



WPI

RBE-550 Motion Planning
Daniel Montrallos Flickinger, PhD

Valet

DUE: 2025-03-24 @ 11:00 UTC

Abstract

A common path planning problem for autonomous vehicles involves maneuvering in tight spaces and cluttered environments, particularly while parking. Create a simulated world and park three vehicles of increasing ridiculousness into a compact space. Planners must take into account vehicle kinematics and collision.

1 Introduction

Use kinematic planning under nonholonomic constraints to efficiently park several vehicles, like in Figure 1. The vehicles range from a skid-steer delivery robot, to a standard car, to a truck with a trailer. Create a simulated world, and implement a kinematic path planner that takes into account the nonholonomic constraints and obstacles.



Figure 1: The parking problem

2 Methods

Refer to Section 13.1.2 in LaValle [2006] for examples of kinematics of wheeled systems, and Section 4.2.1 in LaValle [2006] for modeling configuration space of rigid bodies in two dimensions. A kinematic-only solution is acceptable, though a full simulation utilizing dynamic equations of motion and time-stepping will earn 10% extra credit on this assignment.

The main topic of this assignment concerns how to transition from a discrete planner on a uniform grid to one that works with more constrained kinematics in an environment with continuous states.

There are a few different ways to solve this, such as using a state lattice (Pivtoraiko et al. [2009]), depicted in Figure 2, or Reeds-Shepp paths, (Section 13.1.2 in LaValle [2006]) or path smoothing.

To generate a state lattice, consider a few motion primitives, like straight segments, and curved segments of varying curvature. Generate them linked together similar to a grid to make a discrete graph that can be searched with an algorithm such as A*.

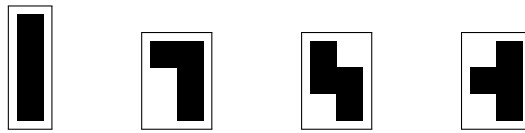


Figure 3: Example free tetromino obstacle shapes (non-standard shapes cost extra).

For a path smoother, consider a high level planner solving on a uniform grid or a visibility graph. Then generate motion primitives to link the path between graph elements with kinematically correct paths. For example, refer to Section 8.4.2 in LaValle [2006], or MATLAB's implementation of Hybrid A*.

Concentrate on creating a kinematically correct path, and animate your vehicles with trajectories with constant/ramped velocities. Full dynamic trajectories are not required unless using a full dynamics-based simulation environment.

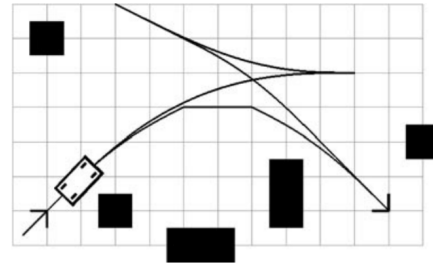


Figure 2: Example state lattice, from Pivtoraiko et al. [2009]

3 Environment

Create kinematic models of cars with different steering mechanisms, then simulate the motion of each of the vehicles within the workspace. Note that this is not a grid-based world, and that each vehicle moves through continuous states with three degrees of freedom (more with a trailer). In each instance, the vehicle starts at the Northwest corner of a bounded two dimensional field. Park the vehicle in a compact space at the Southeast corner, flanked by obstacles on both ends. All wheels remain in contact with the ground, and skidding is only allowed if modeling the full dynamics with contact and friction. Otherwise, nonholonomic constraints are strictly enforced in this environment. Model an oriented rectangular bounding box around each vehicle for collision checking.

3.1 The Parking Lot

As in the previous assignments, create a random obstacle grid utilizing tetrominoes (from Figure 3), as shown in Figure 4. The parking lot grid is 12 x 12, with each cell measuring 3 meters in length, and the obstacle density is 10%. You may adjust the density to tune the simulation outcome. The obstacle placement does not necessarily require that a valid path solution exists, and your planner should handle this situation gracefully.

Place the vehicle in an initial position at the Northwest corner of the environment, while setting the two coincident cells as unobstructed. Also place a parking area in the Southeast corner, as

described in Section 3.2.

