RBE 550: Assignment 3 - Valet

***Abstract*— Autonomous vehicle path planning in tight and congested spaces, particularly during parking maneuvers, poses a common challenge. This paper addresses this issue by proposing a path planning solution for three distinct types of vehicles, while considering both the vehicle kinematics and the obstacles present in the environment. The vehicles of interest are the diwheel robot with differential drive, the car with Ackerman drive, and a trailer. Our approach aims to optimize the path planning process for parking these vehicles in a given environment.**

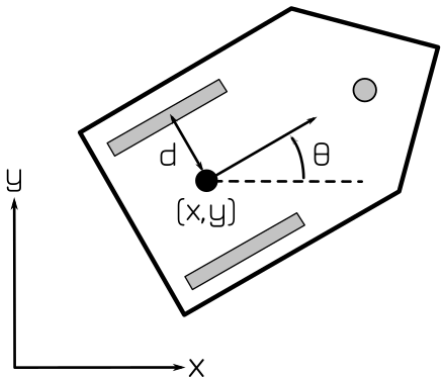
***Index Terms*—Motion planning, Search algorithms.**

ENVIRONMENT

To replicate the real-world scenario of parking vehicles in a constrained environment, we have created a simulation environment using a 200x200 grid. The grid contains a central obstacle (represented by the black color) and two parked cars (represented by the red color) positioned on the southern side of the grid. Our objective is to plan the parking path for all types of vehicles in the available space between the two parked cars, denoted by the red boxes. The obstacle in the center of the grid is indicated by the color code (0,0,0), whereas the parked cars are represented by the color code (254,0,0) on the grid. This simulation environment aims to provide a realistic testing ground for autonomous vehicle path planning algorithms.

THE DELIVERY ROBOT

The delivery robot employs diwheel kinematics, where the input comprises left and right wheel velocities to enable the determination of both the heading angle and position coordinates.



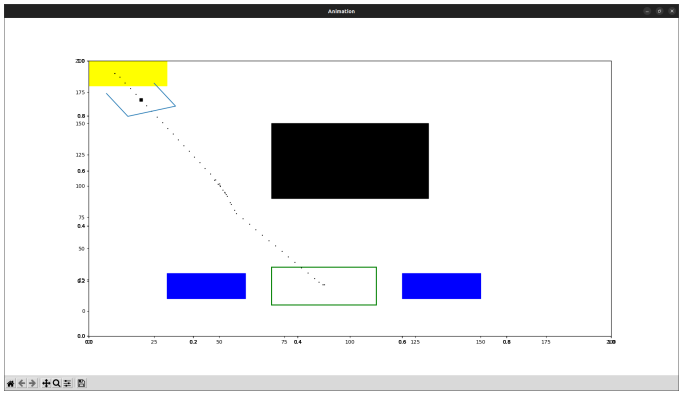
$$\dot{x} = \frac{r}{2}(u_l + u_r)\cos\theta$$
$$\dot{y} = \frac{r}{2}(u_l + u_r)\sin\theta$$
$$\dot{\theta} = \frac{r}{L}(u_r - u_l).$$

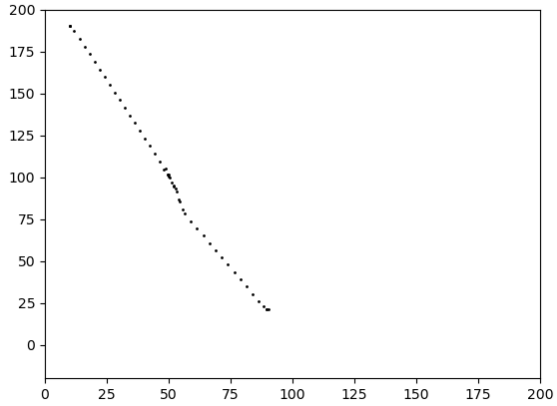
A. Implementation

The starting position for the robot is marked as (20,25) and goal position as (180,85) which is the position between the two parked cars. The length and width of the car is taken as 25 and 12 respectively.

The method used to plan the path for diwheel robot is Hybrid A*. We can't use regular A* because the kinematics of the robot is also included in the system. The code for A* is almost same as Djikstra as done in last assignment only difference will be we have to define a heuristic in A* algorithm. In our case our heuristic is the Eucledian distance between the current position and desired goal position. And at the end we must minimise the sum of cost and heuristic value.

