Multiple Goal A* path planning guided by potential functions

Tiaan Spies[1]

***Abstract*— With the boom in the autonomous car market, focus is being placed on developing reliable algorithms to replace human capabilities. One example of this problem is parking a car in a shopping centre. Here, a suitable path must be found to an open parking spot closest to the entrance. However, this is not the only objective. Some paths may require too much maneuvering or be too close to obstacles causing another parking spot to be more desirable. The algorithm mimics human selection of a parking spot, balancing control effort against the walking distance from the parking spot to the entrance of the shopping center. A-Star is an effective path finding algorithm but will have to be modified to balance prioritizing the shortest path with getting too close to obstacles. To tackle this an A-Star algorithm that uses potential planning as part of the heuristic is implemented. Furthermore, the problem is formulated as a multi-goal problem with a potential function setting weights for different parking spots, depending on their distance to the entrance of the mall. The work improves over prior works by allowing multiple goals with a heuristic to guide the planner to preferred locations. The results will be evaluated on whether a suitable path is found if it exists, and how convenient the path is in terms of walking distance left after parking versus how difficult it is to park the car.**

I. INTRODUCTION

Path planning robotics has been used extensively in industry thus far. However, it is often only used to plan from a single starting node to a single goal node. For example, in warehouse robotics the start and end position are known, the challenge arises in finding the shortest path to link them without colliding with obstructions. The next stage in the development of these algorithms is to allow more freedom in the design, where they are allowed to select one of many goals that satisfy some requirements. This paper develops a modified A* algorithm to find a path for parking a car in a shopping center. Here the car will have many available solutions but some may be more desired than others. Furthermore, the problem is not as simple as finding the closest parking spot to the entrance of the shopping centre. Some parking spots may be difficult to maneuver into, requiring extra control effort. For instance, requiring many changes in direction and driving backwards. The goal of this paper is to use potential methods to impose weight on the Hybrid A* search algorithm to guide it to near optimal goals without drastically increasing run time. More optimal paths could be found if a search is run for each goal and then ranked according to their paths, however, this would be much more costly than the proposed method as shopping centers have many parking spots. The aim of this paper is to use potential planners to guide A* in finding a solution that is simple enough that a non-holonomic robot could follow the path without explicitly planning for the non-holonomic robot. This would allow for much faster search times than with a non-holonomic search and can be used as a starting point for finding a non-holonomic path. The paths also find desired parking spots in a manner that matches human selection of parking spots. An example of the path finding environment is shown in figure 1.

Fig. 1: Example of shopping center parking area.

II. PRIOR WORKS

For this problem different path planners can be considered. Potential planners [1] alone are not well suited as they may not always find a solution if one exists. These planners are susceptible to getting trapped in local minima, causing low success rate in environments with a lot of obstacles. The problem is better suited as a graph search problems such as A* or Dijkstra's algorithm. A* was selected over Dijkstra's algorithm as it returns results faster in most cases [2]. A* has also been shown to be effective for path planning for autonomous vehicles when coupled with the kinematics of the vehicle [3], this method is called Hybrid A* or nonholonomic A*. There has been work to speed up the search by fusing Hybrid A* with visibility diagram planning [4], showing up to 40% speed improvements.

Work using Hybrid A* has been extended to more complex kinematic problems such as a truck and trailer problem [5]. As the kinematics become more complex it becomes exponentially more expensive to find an optimal path as the configuration space becomes much larger. Problems with a truck and trailer not only find a path but find the correct actions leading up to the path to ensure the trailer is in the correct orientation. These types of problems increase

This work was not supported by any organization

[1]Tiaan Spies is with the Department of Mechanical Engineering, Boston University, 110 Cummington Mall, MA 02215, United States spiest@bu.edu

dimensionality drastically. For example a car problem has 3 dimensions, but a truck and trailer problem can have 5-8 dimensions [6].

For large search spaces such as the truck and trailer problem it may be more suitable to use a probabilistic approach. One example is RRT*, this method allows the search to expand by a step size that is not limited to the centroid of a discrete environment which limits A*. For nonholonomic searches RRT is the prefered algorithm and would be a better choice if non-holonomic search was being implemented. However, [7] investigated the differences between modified A* and RRT algorithms for path planning of a simple holonomic robot, finding that in most cases the A* preformed best. [8] applied RRT search to a nonholonomic search using an "Any Angle path" as biasing. Here the Any Angle path is pre-computed in a holonomic environment, which is able to very quickly find a path. This path can then be used to guide RRT through its nonholonomic search. The results show that this search method is effective at finding paths in high dimensional problems. However, it must also be noted that [8] also showed that RRT and RRT* can fail when coupled with angle angle path biasing due to being forced into a certain homotopy class.

