Ryan Spring - rds4
Jayvee Abella – jra8
10/28/2014 COMP 450 - Project 4

Problem Statement

The objectives of this project were to create the RG-RRT planner, to implement the collision detection and dynamics for the pendulum and vehicle systems, and to compare the RG-RRT planner to the RRT and KPIECE1 planners. The RG-RRT planner is similar to the RRT planner except q_{rand}, a random sample from the configuration space is selected when there exists a nearest neighbor state, q_{near} and q_r, a state in qnear's reachability set where the distance between q_r and q_{rand} is less than the distance between q_{near} and q_{rand}.

Environment Description

For the pendulum environment, the space is an open environment with no obstacles. The state space for the environment is SO(2) X R, which represents the orientation and angular velocity of the pendulum. The angular velocity of the pendulum is limited between (-2*PI, 2*PI).

For the vehicle environment, the world space is a bounded box from (-1, -1) to (1, 1). The start state is in the lower-left corner of the bounded box and the goal state is in the upper-right corner of the bounded box. The environment contains two large blocks that create a narrow passage that the robot must navigate in order to move from the start to the goal state. The state space for the environment is R^2 X SO(2) X R, which represents the vehicle's position, orientation, and forward velocity. The velocity of the vehicle is limited between (-1, 1).

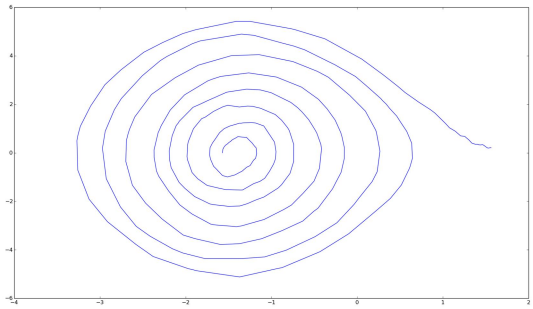

Figure 1 - A point rotating the bottom start position to the top goal position in the pendulum environment
X-Axis: Angle Orientation / Y-Axis: Angular Velocity

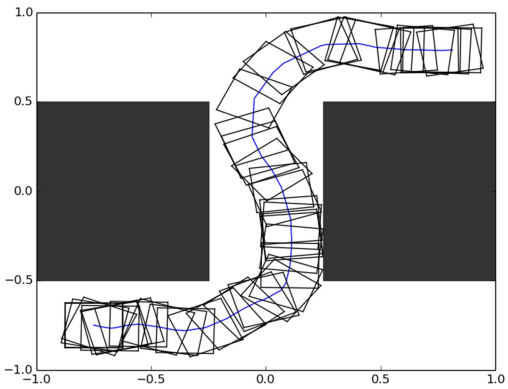

Figure 2 - A rectangle robot navigating through a narrow passage between two obstacles in the vehicle environment

Ryan Spring - rds4
Jayvee Abella – jra8
10/28/2014 COMP 450 - Project 4

Robot Description

There are two robot configurations for this project. For the pendulum scenario, the robot is a point with an R control space. The control space represents the torque applied to the pendulum. The torque is limited between (0, PI). For the vehicle scenario, the robot is a 0.25 square with an R^2 control space. The control space is the orientation of the vehicle's steering wheel and the acceleration of the vehicle. The turning angle limits for the car is (-0.5*PI, 0.5*PI). The acceleration of the vehicle is limited to a range from -1 to 1.

Benchmark Results

We benchmarked the planners based on 20 independent runs with a memory limit of 1000 and max runtime of 20 seconds. Figures 3-8 were generated with 10 controls and a step size of We assessed the performance of these planners based on solution length, time, and memory. All three planners were able to find solutions within the max runtime.

Pendulum

For the pendulum environment, the RRT based planners performed better than KPIECE1. For essentially the same performance in solution length and time, KPIECE1 took at least twice the amount of memory. Between RRT and RGRRT, RGRRT did take less amounts of memory than RRT but the heuristic in the RGRRT algorithm did not benefit much from this simple environment in terms of solution length and time.