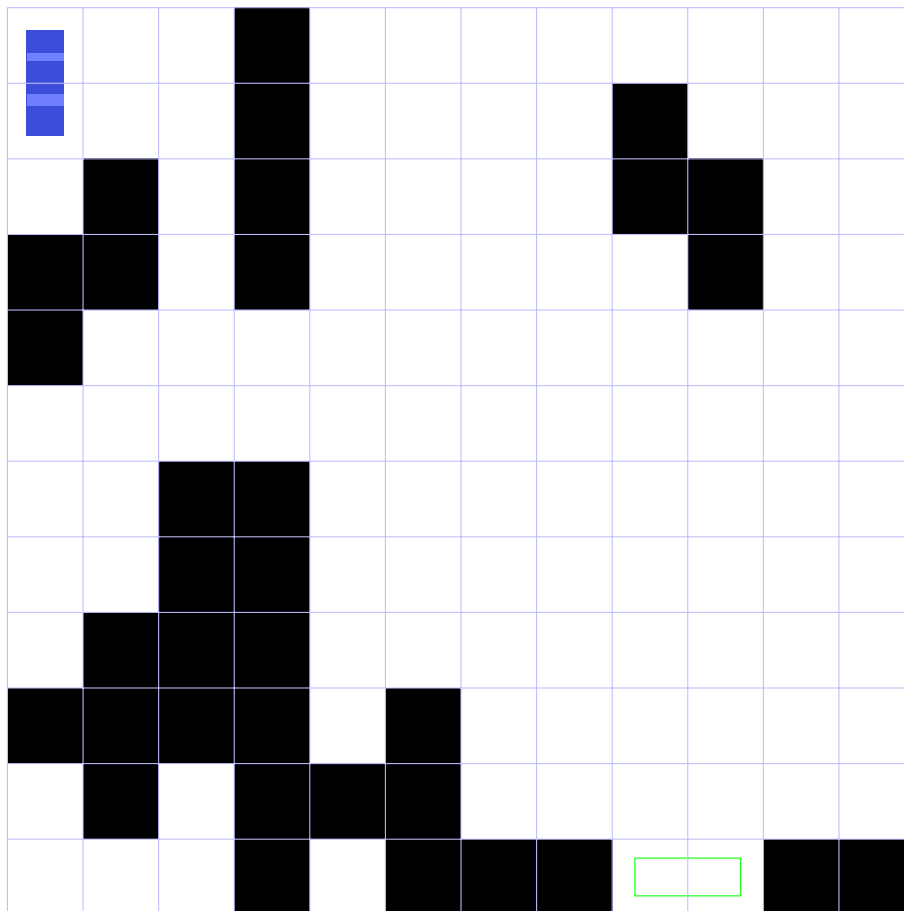


Figure 4: The parking lot

3.2 Parking Spaces

Create a parking area at the Southeast, overwriting any obstacles. Refer to Figure 5a for the exact specifications. The green box denotes the goal pose for each of the vehicles. The exact boundary of this goal box is at your discretion. Use the configuration presented in Figure 5b to park the truck and trailer.

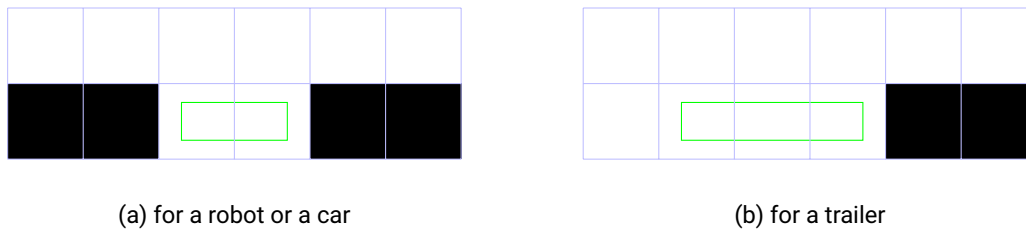


Figure 5: Parking spaces

4 Supporting Software

The most straightforward implementation is to write a basic kinematic simulation system from scratch with Python, MATLAB, or similar. Implement the kinematics, time-stepping, and basic graphical output, and use built-in utilities to generate plots of the path.

Full-featured vehicle simulation systems are generally overkill for this assignment, and might take additional time to set up. However, platforms such as PythonRobotics and Carla (Dosovitskiy et al. [2017]) could be useful.

5 The Vehicles

Our simulation includes three exciting vehicles: the delivery robot, the car, and the truck with a trailer. Simulate each of them individually as they drive to their parking spots.