As this paper is finding a holonomic path, A* is used due to its easy implementation and fast run time in holomomic searches. The paths will be assessed on how close they mimic a nonholonomic path. Furthermore, the paths will be compared to unmodified A*.

III. PRELIMINARIES

A. The A* algorithm

The algorithm was first introduced by [9] and is a directed graph search algorithm. The search space will have to be discretized and translated into a graph before it can be searched. A simple method for creating the graph is to split an environment into a grid and connect surrounding nodes to the center node as neighbors. See figure 2 for examples of a 4 and 8 connected graph. A 8 connected graph allows for shorter paths to be found as traveling diagonally may be a closer approximation to the shortest path. This connectivity is applied to each node, creating a graph that is traversable in all directions. The algorithm implemented uses an 8-connected graph.

(a) 4-Connected (b) 8-Connected

Fig. 2: Examples of grid graph connectivity

When using the A* search algorithm, each node that connects to a new node must have a cost associated with the edge. In the algorithm below it is important to note that O is a priority queue. Therefore, the node with the smallest saved heuristic value will be popped every time. By expanding the node with the smallest heuristic value, A* is able to follow nodes with the best chance of being the shortest path first. Furthermore, A* remains complete by continuing to expand all nodes if dead ends are found.

Algorithm 1 A* Algorithm

Require: $O,\ C,\ f(x),\ n_{start},\ n_{goal}$
function A* SEARCH($x_{initial}$)
  $O \leftarrow n_{start}$ ▷ Add star node to open set
  while O not empty **do**
    $n \leftarrow pop(O)$
    $C \leftarrow n$ ▷ Add node to closed set
    if n=goal **then**
      Break
    end if
    for Each neighbor n' of n not in C **do**
      OrganizeNode() ▷ Update costs if improved
      $O \leftarrow\ n'\ with\ f(n')$ ▷ Add to priority queue
    end for
  end while
  return Path from n_{goal} to n_{start}
end function

The evaluation function for A* can be written as the sum of two parts, see equation 1. Where X is the coordinate of the node position. The equation for the heuristic is shown in equation 2. X is a list with index 0 being the x coordinate, while index 1 is the y coordinate.

$$f(X) = g(X) + h(X) \tag{1}$$

$$h(X) = \sqrt{(X_0 - X_{goal,0})^2 + (X_1 - X_{goal,1})^2} \tag{2}$$

Where $g(x)$ is the cost from the current node, back to the starting node through the graph. $h(x)$ is the heuristic that predicts the cost from the current node to the goal node. By selecting nodes from the stack based on f allows the algorithm to focus on searching nodes that will lead to lowest total cost. Furthermore, selecting a heuristic function that is optimistic permits the algorithm to stop once the lowest f in the stack is larger than the successful path length. No other nodes would achieve a shorter path, even if they travelled in an optimal path to the goal. An example of an optimistic heuristic the euclidean distance between the current node and the goal. The A* algorithm is shown in algorithm: 1. Note that the path returned is a list of nodes between the start and goal.

B. Potential Planners

Potential planners are a simple method to perform path finding. Usually an attractive potential is placed at the goal, with each obstacle receiving a repulsive potential. Equation 3 below shows an example of a quadratic attractive potential. The robot would use the gradient of the potential to traverse towards the goal.

$$a(x) = (x - x_{goal})^2 \tag{3}$$

Figure 3 shows an example of an attractive potential where the goal is set to (0, 25). The value of the potential field is increasing in all directions away from the goal. This is used in the planner to attract the A* algorithm towards the entrance of the shopping center.

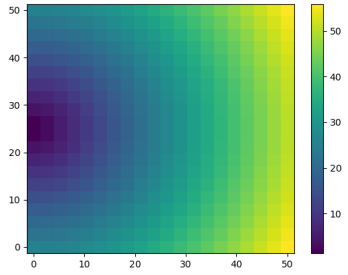


Fig. 3: Attractive potential

A repulsive potential is the inverse of the attractive potential with large values when close to the obstacle and zero when away from the obstacle. Equation 4 is an example of a repulsive potential where it is largest at the obstacle and approaches towards zero as the obstacle distance approaches the influence distance. Figure 4 is an example of the repulsive function with an obstacle at (25, 25). This potential is added to the parking spots to prevent the A* algorithm finding a goal that it drives past. The aim is to only have A* find a goal that is also close to the entrance of the shopping center. Therefore, by adding a slight repulsive potential the A* algorithm will tend to stay away from the spots until another potential also attracts it and overrides the repulsive potential. The potentials are scaled to find a balance between attraction and repulsion.

