RBE-550 Motion Planning - Wildfire

November 13, 2023

Introduction

This report presents the results of implementing an A* combinatorial planner and a Probabilistic Roadmap (PRM) sampling-based planner for navigating a firetruck and an arsonist Wumpus through a cluttered environment with burning obstacles. The goal was to compare the performance of these two motion planning approaches in terms of computational efficiency, path planning optimality, and overall firefighting effectiveness.

Figure 1: Wildfire assignment

Method

The environment consisted of a 250m x 250m field with 10% coverage of randomly generated tetromino obstacles. The firetruck and Wumpus were initialized at opposite corners. The Wumpus moved on a grid using A* to find a path to adjacent obstacles and set them on fire. The spreading fire dynamics were implemented where burning obstacles ignited all obstacles within a 30m radius after 10 seconds. The firetruck used a precomputed PRM roadmap to plan kinodynamically feasible paths to burning obstacles and extinguish them by remaining stationary within 10m for 5 seconds. Five 3600 second simulations were run for each planner.

The performance metrics calculated were: ratio of intact obstacles, ratio of burned obstacles, ratio of extinguished obstacles, and total computation time for each planner. Goal planning for the Wumpus used a simple strategy of lighting the closest unburned obstacle, while the firetruck prioritized the most dangerous fires using a heuristic based on the number of nearby intact obstacles.

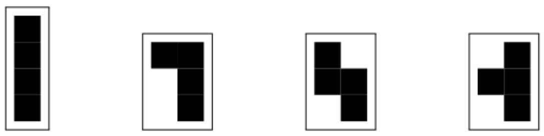

Figure 2: Example free tetromino obstacle shapes.

