Spring 2021 Path Planning with Dynamic Obstacles

Nikilesh Subramaniam, ns4bb@virginia.edu, nikileshsub@gmail.com

May 2021

Contents

1 Objectives

The goal of this project was to develop an autonomous robot that could quickly navigate along a path around moving obstacles. The application in mind for this robot was to pass robots in a highway scenario or race and overtake a manually controlled robot. The robot's planning algorithm needed to have a low computational time in order for the robot to update its plan at every time step (0.1 seconds). In this project, the autonomous Jackal robot received updates about the positions of all the obstacles (other Jackal robots) in the scenario and simply estimated their velocities for its planning.

2 Framework

In implementing this project ROS was used to communicate with the robot. Gazebo was also used to interface the robot in a 3D environment. The robot we used was a jackal robot that was had two control inputs: a linear and angular velocity input. Three environments were made in gazebo, the first was a highway with 6 moving obstacles that moved forward at a linear velocity. The second environment was the same highway, but with one manually controlled obstacle that could move in any direction. The third environment was a racetrack with the manually controlled obstacle. In all of these scenarios, the autonomous robot needs to go along the center path as fast as possible while avoiding obstacles and not going outside of the track.

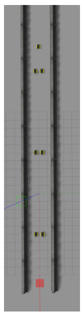

Figure 1: Highway scenario with 6 obstacles, the autonomous robot is at the top of the image

.

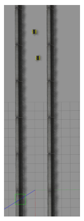

Figure 2: Highway Race Scenario, one autonomous robot and one manual robot ahead of it

.

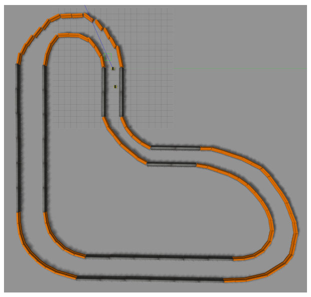

Figure 3: Racetrack Scenario, one autonomous robot and one manual robot ahead of it

.

3 Related Works

Another work that worked on autonomous racing was the game-theoretic planner (GTP) which was a planner that overtook and blocked its competitors during races [1]. The track representation and the tests from that paper influenced this project. Additionally, the particle filter influenced this project as this planning algorithm uses particles to keep track of possible trajectories that the robot could take.

4 Technical approach

The autonomous robots generates a trajectory every time step and then starts to follow that trajectory. To generate the trajectory, the robot uses 5 different primitives of movement: Staying in still, moving straight ahead slowly, moving straight ahead quickly, turning left, and turning right. For N time steps, the planner simulates the obstacles moving at a constant velocity. At each time step, the planner can use any of the primitives to construct a trajectory that avoids the obstacles and progresses the robot along its path. In the highway scenario, the robot wants to go in the +x direction. With 5 primitives and N time steps, the robot has 5^N possible trajectories. Instead of examining each of these trajectories, this planner proposes a method that uses k particles and weighting to only keep track of the k best particles.

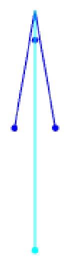

Figure 4: 5 primitives that the robot can use when planning its trajectories: (stay in place, go forward slowly, go forward quickly, turn left, turn right)

.

4.1 Highway Scenario Path Planner

Each particle data structure contains a list of positions, orientations, and control inputs that make up a trajectory. The data structure also has the number of collisions that are expected in this trajectory and the number of particles that share the trajectory. The planner starts with 1 particle data structure that start at the robot's current location with 0 collisions and k particles. This 1 particle data structure represents that there are initially k particles.

For each time step, the planner first simulates where obstacles would be assuming constant velocity. Then, the planner examines each particle data structure. For each particle data structure, the planner determines the next 5 possible locations for the particles by using the 5 primitives and the particles' current position. Each possible location is given a weight based on how safe it is and the progress it makes along the +x direction. If the possible location is within the bounds of the tracks and farther than b units from all obstacles, the location is "safe" and its weight is the new x position minus the old x position. If the possible location is outside of the track bounds, the weight is the negative distance to the track minus β. If the possible location is within b units from an obstacle, the weight is the distance to the obstacle minus b minus β. β is set as the minimum weight of a "safe" location and β is used so that all of the "unsafe" locations have a lower weight than the "safe" locations.

Given that each possible location for a particle data structure is assigned a weight, the particles in the particle data structure are distributed to the new locations. The number of particles assigned to each new location is equal to the number of particles in the data structure multiplied by the new location's weight divided by the total of all positive weights. For every new location that is assigned a particle, a new particle data structure is made. Its trajectory is updated and if the new location was deemed ”unsafe”, its collision number is incremented. If all of the new locations’ weights are negative, the particles are distributed so that the highest weighted location is given half of the particles. The second highest weighted location is given a quarter of the particles and this pattern repeats until all of the particles have been assigned.

