



WPI

RBE-550 Motion Planning
Daniel Montrallos Flickinger, PhD

WILDFIRE

DUE: 2025-04-07 @ 11:00 UTC

Abstract

With two major classes of motion planning algorithms at our disposal, it's time to put them to the test. With both combinatorial and sampling-based planning methods, compare their performance in navigating a firetruck across a deadly obstacle field in an attempt to extinguish as many fires as possible.

1 Introduction



Implement planners utilizing different motion planning algorithms to navigate a firetruck and its adversarial Wumpus through a cluttered, maze-like environment. Use this firefighting simulation to evaluate two types of motion planning algorithms, first a combinatorial search algorithm, specifically A* or a variant, and then a Probabilistic RoadMap planner.

The firetruck must plan paths to the most important fire to fight, as fires spread, and new fires ignite. Simultaneously, the Wumpus navigates the field, setting fires. There is no set algorithm to choose goal points, so use some creativity here. Run the simulation for specified durations, and calculate performance metrics. Use this data to generate plots and a brief report on your findings.

2 Environment

The environment consists of a flat square field, 250 meters on a side, filled with obstacles. The obstacles consist of large patches of un-navigable thick brush, trees, and weeds, suspiciously shaped by the groundskeeper staff to resemble giant tetrominoes. While the environment is not specifically a grid, the base dimension for each obstacle square unit is 5 meters. Inside this field, a Wumpus creeps, and a firetruck drives, attempting to extinguish fires set by the Wumpus. Each player starts on opposite sides of the field (either in random positions, or at set initial points).

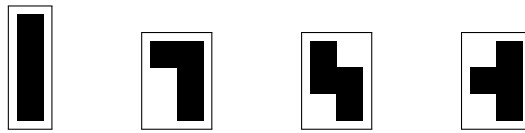


Figure 1: Example free tetromino obstacle shapes (non-standard shapes cost extra).

The arsonist/Wumpus sets a major conflagration instantly at its chosen obstacles. This sets the obstacle state to *burning*. After 10 seconds in this state, the obstacle sets all obstacles within a 30 meter radius to the state of *burning*. Run the simulation for **3600 seconds** – simulation time, not wall clock time. If a steady state is reached, then the simulation may end early.

The truck starts drives around, planning paths to chosen obstacles. If it stops for 5 seconds within 10 meters of a burning obstacle, it sets the state to *extinguished*. Use a path planner to drive the truck to desired locations and attempt to extinguish as many obstacles as possible.

The firetruck cannot perceive the Wumpus, but the Wumpus can perceive the firetruck. But the Wumpus and firetruck do not interact directly. However, as an option for catharsis, if all obstacles are burned, the truck could finally detect the Wumpus and mercilessly hunt it down. Or the Wumpus can burn the firetruck – your choice.

3 Implementation

Use any software to create a simulation of this firefighting world. Python, C++, MATLAB, ROS, or any other systems are acceptable. Third party libraries may be used for any component, including the planner implementations. Be clear in your report and delineate what you wrote versus external sources.

3.1 Obstacles

Create the obstacle field similar to the previous assignments. That is, create obstacles from tetrominoes again (as in Figure 1). Use your magic Tetris generator to fill the environment to 10% obstacle coverage.

The obstacles have the following singular states:

- intact
- burning
- extinguished
- burned

3.2 The Wumpus

The arsonist in this world is a standard issue Wumpus. Aside from its smell, it also has the propensity to light fires in any nearby obstacles. Its motion is confined to a grid, and as such requires a combinatorial motion planning algorithm for movement. Use any variant of A* or similar. (See Chapter 2 of LaValle [2006] for examples.) The grid may be uniform squares, or other variations, such as hexagonal. Any graph density is allowed, it is not necessarily required to match the obstacle grid.

Assume that the Wumpus can light fires on obstacles only in adjacent graph nodes. Once an obstacle state is set to burned, it may not be relit.

