COMP550 - Project 4 Progress Report

Anja Conev - ac121
Dejan Grubisic - dx4

November 7, 2019

1. **A succinct statement of the problem that we have solved:** We have solved the problem of motion planning for two non-holonomic dynamic systems - pendulum system and car-like system, both described by ordinary differential equations dependent on the input controls and current states.
2. **Images of our environments and a description of the start-goal queries we were evaluating.** The pendulum system does not have any obstacles - its environment is 'empty' the only constraints come from the definition of differentials based on the control and previous space. Start state of the pendulum is $\theta = -\frac{\pi}{2}, \omega = 0$, the goal state is $\theta = \frac{\pi}{2}, \omega = 0$. The chosen environment for the car-like system is shown on the figures, the start state is $(x, y) = (-5, -5), \theta = 0, v = 0$, the goal state is $(x, y) = (5, 5), \theta = 1, v = 0$.

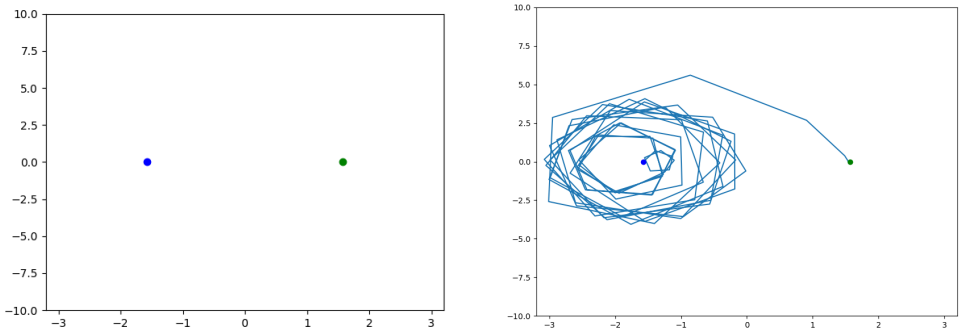


Figure 1: Environment of the pendulum robot along with the solution path generated by RRT planner

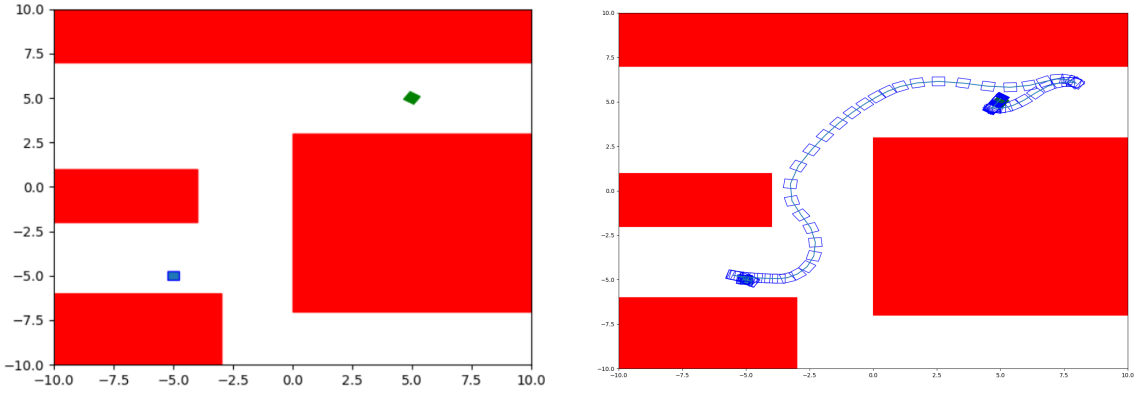


Figure 2: Environment of the car robot along with the solution path generated by RRT planner

3. **A short description of the robots (their geometry) and configuration spaces:**

   (a) **Torque controlled pendulum** - the state of this robot is described by two parameters θ - the angle of the torque relative to a right leaning horizontal position and ω the angular velocity. This makes it a $SO^2 \times R^1$ state space. There is also one control input given to this system - torque τ.