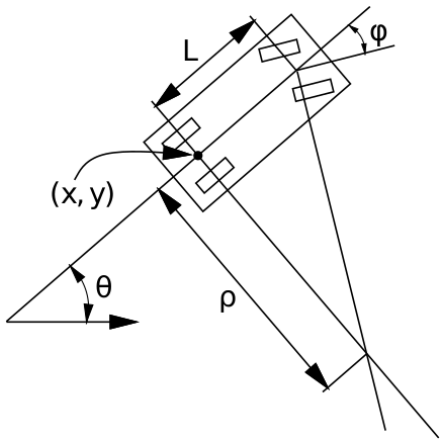
B. Results





THE CAR

The following section pertains to planning for a car, where standard Ackerman steering kinematics constraints have been applied for maneuvering. In Ackerman geometry, both the vehicle velocity and heading angle must be input to determine the final position coordinates of the axle's center. Notably, zero-radius turns are not possible with Ackerman geometry, requiring proper parallel parking execution to park the car correctly. The kinematics for Ackerman geometry in a car have been illustrated.



$$\dot{x} = u_s \cos\theta$$
$$\dot{y} = u_s \sin\theta$$
$$\dot{\theta} = \frac{u_s}{L} \tan u_\phi.$$

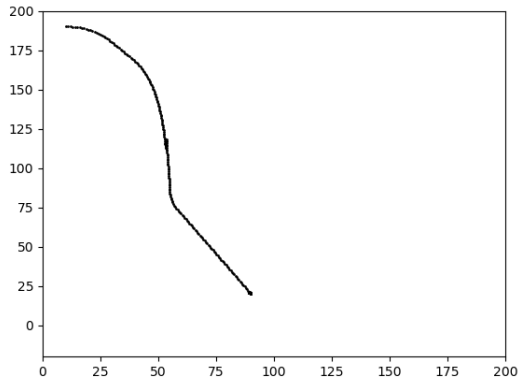
Aditya Nisal
Assignment- Valet

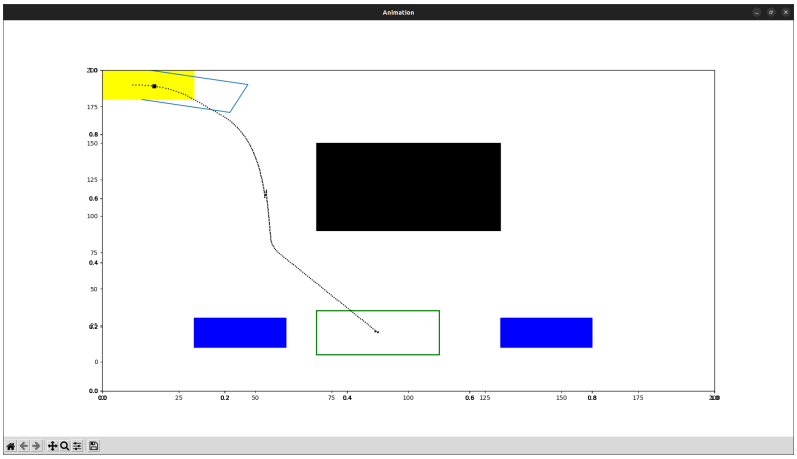
A. *Implementation*

In order to plan the path for a car, the Hybrid A* method is utilized due to the inclusion of the car's kinematic constraints in the system. Unlike regular A*, a heuristic must be defined in the A* algorithm to account for these constraints. In this case, the Euclidean distance between the current position and the desired goal position is used as the heuristic, with the objective of minimizing the sum of cost and heuristic value. The pseudo-code for child node generation in the Hybrid A* method with kinematic constraints for the car is presented below.

The implementation of parallel parking using the normal Hybrid A* algorithm was found to be challenging due to excessive time consumption. One solution to this problem was to use Reeds-Shepps curves, however, the method of dividing the parallel parking path into three intermediate waypoints proved to be even simpler and was employed to find the path between them using the planner. The possible wheel velocities are taken as -1 and 1, while angles are taken in increments of 15 degrees ranging from -30 to 30.

