RBE550 - Valet Path Planning Assignment

Keith Chester

Due date: March 21 2022

Introduction

In this assignment, we are tasked with path planning in challenging continuous spaces with unique vehicle-specific kinematic constraints. To accomplish this, we utilize state lattices to explore a continuous space while adhering to to the kinematics of each vehicle. Specifically, we explore a skid drive robot, an Ackermann drive robot, and an Ackermann drive robot with hitch-attached trailer.

Vehicles and Kinematics

First, we introduce each vehicle type and discuss the kinematics of their movement.

Skid Drive

This robot is called skid drive as the robot simply skids the front caster wheel around while two drive wheels operate its movement. The skid drive robot can turn within its own radius, and can perform curved trajectories by differing the angular velocity of the wheels.
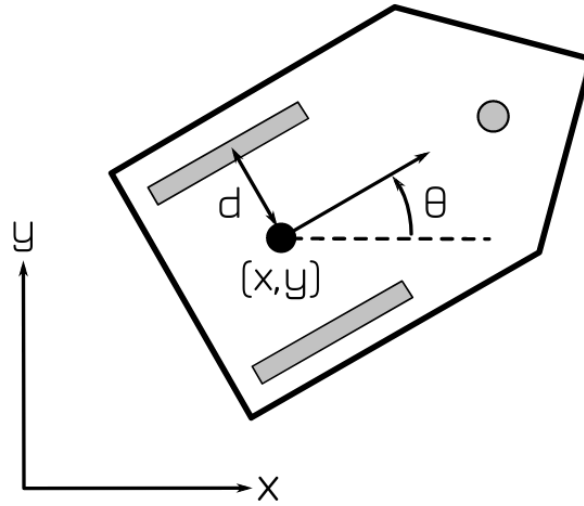

Figure 1: Skid Drive Robot

In our kinematics, we assume that we can directly control the wheel velocities, $U_{l/r}$, where U_l is our left wheel angular velocity and U_r is our right wheel angular velocity. We have a variable L which represents our wheelbase distance (between wheels) and r which is our wheel radius. We use $L = 0.4$ and $r = 0.1$ in meters, estimating a large but still sidewalk manageable delivery robot. For each kinematic step, we are looking at a set time increment, Δt.

$$\dot{\theta} = \frac{r}{L}(U_r - U_l) \tag{1}$$

$$\Delta\theta = \dot{\theta}\Delta t \tag{2}$$

$$\theta = \theta_0 + \Delta\theta \tag{3}$$

$$\dot{x} = \frac{r}{2}(U_r + U_l)\cos(\theta) \tag{4}$$

$$\dot{y} = \frac{r}{2}(U_r + U_l)\sin(\theta) \tag{5}$$

$$\Delta x = \dot{x}\Delta t \tag{6}$$

$$\Delta y = \dot{y}\Delta t \tag{7}$$

$$x = x_0 + \Delta x \tag{8}$$

$$y = y_0 + \Delta y \tag{9}$$

Ackermann

For our robot with Ackermann drive, we have two steerable wheels whose angle are mechanically linked and equivalent. Our rear wheels provide thrust, thus creating a different style of movement from our skid drive. We can no longer turn within our own radius, denoting a need for the path planner to perform K-turns and other maneuvers.
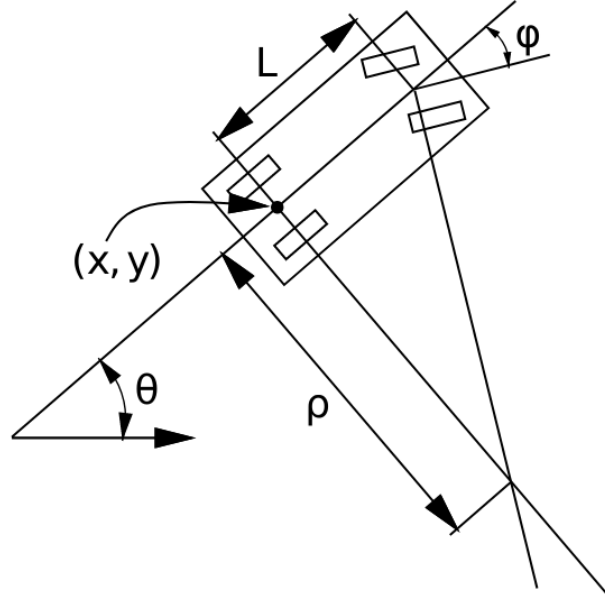