$$r(x) = (\frac{1}{d_{obs}(x)} - \frac{1}{d_{influence}})^2 \tag{4}$$

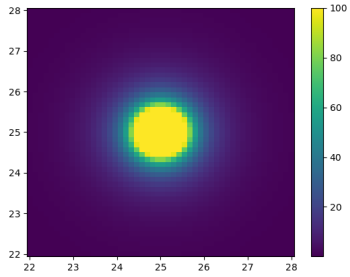


Fig. 4: Repulsive potential

IV. METHODS

A. Simulating parking map

For all tests the simulated shopping center parking setup is shown in figure 5. It contains many goals as well as multiple paths to reach them. The large orange dot is the pedestrian entrance to the shopping center. Therefore, it is desirable to park as close to this orange dot while not needing to take many turns making the parking procedure complicated. Thus minimizing walking distance and control effort.

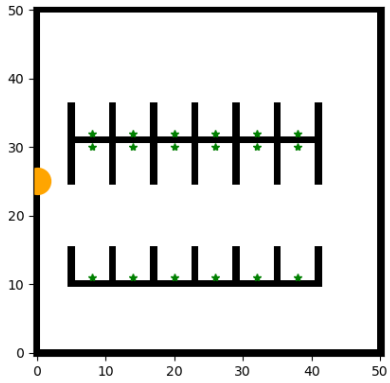


Fig. 5: Simulated parking map

B. Multiple Goal A*

To allow A* to search for multiple goals a sink node is added to the graph. This node does not exist as an actual point. This sink node is then connected to each of the goal nodes with a zero cost edge. Therefore, if the algorithm reaches one of the goal nodes the next node in the priority queue is guaranteed to be the sink node as it has zero cost. The algorithm then only has to check for when it reaches the sink node, once reaching the sink node it can follow its back pointers to find the path it took to reach the goal node. Furthermore, the heuristic function must be adjusted to return the straight line distance to the closest goal.

C. Modified evaluation function

The evaluation function f of the A* algorithm was modified to use the potential planners for guidance. See equation 5. Here $a(X)$ is the attractive potential from equation: 3 and $r(X)$ is the repulsive potential from equation: 4. The coefficients C are used to balance the influence of each potential. Using a large C_1 term causes the planner to prioritize finding parking spots near the entrance of the shopping centre, while large C_2 expands the radius with which the car tries to avoid obstacles.

$$f(X) = g(X) + f(X) + c_1 \cdot a(X) + c_2 \cdot r(X) \tag{5}$$

The plot of the repulsive potential combined with the attractive potential for the simulation map are shown in figure 6. The figure shows that the repulsive potential is generally dominates but only applies close to the obstacles. However, the attractive potential is applied over the entire map. It is important to note that the evaluation function is no longer optimistic after applying the potentials. Furthermore, the paths that are best suited to the problem are not the shortest path. The search will stop once the first goal is found, returning the path with lowest cost to goal.

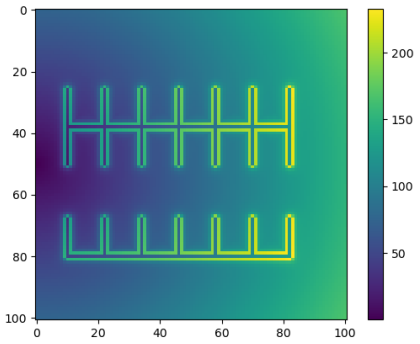


Fig. 6: Repulsive Potential used in map

V. RESULTS

Figure 7 shows the result when starting in the top right of the image. The result gives a good path that is closely represents a path a person may select when parking. There is a parking spot that is closer to the entrance but it would require driving around to the other side of the parking spots while only giving a marginally better result. Therefore, the selected goal can be considered best. The path also stays clear of the obstacles, giving a path that is likely traversable by a nonholonomic system. Analysis could transform the path into one that is nonholonomic but that is left for future works. However, it must be noted that the path can be improved as it takes an detour before returning to the straight path.