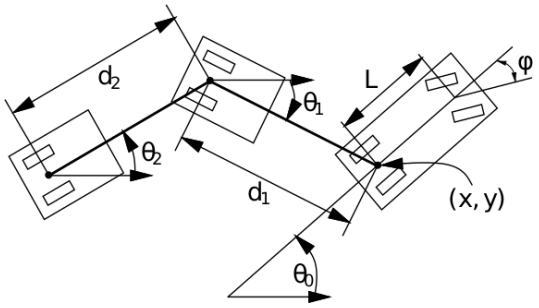
B. *Results*





THE TRUCK

The kinematic constraints for planning the parking of a Truck carrying a trailer at the back include adhering to the standard Ackerman geometry for the truck in the front, along with an added constraint imposed on the heading direction of the trailer at the back. As with the car, the truck and trailer combination cannot make zero radius turns. These constraints are vital for ensuring safe and effective parking maneuvers of the truck and trailer.



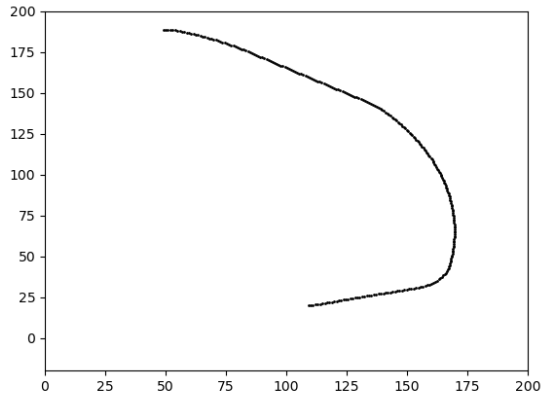
$$\dot{x} = s\cos\theta_0$$
$$\dot{y} = s\sin\theta_0$$
$$\dot{\theta}_0 = \frac{s}{L}\tan\phi$$
$$\dot{\theta}_1 = \frac{s}{d_1}\sin(\theta_0 - \theta_1)$$

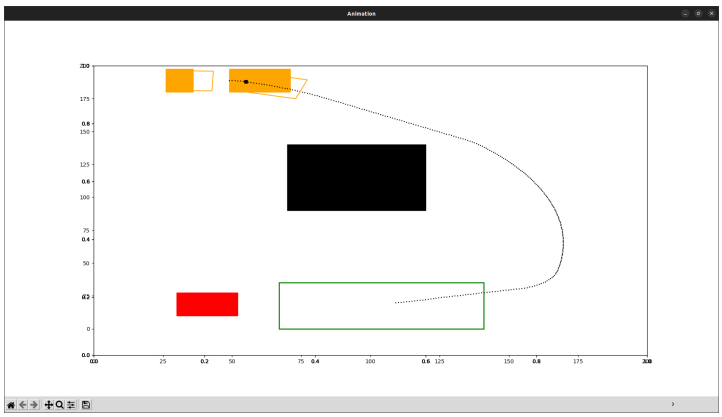
Aditya Nisal
Assignment- Valet

A. *Implementation*

Hybrid A* is employed for path planning of the trailer, as regular A* cannot be utilized due to the inclusion of the robot's kinematics in the system. The algorithm for A* is similar to that of Djikstra's algorithm, with the exception of the addition of a heuristic. In this implementation, the heuristic is calculated as the Euclidean distance between the current position and the desired goal position. The objective is to minimize the sum of cost and heuristic values. Child nodes in the Hybrid A* algorithm are generated while taking into consideration the kinematic constraints for the trailer. The parallel parking feature was challenging to implement using the Hybrid A* algorithm, as it was taking a significant amount of time. In order to reduce the time taken, the parallel parking path was divided into three intermediate waypoints and fed into the planner to find the path between them. Possible wheel velocities are set to -1 and 1, while angles are incremented from -30 to 30.

B. *Results*





PSEUDOCODE:

1. Initialize start and goal states with position, orientation, steering angle and velocity
2. Initialize the start node with 0 cost and heuristic value
3. Initialize the open set with the start node
4. Initialize the closed set as empty
5. While the open set is not empty:
6. Select the node with the minimum f value from the open set
7. If the node is the goal node, return the path to the goal
8. Add the current node to the closed set
9. Generate child nodes for the current node based on kinematic constraints
10. For each child node:
11. If the node is not in the closed set, calculate its cost and heuristic value
12. If the node is not in the open set, add it to the open set
13. Else, if the node's cost is less than the previous cost, update the node's cost and parent
14. End for
15. End while

Aditya Nisal
Assignment- Valet