RBE550(Spring 2023) - Valet

Rohin Siddhartha Palaniappan Venkateswaran

March 20, 2023

This assignment involves planning paths in a constrained environment that pose specific kinematic constraints for each vehicle. To tackle this challenge, state lattices were used to navigate the space while taking into account the unique kinematics of each vehicle. Here, the focus is on three different vehicles: a Skid drive robot, an Ackermann drive robot, and an Ackermann drive robot towing a trailer.

Skid Drive

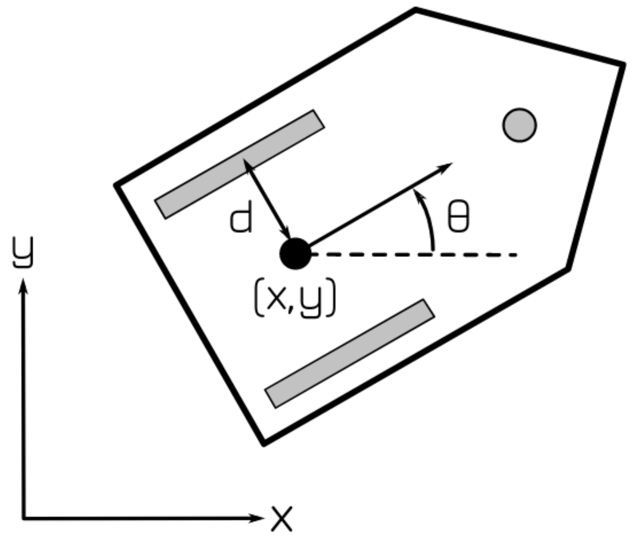

Figure 1: Skid Drive Robot

According to the kinematics, U_l is our left wheel angular velocity and U_r is our right wheel angular velocity, L is the wheelbase and r is the wheel radius. We use $L = 0.4$ and $r = 0.1$ in meters. Let the set time increment be Δt.

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

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

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

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

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

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

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

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

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

Ackermann

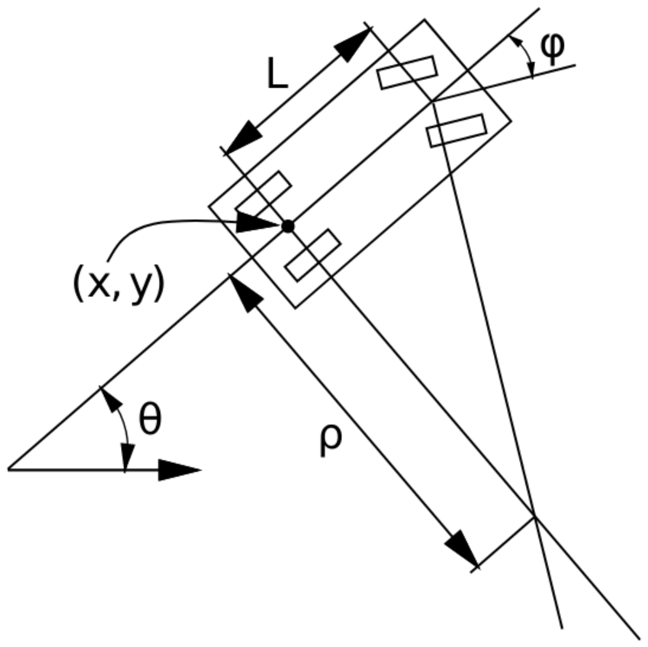

Figure 2: Ackermann Drive Robot

The kinematics consists of velocity v, steering angle ψ. In this assignment, wheelbase $L = 2.8$ meters. Limits of steering angle are also imposed $|\psi| = 60°$.

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

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

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

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

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

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

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

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

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

Ackermann with a Trailer

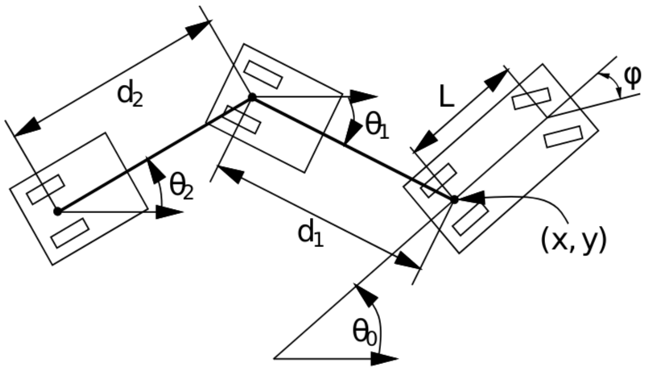

Figure 3: Ackermann Drive Robot with Trailer

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.

