Multi-Robot Planning for Restaurant Environments with Priority-Planning and Active Paths

Syed Adnan Akhtar(4791916), Saket Sarawgi(4809254), Arjan Vonk(4219201)

***Abstract*— This paper presents an approach to automate order delivery to tables within a restaurant environment. The paths of two unicycle robots inside the restaurant are calculated by solving a path planning problem. The restaurant is considered a static environment and collision avoidance is done by determining the collision free path that satisfies the criteria for the shortest distance. The A*-algorithm has been implemented for path planning. Low level controllers control the robots by implementing motion primitives. The robots are shown to navigate the environment using priority planning based on active paths. Simulations have been carried out that indicate that the robots traverse the trajectories and reach the goal without colliding with each other.**

I. INTRODUCTION

Many restaurants want to use robots to deliver food to the tables. Previous research has been done on this subject in restaurants that were predefined [5] and in beforehand unknown environments [6]. This paper expands on the subject of a predefined restaurant. The success of mobile robots in these applications depends on the path planning algorithm. This involves the creation of an optimized collision-free path between two points. Path planning can be divided into categories depending on the nature of the environment: static path planning, where the obstacles do not change their position with time, and dynamic path planning where the obstacles do change their position and orientation with time. The indoor environment used in the restaurant is characterized by the presence of surrounding walls, tables and multiple mobile robots. In the paper, it is assumed that no people are roaming around in the vicinity of both the robots.

To solve this problem, this paper uses the A* algorithm [2] which is a classic algorithm for finding the shortest path between two points due to its optimization capability. The paper also considers the difficulty of two mobile robots passing through the same point without colliding.

The paper is organized as follows. section II briefly describes the forward kinematics of a unicycle robot model. In section III, path planning algorithms are discussed which use predefined cell decomposition. This is followed by the explanation of how the low-level controllers are implemented based on motion primitives and how the centralized controller drives the robots to the goal location without collision using Priority Planning on Active Paths. In section IV the simulation of the running model in the restaurant environment is presented. The results show the ability of the planner to get the robots to the goal locations while providing guarantee on no collision.

II. ROBOT MODEL

This paper uses two differential drive robots due to their ability to rotate at the spot allowing them to maneuver easily in environments containing a lot of obstacles. As such, they are expected to have good performance inside the restaurant environment, even if the environment would be more cluttered with obstacles than the environment of this paper . For the differential drive robots the unicycle model was used to describe the dynamics:

$$v = \frac{r\dot{\phi}_r}{2} + \frac{r\dot{\phi}_l}{2} \tag{1}$$

$$\omega = \frac{r\dot{\phi}_r}{2l} - \frac{r\dot{\phi}_l}{2l} \tag{2}$$

where v is the forward velocity of the robot and ω is the rotational velocity of the robot. $\dot{\phi}_r$ and $\dot{\phi}_l$ are the rotational speeds of the right and left wheels respectively. The wheel diameter is given by r. l represents half of the inter-wheel distance.

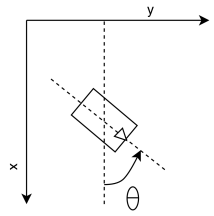


Fig. 1: Orientation of unicycle robot wrt inertial frame

From Figure 1, the equations of motion in the inertial frame are as follows:

$$\dot{x} = \frac{r}{2}(\dot{\phi}_r + \dot{\phi}_l)\cos\theta \tag{3}$$

$$\dot{y} = \frac{r}{2}(\dot{\phi}_r + \dot{\phi}_l)\sin\theta \tag{4}$$

$$\dot{\theta} = \frac{r}{2l}(\dot{\phi}_r - \dot{\phi}_l) \tag{5}$$

with θ the angular difference between the world frame and the robot frame.

Workspace of the robot is given by $W := \mathbb{R}^2$ since the robot moves in $\mathbb{R}^2$ space and the Configuration space of the robot is given by $C := \mathbb{R}^2 \times \mathbb{S}^1$ because of the base position and the orientation in 2D.