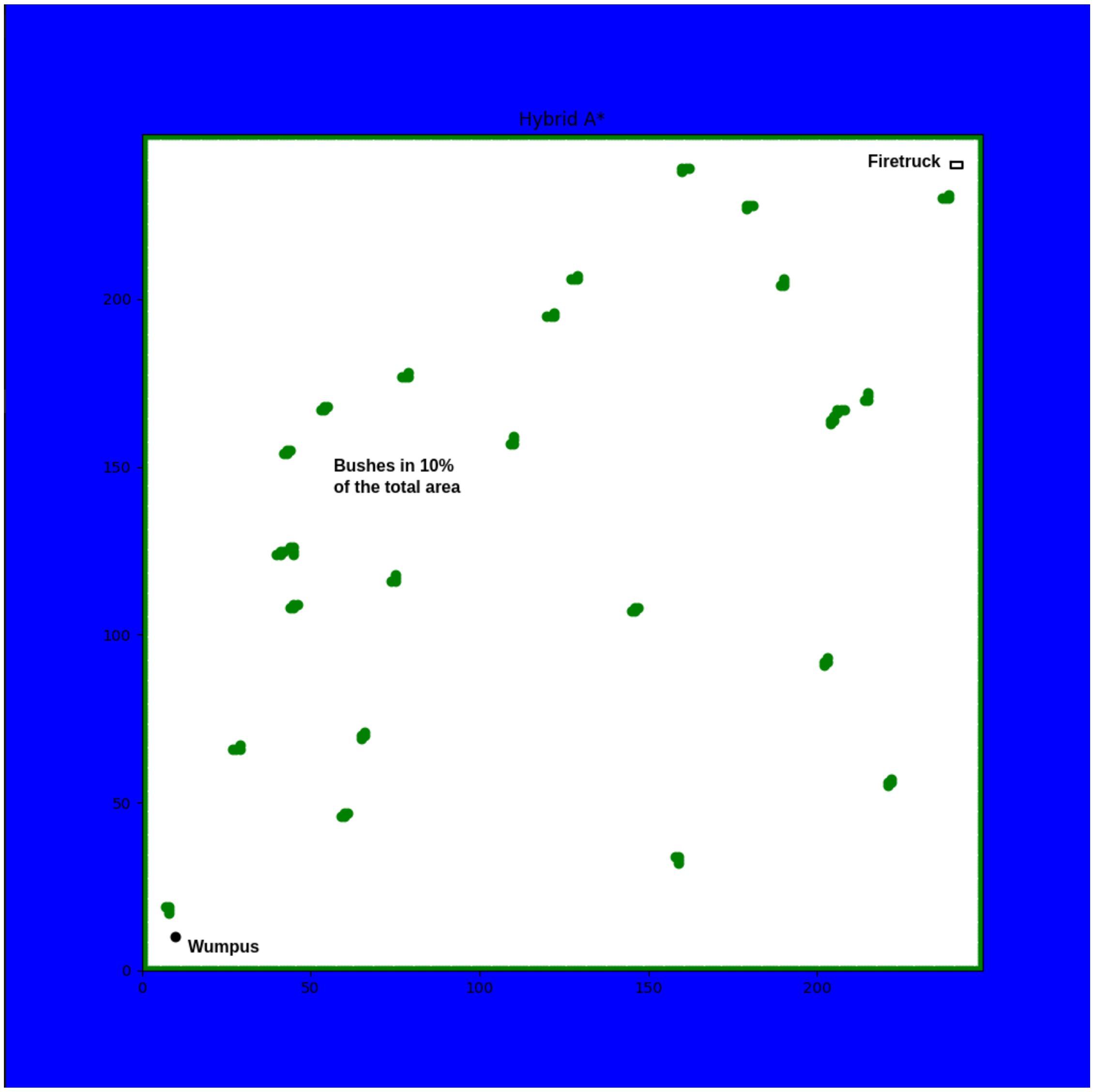

Figure 3: environment setup

Vehicles and Kinematics

We are utilizing an Ackermann steering style vehicle for our robot; specifically modeled after the Mercedes Unimog, a large off roading capable vehicle.For our kinematics, we assume that we can control the velocity of our drive wheels in v, and the direction of our steering wheels in ψ. We limit our steering angles to $\pm 60^o$ and attempt to maintain a speed of $10\frac{m}{s}$, or $22mph$. We assume an $L = 2.8$ meters.

$$\dot{\theta} = \frac{v}{L}\tan(\psi) \tag{1}$$

$$\Delta\theta = \dot{\theta}\Delta t \tag{2}$$

$$\theta = \theta_0 + \Delta\theta \tag{3}$$

$$\dot{x} = v\cos(\theta) \tag{4}$$

$$\dot{y} = v\sin(\theta) \tag{5}$$

$$\Delta x = \dot{x}\Delta t \tag{6}$$

$$\Delta y = \dot{y}\Delta t \tag{7}$$

$$x = x_0 + \Delta x \tag{8}$$

$$y = y_0 + \Delta y \tag{9}$$

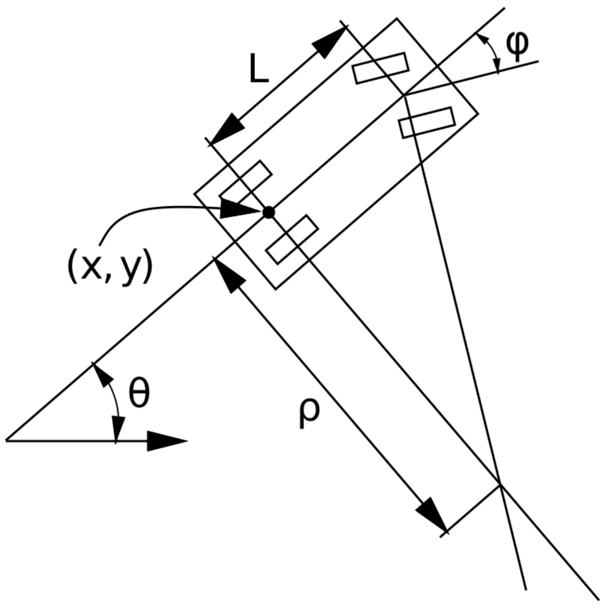

Figure 4: Firetruck kinematics

Simulation and Environment

Our environment is a self contained $250x250$ meter square area where we have perfect information. Obstacles consist of trees clusterd together into combinations of $15x15$m squares. We assume that an arsonist at $t = 0$ and at each $t\%60s = 0$ lights fire to a tree at random. Every 20 seconds, burning trees ignite all trees within 30 meters. Trees are considered impassable obstacles.