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

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

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

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

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

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

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

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

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

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

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

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

The (x, y) of the trailer is calculated from the resulting θ_1 as the distance from the center of rotation for the trailer and truck is a set d meters.

Planner Approach

The standard A* search algorithm has been implemented for the purposes of this assignment. Kinematic time stepping for each step is used to create a state lattice, hence converting the continuous space to discrete space.

A priority queue has been implemented for storing the neighbouring as well as selected states. The cost function is the euclidean distance between states from start to this point, and a theta penalty has also been implemented to ensure an efficient path generation. With regards to the heuristic, the euclidean distance to the goal is penalized by simply multiplying it by a constant.

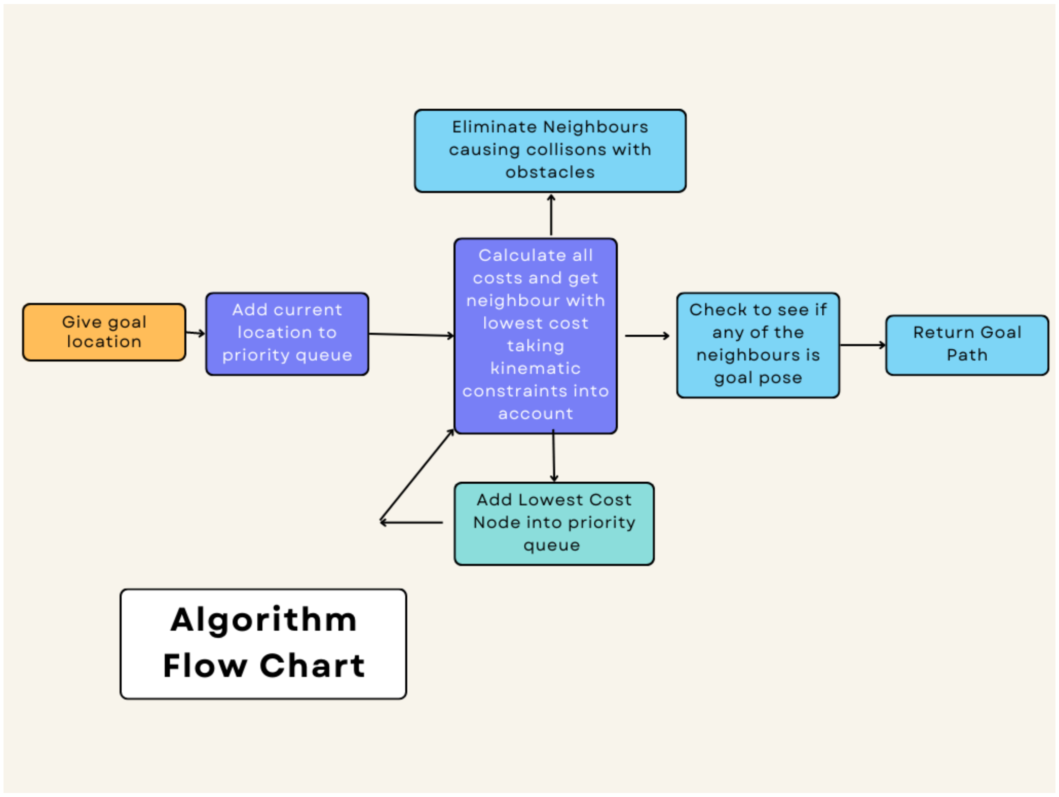
Eliminate Neighbours causing collisons with obstacles
Give goal location
Add current location to priority queue
Calculate all costs and get neighbour with lowest cost taking kinematic constraints into account
Check to see if any of the neighbours is goal pose
Return Goal Path
Add Lowest Cost Node into priority queue
Algorithm Flow Chart

Figure 4: Path planning flow chart

Results

Here the path has been generated to match a requested pose in an obstacle filled environment as instructed in the assignment. The green line represents a path leading to the goal pose.