5.1 The Delivery Robot

First up is the delivery robot, seen in Figure 6. Is it delivering burritos? Is it delivering essential medicine? Who knows. All we know is that it should avoid Philadelphia. It has skid steering, but assume that it has diwheel kinematics for this simulation. Refer to Figure 6b for the kinematics.

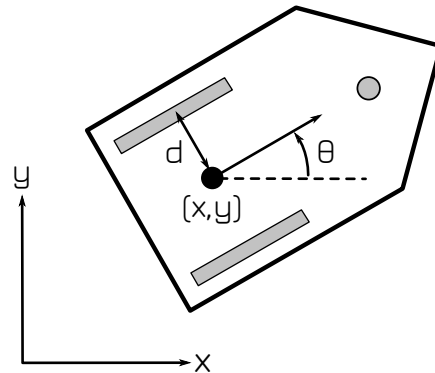
The dimensions for the delivery robot are 0.7 x 0.57 meters.

5.2 The Car

Next, the trusty old police car in Figure 7 gets a try. If it can move under its own power, maybe it won't get scrapped for another month. Standard Ackerman steering affords some maneuverability, but it can't execute zero radius turns. Wheelbase (L) is 2.8 m. Kinematics are shown in Figure 7b. The dimensions are 5.2 x 1.8 meters.



(a) Burrito on board



(b) Differential drive (or skid steer) kinematics

Figure 6: The delivery robot

5.3 The Truck

Finally, it's truck time. The neighbors really want it gone, along with all the beat up old refrigerators and air conditioner condensers littered about the property. It also has Ackerman steering. Dimensions are 5.4 x 2.0 meters, wheelbase is 3.4 meters. The trailer is attached at the rear of the truck, and the distance between the hitch point and axle center of the trailer (d_1) is 5.0 meters. The trailer dimensions are 4.5 x 2.0 meters, with the axle center at the midpoint. The kinematics for a vehicle with multiple trailers is given in Figure 9.

6 Results

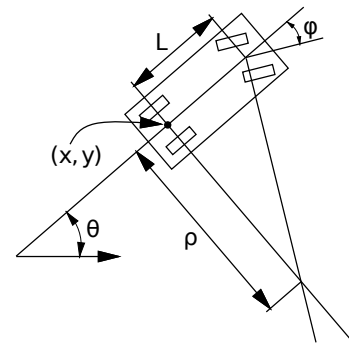
Complete the components discussed in the following sections to receive full credit on this assignment. Refer to Section 7 for submission guidelines, and Section 8 for a summary of the grading rubric. Consider submitting your work as you complete each component, as partial credit is preferred to no credit.

6.1 20% Full Source Code

Submit all source code, organized into functions, classes, associated files, etc. Include references to any 3rd party libraries utilized, and cite any significant contributions from AI-generated code.



(a) Needs to move this month



(b) Ackerman steering kinematics

Figure 7: The car

6.2 20% Planner Algorithms

Submit pseudocode, flow charts, or any other artifacts for each of your planner algorithm implementations. Submit this material individually, but also include it in your report.

6.3 20% Collision Checker

Demonstrate your collision checker implementation with annotated diagrams depicting overlap checks with a vehicle and obstacle region. Submit these diagrams individually, but also include them in your report.

6.4 20% Report

Write brief report detailing your design goals, environment, planner design and implementation, obstacle avoidance, and simulation results. Discuss design decisions, challenges, solutions, and interesting aspects of your project.

6.5 20% Path Animation

For each simulation run, create a short video animating the trajectory from start to finish. Alternatively, you may superimpose periodic snapshots onto an image to depict the movement of the vehicle. Post your results to the discussion forum. Include your path animations, and address the following topics in your post:



Figure 8: The truck

- How was the path generated? (e.g. state lattice, smoothing, visibility graph)
- What is being delivered by the delivery robot?
- What is the most egregious parking failure you've ever seen for a wheeled robot? (Or a human-driven vehicle for that matter.) Include photo or video evidence to back up your claim.
- What are some of the major challenges in creating an accurate path to the parking spot?
- If you had more time, how would you improve your simulation?

7 Submission














This assignment is due 2025-03-24 @ 11:00 UTC. *Late submissions are not accepted.* Upload completed assignment components (as individual files, **not** a single ZIP or TAR file) to the course site on Canvas.

Submit work early, as partial credit for incomplete work is preferred to earning a zero for missing work. Submissions may be amended up until the deadline.