Figure 2: Ackermann Drive Robot

The kinematics of our Ackermann drive robot can be expressed utilizng a controlled velocity v. Our steering is controllable as well, expressed as an agle ψ. For our robot, we chose a wheelbase $L = 2.8$ meters. We allow a maximum $|\psi| = 60°$.

$$\dot{\theta} = \frac{v}{L}\tan(\psi) \tag{10}$$

$$\Delta\theta = \dot{\theta}\Delta t \tag{11}$$

$$\theta = \theta_0 + \Delta\theta \tag{12}$$

$$\dot{x} = v\cos(\theta) \tag{13}$$

$$\dot{y} = v\sin(\theta) \tag{14}$$

$$\Delta x = \dot{x}\Delta t \tag{15}$$

$$\Delta y = \dot{y}\Delta t \tag{16}$$

$$x = x_0 + \Delta x \tag{17}$$

$$y = y_0 + \Delta y \tag{18}$$

Ackermann with a Trailer

We expand our exploration of Ackermann drive robots by adding a hitched trailer, as if our robot was a large truck. The trailer has no control over its movement or velocity - it is pulled at an angle θ_1 relative to the movement of the towing robot. For this assignment we only explored a singular trailer.
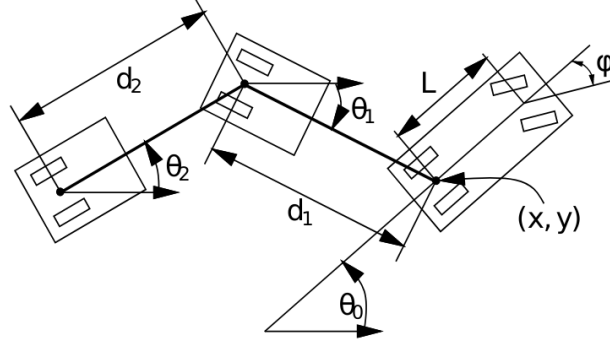

Figure 3: Ackermann Drive Robot with Trailer

We assume a wheelbase $L = 3$ meters and the distance of the trailer hitch to the center of rotation for the robot for a $d = 5$ meters.

$$\dot{\theta}_0 = \frac{v}{L}\tan(\psi) \tag{19}$$

$$\Delta\theta_0 = \dot{\theta}_0\Delta t \tag{20}$$

$$\theta_0 = \theta_{0_0} + \Delta\theta \tag{21}$$

$$\dot{x} = v\cos(\theta_0) \tag{22}$$

$$\dot{y} = v\sin(\theta_0) \tag{23}$$

$$\Delta x = \dot{x}\Delta t \tag{24}$$

$$\Delta y = \dot{y}\Delta t \tag{25}$$

$$x = x_0 + \Delta x \tag{26}$$

$$y = y_0 + \Delta y \tag{27}$$

$$\dot{\theta}_1 = \frac{v}{d}\sin(\theta_0 - \theta_1) \tag{28}$$

$$\Delta\theta_1 = \dot{\theta}_1\Delta t \tag{29}$$

$$\theta_1 = \theta_{1_0} + \Delta\theta_1 \tag{30}$$

Planner Approach

Here we present a flow chart for the path planner as built. We utilize an A* search algorithm. The act of expanding neighboring cells utilizes kinematic equations for each step to create an in-memory state lattic - this allows us to explore the continuous space as if it was a discrete space.