3.3 The Firetruck

The firetruck, being a Mercedes Unimog, is a car-like robot, with standard Ackerman steering. See Table 1 for the vehicle parameters. A full dynamic model is optional, but kinematics (including collision) must be considered. See Figure 2 for an example of the kinematics. Assume that the firetruck robot has air support from a small drone, in that an omniscient perception of the environment is provided to the planner. The maximum velocity and minimum turning radius must be enforced, but instantaneous acceleration, while allowed in your implementation, is destructive to the transmission.

Table 1: Truck parameters

width	2.2 meters
length	4.9 meters
wheelbase	3 meters
minimum turning radius	13 meters
maximum velocity	10 meters per second

Use a sampling-based planner for the firetruck. Specifically, implement a Probabilistic RoadMap Planner, as discussed in Chapter 5 in LaValle [2006]. The RoadMap itself may be created a priori before simulation start, or generated as needed. Note that a local planner must be used to generate kinematically correct paths. Pretend that we live in a dystopian alternate reality where RRT was never invented, because we're saving that algorithm for the next assignment.

4 Simulation

Run five simulations of each firefighting scenario, for 3600 seconds simulation time each run. Each iteration shall have a different random obstacle field. Take care with simulation parameters. Your

implementation should run an iteration within a few minutes or less, and does not need to run at real time.

5 Results

Complete the components discussed in the following sections to receive full credit on this assignment. Refer to Section 6 for submission guidelines, and Section 7 for a summary of the grading rubric. Consider submitting your work as you complete each component, as partial credit is preferred to no credit.

5.1 10% Source Code

Submit full source code, include citations of any third party libraries or external tools (e.g. LLM code generators) used.

5.2 20% Discrete Planner

Implement a discrete, grid-based planner for the Wumpus, and submit your source code (or a link) with your assignment. Also include a flow diagram or pseudocode with your report.

5.3 20% Sampling-based Planner

Implement a sampling-based planner, utilizing a probabilistic roadmap or similar algorithm, and submit your source code (or a link) with your assignment. Also include a flow diagram or pseudocode with your report.

5.4 10% Simulation Results

In each simulation run, generate a score for each player (Wumpus, firetruck, and any bystanders). For each obstacle ignited, award 1 point to the Wumpus, then an additional point if an obstacle is burned. For each obstacle extinguished, award the firetruck with 2 points. Create a table with the final results for each round, and award a fancy trophy to the champion who wins three out of five runs.

5.5 10% computational resources figure

Instrument each planner implementation to measure the execution time. (This includes roadmap generation). Create a bar chart comparing the CPU time required for each player (Wumpus and firetruck), summed over all of the iterations.

5.6 20% brief report

Discuss the goal planning methods for both players, with details on if there is any reaction, avoidance, pre-planning, or other features.

Create an appendix listing all third party software libraries utilized in the project.

5.7 10% Simulation Animation

Additionally, pick a particularly interesting or representative excerpt for each player, and create a brief animation (edited for clarity and length, and sped up where required). Write a brief discussion of your results, including the scores and computation resource plots, and post to the discussion forum on Canvas.

Which method is better suited? Does one method plan more efficient paths? Are there advantages to having a roadmap computed a priori? Which method requires more computational resources? And which method provides the best performance accounting for computational resources? (e.g. considering total score over computation required.)

6 Submission

This assignment is due 2025-04-07 @ 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.

7 Grading Rubric

Refer to Table 2 for the grading rubric.