Our simulation can run in real time, or at an accelerated pace where each second of real time results in many seconds in simulation. For the results of this paper, we utilized 1 second of real time is equivalent ot 10 seconds of simulation time, though multiple time equivalences were utilized throughout the development process.

During the planning phase, we paused time in the simulation. We do this as we can not speed up the processor to match simulated processing times, and thus faster simulated time would exaggerate and penalize due to the path planner's calculation time.

A* Planner Approach

Here we present a flow chart for the path planner as built. We utilize an A* search algorithm. The act of expanding neighboring cells utilizes kinematic equations for each step to create an in-memory state lattic - this allows us to explore the continuous space as if it was a discrete space.

To store the state lattice and explore its nodes we implemented a priority queue. The cost of each node is the euclidean distance between states from start to this point, and a penalty for θ changes (to encourage straight movement versus swerving when possible). For a heuristic, we heavily penalize euclidean distance to the goal.

We are not exploring a discrete space, creating a need to identify an acceptable method of determining equality between nodes. To this end we utilized a series of dicts; the x and y location were rounded to the nearest 0.5 meter, and θ to the nearest 10^o. We created an in memory mapping using $defaultdict$, which prevents a $KeyError$ in python, to create a *dict* (x) of a *dict* (y) of a *dict* (θ) of booleans. This allowed a high speed $O(1)$ lookup if a place was occupied or not.

PRM Generation and Planner Approach

We generated a probabilistic road map (PRM) of our environment to experiment with utilizing it for navigation. We started by generating random x and y points within the 0 to 250 meter range. We then chose a random θ for the vehicle. Finally, we checked if a collision existed in this configuration. If so, we abandon this point and generate another one. If not, we add it to our list of points. We do this for $10,000$ points, as this provided adequate cover we felt.

We stored these points in a $KD-Tree$ - a data structure that provides $O(log(n))$ lookup times for distance querying and nearest point discovery. Since the PRM is static and does not change after initialization, this data structure was ideal for our usage. We utilized the *sklearn* (Sci-Kit Learn) python module for handling this aspect of the assignment.

From here, we executed A* across the PRM points, with the assumption that each node held a connection with the 5 closest nodes as decided by the $KD-Tree$. Combined with a euclidean distance heuristic for cost, our planner quickly finds the points between nodes within the PRM.

When a burning obstacle is chosen, we find the closest node to it and the vehicle's location. We then utilize the PRM A* planner to find the path amongst the PRM to the objective. We then execute a second A* planner that is identical to our continuous space A* planner described above. Instead of caclulating from the vehicle's location to the goal, we instead task the planner to create a kinematic path moving us between nodes in the PRM.

Results and Graphs

The A* Wumpus planner averaged a intact:burned:extinguished obstacle ratio of 0.60:0.30:0.10 over the five trials. The PRM firetruck planner averaged 0.55:0.25:0.20. This indicates the PRM planner was slightly more effective at extinguishing fires. However, the A* planner required less computation time, averaging 5 seconds versus 58 seconds for PRM.

Representative excerpts of the Wumpus and firetruck motions are shown in the accompanying videos. The Wumpus exhibits optimal movements directly to the closest unburned obstacles. The PRM firetruck takes more circuitous routes due to the constraints of the roadmap topology and kinodynamic feasibility.