   (b) **Car-like vehicle** - the state of this robot is described by its 2D position - x and y parameters, heading θ and forward velocity v. This makes it a $R^2 \times SO^2 \times R^1$ state space. Input controls are angular velocity ω and forward acceleration $\dot{v}$.

4. **Explain the differences in solution paths when torque is limited to 3, 5, and 10 for the pendulum problem.** The biggest difference between the paths generated by different limits of torque visually is the number of "circles" generated before reaching the goal state. With lower torque limit more oscillations are made. This is the consequence of the fact that with higher available torque we have more force in the control of the pendulum and we can get it into the goal state "faster".

5. **A summary of benchmarking data for the RRT, KPIECE and RG-RRT planners for both systems. Use a torque value of 3 for the pendulum for the other benchmark and comparison exercises. This data must be presented in a quantitative manner (i.e., plots or tables) and any conclusions must be supported from this data. Consider the following metrics: computation time, path length, number of tree nodes, and success rate. When referencing benchmarking results you must include the referenced data as figures.**

   Comparison between RRT, KPIECE1 and RG-RRT for Pendulum and Car is based on benchmark that runs 50 runs with time limit of 20s per planning for parameters *execution time, solution length, number of states* and *number of correct solutions*.

   Computation time:
   For the pendulum problem the RRT planner is faster than RGRRT and KPIECE as shown on Figure 3a. RGRRT is slower than RRT because it keeps track of the reachable set and that is an expensive operation since it involves doing multiple steps of integration for each new node we add. KPIECE is also slower than RRT, its speed is mainly influenced by the efficiency of the chosen projection which determines how efficiently it explores the space.
   For the car problem all three planners only come to the approximate solutions in the given timeframe of 20s, so they all have on average the same computational time.

   Path length:
   For pendulum RRT and RG-RRT generate similar path lengths on average while KPIECE generates the longest paths (Figure 3b). Again, the performance of KPIECE is mainly influenced by the choice of the projection function, and here we could agree that our choice of a projection did not improve performance of KPIECE compared to RRT.
   For the car problem KPIECE on average finds the solution with the smallest length, this indicates that we have a good projection function (Figure 4b).

   Number of tree nodes:
   As expected, the smallest number of nodes for both the car and the pendulum problem is generated by RG-RRT (Figure 3c, Figure 4c). This is the main advantage of RG-RRT planner and it is achieved by keeping the reachable set for every node in the graph, which prevents it from extending the nodes in wrong directions.

Success rate: Figure 3d and Figure 4d clearly show that all planners managed to find correct approximate solutions within a given time limit.

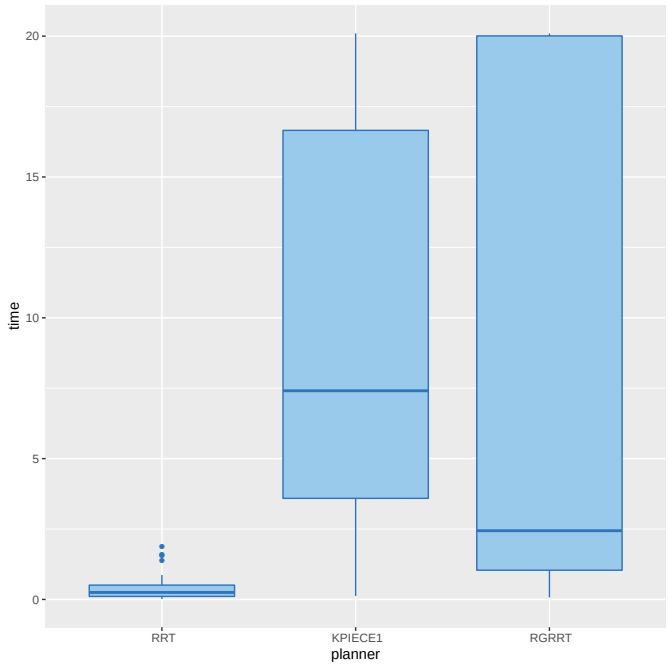
6. **A head-to-head comparison of RRT and RG-RRT. Discuss the performance trade-offs of RG - RRT for computing reachable states in terms of the length of the time period and the number of controls. Support your claims with images and/or benchmark data.**