We utilize a priority queue utilizing a heap memory structure. The cost of each node differs for each robot type, but typically is euclidean distance between states and a penalty for θ changes (to encourage straight movement versus swerving when possible). For a heuristic, we heavily penalize euclidean distance ot the goal.
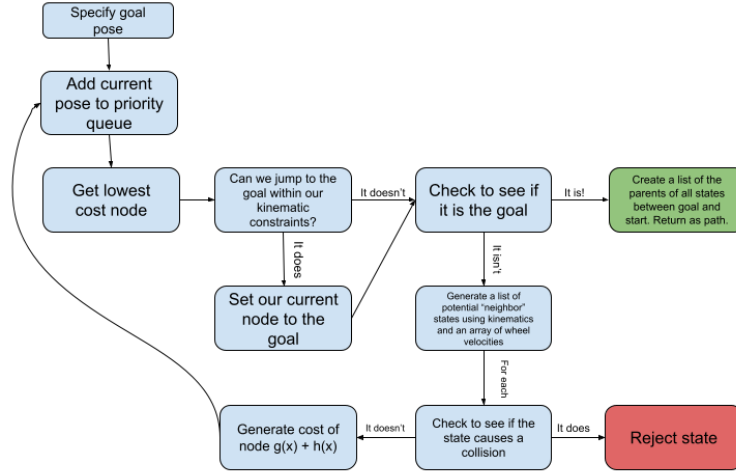
Figure 4: Path planning flow chart

Results

Here we provide examples of a path being generated to match a requested pose in an obstacle-laden environment. Each dot represents a considered state reached via the kinematic equations described above. The green line represents a path dictating which states ultimately led to the goal pose. We provide two examples for each vehicle.
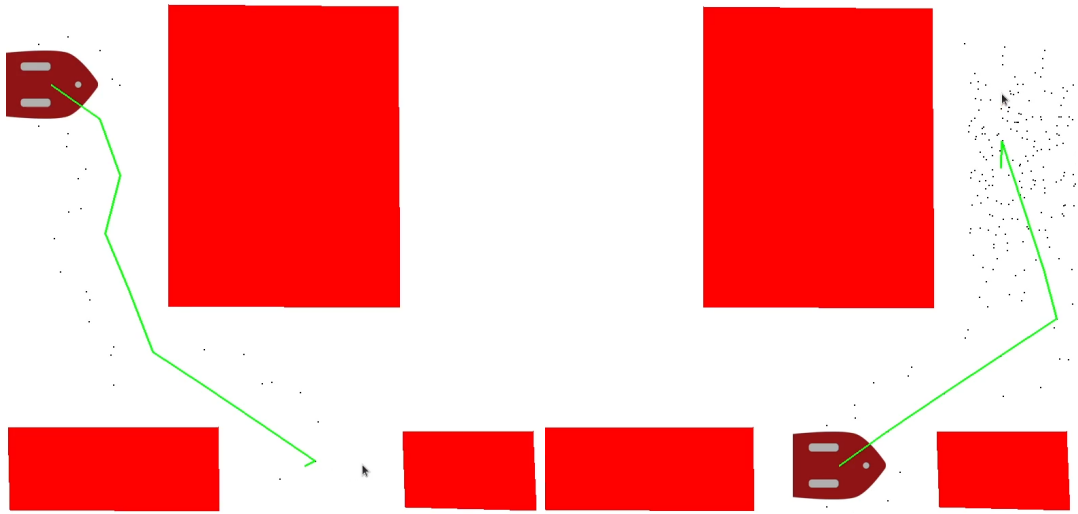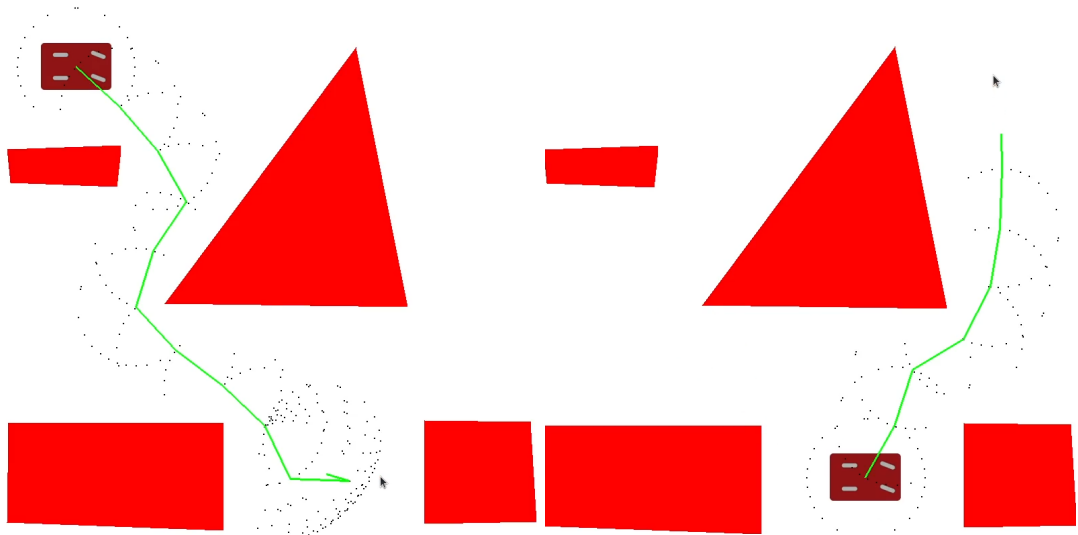
Figure 5: Skid Drive Robot Path Planning