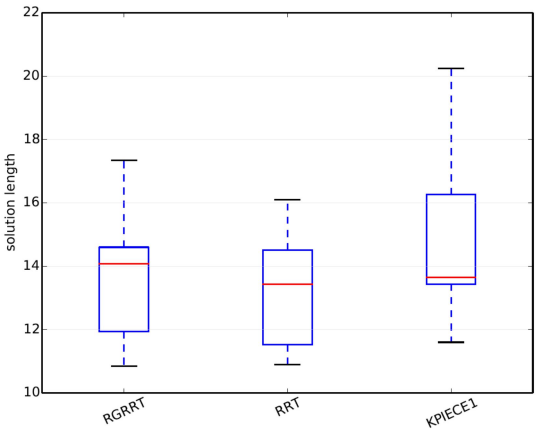

Figure 3 Pendulum Scenario - Solution Length

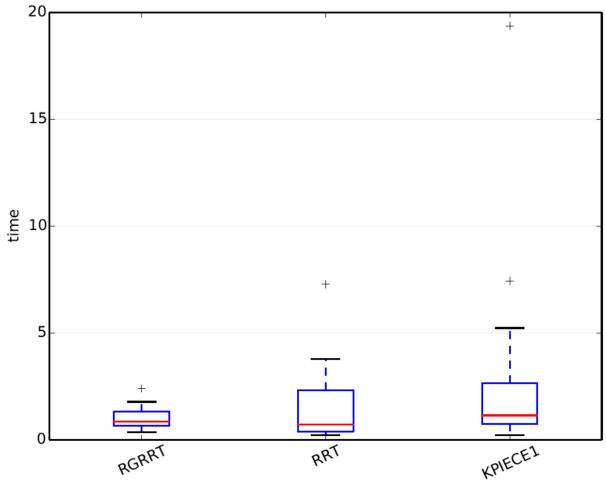

Figure 4 Pendulum Scenario – Time

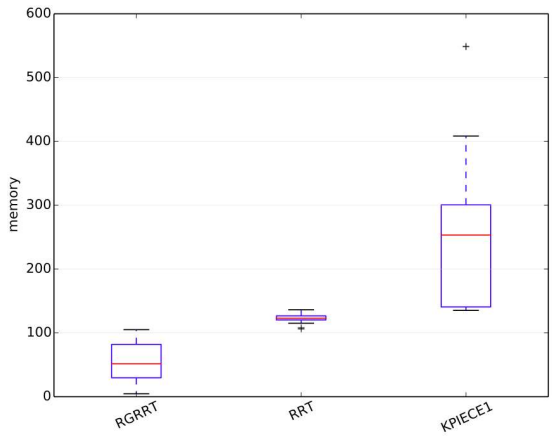

Figure 5 Pendulum Scenario – Memory

Car

The car environment presented these planners with a different challenge in producing a path between a passage given the constraints. KPIECE1 produced the shortest solutions but at the cost of spending more time and memory than the other planners. The performance between RRT and RGRRT was similar to the pendulum environment in that RGRRT took less amount of memory but both planners created similar solution lengths in similar times.

