Rochester Institute of Technology

RIT Scholar Works

Theses

5-2020

Turn and Orientation Sensitive A* for Autonomous Vehicles in Intelligent Material Handling Systems

Rashmi Ballamajalu
rb4609@rit.edu

Follow this and additional works at: <https://scholarworks.rit.edu/theses>

Recommended Citation

Ballamajalu, Rashmi, "Turn and Orientation Sensitive A* for Autonomous Vehicles in Intelligent Material Handling Systems" (2020). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

Turn and Orientation Sensitive A* for Autonomous Vehicles in Intelligent Material Handling Systems

by

Rashmi Ballamajalu

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science
in Electrical and Microelectronic Engineering

Supervised by

Professor Dr. Ferat Sahin
Department of Electrical and Microelectronic Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
May 2020

Approved by:

Dr. Ferat Sahin, Professor
Thesis Advisor, Department of Electrical and Microelectronic Engineering

Dr. Clark Hochgraf, Associate Professor, Committee Member
Dept. of Electrical, Computer, and Telecommunications Engineering Technology

Dr. Jamison Heard, Assistant Professor, Committee Member
Department of Electrical and Microelectronic Engineering

Dr. Sohail Dianat, Professor
Department Head, Department of Electrical and Microelectronic Engineering

Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title:

Turn and Orientation Sensitive A* for Autonomous Vehicles
in Intelligent Material Handling Systems

I, Rashmi Ballamajalu, hereby grant permission to the
Wallace Memorial Library to reproduce my thesis in whole
or part.

Rashmi Ballamajalu

Rashmi Ballamajalu

05/01/2020

Date

Dedication

I dedicate this work to all the people who stood by me as I
explored this dream.

To my parents, in-laws and my brother, the unconditional
affection and encouragement gave me strength each day.

To my teachers and mentors, every lesson has been a
privilege and an honor.

To my friends, who rallied with me at every dead-end so I
could reach the goal.

To my husband, your love, patience and undying support is
a debt I can never repay.

Acknowledgments

I would like to thank Dr. Ferat Sahin for never saying no, and encouraging me to pursue this research. I thank Dr. Michael Kuhl and all the professors in the iMHS team for giving me the opportunity to work on an esteemed project. I am extremely fortunate for such an opportunity to learn and gain experience and be a part of the research funded in part by Toyota Material Handling North America and The Raymond Corporation.

I thank Dr. Clark Hochgraf for being an inspirational mentor and guide at every level of the project. I truly believe there isn't a topic that you haven't explored.

I want to thank my friends Lipisha and Saurav for giving me your time and energy whenever I needed, leaving your own work behind. Robert and Patrick, thank you for the conversations in lab, I learnt a lot.

I thank Rochester Institute of Technology for providing students with resources, labs, and materials for the progress of the research.

I thank everyone who was involved and interacted with me to help shape my life as a master's student.

Abstract

Turn and Orientation Sensitive A* for Autonomous Vehicles in Intelligent Material Handling Systems

Rashmi Ballamajalu

Supervising Professor: Dr. Ferat Sahin

Autonomous mobile robots are taking on more tasks in warehouses, speeding up operations and reducing accidents that claim many lives each year. This paper proposes a dynamic path planning algorithm, based on A* search method for large autonomous mobile robots such as forklifts, and generates an optimized, time-efficient path. Simulation results of the proposed turn and orientation sensitive A* algorithm show that it has a 94% success rate of computing a better or similar path compared to that of default A*. The generated paths are smoother, have fewer turns, resulting in faster execution of tasks. The method also robustly handles unexpected obstacles in the path.

List of Contributions

- Proposed a modified heuristic path planning algorithm termed, turn and orientation sensitive A* for large autonomous mobile robots in material handling applications.
- Implemented the algorithm and simulation results show that it generates time-efficient paths by reducing the number of turns and heading changes compared to default A* with Euclidean distance as the heuristic function.
- The generated paths are smoother with fewer turns thereby reducing the chances of tipping over as well. It considers the direction the vehicle is heading in and attempts to maintain the same. This results in faster execution of tasks and more energy effective operation of the robot suitable for the fast paced environment in the industry application.

Publication Submission:

R. Ballamajalu, M. Li, F. Sahin, C. Hochgraf, R. Ptucha, M.E. Kuhl, "Turn and Orientation Sensitive A* for Autonomous Vehicles in Intelligent Material Handling Systems", *2020 IEEE International Conference on Automation Science and Engineering*

Contents

1	Introduction	1
1.1	Intelligent Material Handling Systems	1
1.2	Path Planning Algorithm for Industrial AMRs	2
2	Literature Review	7
3	A-star Algorithm	13
3.1	Working Principle of A*	14
4	TOS-A* Algorithm	18
4.1	Minimizing Number of Turns in the Path	18
4.1.1	Considering Obstacles in Line-of-Sight	22
4.1.2	Considering Heading Direction	23
5	Simulation Results and Real-time Implementation	25
5.1	TOS-A* Path Planning Simulation	26
5.1.1	Dynamic Re-planning Using TOS-A*	35
5.2	ROS Implementation	37
5.2.1	Tricycle Robot Kinematic Model	38
5.2.2	Gazebo Simulation	39
5.3	Hardware Specifications	44
5.3.1	Communication to the Forklift	45
6	Conclusion	48
7	Future Work	49
8	Appendix	55

