_angle$ : Checks if there is a turn of 45 or 90 degrees in the
       path considering last three cells in the path.
7: $update\_values(floor\_map, start, goal, total)$ : Intial floor_map is taken and
   the values are modified as discussed in Distance Transform Algorithm.
   Nodes adjacent to goal cell including diagonal cells are marked 1 and the
   wave front traverses through all nodes until all the empty cells are covered
8: $next\_maxNode(node, visited, adj\_list, floor\_map)$ :Checks all the adjacent
   nodes of a cell and gives the index of the not visited cell having maximum
   number in the modified floor_map.
9: $min\_backtrace(node, visited, adj_list, floor_map, goal)$ : If two or more un
   visited adjacent cells are same one particular cell is selected according to
   the priority.
10: $find\_path(start, floor_map, adj_list, goal)$ :
       $visited[]-$Nodes traversed in a path are added to visited list.
       $root-$TreeNode object.Start cell is added to root node
       $current-$Current node in a path
       $while(current.data! = goal)$ :
       Nodes are added to the path in this loop by getting the results from
   next_maxNode and min_backtrace methods. Final path is obtained.
```

Chapter 4

Results

Complete path obtained from the two algorithms are discussed here

4.1 Results from Algorithm1

Result 1: It is a case where each cell is traversed once (there is no back traversing of cells). Consider floor_map=[[1,0,1,1,1,1],[1,0,1,1,1,1],[1,1,0,1,1,1],
[1,1,1,1,0,1],[1,0,1,1,1,1],[1,1,1,1,1,1]]

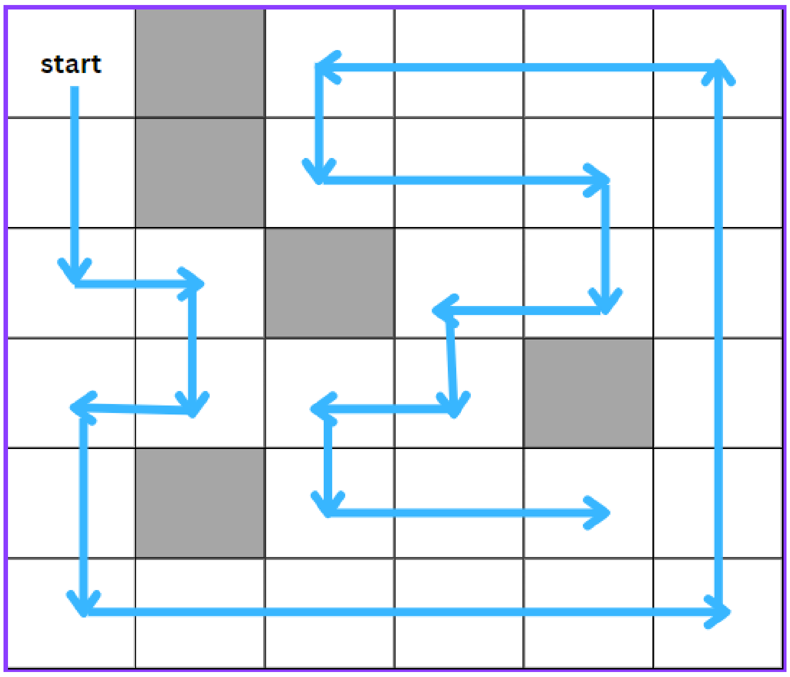


Figure 4.1: Algorithm1-Result1

The final path calculated from this algorithm is as follows:

Total No of empty cells : 31

Cost for 90 degrees turn is considered as cost of travelling a distance of one cell = 1 unit

Total No of turns= 13

Total cost of complete path coverage = 31+13=44 units

Here we consider robot can turn only 90 degrees. Diagonal cells are not counted as adjacent cells

Result 2: It is a case where the robot has to traverse certain cells twice in order to ensure complete path coverage.