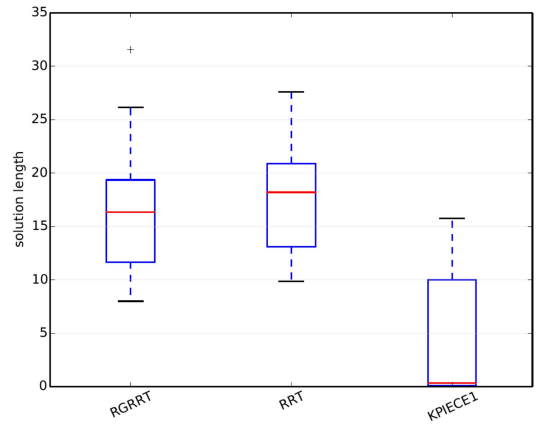

Figure 6 Car Scenario - Solution Length

Ryan Spring - rds4
Jayvee Abella – jra8
10/28/2014 COMP 450 - Project 4

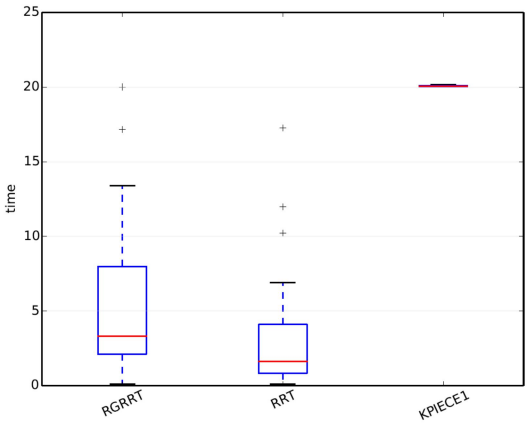

Figure 7 Car Scenario – Time

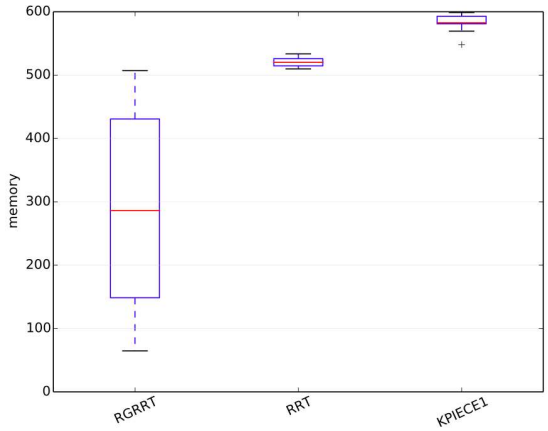

Figure 6 Car Scenario – Memory

RG-RRT / RRT Comparison

As the number of controls increases, the size of R(q) in the RG-RRT planner increases. In the original experiments, we divided the torque control space into 10 controls (see figure 7). We increased the number of controls to 15, causing the computation time for the RG-RRT planner to increase (see figure 9). We assume the length of the time period to be the number of time steps to propagate the controls in the reachable set R(q). The original time period was the minimum value of 1 (see figure 7). We increased the time period to the maximum value of 10 (see figure 10). This time period increase caused the RG-RRT performance to decrease in terms of computational time. The RG-RRT planner was slower than the RRT planner in every case by a few seconds. The average computational time for increasing the number of controls and the time period was about 6 seconds. With the original settings, the average computational time was about 3 seconds.

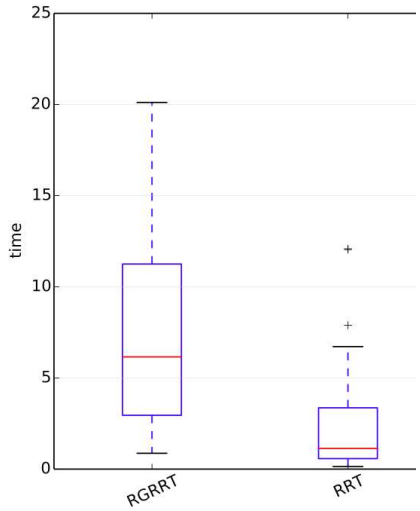

Figure 9 Car Scenario – Time (controls = 15)

Ryan Spring - rds4
Jayvee Abella – jra8
10/28/2014 COMP 450 - Project 4

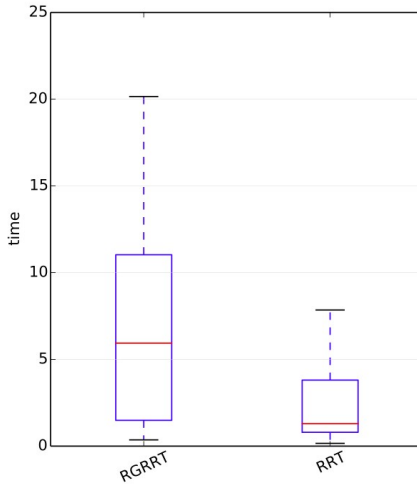

Figure 10 Car Scenario – Time (Max propagation time – 10)

Conclusion

An interesting ambiguity is how to add q_{new} to the tree using the q_{rand}, q_{near}, and q_r samples. One choice is to add the q_r sample because the state moves the tree towards q_{rand} and time was spent creating R(q) and then finding the q_r sample. The other choice is the default RRT procedure, which finds a control that will attempt to reach q_{rand} state from the q_{nea} state and creates the q_{new} state. The interesting part is the former choice significantly decreased the amount of time and states required to solve the pendulum scenario but caused the car scenario to always fail. The latter choice solved both scenario but caused a performance penalty to the pendulum scenario. I suspect that the dynamics of the pendulum scenario creates a precise pattern for the pendulum swings, as noted by the intricate spiral in Figure 1. In this case, the q_{new} sample moves the tree closer to q_{rand}, which is not necessarily closer to the goal. In the car scenario, there is more free space for car to move around and less restrictive dynamics, which means that using the q_{new} sample will cover a larger distance and move the tree closer to the goal.

In addition, we established a limit on selecting the q_{rand} sample to 100 attempts. This limit prevented the planner from wasting large amounts of time, trying to find a random sample that met the criteria. We added the limit because the while loop was taking too much time and the 100 try limit worked well for the planner.

Difficulty / Time

Ryan Spring: 5 out of 10 – 20 hours
Jayvee Abella: 4 out of 10 – 8 hours

We didn't have too much trouble creating the environments and understanding the RG-RRT planner. Most of the time was spent tweaking the RRT planner's propagation step size and debugging pointer issues in the RG-RRT planner.