List of Figures

1.1	A Smart Warehouse Simulation Setup	3
1.2	Real Warehouse setup	4
1.3	(a) Map of a working warehouse (b) Path planning Challenge for a Forklift	5
2.1	A Guide to Selecting Path Planning Algorithm Described in [1]	8
3.1	(a) Path Planned by Default A* (b) Euclidean distance Heuris- tic Map	16
3.2	Exploration of the Map by (a) Dijkstra's Algorithm (b) A* Euclidean Algorithm	17
4.1	Top view of the Vehicle	19
4.2	Smooth 90-degree Turn Paths	19
4.3	Exploration of Immediate Neighbors: (a) All neighbors (b) Four neighbors	20
4.4	Robot as a Point Mass to Plan Paths around Obstacles	20
5.1	Map of the Warehouse Layout.	26
5.2	TOS-A* Grid Map with Given Start and Goal Locations	27
5.3	Grid Map Exploration by (a) TOS-A* (b) Default A*	28
5.4	Paths Generated by (a) TOS-A* (b) Default A*.	28
5.5	Cost Map of (a) TOS-A* Heuristic Function (b) Default A* Euclidean Distance Heuristic.	29
5.6	Change of Velocity over Time as robot traverses the Generated Path.	31
5.7	TOS-A* Path Considering only the Turning Heuristic Function in (4.1)	33
5.8	TOS-A* Path Considering both Turning and Direction of Head- ing Heuristic Function in (4.1)	35

5.9	Path Re-planning with (a) Obstacle highlighted in Red (b) Updated Grid Map including Obstacle and New Start Location. . .	36
5.10	TOS-A* (a) Explores New Grid to Recompute Path (b) Generates a New Path with Updated Start Location.	37
5.11	Tricycle Drive Kinematic Model	38
5.12	Forklift Simulation in Gazebo	40
5.13	Forklift Simulation in Gazebo	40
5.14	Setup of the Warehouse in Gazebo	41
5.15	Setup of the Warehouse in Gazebo	42
5.16	Rviz Simulation of the Forklift	42
5.17	Rviz Simulation of the Forklift	43
5.18	Mapping of the Warehouse in Rviz	43
5.19	Navigation to Goal Location	44
5.20	CAN Communication over ROS-serial	45
5.21	Wiring setup to Communicate with the Forklift	46
8.1	Automated Forklift at the Warehouse	55
8.2	CAN Wiring to Establish Communication with Forklift	56

List of Tables

5.1	Performance of TOS-A* over Default A* for 50 Trials considering only Number of Turns.	32
5.2	TOS-A* comparison with Default A* for 50 Trials considering only Number of Turns.	32
5.3	Performance of TOS-A* over Default A* for 50 Trials considering Number of Turns and Direction of Heading.	33
5.4	TOS-A* comparison with Default A* for 50 Trials considering Number of Turns and Direction of Heading.	34

Chapter 1

Introduction

Autonomous mobile robots (AMR) are increasingly being deployed in industry applications and are rapidly moving towards automation, with focus on large vehicles such as forklifts. As the size of these mobile robots increase and their applications diversify, it is important to have safe and reliable navigation while maintaining productivity.

Autonomous forklifts used in intelligent material handling applications such as smart warehousing are the focus of this research.

1.1 Intelligent Material Handling Systems

Warehousing applications in industry require significant trained human labor. The management, movement, and storage of products is a fast paced and demanding environment where tasks are arduous and time sensitive. The fast pace of warehouse operations can lead to accidents.

Every year, about thirty five thousand people get seriously injured [2] in factories or warehouses while handling material and out of which 24% of these accidents occur due to forklifts tipping over and injuring human operators. A tip-over is caused when a turn is executed with excessive speed. A significant percent of these accidents could be prevented with the use of AMRs.

The material handling industry is currently facing another challenge due to a shortage of experienced forklift operators. The limited availability of human operators during peak supply seasons is adversely affecting the production and delivery. If some of these tasks were automated, industry's leading services could keep up the same level of production and supply throughout the year. Fig. 1.1 shows the simulation setup of a smart warehouse where the fleet of automated forklifts perform various pick-up and drop-off tasks.

1.2 Path Planning Algorithm for Industrial AMRs

AMRs are intended to assist human operators perform routine tasks with similar efficiency while co-existing in an environment with material, equipment, and people. A real warehouse setup with a forklift is shown in Fig. 1.2. The safety of the vehicle's behaviour in autonomous mode is paramount, as these robots will be interacting with humans

Figure 1.1: A Smart Warehouse Simulation Setup

and materials. In the case of an autonomous forklift, the vehicle's size and kinematics make it difficult to use existing navigation solutions that are largely meant for small differential drive robots. This is due to autonomous robots being largely deployed and used in small scale applications. However, few approaches have been proposed for Automated Guided Vehicles (AGVs) and these have been detailed in chapter 2. The non-holonomic nature of the vehicles make it challenging to maneuver and achieve any desired orientation and position within the warehouse.

The path planning and navigation must be dynamic because the warehouse environment is fast paced and constantly changing to meet fluid demand and supply requirements. The planner is made aware of the

Figure 1.2: Real Warehouse setup

surroundings using a LiDAR sensor that generates point-cloud depth information. The path planner is aware of the changes as they occur and re-plans accordingly. Given a task to be completed and the associated pick up and drop off locations within the mapped region, the goal of algorithm is to not just find the shortest path but to find the most time-efficient path, constrained by the need to slow down the vehicle during turns.

Consider the example shown in Fig. 1.3a which shows the map of a warehouse. Suppose the forklift is in one aisle and must pick-up a pallet and drop it off in aisle two as shown in Fig. 1.3. There are multiple ways that this can be done and each results in a different cost with respect

Figure 1.3: (a) Map of a working warehouse (b) Path planning Challenge for a Forklift

to the number of turns taken, whether the vehicle is moving backwards than forwards or its speed. All these factors affect the quality of the task that is being handled and in turn the productivity.

The focus of this research is developing such a global path planner that takes into account, (i) the characteristics of a large AMR such as its substantial footprint, requirement to slow down during turns, and its orientation; (ii) distance to the target; and (iii) potential obstacles. We propose a modified version of the A* search method to accomplish this complex path planning task.

The rest of the report is organised as follows. Chapter 2 briefly explores the existing and upcoming path search technologies. The A* al-

gorithm is discussed in chapter 3. The proposed Turn and Orientation Sensitive (TOS) A* algorithm and the design modifications are detailed in chapter 4. Simulation experiments and results are presented in chapter 5, and conclusions are presented in chapter 6.