Consider floor_map=[[1,1,1,1,1,0],[1,1,0,0,0,0],[1,1,0,0,0,0],
[1,1,0,0,1,1],[1,1,0,0,1,1],[1,1,1,1,1,1]]

The final path calculated from this algorithm is as follows:

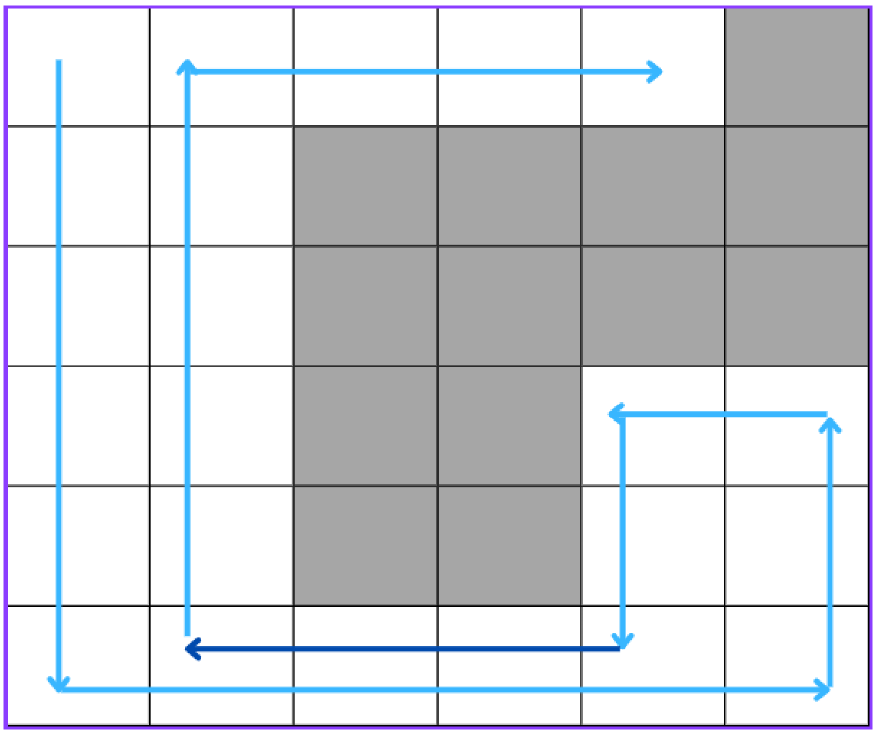


Figure 4.2: Algorithm1-Result2

This is a case where back traversing of cells is mandatory for the coverage of complete path Total No of empty cells : 23

Cost for 90 degrees turn is considered as cost of travelling a distance of one cell = 1 unit

Total No of turns= 7 No of cells traversed two times =4

Total cost of complete path coverage = 23+7+4=34 units

Here we consider robot can turn only 90 degrees. Diagonal cells are not counted as adjacent cells

Algorithm 1 ensures that the cost of complete path calculated is minimum. But the disadvantage is it has high time complexity (exponential) .

It is not feasible to use this algorithm for higher number of cells

It can be used to generate data set to be given to train Neural Network

4.2 Results from Algorithm 2

Using modified distance transform for CCPP.

Result 1:

For floor_map=[[1,0,1,1,1,1],[1,0,1,1,1,1],[1,1,1,1,1,1],
[1,1,1,1,0,1],[1,0,1,1,1,1],[1,1,1,1,1,1]]

The final path calculated from this algorithm is as follows:

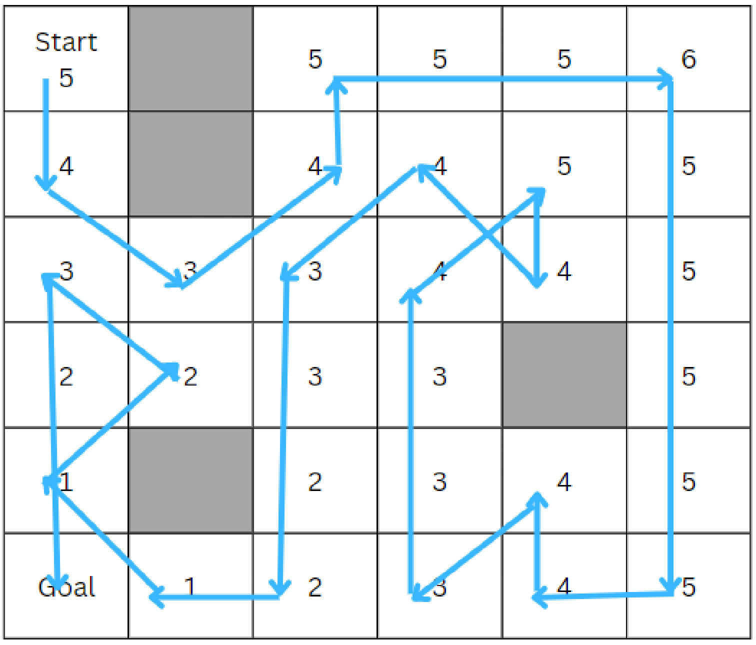


Figure 4.3: Algorithm2-Result1

Total No of empty cells : 32

Cost is turn is not taken into account in this algorithm.