The planner updates the list of particle data structures at each time step and it ends with k different particle data structures that represent the k best trajectories that the robot could take. The planner then goes through each particle and chooses the set of trajectories that have the least amount of collisions (usually 0 collisions in a non-crowded environment). Given this set of trajectories, the planner selects the trajectory that moves the farthest along the +x direction. The figure below shows the different trajectories considered by the particle planner. The light blue trajectory is selected because it avoids the obstacles while moving far along the +x direction.

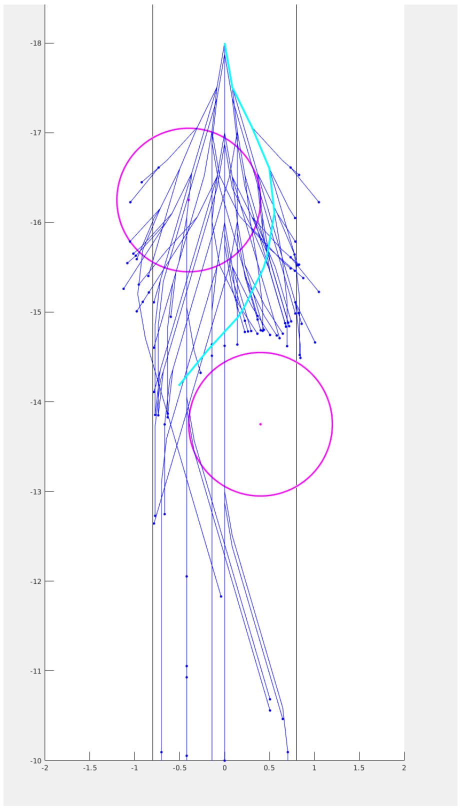

Figure 5: Final particles of the planner (k = 200) Light Blue trajectory is selected as the best

.

4.2 Planner Modifications for Racetrack

The planner was modified slightly for the racetrack scenario. Since the racetrack scenario is not a straight path along the x axis like the highway scenario, the planner's measurement of progress needed to be changed. Given a dense list of way points for the racetrack's center path, the planner's weighting formula was changed to weight "safe" locations based on their progress along the center path. Additionally, at the end of the planning stage, the planner chooses the particle with the least amount of collisions that progresses the most along the center path. It is important that the list of way points for the center path is dense or it will be harder for the planner to determine the relative progress of each trajectory.

5 Experimental Validation

In the highway scenario with 6 moving obstacles, the autonomous robot was able to successfully navigate around the obstacles and overtake them while staying in bounds. The moving obstacles moved at an eighth of the max speed of the autonomous robot. Another test was done with 12 obstacles in the highway and the autonomous robot struggled with this scenario. It would sometimes try to overtake a robot, but it would crash into a robot in the opposite lane.

In both racing scenarios, the autonomous robot was placed behind a manually operated robot. The manually operated robot had a max speed that was half of the autonomous robot, so that the robot would have the ability to overtake the manual robot. In both the highway scenario and the racetrack scenario, the autonomous robot was able to overtake the manually operated robot. The autonomous robot succeeded even though the manually operated robot did not have a constant linear velocity. The manual robot was constantly turning and trying to block the autonomous robot.

Another scenario was tested where the autonomous robot was in front of the manually controlled robot. The manually operated robot had a max speed that was double that of the autonomous robot, so that the robot would have the ability to overtake the autonomous robot. In this scenario, the autonomous robot failed to block the manual controlled robot from overtaking it.

6 Conclusions and future works

Overall, the planner was successful in quickly picking a trajectory to avoid obstacles and quickly progress along a path. When obstacles trajectory differed from a constant linear velocity, the autonomous robot was able so successfully adjust its trajectory.

For future works, this project proposes incorporating game theory into its planner like the GTP [1]. The planner can focus on improving its progress while limiting the progress of its competitors. Additionally, this project proposes more complex obstacle tracking. Instead of simple linear velocity estimations, a more complex obstacle movement model will be more accurate. Additionally, the uncertainty from this movement model should be incorporated into the trajectory planner. Currently, the planner treats all obstacles as circular obstacles even though elliptical or rectangular models are more accurate. Finally, sensing from a LIDAR and/or camera to detect obstacles should be added to this project.

References

[1] Mingyu Wang, Zijian Wang, John Talbot, J. Christian Gerdes, and Mac Schwager. Game-theoretic planning for self-driving cars in multivehicle competitive scenarios. *IEEE Transactions on Robotics*, 2021.