III. Motion planning

The restaurant-like environment with four tables used for the assignment is shown in Figure 2. Dividing the environment into a grid like structure, as discussed in [5] makes it possible to navigate the environment in an easy way. To make the path planning computationally cheap a graph planning algorithm with a simple grid like graph was used to guide the differential robots from a start position to the goal location (or goal table) while avoiding obstacles and avoiding collisions with the other robot.

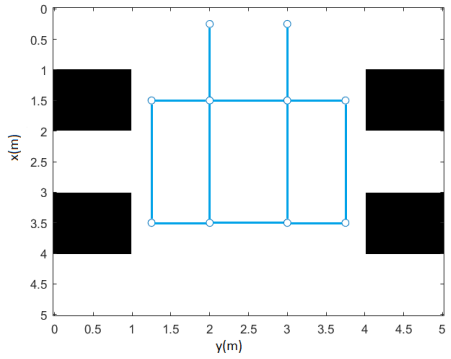


Fig. 2: Restaurant like environment

Due to the small size of the restaurant environment it's still possible to achieve a high performance with a relatively small graph map of the environment. Since the restaurant environmental specifications were static and known beforehand - the graph doesn't change over time and is embedded into the robots, therefore, a pre-defined cell-decomposition is used instead of a sampling based method. The path the robots need to take is then calculated with the A* algorithm.

For simplification, we assume that no dynamic obstacles are present in the workspace except for the other robot. The current motion planning algorithm is in a relatively simple environment with four tables and two robots. It can however be extended to more robots and a more complex environment if no dynamic obstacles are introduced.

$Path \leftarrow Astar(Map, Start, Goal)$
$\omega_0 \leftarrow k_1$ (some constant)
$v_0 \leftarrow k_2$ (some constant)
while $(x, y) \neq Goal$ **do**
  $(x_n, y_n) = getNextCheckpoint(x, y)$
  if $\theta \neq \arctan \frac{y_n - y}{x_n - x}$ **then**
    $\omega = sign(\arctan \frac{y_n - y}{x_n - x} - \theta)\omega_0$
    $v = 0$
  else
    $\omega = 0$
    $v = v_0$
  end
  $(x, y, \theta) \leftarrow$
   $updateRobotState(CurrentState, v, \omega)$
end

Algorithm 1: Controller based on Motion Primitives

The graph used for the motion planning algorithm of the workspace is predefined as can be seen in Figure 2. The hollow circles are checkpoints or nodes and the blue lines are the edges connecting them. Each robot is equipped with a low-level controller based on motion primitives. The controller forward simulates the commands it gets from the path planner and issues those commands which will make the robot move towards the next checkpoint. The controller based on motion primitives is shown in **Algorithm 1**. The four motion primitives are as follows

- 0.1m Forward
- 0.1m Backward
- 90^0 Right Turn
- 90^0 Left Turn

The A* Algorithm, with the euclidean distance to the destination as the heuristic, first calculates the path for each of the two robots independently without taking into account possible collisions. The output of the A* algorithm are the successive checkpoints that the robots must pass through to reach the destination.

A centralized controller is used which issues commands to the robots in real time. As such, it is aware of the location of both the robots. The paper presents Priority Planning based on Active Paths to drive the robots to goal locations without collision. Robot 1 has been allotted priority 1 whereas Robot 2 has been allotted priority 2.