Chapter 2

Literature Review

Path planning algorithms for mobile robots have largely been explored considering small holonomic robots with the path having the shortest travel distance, shortest time, and conflict free as some of the optimal attributes. For the application considered in this research, the optimal path is redefined to include the maneuverability and time taken in executing the path.

The importance and need for adapting path planning algorithms based on specific applications are detailed in Souissi et al. [1]. The survey illustrates how a path planning algorithm can be selected based on the levels shown in Fig. 2.1. Each level helps to choose the attribute pertaining to the specific application at hand and narrows the selection process.

They explore the most current and effective algorithms, specifically

Figure 2.1: A Guide to Selecting Path Planning Algorithm Described in [1]

those that cater to real-time applications and dynamic re-planning in case of unplanned obstacles in the environment. Algorithms that require a pre-defined map such as the Dijkstra and A-star (A*) algorithms, and those that do not require a map such as Rapidly exploring Random Trees (RRTs) and ant colony algorithms are explained. In path planning, whether in 2-dimensions or 3-dimensions, the complexity and kinematic details of the robot's movements need to be analyzed before selecting a planning algorithm [3]. A thorough survey of 50 path planning algorithms is described in Rajchandara et al. [4]. Each algorithm has been individually considered and their objective, use case and advantages tabulated. This helps to determine the various options already researched

in the community.

Genetic Algorithms (GAs) have been quite widely used in the last decade. Tuncer and Yildirim [5] introduce a new mutation operator to adapt the algorithm to dynamic environments. The GA also offers flexibility that is utilized by Yun et al. [6] to implement dynamic planning that helps the robot move, identify obstacles, and navigate in an unknown environment. Their algorithm has also been implemented on the AmigoBot robot for real-time validation.

Deep learning methods are also slowly taking root in path planning as illustrated in Li et al. [7] where an improvised Q-learning algorithm is used for dynamic path planning. Furthermore, algorithms such as Pattern Search (PS), Particle Swarm Optimization (PSO) and other evolutionary methods have been explored by Fetanat et al. [8] to improve dynamic path planning in mobile robots. A method using potential fields for dynamic planning when the target and obstacles in an environment are moving is presented in Gi and Cui [9]. However, all these methods have a computational overhead that is too high for the safety and quick responsiveness needed for our application. The warehouse environment although has static infrastructure, it is constantly evolving and changing to suit the demand and supply of products. This makes it difficult for machine learning approaches to be generalized to suit any warehouse

environment.

Although these algorithms solve the path planning problem quite effectively, the time taken to train them has to be considered and the need to be re-trained when the map of the environment changes. A review of motion planning techniques currently being explored in the research community is presented in Gonzalez Bautista et al. [10]. They conclude that graph based search algorithms are most popular when it comes to real world implementations and are quite adaptable to most use cases. The A* (A-star) and D* (D-star) algorithms seem to be the most popular among graph based methods. A comparison of these two algorithms for differential drive robots is presented in Setiawan et al. [11]. Based on the simulation and experimental results, they observe that D* Lite can plan a path in lesser computational time than A* by retaining the memory of previously explored paths. However, another such comparison [12] shows that the D* Lite algorithm is less effective than the A* algorithm in relatively smaller and less complex environments.

It is therefore important to consider the characteristics of the system in which the algorithm is going to be applied and the nature of the system, whether static, dynamic or a mix of both. The application discussed in this research has a map already in place due to the generally static nature of the overall layout of a warehouse. However, there may be

dynamic obstacles, such as humans or material, in the path of the robot. Due to these factors, we hypothesize that a graph-based A* search algorithm that can be implemented dynamically is best suited for the given application.

Several variations and implementations of A* can be found such as the algorithm developed by Duchon et al. [13] where the modification is focused on the computational time and optimal path. These modifications are individually evaluated with varied levels of complexity in the environment. Additionally, vehicle characteristics such as turning radius are considered by Yang and Wushan [14] where a grid-based path smoothing method is proposed and applied to the path provided by A*. This satisfies the robot's turning radius, makes a smooth transition during turning, and considers the deviation from the path as well.

This is important especially in material handling applications where the robots carrying a payload have a higher risk of tipping over while making sharp turns. However, Guruji et al. [15] show that the computational time of the algorithm tends to increase exponentially with the size of the environment. They also introduce modifications to reduce the overall computation time. An interesting take on irregular grids and utilization of visibility graphs for A* are presented in Daniel et al. [16]. The visibility graphs provide the line-of-sight and the angle to the goal.

Paths are generated considering this criterion.

An improvisation of A* specifically intended for autonomous mobile robots has been presented in Wang et al. [17] which includes factors such as turning radius, number of turns taken in a path and the shortest path. The algorithm computes paths and later counts the number of edges as the number of turns and stores the path. The path with the shortest distance and least number of turns is selected. However, this process is computationally expensive. The TOS-A* algorithm proposed in this report tries to intuitively reduce the number of turns as the path is being explored and gives the best solution in a single run.

Chapter 3

A-star Algorithm

The algorithm was developed by Peter E. Hart, Nils Nilsson and Bertram Raphael in 1968 and was extended from the famous “Dijkstra’s algorithm”. It aims to reduce the number of graph nodes explored by using a heuristic function or an estimate leading to the goal node. Some of the dynamic path planning implementations are D*(D-star), Focused D*, Lifelong Planning A* (LPA) mentioned in chapter 2.

The proposed TOS-A* planner has been built upon A* search method to generate a path for an autonomous robot in a warehouse environment. The working principle of the default A* algorithm is explained below along with the key elements in the algorithm that have been modified to suit the application.

3.1 Working Principle of A*

The A* search algorithm requires a pre-defined map of the environment that can be a static or a dynamic map. In this research, we consider a regular grid with equidistant grid spacing of one unit each, that is each step taken covers a distance of one meter. The A* algorithm considers the blocked regions, the regions that the robot can traverse, and the pre-defined start and goal points. The pseudo code for A* has been illustrated in Algorithm 1.