   The main benefit of RG-RRT is in the 'book-keeping' of reachable nodes. This helps RG-RRT not grow the tree in 'wrong' directions. On one hand this leads to less states being explored. On the other hand, RG-RRT requires much more space than RRT. Another thing that is worth noting is that for each new node added to RG-RRT its reachability set needs to be computed - this means more integrations, which in theory means longer execution time. The fact that the RG-RRT is guided well sometimes actually pays off for the number of integrations per new node needed. For the systems with more controls RRT 'more often' ends up straying away from the goal - which means that the difference in speed between RG-RRT and RRT is even greater for systems with more controls. For example, for systems with zero controls RG-RRT does not make any sense and does not really bring any benefits, since the reachability of states is no longer constrained by the dynamics of the system.

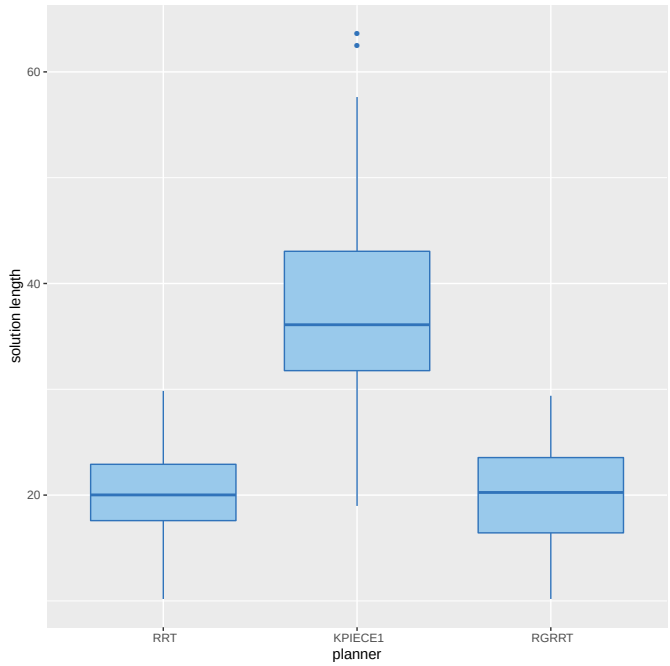
   On Figure 5a and Figure 5b, we can see the evidence for these claims in case of the pendulum. Here, RGRRT has higher computation time due to expensive integrations for generating reachability sets, but makes up for that in the smaller number of states that it generates. As we look at the systems with higher number of controls and more complex environments (like the car problem), this trade-off goes in the advantage of RGRRT - the time it takes to keep track of reachability sets ends up being smaller than the time that RRT spends on heading in wrong directions, so in practice it is a win-win.

   Rate the difficulty of each exercise on a scale of 1–10 (1 being trivial, 10 being impossible). Give an estimate of how many hours you spent on each exercise, and detail what was the hardest part of the assignment. Additionally, for students who completed the project in pairs, describe your individual contribution to the project.