A typical scenario for a two robot system when no solutions exist is shown in Figure 3. The Red Robot, which is assigned priority 1, calculates its path based on graph search. While planning for Robot 2 (Blue in the figure), the path of Robot 1 is considered an obstacle to avoid collision. Therefore, no solution will exist for Robot 2 to reach its goal position. Even if there is a path available after considering the trajectory of Robot 1 as an obstacle, it will be longer than required and hence not optimal. The subsequent paragraphs will present a method wherein the blue robot will wait for the red robot to pass, and eventually start moving towards it own goal. Now we come back to our

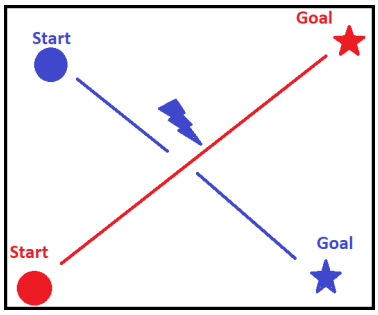


Fig. 3: Two Robot System with Priority based Planning

restaurant-like environment setup as shown in Figure 2. For Robot 1, the path is calculated using A* and the velocity commands are issued using the low level controller based on motion primitives. Therefore, Robot 1 traverses the trajectory as if Robot 2 is not present.

Robot 2, which has the second priority, needs to check for obstacle avoidance. The following is how it does. Given the current position, the robot checks if it is within ϵ distance of the next checkpoint. If yes, it checks whether the next checkpoint is a part of the first robot's path and whether it has already passed that checkpoint or not. If the checkpoint is present in the path of the first robot and it has not yet passed through that, then the second robot stops and waits till the first robot passes that point. If the previous conditions was not true, then the low-level motion primitive controller is activated. Robot 2 does that before arriving at every checkpoint. The Controller for Robot 2 is shown in **Algorithm 2**

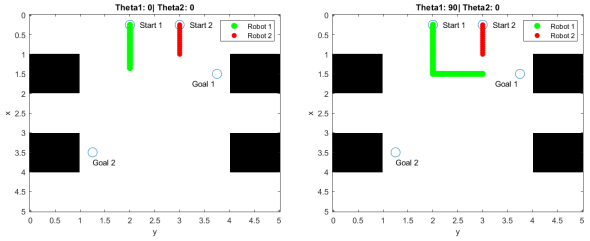
$Path1 \leftarrow Astar(Map, Start1, Goal1)$
$Path2 \leftarrow Astar(Map, Start2, Goal2)$
while $(x_i, y_i) \neq (Goal2.x, Goal2.y)$ **do**
  $j \leftarrow indexOf(Robot2.NextCheckpoint)$
  $(x_1, y_1, \theta_1) \leftarrow$ Robot1.pose
  $(x_2, y_2, \theta_2) \leftarrow$ Robot2.pose
  if $dist([x_2, y_2], path2(j, :)) \leq \epsilon$ **then**
    $m = indexOf(Path1(:, :) == Path2(j, :))$
    if $ActivePath1(m) == 1$ **then**
      $(v, \omega) = (0, 0)$
    end
    else
      $(v, \omega) =$
      $calcMotPrim(Robot2.pose, Path(j, :))$
    end
  end
  $updateRobotState(Robot2, v, \omega)$
end

Algorithm 2: Robot 2 Controller based on Priority Planning with Active Paths

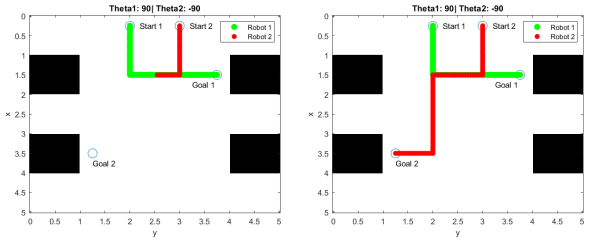
IV. RESULTS

The controllers were implemented in MATLAB. The position of the robots were updated at a timestep of $0.05s$ and their positions were successively plotted. As stated before, the environment includes four tables and two robots to deliver the food with robot 1 being prioritized. Robot 2, when in ϵ radius of the next node, will check whether that node is in the path of Robot 1 or not. If yes, it will wait for Robot 1 to cross that node. After Robot 1 has crossed that node, it will follow the pre-computed trajectory using the motion primitive controller. In Figure 4a, Robot 2 is waiting for Robot 1 to cross the node which is common to both the trajectories. Robot 2 starts moving towards its goal as soon as Robot 1 crosses the common node as shown in Figure 4b where robot 2 is seen waiting for robot 1 to cross that node. Figure 4c and Figure 4d show that once there are no more common trajectories or nodes to both the robots, they will eventually reach their goal and stop.