References

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

Last update: March 15, 2025

Table 2: Grading rubric









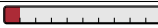



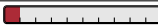






Weight	Type	Component (See Table 3)
 10%		Source Code
 20%		Discrete Planner
 20%		Sampling-based Planner
 10%		Simulation Results
 10%		computational resources figure
 20%		brief report
 10%		Simulation Animation

Table 3: 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

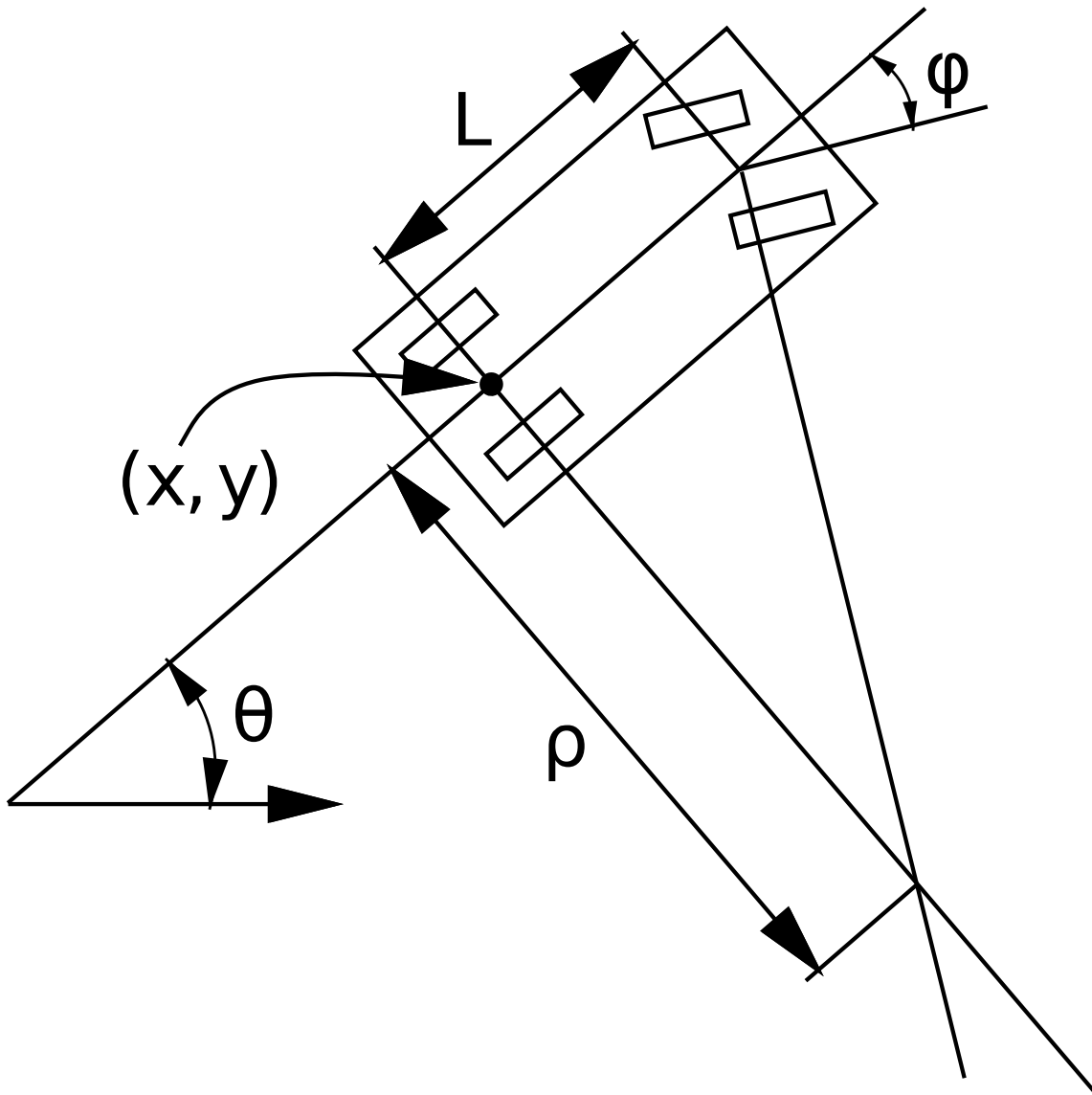


Figure 2: Trailer kinematics