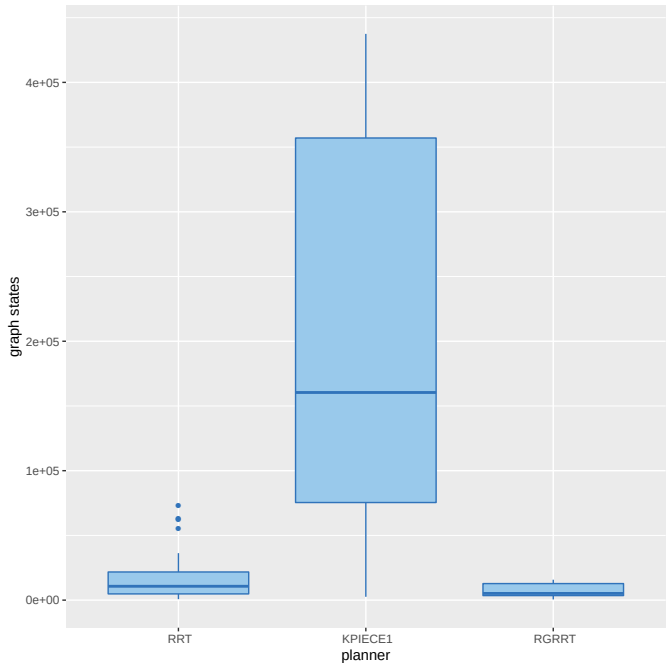
Project Exercises	Difficulty(1-10)	Time(hours)	Problems
1. Implementation of RG-RRT	5	5	We had no problems
2. Pendulum Problem	4	4	We had no problems
3. Car Problem	3	5	We had no problem
4. Benchmark	3	5	We had no problem