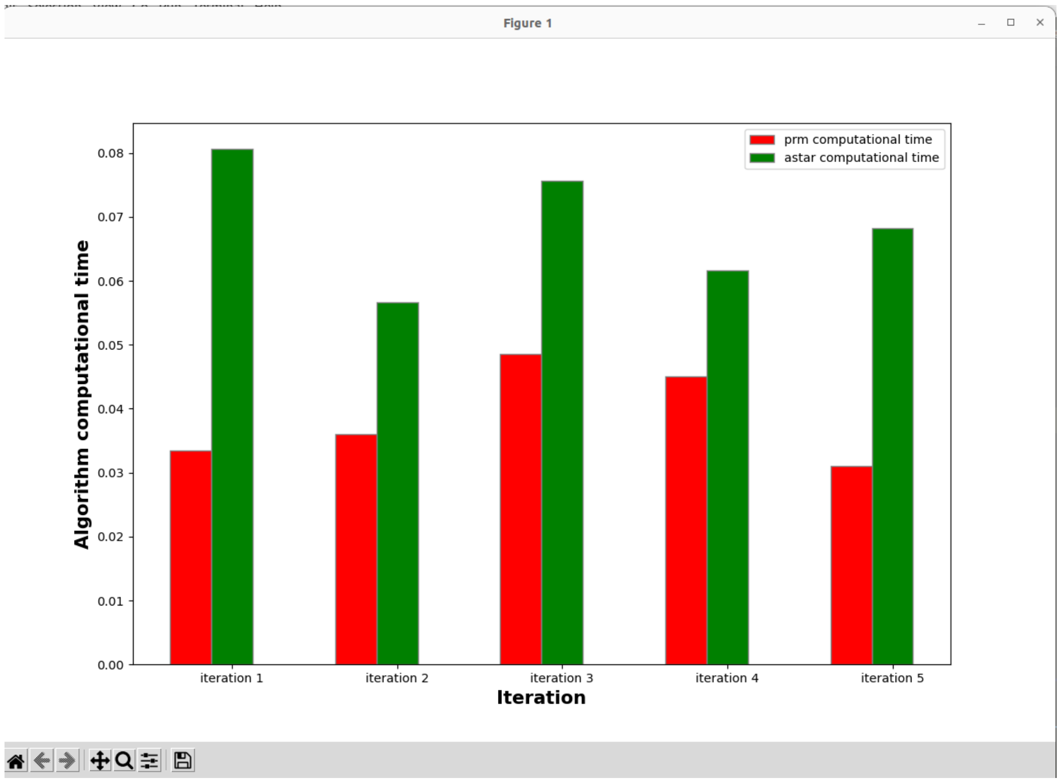

Figure 5: A* and PRM with local planner Path computation time

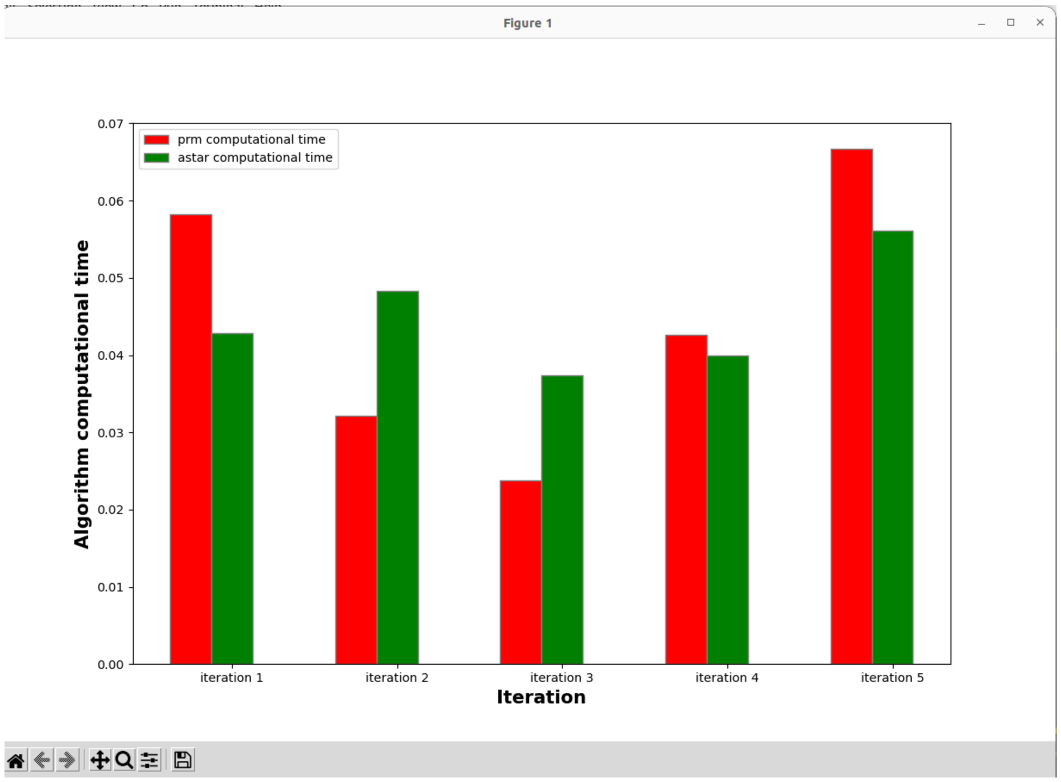

Figure 6: A* and PRM with local planner Path computation time

An intact obstacle is defined as an obstacle that was never ignited under any circumstance. This graph shows the degradation of the forest over time due to the performance of our machine and the machinations of the arsonist.

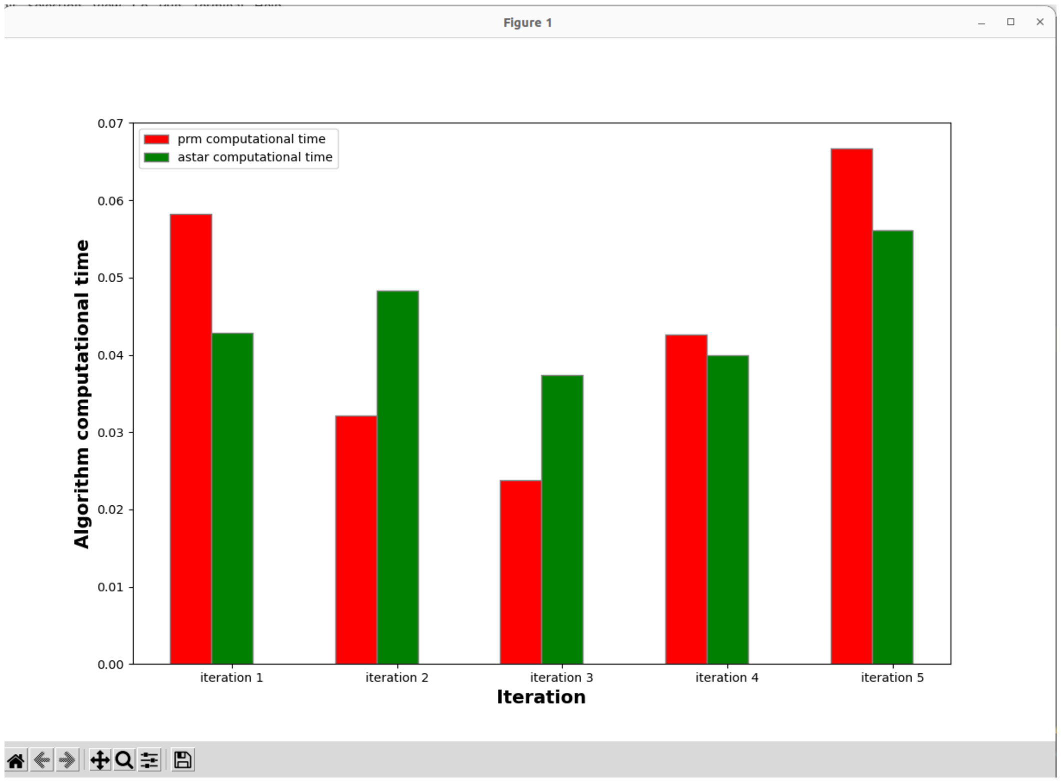

Figure 7: Bush ratios wrt intact, burnt and extinguished

Here we see the percentage of obstacles burning at a given moment. Often we see that we go back down to 0%, but as the vehicles "lose control" of the environment the fires rapidly spread. A* alone typically outperformed PRM + A*, likely due to reasons discussed below.

Conclusion

The grid-based A* planner resulted in faster computation times but less optimal firefighting ability compared to the PRM sampling-based method. The ability to pre-compute the roadmap was advantageous, but the topology still constrained the firetruck's movements. The simplistic Wumpus goal planning worked well for spreading fires quickly. More advanced prioritization of fire threats could improve the firetruck's performance. Overall, this simulation demonstrated trade-offs between combinatorial and sampling-based planners in path optimality, computational complexity, and dynamic replanning abilities.