Simulations were run for another case when the goal positions of both the robots were changed as in Figure 5.

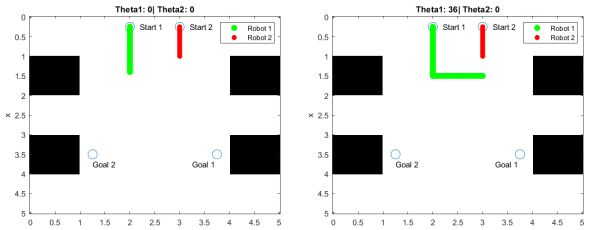


(a) Position at $t = 4.1$s (b) Position at $t = 7.0$s

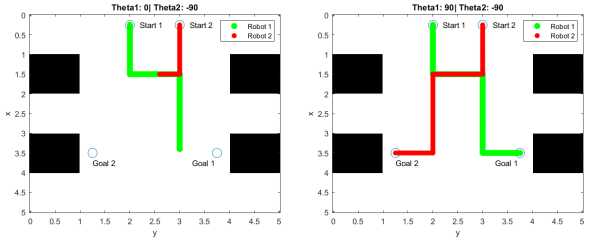


(c) Position at $t = 8.6$s (d) Position at $t = 18.2$s

Fig. 4: Case 1: Position at different time steps



(a) Position at $t = 4.1$s (b) Position at $t = 7.0$s



(c) Position at $t = 9.2$s (d) Position at $t = 18.2$s

Fig. 5: Case 2: Position at different time steps

Robot 2 sometimes has to wait a long time for Robot 1 to pass, if they are to return to their start position after serving a table. To counter this problem a different area graph was proposed, that only contains one-way streets as shown in Figure 6.

Figure 7 to Figure 11 show the results of implementing one-way street. Robot 1 reaches the bottom right table and back, whereas Robot 2 reaches the bottom left table and back in 23 seconds. The two way street take 43 seconds to perform the same task.