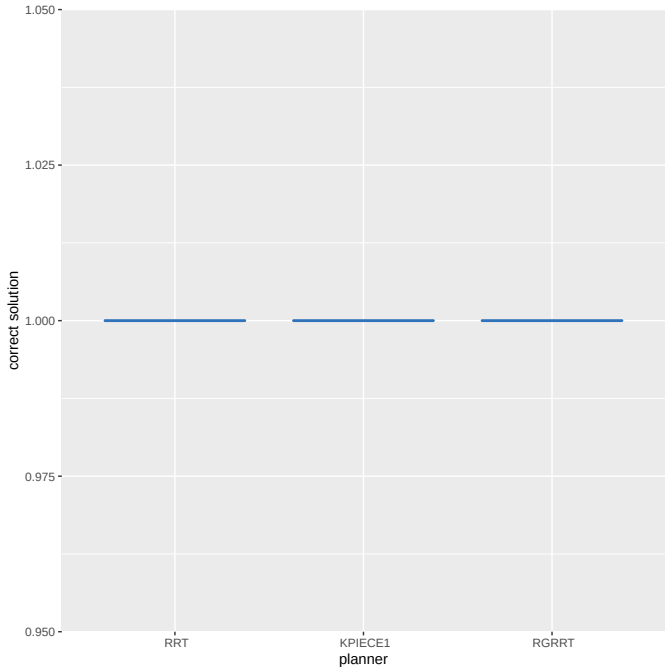
(a) computation time



(b) path length

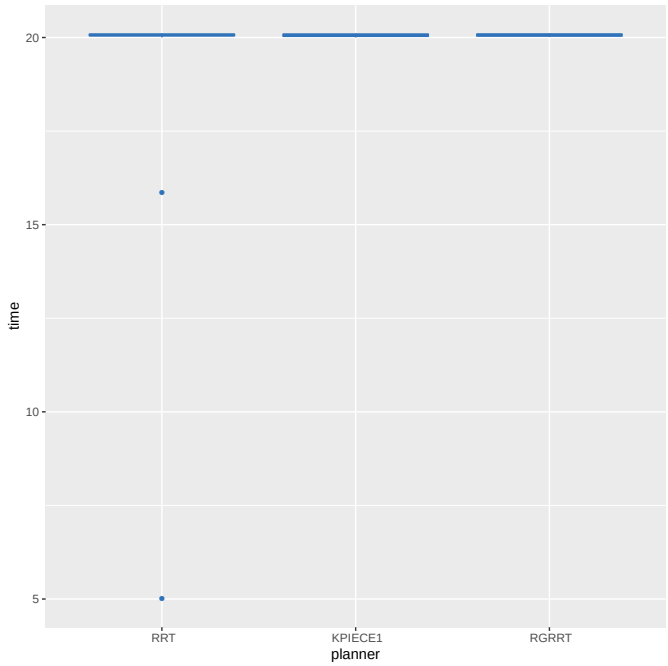


(c) number of states

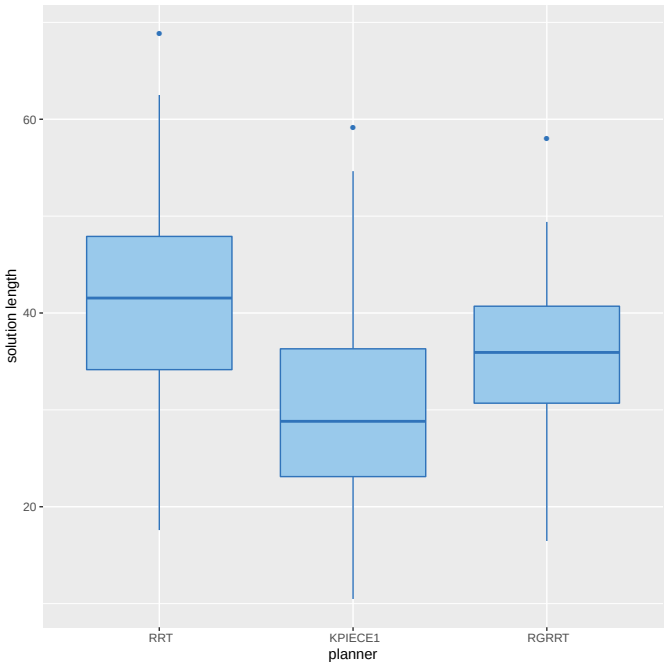


(d) correct solutions

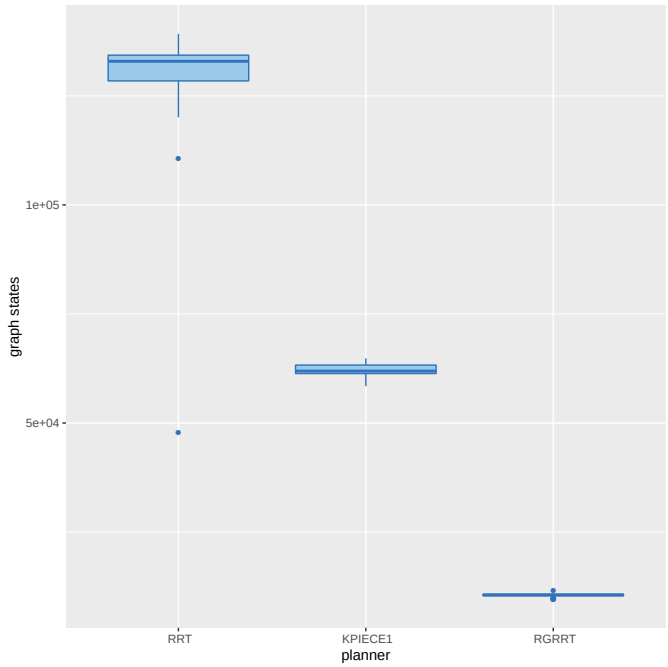
Figure 3: Pendulum Benchmark



