2020 16th IEEE International Conference on Automation Science and Engineering (CASE)
August 20-21, 2020, Online Zoom Meeting

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

Rashmi Ballamajalu[1], Maojia Li[2], Ferat Sahin[3], Clark Hochgraf[4], Raymond Ptucha[5] & Michael E. Kuhl[6]

Abstract— 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.

I. Introduction

Warehousing applications that require significant human labor are rapidly moving towards automation, with focus on large vehicles such as forklifts as Autonomous Mobile Robots (AMRs). 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 paper.

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 [1] 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.

[1], [3]Department of Electrical Engineering, Rochester Institute of Technology, Rochester, NY (rb4609,feseee)@rit.edu

[4]Electrical, Computer, and Telecommunications Engineering, Rochester Institute of Technology, Rochester, NY cghiee@rit.edu

[2], [6]Industrial Systems and Engineering, Rochester Institute of Technology, Rochester, NY (mxl8487,mekeie)@rit.edu

[5]Computer Engineering, Rochester Institute of Technology, Rochester, NY rwpeec@rit.edu

AMRs are intended to assist human operators perform routine tasks with similar efficiency while co-existing in an environment with material, equipment, and people. The safety of the vehicle's behaviour in autonomous mode is paramount, as these robots will be interacting with humans 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. 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 path planner must be aware of the changes as they occur and re-plan 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.

The focus of this paper is developing 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 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 paper is organised as follows. Section II briefly explores the existing and upcoming path search technologies. The A* algorithm is discussed in section III and the proposed Turn and Orientation Sensitive (TOS) A* algorithm and the design modifications are detailed in section IV. A simulation experiment and results are presented in section V, and conclusions are presented in section VI.

II. Related Work

Path planning algorithms for mobile robots have largely been explored considering small holonomic robots with the path having the shortest travel distance considered to be optimal. For the application considered in this paper, 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. [2]. The survey illustrates how uniformly spaced, irregular, or a mesh grid can be used for planning paths, specifically those that cater to real-time applications and dynamic re-planning in case of unplanned obstacles in the environment. 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.

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.

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.

Although these algorithms solve the path planning problem quite effectively, the time taken to train them is high, and they 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 shorter path in faster computational time than A*. 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 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 paper 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].

An improvisation of A* intended to suit a specific application of parking autonomous vehicles has been shown in Dolgov et al. [17]. They use a modified A* along with an optimization method to find paths faster. Another similar solution to complex urban path planning has been presented in Ferguson et al. [18], where a hierarchical approach with a high level planner first and then a low level planner have been shown through experiments. A modified A* specifically intended for autonomous mobile robots has been presented in Wang et al. [19] 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 paper tries to intuitively reduce the number of turns as the path is being explored and gives the best solution in a single run.

III. Default A* Algorithm

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.

The A* search algorithm requires a pre-defined map of the environment that can be a static or a dynamic map. In this paper, 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 interesting aspect here that makes A* a smart search algorithm is the heuristic function. The exploration is concentrated in the direction of the goal rather than a breath-first search approach such as that used in Dijkstra's algorithm. The heuristic function can vary by implementation. Some examples of which were discussed in section II. The most commonly used function is the Euclidean distance which calculates the shortest path from the start to goal location. In the next section, the various modifications to the heuristic function is proposed with regard to the kinematics of the robot and the vehicle characteristics.

```
Algorithm 1: A* Algorithm Pseudo-code
Result: Search for a Path from Start to Goal
Initialize $open\ list = [start]$;
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:
                $f(n) = g(m,n) + h(n, goal)$
            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
```

Where $f(n)$ is cost of the current neighbour 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 $goal$. In the algorithm, m is the current node under consideration whose neighbours are n.

IV. 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.

A. Minimizing Number of Turns in the Path

A forklift's turning pivot point is near the fork-tip, similar to Ackermann steering, thus the algorithm has been modified to give turn paths with a non-zero radius and assumes 90-degree turns. The algorithm explores only four of its neighboring nodes, instead of all eight. The local trajectory planner smoothens these 90-degree turn paths and also ensures that the vehicle can make a turn given its footprint within the available space in an aisle.

In attempt to minimize the number of turns in the path that is generated, several existing approaches count the edges present in the planned 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 paths that are close to the edges of blocked cells.

Additionally, AMRs such as forklifts cannot perform the turns greater than 90 degrees in an aisle, and local planners often cannot recover the vehicle's behaviour and get it back on track. 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,

$$h(m,n,t,goal,p) = [w_0 + w_1 h_1(m,n,t) + w_2 h_2(p,m,n)] \\ (h_0(n, goal)) \quad (1)$$

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 and updated based on multiple trials as the heuristic function is an approximation of the cost from the current location to the goal.

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

The 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. However, the disadvantage with using only the Euclidean distance function is that multiple neighboring cells have the same cost.

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

$$h_1(m,n,t) = 1\ or\ 0 \quad (3)$$

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. However, this does not consider the prior heading of the vehicle.

