



# WPI

RBE-550 Motion Planning  
Daniel Montrallos Flickinger, PhD

# Wildfire

DUE: 2023-04-03 @ 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, and discuss it in your report. The simulation is run for a specified time, after which performance metrics are calculated. 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 like 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).

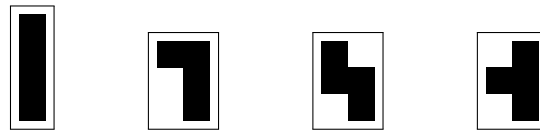


Figure 1: Example free tetromino obstacle shapes.

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 grid world assignment. That is, create obstacles from tetrominoes again (as in Figure 1). Use your magic Tetris generator to fill the environment to 10% obstacle coverage. Obstacles don't necessarily need to be orthogonal to the world coordinate frame, but it might be easier.

#### 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. Extinguished obstacles can be relit.

### 3.3 The Firetruck

The firetruck, being a Mercedes Unimog, is a car-like robot, with standard Ackerman steering (Explained). 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 is allowed.

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 should be created a priori before simulation start. Note that a local planner must be used to generate kinematically correct paths. Pretend that you have never heard of RRT, because we're saving that for the next assignment.

## 4 Results

Run five 3600 second iterations of each firefighting scenario. Each iteration should have a different random obstacle field. Take care with simulation parameters. This should run an iteration within a few minutes or less, and does not need to run at real time.

At the end of each scenario, sum up the number of obstacles in each state (*intact*, *burning*, *extinguished*). Calculate three ratios:  $\frac{N_{intact}}{N_{total}}$ ,  $\frac{N_{burned}}{N_{total}}$ , and  $\frac{N_{extinguished}}{N_{total}}$ . Where  $N_{burned}$  is the total number of obstacles burned completely. Average these metrics over all iterations, and create a bar chart showing the results. (The performance 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.


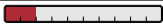





Additionally, pick a particularly interesting or representative excerpt for each player, and create a brief animation (sped up to 10x realtime if required). Write a brief report discussing the results of this experiment. Which method is better suited? Does one method plan more efficient paths?

Are there advantages to having a roadmap computed a priori? Which method requires more computational resources? Discuss the goal planning methods for both players, with details on if there is any reaction, avoidance, pre-planning, or other features.

Submit full source code (not including third party code) as an appendix to your report.

## 5 Grading and Submission

This assignment is due 2023-04-03 @ 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.

Weight	Type	Component
 20%	source	combinatorial planner implementation
 20%	source	sampling-based planner implementation
 10%	PDF	performance figure
 10%	PDF	computational resources figure
 20%	PDF	brief report
 10%	video	combinatorial planner animation
 10%	video	sampling-based planner animation

## 6 References

Engineering Explained. Ackerman steering - explained. URL <https://www.youtube.com/watch?v=oYMMdjbmqXc>.

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

Last update: January 3, 2023

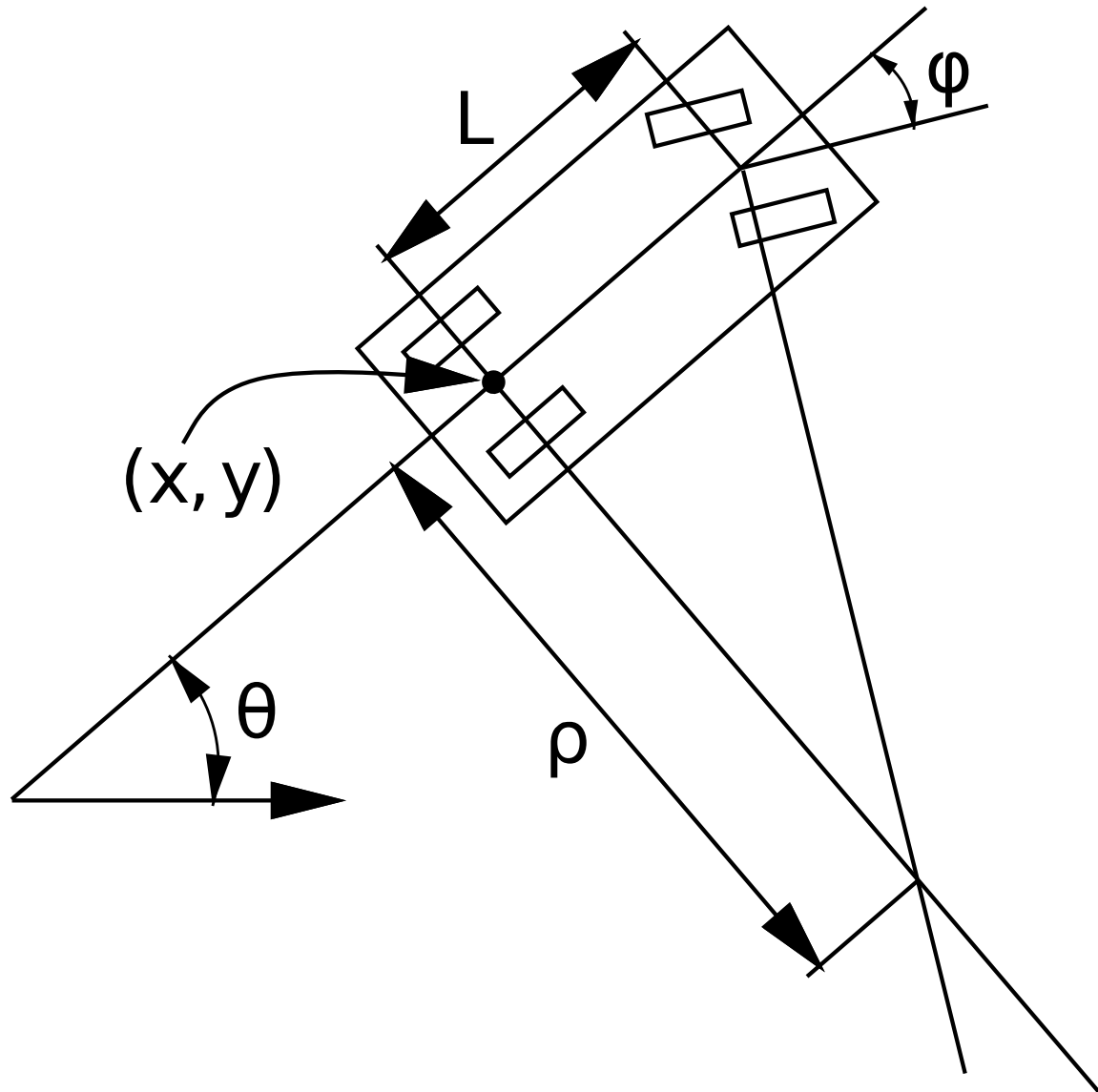


Figure 2: Trailer kinematics