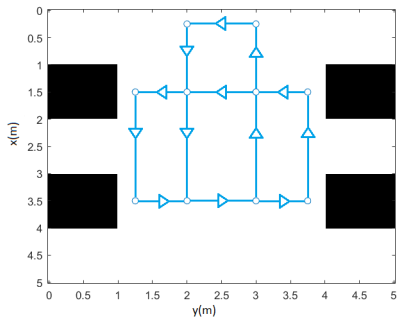


Fig. 6: Graph of the environment with only one way edges

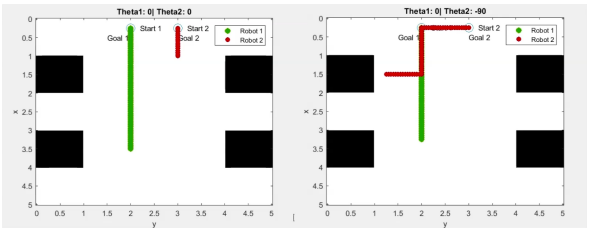


Fig. 7: One way system at $t = t_1$

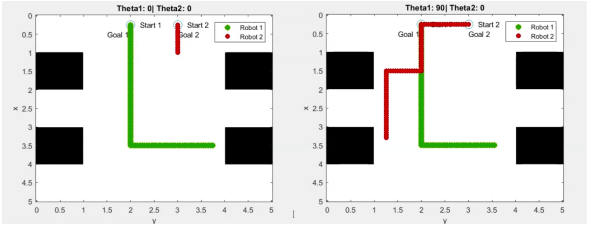


Fig. 8: One way system at $t = t_2$

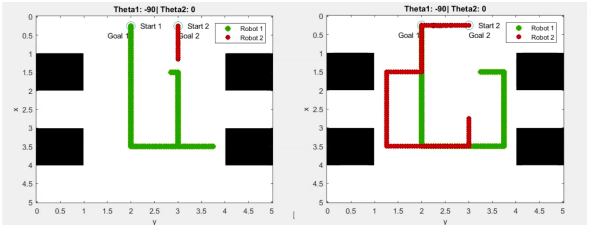


Fig. 9: One way system at $t = t_3$

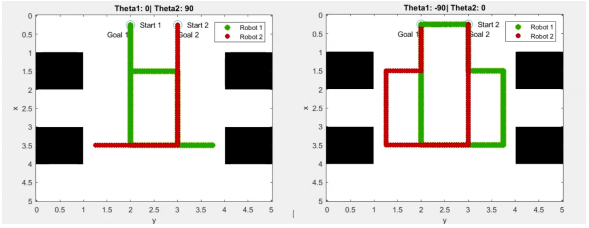


Fig. 10: One way system at $t = t_4$

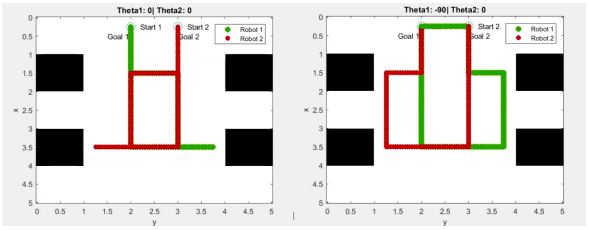


Fig. 11: One way system at $t = t_5$

V. DISCUSSION

The controllers presented are very computationally cheap. This is because the environment is well known a priori and only a few points of interests are included in the graph which makes graph search using dijkstras or A* really fast.

The presented algorithms were able to drive the robots towards the goal without colliding. The paper assumes that the low level controller implements the motion primitives perfectly. However, in reality, there will be some offset from the desired position, orientation or velocity. Therefore, some method for path tracking needs to be present such as a Model Predictive Controller or PID. The authors believe that use of magnetic strips along the edges of the graph may be a good way for path tracking. However, this has not been tried yet by the authors. Moreover, the presented controller for Robot 2 requires the checkpoint navigation data of Robot 1. This needs a communication module as well. For this paper the robots were assumed to be dots moving on a graph. More advanced robots [4] can also be used by using this path planning algorithm making it possible to provide great service, and also more diverse tasks.

As future work, the controllers can be expanded with local sensor data, avoiding dynamic obstacles as well as develop path-tracking controller.

REFERENCES

[1] M. Erdmann, T. Lozano-Perez; On Multiple Moving Objects. Algorithmica 2(4), pp. 477-521, 1987.

[2] P. E. Hart, N. J. Nilsson, B. Raphael, “A formal basis for the heuristic determination of minimum cost paths”, IEEE Transactions on Systems Science and Cybernetics, vol. SSC-4, no. 2, pp. 100-107, 1968.

[3] A hierarchical approach for primitive-based motion planning and control of autonomous vehicles David J. Grymin, Charles B. Neas, Mazen Farhood

[4] K. Kim, Y. Cha, J. Park, J. Lee and B. You, ”Providing services using network-based humanoids in a home environment,” in IEEE Transactions on Consumer Electronics, vol. 57, no. 4, pp. 1628-1636, November 2011.

[5] G. Huang and Y. Lu, ”To build a smart unmanned restaurant with multi-mobile robots,” 2017 International Automatic Control Conference (CACS), Pingtung, 2017, pp. 1-6.

[6] J. Zhang, Y. Ou, G. Jiang and Y. Zhou, ”An approach to restaurant service robot SLAM,” 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), Qingdao, 2016, pp. 2122-2127.