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. 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,

$$h_2(p, m, n) = 1 \ or \ 0 \tag{4}$$

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 has higher priority in (1) than w_1 based on experiment results and has been explained in Section V. It 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 (1) are added before multiplying with the Euclidean heuristic in order to normalize the equation and scale it down to cost of distance.

Section V details and illustrates the advantages of the TOS-A* and the use cases a associated with it.

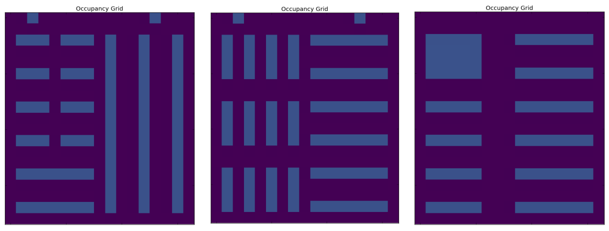
V. Simulation and Results

The implementation of the proposed method has been done using Python and Matplotlib [20] for simulating the path planning and movement of the robot.

The experiment setup consists of 6 different warehouse maps of varying levels of density in terms of the shelves placement and area between the aisles. 50 trials were conducted for each map and the number of turns taken, distance travelled and the number of iterations in which the algorithm found a path was recorded. The weights assigned to each term were selected based on the experiments conducted on the first 2 maps and the same were validated on the remaining 4. The parameter t relates to the number of steps clear for movement in the direction of heading, this could be forward or in reverse. This must be changed for each map based on the aisle width.

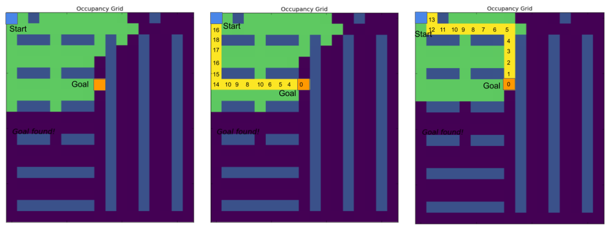
An example of the warehouse layouts setup is shown in Fig. 1. The top two boxes correspond to loading and unloading bay areas. The rest of rectangular boxes represent the shelves or blocked regions of the grid. The remaining region is open for exploring paths. Each warehouse setup consists of about 200 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).

In the scenario under consideration using map 1 of Fig. 1, (0,0), the origin of the grid is the start location and (6,8) is the goal location as shown in Fig. 2a. 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. Green region in Fig. 2a shows the algorithm exploration of the map based on the heuristic functions illustrated in section IV-A and IV-A.2. Once the goal is found, the path is formed from the goal point to the start point choosing those grid cells that have the least overall cost.



(a) (b) (c)

Fig. 1: Simulation Example of Real-world Warehouse Layouts (a) Map 1 (b) Map 2 (c) Map 3.



(a) (b) (c)

Fig. 2: (a) TOS-A* Grid Map with Start and Stop Locations. Generated Paths with Cost by (b) TOS-A* (c) Default A*.

The path generated by default A* is shown in Fig. 2c where nodes with the shortest distance (cost) to the goal are successively chosen until goal is reached. This contains 3 turns and not suitable for a large vehicle such as a forklift. Fig. 2b shows path generated by TOS-A* has only one turn, with updated cost based on the modified heuristic function.

Fig. 3 shows the plot of velocity of the robot as it traverses the above shown paths generated by both algorithms. 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. 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 affects 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.

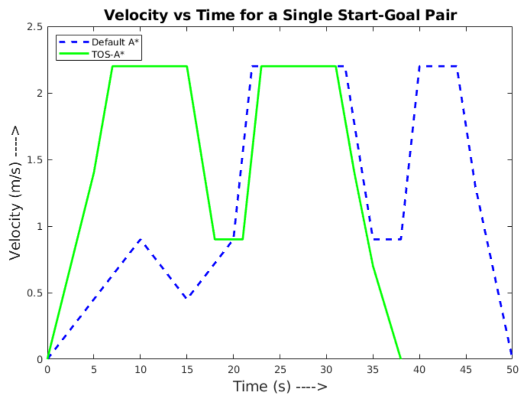


Fig. 3: Change of Velocity over Time as robot traverses the Generated Path.

A. Optimization of Parameters

Table I shows summary of 50 trials over 3 varying maps where TOS-A* algorithm has a much better performance over default A*. And table II shows the summary of parameters such as distance of the generated path, number of turns and number of iterations that the algorithm needed to explore in order to generate a path.

TABLE I: Performance of TOS-A* over Default A* for 50 Trials (Map 1, 2 & 3) considering Number of Turns and Direction of Heading.

Performance	Map 1	Map 2	Map 3
Better	17	13	18
Similar	30	33	29
Worse	3	4	3

TABLE II: TOS-A* comparison with Default A* for 50 Trials (Map 1, 2 & 3) considering Number of Turns and Direction of Heading.