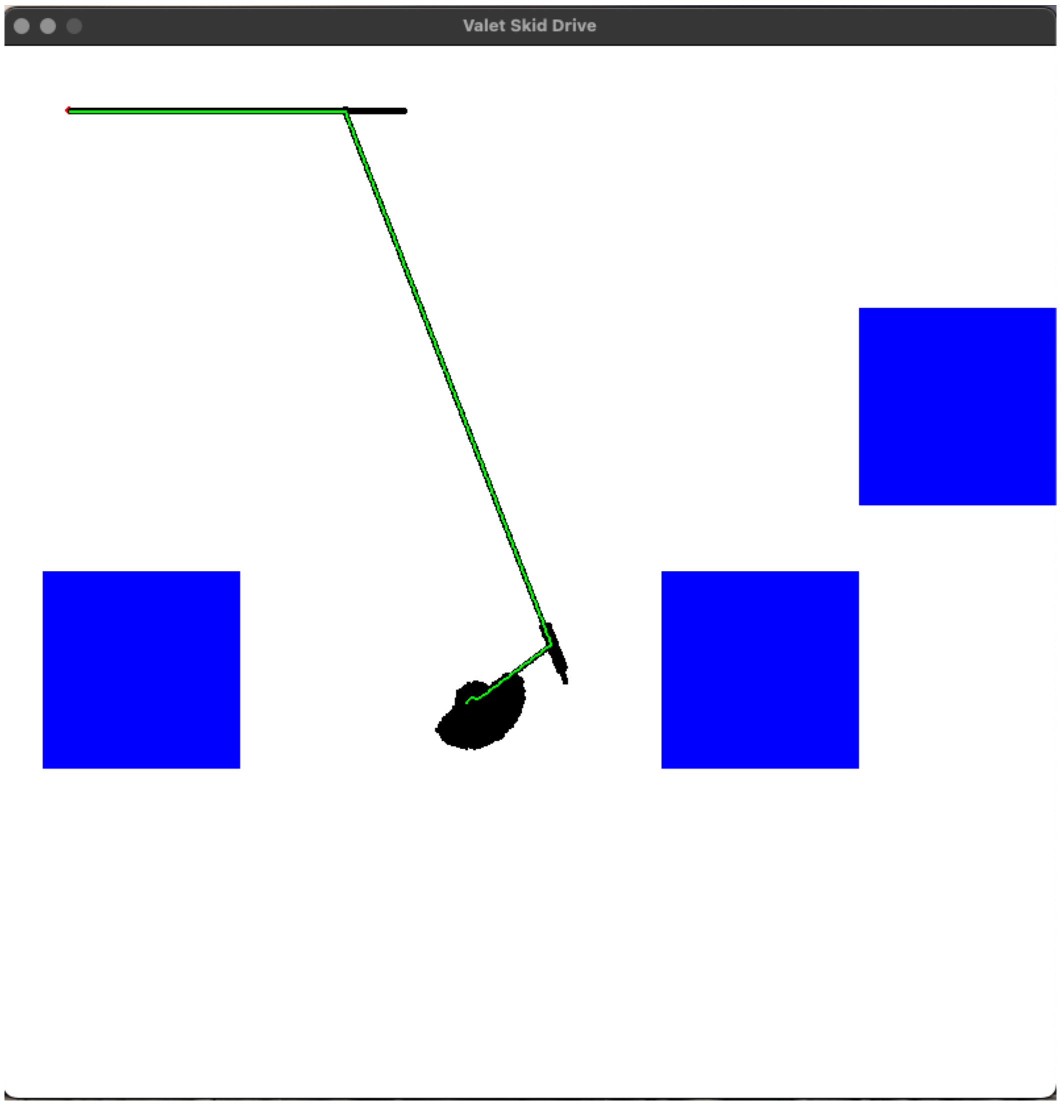

Figure 5: Skid Drive Robot Path Planning

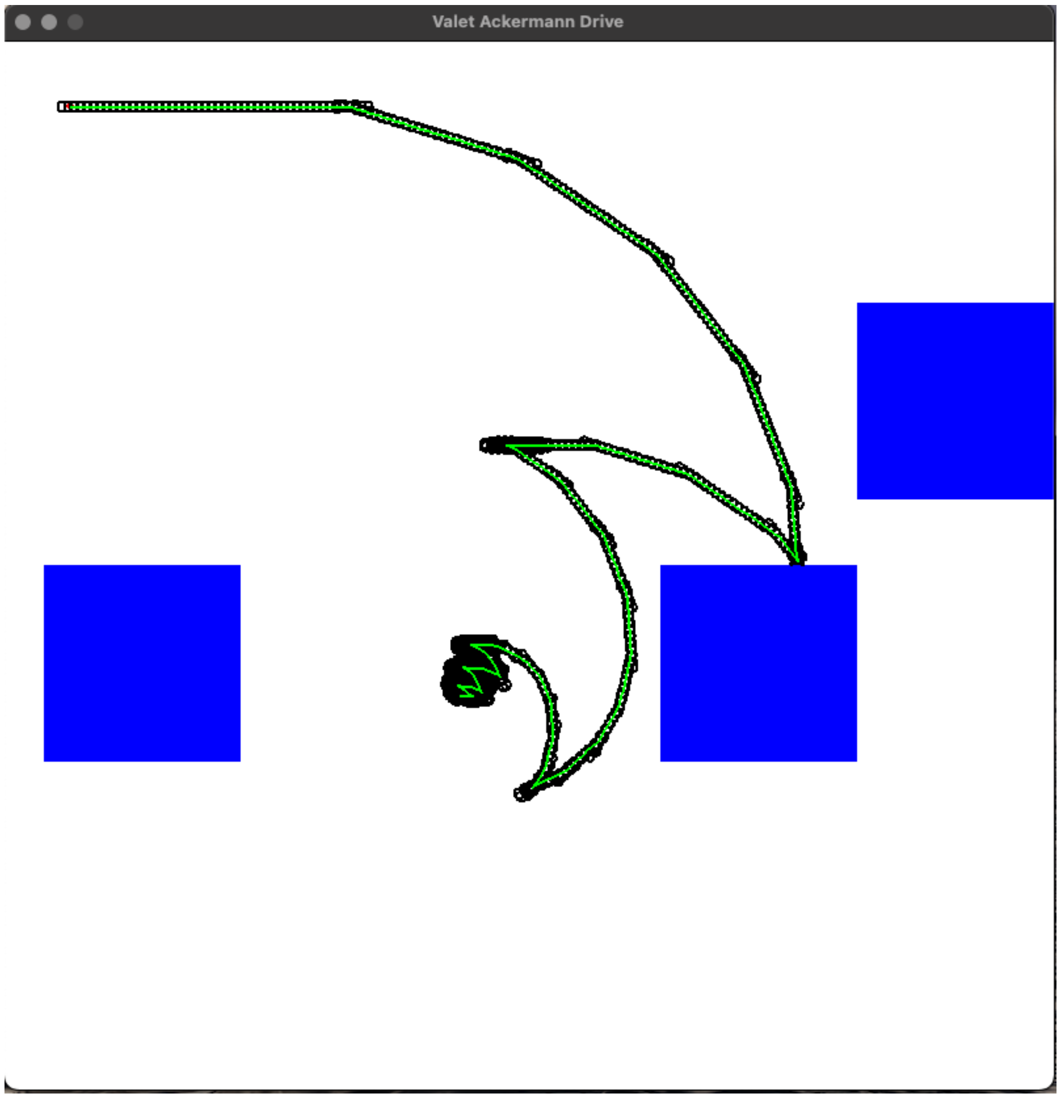

Figure 6: Ackermann Robot Path Planning

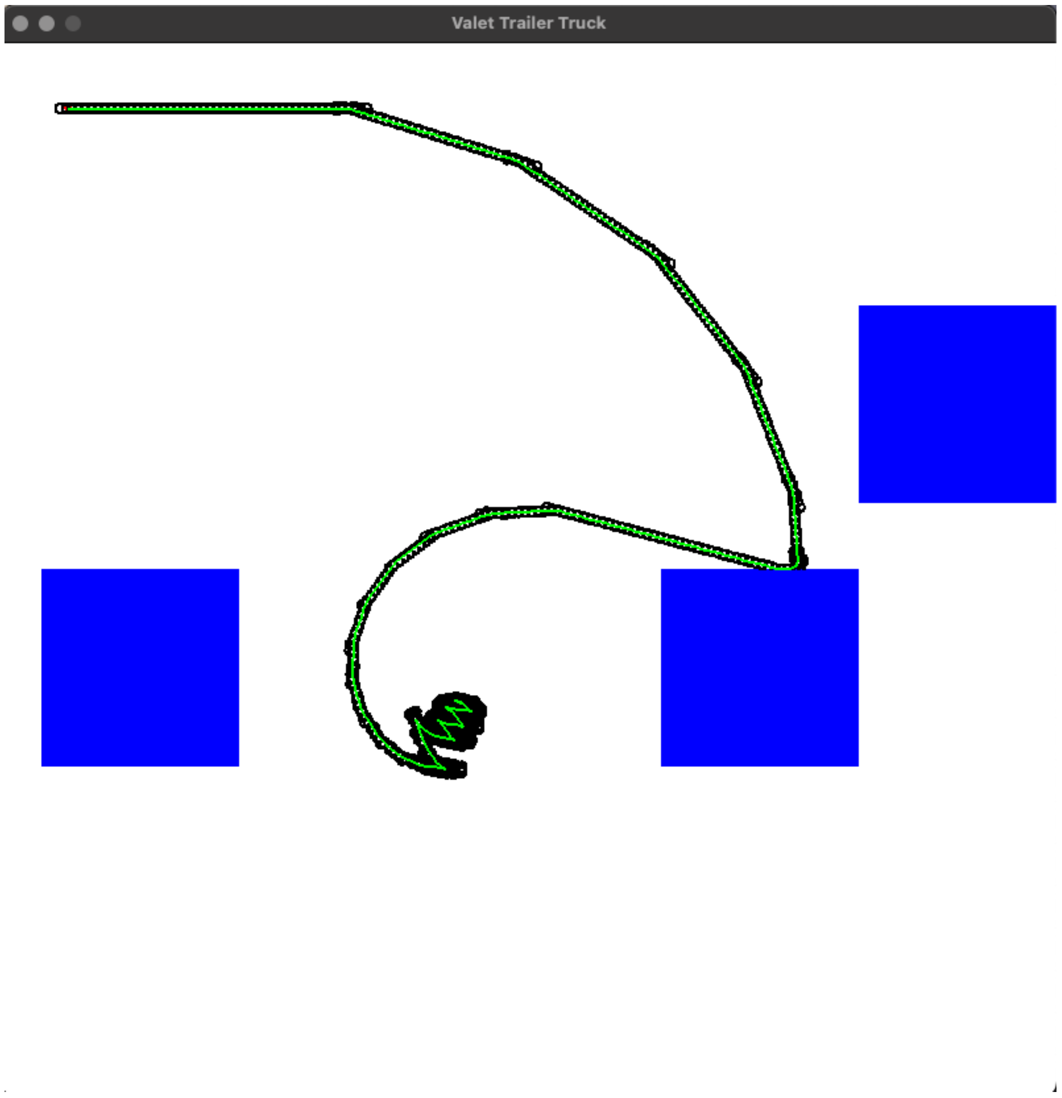

Figure 7: Ackermann Robot with Trailer Path Planning

Final Notes Conclusions

For each type of robot, run the "drive_ackermann.py", "drive_diff_drive.py", "trailer_drive.py". Observe the generated path, and close the generated pygame window. Another pygame window opens which shows the robot being run in the generated path. Closing the first generated pygame window is a must so that the next pygame window opens in which we can see the robot follow the path to the goal

Further, each program is not optimized to run in the fastest time possible, hence, it will take some time to run, depending on the machine on which the code is being run.

Finally, in the trailer truck simulation, observe that the trailer is not simulated to be behind the truck, but it follows the kinematics mentioned in the initial part of the report, as you can see it rotating as the truck also rotates. This is a small simulation issue, which could not be debugged before the submission deadline.