A list consisting of the start node is created. The node m having the lowest cost value is chosen to start the search. Each neighboring child node denoted by n is explored and the cost function denoted by $f(n)$ is updated based on the condition that it has a lower heuristic value indicating that the search is nearing the goal. Considering an example start and goal pair, the path selected by A* has been illustrated in Fig. 3.1a. The corresponding Euclidean distance heuristic map has been shown in Fig. 3.1b. The nodes with the smallest overall cost function as successively chosen to form the path. This is the shortest path planning algorithm that we compare our proposed algorithm with in the next chapters.

The interesting aspect here that makes A* a directed search algorithm is the heuristic function. The exploration is concentrated in the direction

Algorithm 1: A* Algorithm Pseudo-code

Result: Search for a Path from Start to GoalInitialize *open list* = [*start*];**while** *open list* ≠ [] **do** Select *m* with lowest cost; **if** *m* ≠ *goal* **then** Remove *m* from *open list*; **for all** *n* in *child*(*m*) **do**

Compute cost:

$$f(n) = g(m, n) + h(n, goal)$$

Where $f(n)$ is cost of the current child n , $g(m, n)$ is actual cost to move from m to n , and $h(n, goal)$ is heuristic estimation to move from n to $goal$;

if $g(m, n) < g'(n)$ **then**

 Append node to *open list* where $g'(n)$ is the previous node's cost;

else

exit loop;

end **end** **else** *goal found*; **end****end**

of the goal rather than a breath-first search approach such as that used in Dijkstra's algorithm. Fig. 3.2 shows the difference between the A* search and the Dijkstra's and A* Euclidean distance algorithms. It can be observed that A* is faster and explores less number of nodes and derives a shortest path.

The heuristic function can vary by implementation. Some examples

Figure 3.1: (a) Path Planned by Default A* (b) Euclidean distance Heuristic Map

of which were discussed in chapter 2. The most commonly used function is the Euclidean distance which calculates the shortest path from the start to goal location. This is because the Euclidean distance is an admissible heuristic function and widely accepted. At any given instance, the resultant path is the optimized path for that scenario. The heuristic proposed in this work has been compared to the Euclidean heuristic performance. Although the performance of the modified heuristic function shows promising results, it cannot be categorized as admissible due to the factors such as number of turns considered in deeming that a path as optimized.

In the next section, the various modifications to the heuristic function

Figure 1 consists of two 10x10 grids, (a) and (b), illustrating a pathfinding problem. In both grids, the top-left cell (1,1) is blue and labeled 'Start'. The bottom-right cell (10,10) is green and labeled 'Goal'. Brown cells represent 'Blocked Regions'. In grid (a), the blocked regions are a vertical bar at column 2 (rows 2-10), a horizontal bar at row 3 (columns 4-7), and a horizontal bar at row 5 (columns 4-7). In grid (b), the blocked regions are a vertical bar at column 2 (rows 2-10), a horizontal bar at row 3 (columns 4-7), and a horizontal bar at row 5 (columns 4-7). A dashed line with arrows indicates the path from Start to Goal. In grid (a), the path is (1,1) → (2,2) → (3,3) → (4,4) → (5,5) → (6,6) → (7,7) → (8,8) → (9,9) → (10,10). In grid (b), the path is (1,1) → (2,2) → (3,3) → (4,4) → (5,5) → (6,6) → (7,7) → (8,8) → (9,9) → (10,10). Labels 'Open Regions' and 'Blocked Regions' are placed near the respective areas in both grids.

Figure 3.2: Exploration of the Map by (a) Dijkstra's Algorithm (b) A* Euclidean Algorithm

Chapter 4

TOS-A* Algorithm

The AMR used for the experiments in this paper is a tricycle model forklift. This section details the key factors considered while applying the modified A* algorithm for this application.

4.1 Minimizing Number of Turns in the Path

The forklift has a front steer drive wheel as shown in Fig. 4.1, similar to Ackermann steering, thus the algorithm has been modified to give turn paths with a non-zero radius and assumes 90-degree turns.

As illustrated in Fig. 4.3b, the algorithm explores only four of its neighboring nodes, namely, n2, n4, n6 and n8, instead of all eight as shown in Fig. 4.3a. In order to compare the results, both A* and TOS-A* have been implemented in this work to explore four neighboring nodes.

Figure 4.1: Top view of the Vehicle

The local trajectory planner in turn smooth-ens these 90-degree turn paths as shown in Fig. 4.2. It also ensures that the vehicle can make a turn given its footprint within the available space in an aisle.

Figure 4.2: Smooth 90-degree Turn Paths

In attempt to minimize the number of turns in the path that is generated, several existing approaches count the edges present in the planned

Figure 4.3: Exploration of Immediate Neighbors: (a) All neighbors (b) Four neighbors

path. It is then analysed and a new path is computed to propose one with fewer number of turns. Another approach is theta* [16] which uses a line of sight method to derive the shortest path. However, it does not focus on the number of turns taken and proposes greedy paths that are close to the edges of blocked cells.

Additionally, path planning algorithms consider a robot to be a point mass and assume they can traverse in any direction as shown in Fig. 4.4.

Figure 4.4: Robot as a Point Mass to Plan Paths around Obstacles

However, AMRs such as forklifts cannot perform the turns greater than 90 degrees in an aisle [18], and local planners often cannot recover the vehicle's behaviour once it has been backed into an aisle that is blocked on the other end. In order to minimize the number of turns, and consider the direction of heading in a single planning iteration, we introduce a modified heuristic function that is a weighted linear combination of three individual heuristics as follows,

$$h(m, n, t, goal, p) = [w_0 + w_1 h_1(m, n, t) + w_2 h_2(p, m, n)](h_0(n, goal)) \quad (4.1)$$

where $h(m, n, t, goal, p)$ is the modified heuristic that is a function of starting grid point m , current neighbor n (e.g. n2, n4, n6, n8), t steps explored in the forward direction, goal grid point $goal$ and the previous explored node p . The weights associated with each term are static, with w_0 always being a non-zero value. They are updated based on results of multiple trials as the heuristic function is an approximation of the cost from the current location to the goal. Due to the addition of the weights in the heuristic and algorithm's behaviour changes as the weights are changed, the heuristic function cannot be classified as admissible.

$$h_0(n, goal) = ||n - goal||^2 \quad (4.2)$$

The first parameter $h_0(n, goal)$ represents the Euclidean distance measurement, that is, the shortest distance to the goal. This has proven to be the fastest depth-first search approach, and forms the foundation for the cost function. It is also a linear function. However, the disadvantage with using only the Euclidean distance function is that multiple neighboring cells have the same cost. The Manhattan distance measurement has not been explored in this work and could be a good alternative since the 4 neighbors are explored.