It is considered as 0.

Diagonals cells are also considered as adjacent cells in this case and the distance between diagonal cells are also considered as 1 unit.

Robot can take turn of 90 or 45 degrees

No of cells traversed backwards = 1

Total cost of complete path coverage = 32+1 =33 units

Result 2: Using Algorithm 2 for large no of cells

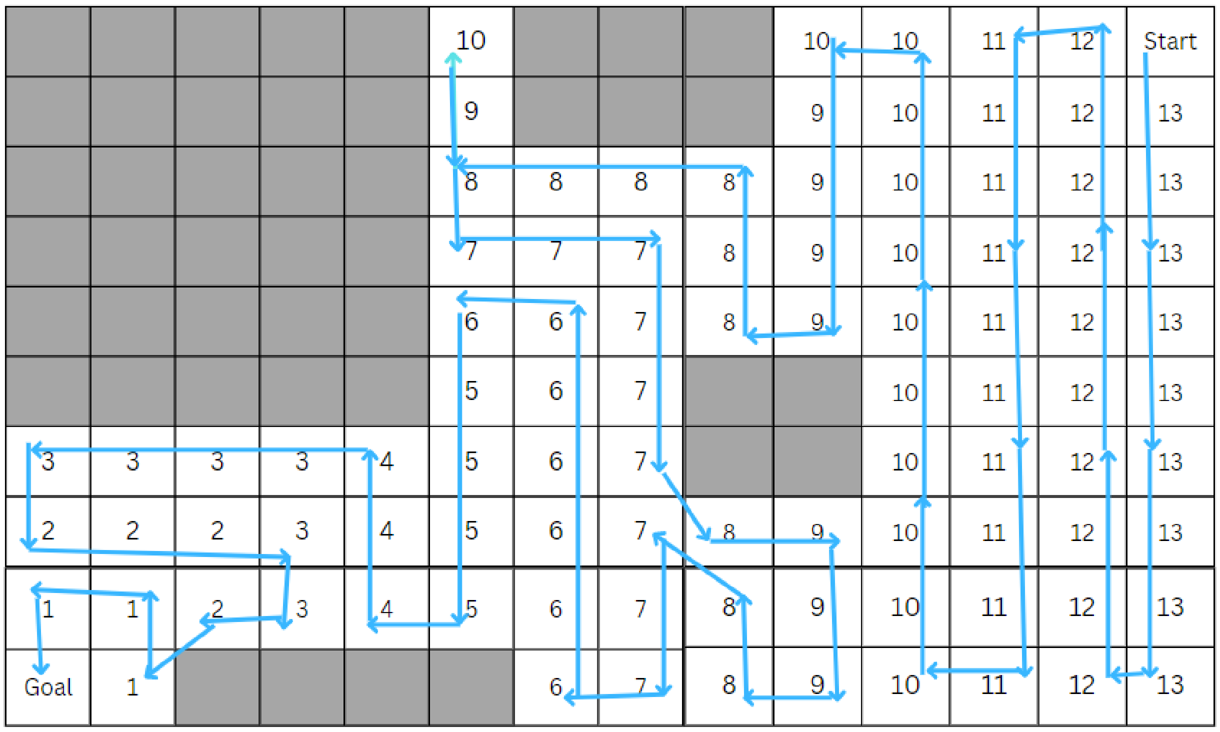


Figure 4.4: Algorithm2-Result2

The time taken for CCPP is less. But it does not consider cost for taking turns and distance between diagonal cells are also considered as 1 unit

The path length obtained from this algorithm may not be optimal

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Complete path coverage by an autonomous vehicle is studied in this report.CCPP is an NP Hard problem. Different algorithms are explored. They include Branch and Bound,Path Transform Algorithm, Distance Transform Algorithm, Modified DT algorithm, Complete Coverage D*(CCD*), Minimum Spanning Tree, 2 opt algorithm. Two algorithms are developed by taking the reference of Brand-Bound and distance Transform Algorithms. In first algorithm both Breadth First Search and Depth First Search concepts are used for CCPP. The path calculated from this algorithm is of minimum cost .But the time taken for calculating path is very high. It increases exponentially with increase in no of levels. So it is not feasible for a practical scenario. The second algorithm uses minimum time to calculate the path but it does not account for cost of taking turn. So the path developed by this algorithm may not be optimal. Different grid maps are created with obstacles ranging from 10% to 50% and tested the results of both the algorithms.

5.2 Future Work

A graph based neural network model has to be created by taking the data obtained from the two designed algorithms. Also the NN model has to be developed for dynamic environments. These neural network models have to be interfaced on ROS-Gazebo simulation and the results should be verified. Later CCPP has to be implemented and tested on an actual hardware.

References

[1] X. Qiu, J. Song, X. Zhang, and S. Liu, "A complete coverage path planning method for mobile robot in uncertain environments," in *2006 6th World Congress on Intelligent Control and Automation*, vol. 2, pp. 8892–8896, IEEE, 2006.