8 Grading Rubric

Refer to Table 1 for the grading rubric.

Table 1: Grading rubric

Weight	Type	Component (See Table 2)
 20%		Full Source Code
 20%	 	Planner Algorithms
 20%	 	Collision Checker
 20%		Report
 20%	 	Path Animation

9 References

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, May 2006. ISBN 9780521862059. URL <http://lavalle.pl/planning/>.

Mihail Pivtoraiko, Ross A Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

10 List of URLs

https://atsushisakai.github.io/PythonRobotics/modules/5_path_planning/hybridastar/hybridastar.html p. 1






https://atsushisakai.github.io/PythonRobotics/modules/5_path_planning/reeds_shepp_path/reeds_shepp_path.html p. 1

https://atsushisakai.github.io/PythonRobotics/modules/5_path_planning/state_lattice_planner/state_lattice_planner.html p. 1

https://atsushisakai.github.io/PythonRobotics/modules/6_path_tracking/path_tracking.html p. 4

<https://canvas.wpi.edu> pp. 7, 8

Table 2: Submission legend

	report or brief writeup
	source code (or link to revision control)
	video presentation, or simulation animation
	data plot or image
	post in the discussion forum on Canvas

<https://carla.org/> p. 4
<https://docs.nav2.org/configuration/packages/smac/configuring-smac-lattice.html>
 p. 1
https://en.wikipedia.org/wiki/Ackermann_steering_geometry pp. 4, 5
<https://www.mathworks.com/help/nav/ref/plannerhybridastar.html> p. 2
<https://www.phillymag.com/news/2015/08/03/philadelphia-killed-hitchbot/> ... p. 4
<https://youtu.be/nKj30ikdzws?si=LTdgabwmhkeRj3wd> p. 5

Last update: February 26, 2025

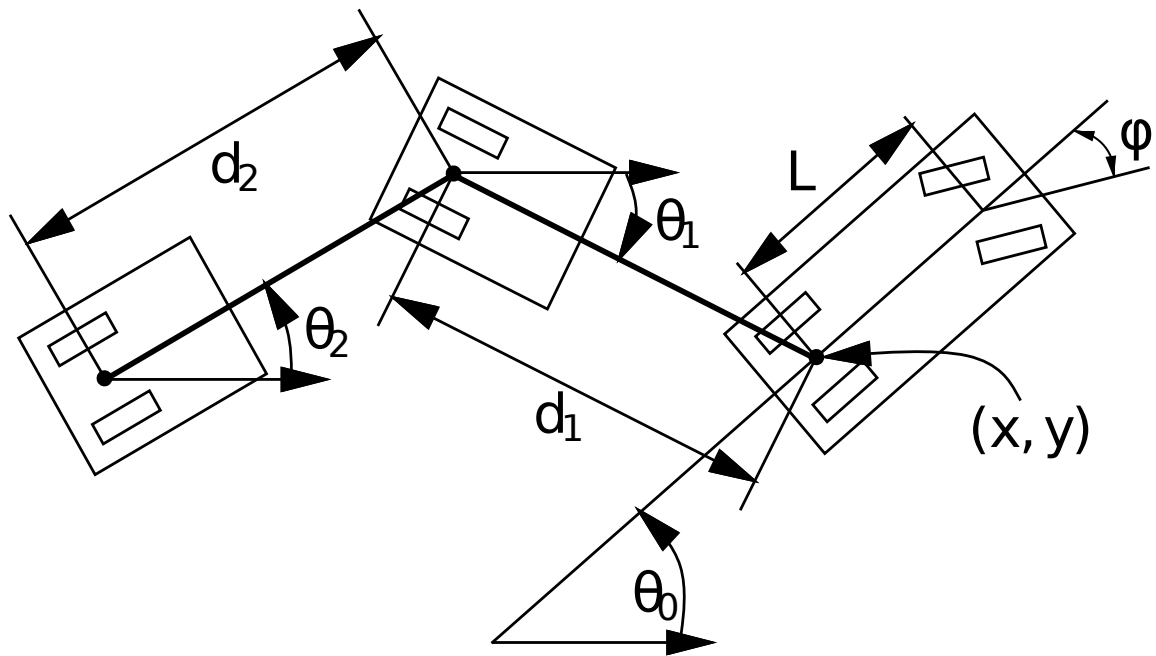


Figure 9: Trailer kinematics