Name: Camilo Girgado & Jason Patel

Class: RBE 550

Assignment: Valet

**Task**

In this assignment, we were tasked with using kinematic planning to effectively park vehicles of various drivetrains and configurations. We were to create a world environment and implement a path planner to navigate the nonholonomic constraints and obstacles present.

**Approach**

The first vehicle for this assignment incorporated skid steering (diwheel kinematics), the second vehicle incorporated ackerman steering, and the third vehicle incorporated a trailer pulling application. We’ll get into more detail about the scripting for those drivetrains below.

Our environment was built up using pygame and have the details of the environment stored in our environment.py class. We define a list of colors that can be used as tuples, as well as the overall map dimensions in width and height.

Our robots are initialized with their start positions, initial velocities and orientations, as well as their image file in the classes shown below.

**A screen shot of a computer program

Description automatically generated A screen shot of a computer program

Description automatically generated**

- Figure 1: Differential Drive - - Figure 2: Ackerman Drive -

**Ackermann**

In this configuration, the front of our robot has steerable wheels whose pose is shown with the variable psi. Because this robot has steering independent from its drivetrain, it is no able to make turns of small radius like the differential drive robot can.

A screenshot of a computer

Description automatically generated A screenshot of a video game

Description automatically generated

- Figure 3: Planner through iteration 336 - - Figure 4: Approaching the goal (green circle) -

A screenshot of a video player

Description automatically generated

- Figure 5: Final parked state –

**Differential Drive**

In this configuration, we control both rear wheel velocities independently with our vL and vR variables (these velocities are then converted into pixels). Because each wheel is able to be independently driven, the robot can make zero radius turns, a fault that the Ackermann robot needed to overcome above.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

- Figure 6: Planner through iteration 500 - - Figure 7: Approaching the goal (green circle) -

A screenshot of a computer

Description automatically generated

- Figure 8: Final parked state –

**Trailer**

The third configuration compounds on the Ackermann robot with the addition of a trailer. The trailer position is calculated with differential x and y terms and displays its pull angle with the variable theta2. These differential positions are calculated using the theta2 angle.

**Lattice Planner**

Our planner consists of the A\* search algorithm and a looping lattice. The nodes of the lattice are placed into a priority queue, with a cost function that calculates the distance between each state.