4.1.1 Considering Obstacles in Line-of-Sight

Attributes such as turning and orientation which result in a heuristic function that assigns a unique cost to each neighbor cell have been introduced. Penalizing paths that have future obstacles (paths that would result in a turn) gives a more directed path search. The heuristic function that considers the obstacles in the path

$$h_1(m, n, t) = 1 \text{ or } 0 \quad (4.3)$$

where $h_1(m, n, t)$ is a binary variable that is 1 if moving from m for t steps along the direction of m contains an obstacle. In our experiments, $t = 2$ where the vehicle looks ahead 2 steps in the forward direction and the weight $w_1 = 0.3$ is assigned based on experimental trials. However,

this does not consider the prior heading of the vehicle.

4.1.2 Considering Heading Direction

For automated forklifts, the forks-first or face-first heading affects the execution of the task, be it pick up or drop off. Most warehouses have single-width aisles where a 180 degree turn is not possible without backing up multiple times. The direction the AMR is originally heading is therefore a major contributing factor while planning the path. and is represented by the heuristic function as,

$$h_2(p, m, n) = 1 \text{ or } 0 \quad (4.4)$$

where p is the previous node to current node m and n is the next neighbor. $h_2(p, m, n)$ is a binary variable that is 0 if p , m and n of the vehicle's previous heading and the next are in the same direction, and 1 if they are not aligned. The weight $w_2 = 0.5$ has higher priority in (4.1) than w_1 and influences the decision of whether the path must take a turn or continue straight. In case of large scale warehouses, the distance between points might be too large and the weights associated with the other heuristics need to be scaled as well. Therefore, the weights in (4.1) are added before multiplying with the Euclidean heuristic in order to normalize the equation by scaling it with the factor of the distance measurement.

Algorithm 2: TOS-A* Algorithm Pseudo-code

Result: TOS-A* Search for a Path from Start to GoalInitialize *open list* = [*start*];**while** *open list*! = [] **do** Select *m* with lowest cost; **if** *m* != *goal* **then** Remove *m* from *open list*; **for all** *n* in *child*(*m*) **do** **if** (*Obstacle in t steps*) **then** | update $h_1(m, n, t)$; **else**

| pass

end **if** (*heading change*) **then** | update $h_2(p, m, n)$; **else**

| pass

end

Compute cost:

$$f(n) = g(m, n) + h(m, n, t, goal, p)$$

Where $f(n)$ is cost of the current child *n*, $g(m, n)$ is actual cost to move from *m* to *n*, and $h(m, n, t, goal, p)$ is TOS-A* heuristic estimation to move from *n* to *goal* as shown in (4.1);

if $g(m, n) < g'(n)$ **then**

 Append node to *open list* where $g'(n)$ is the previous node's cost;

else

| exit loop;

end **end** **else** | *goal found*; **end****end**

Chapter 5

Simulation Results and Real-time Implementation

The implementation of the proposed method has been done using Python and Matplotlib [19] for simulating the path planning and movement of the robot.

A simulation of the warehouse layout has been setup as shown in Fig. 5.1. The top two boxes correspond to loading and unloading bay areas. The rest of rectangular boxes represent the shelves or blocked regions of the grid. The remaining region is open for exploring paths. This is a small warehouse setup with about 200 combinations of different start and goal locations. These have been randomized to have an arbitrary start and goal points, with a random direction of heading among the 4

options, namely, $+Y$ (up), $-Y$ (down), $+X$ (right) and $-X$ (left).

Figure 5.1: Map of the Warehouse Layout.

5.1 TOS-A* Path Planning Simulation

In the example scenario under consideration, $(0,0)$, the origin of the grid is the start location and $(6,8)$ is the goal location as shown in Fig. 5.2. The start point of the robot is on the top left corner near the first loading station, and the goal point 2 aisles away in the middle. The forklift is picking up a package, heading forward towards the goal and dropping off

at the middle aisle.

Figure 5.2: TOS-A* Grid Map with Given Start and Goal Locations

Fig. 5.3a shows the TOS-A* algorithm exploration of the map based on the heuristic functions illustrated in section 4.1. Fig. 5.3b shows the Euclidean distance heuristic exploration of the map. Both the algorithms take 44 iterations for the chosen example, however, TOS-A* algorithm explores options for more alternate routes. This is due to the obstruction of the shelf in the middle next to the goal location. The weights $w_0 = w_1 = 1$ assigned to the heuristic predicts the possibility for a turn and therefore avoids that path altogether. Once the goal is found, the path

Figure 5.3: Grid Map Exploration by (a) TOS-A* (b) Default A*

Figure 5.4: Paths Generated by (a) TOS-A* (b) Default A*.

is formed from the goal point to the start point choosing those grid cells that have the least overall cost.

Figure 5.5: Cost Map of (a) TOS-A* Heuristic Function (b) Default A* Euclidean Distance Heuristic.

The path generated by default A* is shown in Fig. 5.4b where nodes with the shortest distance to the goal are successively chosen until goal is reached. The associated Euclidean distances from each point to the goal is shown in Fig. 5.5b. However, considering only the shortest distance results in a sub-optimal path that is not suitable for a large vehicle such as a forklift. The shortest path also does not consider the smoothness and the number of turns. Fig. 5.4a shows that the path generated by TOS-A* and can be observed that it has fewer turns than the one generated by default A*.