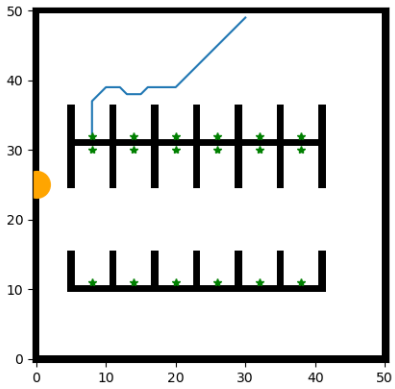
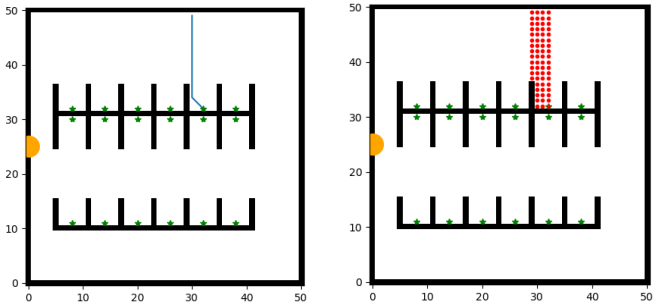


Fig. 7: Resultant path with start point in top right of map.

For comparison figure 8 was complete with A* without modifications. The paths efficiently finds the closest goal node to the starting position. However, this parking spot is not ideal as there is a large distance left to walk to the entrance of the mall. The comparison shows how the updated heuristic has improved goal finding based on the adjusted requirements.

It is important to analyze the nodes that were expanded as well as the path that is found. Figure 9 steps through how the nodes progress. During the first 40 steps the path initially moves directly towards the entrance before turning into the parking spots. The nodes then reach the repulsive

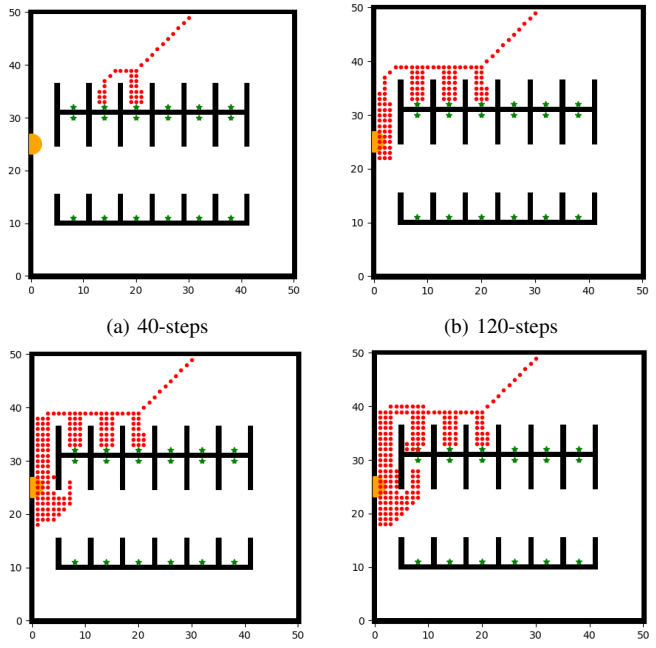


(a) Normal A* path, no modifications

(b) Normal A* node expansion, no modifications

Fig. 8: A* without modification

potential as the approach the obstacle, stoping the nodes from expanding onto the goal node. After which the nodes start expanding towards the next parking spots again. At 120 steps the nodes have expanded to the mall entrance and begin expanding everywhere along the path. The nodes keep expanding until it the cost of the repulsive function is not large enough to prevent the node expanding onto the goal nodes. When complete we see that the goal did not select the spot that was the closest to the entrance. This is because the node closest to the entrance has a larger g in it's evaluation function (f) due to the longer path between the goal and start.