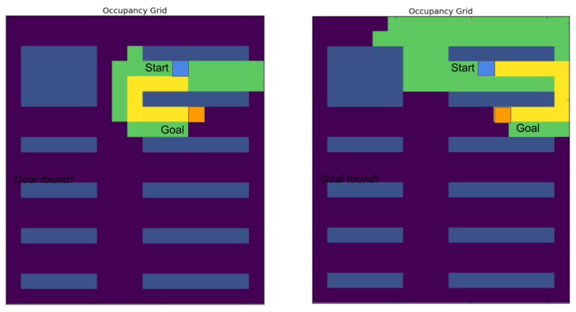
Algorithm	Map	Total Distance Travelled (m)	Number of Turns Taken (units)	Number of Iterations
TOS-A*	1	742	97	2466
Default A*	1	742	108	2513
TOS-A*	2	584	93	2209
Default A*	2	530	100	1879
TOS-A*	3	613	85	2337
Default A*	3	590	97	2285

Additionally, 50 trials considering only the turning heuristic function were conducted using (1) where $w_2 = 0$. The 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.

It was observed that the effect of direction on the overall path is considerable. Based on the experiments using the first 2 maps, the weights assigned for the experiments are $w_0 = 1$, $w_1 = 0.3$ and $w_2 = 0.5$ using these previous trials. 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 as shown in table I.

Furthermore, it can be observed from the two tables that when the map is simpler with more aisle width, TOS-A* performs better than default A*. However, in a more complex environment such as in Map 2, although TOS-A* performs better, it is only marginally better and takes more number of iterations in each run. However, paths proposed by TOS-A* algorithm within an aisle are straight compared to those proposed by default A*. This is especially useful for large vehicles such as forklifts since these vehicles cannot execute turns within an aisle or a 180° turn. Additionally the local planners can only smoothen these paths but do not change the original path given by the global planner. In most cases, the paths proposed by global planners become sub-optimal for the application and if the vehicle is backed into an aisle, recovery by the local planner is usually ineffective.

The parameter t is the number of steps the algorithm looks ahead at every step it takes. For example, if the aisle width is 2 and the vehicle is starting off right beside a shelf, it has a gap of 1 step before it will need to turn due to the shelf opposite to it. In this case $t = 2$ is the optimized value that gives best results. If the aisle width is 3, then $t = 3$ is the optimal value. Consider the example shown in Fig. 4a where the robot can look ahead only one step and in Fig. 4b where it can look ahead 2 steps and look for a better route with less number of turns and suitable for a large robot. This value could potentially be tied in with a sensor such as a LiDAR to move away from manually modifying it for each type of map or environment.



(a) (b)

Fig. 4: Parameter t in Map 3 (a) $t = 1$ (b) $t = 2$.

B. 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. 5a shows the obstacle in the middle of the already explored path. As the robot traverses the path shown in Fig. 2b, the obstacle is identified and the grid is updated with this information as shown in Fig.

5b. A new start location is created which is one grid cell behind the obstacle point.

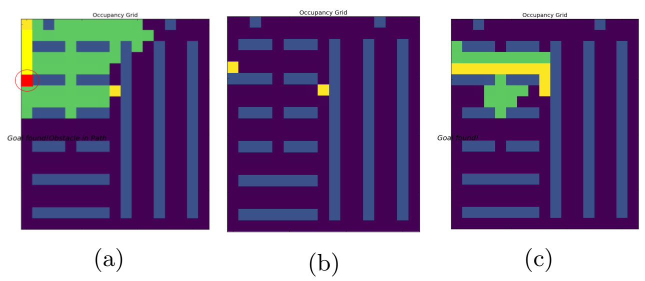


Fig. 5: Dynamic Path Planning with (a) Obstacle highlighted in Red (b) Updated Grid Map including Obstacle and New Start Location (c) New Path with Updated Start Location.

Once the new start point is defined, the algorithm has to repeat the search since the heuristic function results in a non-uniform cost map and the prior values cannot be reused. Fig. 5c shows the new explored cells from the updated starting point and computes a new path.

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. Communication with the real forklift has been setup with vehicle kinematics model encoded into the navigation stack. TOS-A* is currently being incorporated into the ROS navigation stack for real-time path planning and navigation.

VI. 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 path planning algorithm 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.

VII. Acknowledgement

This research is funded in part by a grant from Toyota Material Handling North America and The Raymond Corporation.

References

[1] "Forklift Accidents Statistics." https://www.mccue.com/content/forklift-accident-statistics.

[2] 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.

[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, "Multiplerobot 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 Automation 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] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," The International Journal of Robotics Research, vol. 29, no. 5, pp. 485–501, 2010.

[18] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," Journal of Field Robotics, vol. 25, no. 11-12, pp. 939–960, 2008.

[19] 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.

[20] "Reality Bytes, Robot Accessories." https://realitybytes.blog/2018/08/17/graph-based-path-planning-a/.