Fig. 5.5a shows the modified heuristic cost map for TOS-A* algorithm. Due to the high cost on the column to the right, the path explores other options. Also, this is convenient since the vehicle is oriented downward in the $-Y$ direction. However, this heuristic function is inadmissible due to the varying behaviour based on the local environment around the robot.

Fig. 5.6 shows the plot of velocity of the robot as it traverses the path generated by both algorithms based on the Occupational Safety and Health Administration (OSHA) standards [18]. It can be observed that time taken to execute TOS-A* path is 14 seconds lesser than that of default A*. Also, OSHA mandates all forklifts carrying cargo to limit maximum straight path speed to 2.2 m/s and 0.9 m/s at the turning. This is because the regulation requires the forklifts to slow down, sound horn and then proceed at all major intersections and turns. This causes additional deceleration, braking and subsequent acceleration which in turn affect the overall performance of the vehicle.

Additionally, fewer number of turns in the path will reduce the operations of the vehicle in every duty-cycle. The resulting increase in speed of operations will lead to increased productivity, reduce number of maintenance cycles, and longer life of these vehicles.

The algorithm was tested in 2 stages. This was to determine the ap-

Figure 5.6: Change of Velocity over Time as robot traverses the Generated Path.

propriate weights for each heuristic based on a trial and error method and also to determine the effect of the direction of heading on the planning. Initially, 50 trials considering only the turning heuristic function were conducted using (4.1) where $w_2 = 0$. By doing so, the direction of heading was not considered. Table 5.1 shows the number of times the TOS-A* outperformed the default algorithm. A summary of the 50 trials with comparison on parameters such as distance of the generated path, number of turns and number of iterations that the algorithm needed to generate the path is described in table 5.2.

Table 5.1: Performance of TOS-A* over Default A* for 50 Trials considering only Number of Turns.

Performance	TOS-A* over Default
Better	14
Similar	29
Worse	7

Table 5.2: TOS-A* comparison with Default A* for 50 Trials considering only Number of Turns.

Algorithm	Total Distance Travelled (m)	Number of Turns Taken (units)	Number of Iterations
TOS-A*	697	116	2064
Default A*	660	124	2092

Consider another example where (10,6) shown in blue is the start location and (12,16) shown in orange is the goal location. It can be seen in Fig. 5.7 that the path is far too deviated due to the number of turns heuristics having an equal weight to Euclidean distance heuristic. This behaviour was observed in 7 such scenarios.

These cases where the algorithm did not perform well were analysed and deduced that there is a need for prior information such as the direction of heading and the direction in which the goal is located. After incorporating this information, the modified heuristic function (4.1) contains the shortest distance, the number of turns, current grid information, line-of-sight and the previous position and heading.

Figure 5.7: TOS-A* Path Considering only the Turning Heuristic Function in (4.1)

Table 5.3 shows that the improved TOS-A* algorithm over 50 trials and has a much better performance with 17 trials outperforming default A*, 30 trials having the same path distance and turns as Euclidean heuristic and only 3 poor cases. And table 5.4 shows the summary of the new TOS-A* over the same parameters.

Table 5.3: Performance of TOS-A* over Default A* for 50 Trials considering Number of Turns and Direction of Heading.

Performance	TOS-A* over Default
Better	17
Similar	30
Worse	3

Table 5.4: TOS-A* comparison with Default A* for 50 Trials considering Number of Turns and Direction of Heading.

Algorithm	Total Distance Travelled (m)	Number of Turns Taken (units)	Number of Iterations
TOS-A*	742	97	2466
Default A*	742	108	2513

It was observed that the effect of direction on the overall path is considerable. The weights assigned for the experiments are $w_0 = 1$, $w_1 = 0.3$ and $w_2 = 0.5$ based on previous trials by experimenting on a trial and error method. The paths generated when w_2 is zero were too cautious and resulted in paths that are too far diverted from the goal.

However, considering the previous heading information and assigning a higher weight to w_2 than w_1 resulted in optimal paths for most cases. Considering the example where (10,6) shown in blue in Fig. 5.8 is the start location and (12,16) shown in orange is the goal location. The path generated taking into considering both turning and direction of heading heuristic consists of only 3 turns as supposed to the 5 in the previous example. The vehicle is oriented to the right in this example and the path generated has fewer turns compared to the one shown in Fig. 5.7.

Figure 5.8: TOS-A* Path Considering both Turning and Direction of Heading Heuristic Function in (4.1)

5.1.1 Dynamic Re-planning Using TOS-A*

In case of warehouse applications, anticipation of obstacles, be it static such as fallen boxes, oil spills, etc, or dynamic such as human beings or other vehicles, is a must. Fig. 5.9a shows the obstacle in the middle of the already explored path. As the robot traverses the path shown in Fig. 5.4a, the obstacle is identified and the grid is updated with this information as shown in Fig. 5.9b. A new start location is created which

is one grid cell behind the obstacle point.

Figure 5.9: Path Re-planning with (a) Obstacle highlighted in Red (b) Updated Grid Map including Obstacle and New Start Location.

Once the new start point is defined, the algorithm has to repeat the search since the heuristic function results in a new cost map as the environment around the robot changes and the prior values cannot be reused. Fig. 5.10a shows the new explored cells from the updated starting point and compute a path illustrated in Fig. 5.10b. The obstacle considered in this example is a static one which is detected using on-board sensor such as the LiDAR. However, in order to plan for a moving obstacle, such as a human or another forklift, prediction based algorithms need to be used considering time as another dimension in the heuristic function.

Figure 5.10: TOS-A* (a) Explores New Grid to Recompute Path (b) Generates a New Path with Updated Start Location.

5.2 ROS Implementation

A Gazebo simulation using Robot Operating System (ROS) has been set up with the model of the forklift, to simulate real-time operation of the vehicle in a warehouse. In order to simulate the forklift, it is necessary to understand its kinematics and the dynamics of the vehicle. Section below details the tricycle model's kinematic model and the equations used in the simulation and the real-time implementation.