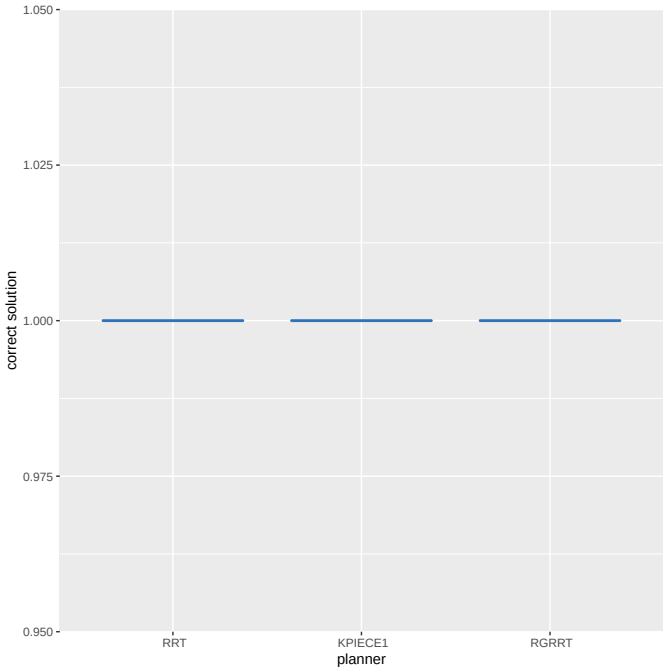
(a) computation time



(b) path length

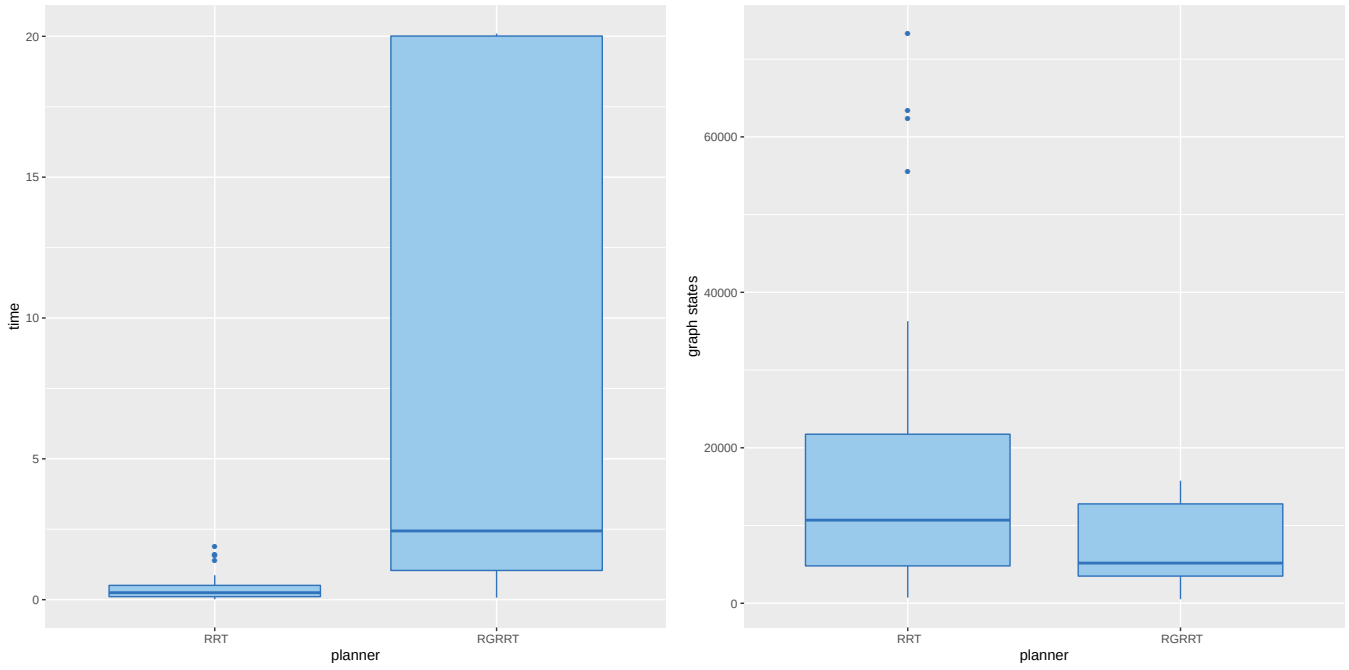


(c) number of states



(d) correct solutions

Figure 4: Car Benchmark



(a) computation time

(b) number of states

Figure 5: RRT vs RG-RRT - pendulum