[2] Z. Sheny, P. Agrawal, J. P. Wilson, R. Harvey, and S. Gupta, "Cppnet: A coverage path planning network," in *OCEANS 2021: San Diego–Porto*, pp. 1–5, IEEE, 2021.

[3] M. Dakulović, S. Horvatić, and I. Petrović, "Complete coverage d* algorithm for path planning of a floor-cleaning mobile robot," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 5950–5955, 2011.

[4] A. Zelinsky, R. A. Jarvis, J. Byrne, S. Yuta, *et al.*, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proceedings of international conference on advanced robotics*, vol. 13, pp. 533–538, 1993.

[5] G. Sanna, S. Godio, and G. Guglieri, "Neural network based algorithm for multi-uav coverage path planning," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1210–1217, IEEE, 2021.

[6] P. T. Kyaw, A. Paing, T. T. Thu, R. E. Mohan, A. V. Le, and P. Veerajagadheswar, "Coverage path planning for decomposition reconfigurable grid-maps using deep reinforcement learning based travelling salesman problem," *IEEE Access*, vol. 8, pp. 225945–225956, 2020.

[7] K. Chen and Y. Liu, "Optimal complete coverage planning of wall-climbing robot using improved biologically inspired neural network," in *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pp. 587–592, IEEE, 2017.

[8] A. Bagnitckii, A. Inzartsev, and A. Pavin, "Planning and correction of the auv coverage path in real time," in *2017 IEEE Underwater Technology (UT)*, pp. 1–6, IEEE, 2017.

[9] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[10] S. X. Yang and C. 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.

[11] J. Song and S. Gupta, "Care: Cooperative autonomy for resilience and efficiency of robot teams for complete coverage of unknown environments under robot failures," *Autonomous Robots*, vol. 44, no. 3, pp. 647–671, 2020.

[12] B. Sun, D. Zhu, C. Tian, and C. Luo, "Complete coverage autonomous underwater vehicles path planning based on glasius bio-inspired neural network algorithm for discrete and centralized programming," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 1, pp. 73–84, 2018.

[13] C. Yang, Y. Tang, L. Zhou, and X. Ma, "Complete coverage path planning based on bioinspired neural network and pedestrian location prediction," in *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 528–533, IEEE, 2018.