Figure 5.11: Tricycle Drive Kinematic Model

5.2.1 Tricycle Robot Kinematic Model

Assume a tricycle model as shown in Fig. 5.11, where the steering wheel is at an angle α , enabling the robot to rotate with an angular velocity $\omega(t)$ about the point on Instantaneous Center of Curvature (ICC) that is R units away along the line perpendicular to and passing through the rear wheels. The distance between the steering wheel and the pivot point at the rear wheels is d . The linear velocity $v_s(t)$ of the steering wheel is given by,

$$v_s(t) = \omega_s(t)r \quad (5.1)$$

where r is the radius of the wheel and $\omega_s(t)$ is the angular velocity of the steering wheel. Also, it is assumed that the linear velocity is present only

in the x direction and zero in y direction due to zero slippage of wheels.

The angular velocity $R(t)$ about the pivot point where ICC meets the center line passing through the robot is given by,

$$R(t) = d \tan(\pi/2 - \alpha(t)) \quad (5.2)$$

The angular velocity of the robot with respect to the base frame is derived using (5.1) and (5.2) as,

$$\omega(t) = (\omega_s(t)r/(\sqrt{d^2 + R(t)^2})) \quad (5.3)$$

$$\omega(t) = (V_s(t)/d) \sin \alpha(t) \quad (5.4)$$

The kinematic equations have been encoded to simulate the tricycle drive of the robot using Robot Operating System (ROS) and Gazebo.

5.2.2 Gazebo Simulation

A Gazebo simulation has been setup to model the vehicle's dynamics and behaviour. Additionally, the navigation stack has also been implemented to incorporate the movement of the robot within a warehouse.

Fig. 5.12 and 5.13 show the representation of the forklift model within Gazebo environment.

Fig. 5.14 shows the setup of the warehouse environment in Gazebo.

Figure 5.12: Forklift Simulation in Gazebo

Figure 5.13: Forklift Simulation in Gazebo

The blue light is from the single channel LiDAR sensor which is a primary sensor input to the navigation stack. The shelves on either side of the

Figure 5.14: Setup of the Warehouse in Gazebo

truck represents the shelf placement of a small warehouse setup similar to the one used in the simulation shown in Fig. 5.1.

A top view of the warehouse setup has been shown in Fig. 5.15. The kinematics of the model has been encoded as mentioned in the previous section. Additionally, in order to emulate the movement of the robot and its dynamics, the model has been implemented in Rviz as well. Fig. 5.16 shows the forklift with the mapping of the warehouse. Fig. 5.17 shows the front view of the vehicle in Rviz with the sensor enabled. A camera sensor has been mounted on top of the truck for visualization as well.

Fig. 5.18 shows the mapping of the warehouse in Rviz. The TIAGo navigation stack [20] has been used for the mapping and visualization.

Figure 5.15: Setup of the Warehouse in Gazebo

Figure 5.16: Rviz Simulation of the Forklift

After the environment has been mapped, the ROS navigation stack uses particle filters to help in localization. Additionally, the AMCL pack-

Figure 5.17: Rviz Simulation of the Forklift

Figure 5.18: Mapping of the Warehouse in Rviz

Figure 5.19: Navigation to Goal Location

age also helps to localize the robot. Once the robot is aware of where it is in the map, a goal pose and location is given in Rviz. This is shown in Fig. 5.19, with the red arrow representing the position and it is pointing towards the desired orientation of the destination. The blue line is the map generated by the global planner. This is the path that the robot will take in order to reach its destination.

5.3 Hardware Specifications

This work has been conducted in conjunction with an on-going research with Raymond Corporation. The project aims to automate forklifts in

a real warehouse environment for optimizing day-to-day tasks and preventing hazards to human lives. The proposed algorithm has potential to be tested on a real forklift and the following section details the work done so far.

Communication with the real forklift has been setup and detailed in the section 5.3.1. The kinematics model has been explained in this section and encoded into the navigation stack to take in velocity commands and execute them.

5.3.1 Communication to the Forklift

The communication to the vehicle used is over the CAN protocol as shown in Fig. 5.20. The protocol specifications are proprietary of Raymond Corporation.

Figure 5.20: CAN Communication over ROS-serial

The Pallet trucks' or the Forklift's vehicle manager (VM) interacts with RIT's intelligent management system to perform various tasks in the warehouse. It consists of the Arduino Mega as the micro-controller

and a Sparkfun CAN-bus shield that has MCP2515 CAN controller with the MCP2551 CAN transceiver on-board. Communication between the forklift's VM and RIT's AGV is through a DB9 connector with pins connected for CAN-High (CAN-H), CAN-Low(CAN-L) and ground (GND). Additionally, wiring has been done for the floor mat signals, brake switch control, and emergency stop and the required voltage levels have been supplied to these ports.

Fig. 5.21 shows the wiring setup to communicate with the Forklift.

Figure 5.21: Wiring setup to Communicate with the Forklift

The system sends out CAN messages to the AGV. The control messages are for parameters such as traction motor RPM, steering angle, and operation modes of the truck are sent over CAN.

The communication has been made more reliable and robust by upgrading to industry grade DIN mounted wiring. The truck encoder has been interfaced to read odometer data (traction motor RPM, steering angle) and pushed to ROS (Robot Operating System) and will be the basis for our navigation stack that will facilitate navigating and localizing the truck in the warehouse environment. Additionally, the sensors needed on the truck have been integrated and a simulation of the area of coverage and view of each sensor has been studied. This is important to cover all the blind spots on the truck to ensure safe obstacle avoidance, human and material safety at the warehouse.

Chapter 6

Conclusion

Autonomous mobile robots can safely and reliably navigate a warehouse in minimum time, by reducing the number of turns and heading changes in the path. A new heuristic function was proposed considering these attributes in addition to the path length. The proposed TOS-A* algorithm was simulated, and results show that it outperforms default A* in producing time-efficient paths, considering the need to slow down for turns. Also, the generated paths are smoother with fewer turns thereby reducing the chances of tipping over. It considers the direction the vehicle is heading in and attempts to maintain the same. This results in faster execution of tasks and more energy effective operation of the robot.