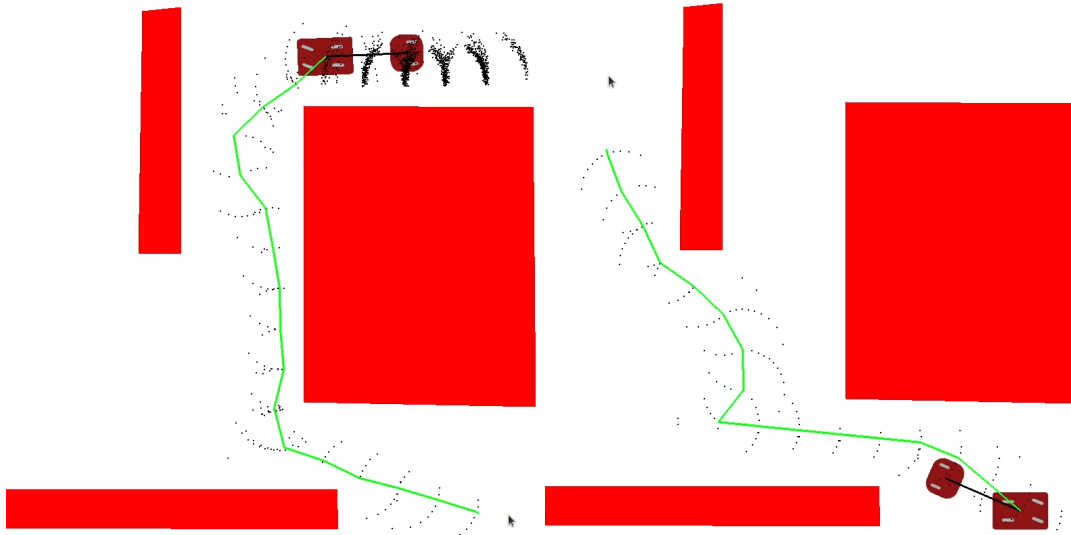Figure 6: Ackermann Robot Path Planning

Figure 7: Ackermann Robot with Trailer Path Planning

Conclusions

In this assignment we created a path planning algorithm that worked in a continuous space utilizing the unique kinematic properties and constraints for each explored vehicle type.

We successfully managed to manuever around the continuous environment while avoiding collisions with obstacles.

During the completion of this project, several challenges were encountered. Our attempts to discretize the continuous space could improve, as we regularly see repeated exploration of previously explored areas; it is difficult to identify and confirm that you've considered a given area prior since we can not purely discretize a given space.

We also witnessed the "tunneling" problem - since we move the vehicle with a given time increment, performing collision checks only at the start and finish poses for a given increment in the planner, we see chosen paths have small momentary collisions occasionally. Checking smaller intervals or performing collision checks for each intermediate step would be computationally expensive, but could be possible solutions.

We also see our simulation occasionally move oddly, identifiable as "sliding". This is due to our methodology of animating changes between poses. When animating our calculated pose, we utilize linear extrapolation of a given delta pose's (x, y, θ) instead of the actual curved trajectory that would have been performed. This was ignored as our simulation was acceptable enough for the time constraints.

It is this author's opinion that A* was, at times, not suitable for exploring the environment. There were times where a complicated environment would result in the algorithm exploring a given area too thoroughly, especially with our problems discretizing the continuous space. Algorithms that are more thorough in exploration, such as RRT*, could have performed better.

Finally, we could have improved A* performance by experimenting with other node transition and heuristic costs.