(a) 40-steps

(b) 120-steps

(c) 160-steps

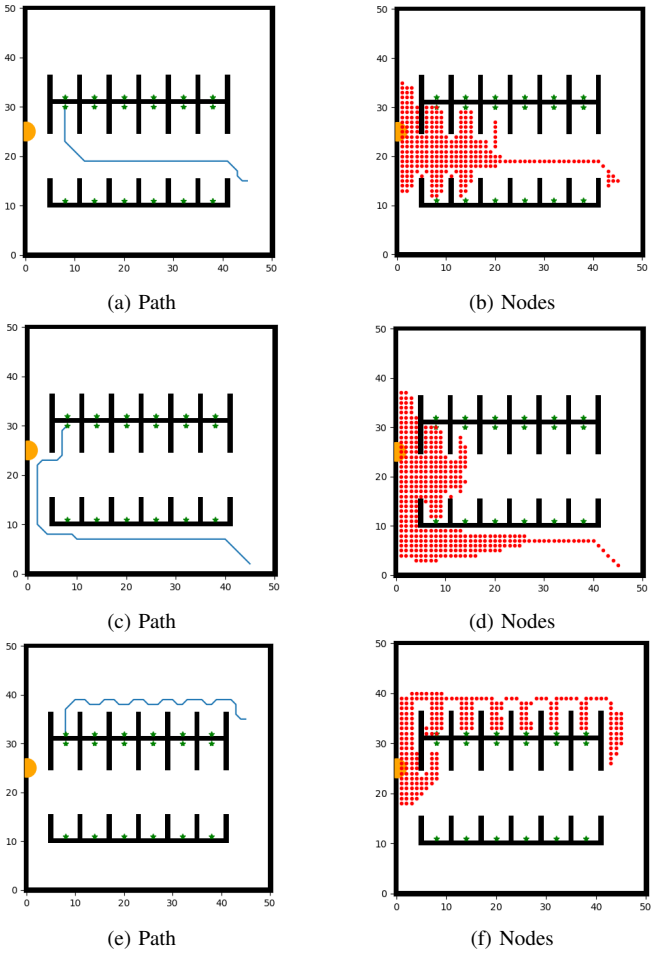
(d) Path complete

Fig. 9: Node expansion example

Different starting positions are tested and shown in figures 10-11, we see that the issue of the paths following a "zig-zag" path when driving alongside the parking spots appears multiple times. We also see figure 11) where a couple of goals are blocked by an obstacle. This scenario is for if cars

are parked there. The path found is a suitable path and can be seen as a good compromise between closest to the goal and easiest path to traverse. We see that the algorithm is robust to obstacles without the need to remove the goals. However, the algorithm expands many more nodes in order to find a solution as the optimal goal moves further from the entrance.

In general the results are promising. The paths stay clear of the obstacles while selecting good parking spots. A simple modification to the A* algorithm gives very different behaviour. This method will compute faster than finding an A* path for each goal node then selecting the best path based on the requirements.



(a) Path (b) Nodes

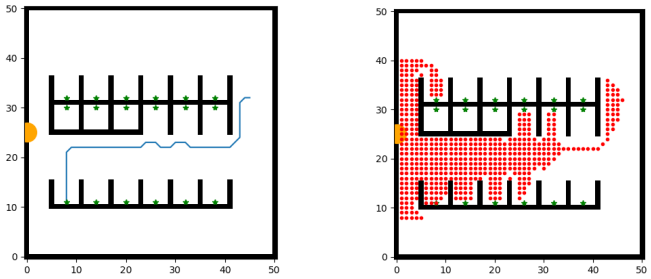
(c) Path (d) Nodes

(e) Path (f) Nodes

Fig. 10: Generated Paths

VI. CONCLUSION

The proposed adjustment to the A* algorithm successfully finds parking spots in a similar method to which humans with it. This method minimized multiple parameters where regular A* only minimizes the path length. Furthermore, the paths found are more easily translated into a nonholonomic path as they tend to not have sharp turns. All of these complex behaviours arise from a simple adjustment to the evaluation function. These results show that A* can be modified to work in many different scenarios.



(a) Path (b) Nodes

Fig. 11: Path with obstacles

REFERENCES

[1] S. Ge and Y. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 615–620, 2000.

[2] X. Liu and D. Gong, "A comparative study of a-star algorithms for search and rescue in perfect maze," in *2011 International Conference on Electric Information and Control Engineering*, pp. 24–27, 2011.

[3] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[4] S. Sedighi, D.-V. Nguyen, and K.-D. Kuhnert, "Guided hybrid a-star path planning algorithm for valet parking applications," in *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, pp. 570–575, 2019.

[5] G. L. Woods, *Evaluation of Local Kinematic Motion Planning Algorithms for a Truck and Trailer System*. PhD thesis, 2020.

[6] P. Zips, M. Böck, and A. Kugi, "An optimisation-based path planner for truck-trailer systems with driving direction changes," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 630–636, 2015.

[7] L. da Silva Costa and F. Tonidandel, "Dvg+ a* and rrt path-planners: a comparison in a highly dynamic environment," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, pp. 1–20, 2021.

[8] L. Palmieri, S. Koenig, and K. O. Arras, "Rrt-based nonholonomic motion planning using any-angle path biasing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2775–2781, 2016.

[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.