Chapter 7

Future Work

The current implementation for obstacle avoidance is computationally expensive. A time-window based obstacle prediction is necessary for optimization. Also, the prior heading information could be added to the g cost rather than the heuristic, since this is absolute information that has occurred.

The algorithm must be tested in the navigation stack before it can be deployed in real time. There are a few issues due to the way the current ROS navigation stack is implemented. The costmap of the occupancy grid is calculated based on the configured options such as A-star or Dijkstra's methods. Once the method is configured, the associated formula is applied to the occupancy grid and a costmap is derived. This map is used by the robot and subsequently updated as it moves around this

environment.

The navigation stack configured and implemented so far in Gazebo simulation environment has a modified costmap. The modified heuristic function updates the costmap and generates a global path. However, it was observed that the robot motion is not smooth and results in jerky motion around the edges of the shelves. Further debugging is necessary to understand the behaviour of the robot and modify the implementation to include the path planning as an additional layer over the costmap, rather than directly modify it. This is because, in case of Euclidean distance heuristic, irrespective of the robot's position to the goal, the cost does not change. However, in case of the proposed TOS-A* algorithm, the modified heuristic is subject to change as the environment around the robot changes.

The next step in the course of this work is to run the implementation on the real forklift and test it in the warehouse with a real task. The time taken for execution of default A* and TOS-A* must be recorded and compared. The weights associated with each heuristic can be adjusted as more experiments are conducted. That will be an interesting challenge to explore.

Bibliography

- [1] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau, “Path planning: A 2013 survey,” *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management, IEEE - IESM 2013*, 01 2013.
- [2] Statistics, “Forklift Accidents Statistics.” <https://www.mccue.com/content/forklift-accident-statistics>.
- [3] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, “Survey of robot 3d path planning algorithms,” *Journal of Control Science and Engineering*, vol. 2016, pp. 1–22, 01 2016.
- [4] K.Rajchandara, R. Baskaranb, and K. P. Panchu, “Multiple robot path planning algorithms for static environment and dynamic environment: A review,” *International Journal of Pure and Applied Mathematics*, vol. 119, 2018.

- [5] A. Tuncer and M. Yildirim, “Dynamic path planning of mobile robots with improved genetic algorithm,” *Computers and Electrical Engineering*, vol. 38, no. 6, pp. 1564 – 1572, 2012.
- [6] S. Yun, S. Parasuraman, and V. Ganapathy, “Dynamic path planning algorithm in mobile robot navigation,” in *2011 IEEE Symposium on Industrial Electronics and Applications*, pp. 364–369, Sep. 2011.
- [7] S. Li, X. Xu, and L. Zuo, “Dynamic path planning of a mobile robot with improved q-learning algorithm,” in *2015 IEEE International Conference on Information and Automation*, pp. 409–414, Aug 2015.
- [8] M. Fetanat, S. Haghzad, and S. B. Shouraki, “Optimization of dynamic mobile robot path planning based on evolutionary methods,” *2015 AI & Robotics (IRANOPEN)*, Apr 2015.
- [9] S. Ge and Y. Cui, “Dynamic motion planning for mobile robots using potential field method,” *Autonomous Robots*, November 2002.
- [10] D. Gonzalez Bautista, J. Pérez, V. Milanes, and F. Nashashibi, “A review of motion planning techniques for automated vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 11 2015.

- [11] Y. D. Setiawan, P. S. Pratama, S. K. Jeong, V. H. Duy, and S. B. Kim, “Experimental comparison of a* and d* lite path planning algorithms for differential drive automated guided vehicle,” in *AETA 2013: Recent Advances in Electrical Engineering and Related Sciences* (I. Zelinka, V. H. Duy, and J. Cha, eds.), (Berlin, Heidelberg), pp. 555–564, Springer Berlin Heidelberg, 2014.
- [12] D. H. Kim, N. Trong Hai, and W. Joe, “A guide to selecting path planning algorithm for automated guided vehicle (agv),” in *International Conference on Advanced Engineering Theory and Applications*, pp. 587–596, 01 2018.
- [13] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, “Path planning with modified a star algorithm for a mobile robot,” *Procedia Engineering*, vol. 96, 12 2014.
- [14] X. Yang and C. Wushan, “Agv path planning based on smoothing a* algorithm,” *International Journal of Software Engineering & Applications*, vol. 6, pp. 01–08, 09 2015.
- [15] A. K. Guruji, H. Agarwal, and D. Parsediya, “Time-efficient a* algorithm for robot path planning,” *Procedia Technology*, vol. 23, pp. 144 – 149, 2016. 3rd International Conference on Innovations in Au-

tomation and Mechatronics Engineering 2016, ICIAME 2016 05-06 February, 2016.

- [16] K. Daniel, A. Nash, S. Koenig, and A. Felner, “Theta*: Any-angle path planning on grids,” *J. Artif. Intell. Res. (JAIR)*, vol. 39, 01 2014.
- [17] C. Wang, L. Wang, J. Qin, Z. Wu, L. Duan, Z. Li, M. Cao, X. Ou, X. Su, W. Li, Z. Lu, M. Li, Y. Wang, J. Long, M. Huang, Y. Li, and Q. Wang, “Path planning of automated guided vehicles based on improved a-star algorithm,” *2015 IEEE International Conference on Information and Automation*, pp. 2071–2076, 2015.
- [18] OSHA, “Occupational Safety and Health Administration.” <https://www.osha.gov/>.
- [19] R. Bytes, “Reality Bytes, Robot Accessories.” <https://realitybytes.blog/2018/08/17/graph-based-path-planning-a/>.
- [20] “TIAGo Navigation Stack.” <http://wiki.ros.org/Robots/TIAGo>.

Chapter 8

Appendix

Figure 8.1: Automated Forklift at the Warehouse

The following figure is the CAN wiring on the truck to establish communication.

Figure 8.2: CAN Wiring to Establish Communication with Forklift