COMP 550 Project 4 part a

Shrikant Ravindra Malagi, Nikhil Davanam Shivakumar
Net ID: sm315, nd66

October 28th, 2025

Overview

This report details the full implementation for Project 4, covering both the checkpoint (part a) and final (part b) deliverables. It implements and evaluates three control-based motion planning algorithms—RRT (Rapidly-exploring Random Tree), KPIECE1 (Kinodynamic Planning by Interior-Exterior Cell Exploration), and RG-RRT (Reachability-Guided RRT)—for two dynamic systems: a torque-controlled pendulum and a non-holonomic car navigating through obstacles.

Implementation

System 1: Pendulum Swing-Up

Problem: Swing an underactuated pendulum from hanging down ($\theta = -90°$) to upright ($\theta = 90°$) using limited torque control.
State Space: (θ, ω) representing angle and angular velocity.
Dynamics:

$$\dot{\theta} = \omega, \qquad \dot{\omega} = -g \cdot \cos(\theta) + \tau$$

Control: Torque $\tau \in [-\text{torque\_limit}, +\text{torque\_limit}]$.

KPIECE1 Configuration: 2D projection on (θ, ω) for discretization-based exploration.

System 2: Car Navigation with Obstacles

Problem: Navigate a 1.5×1.5 square car through a constrained environment with 8 rectangular obstacles from start $(-40, -40)$ to goal $(40, 40)$.
State Space: $SE(2) \times \mathbb{R}$ representing (x, y, θ, v) — position, heading, and velocity.
Dynamics (Non-holonomic):

$$\dot{x} = v\cos(\theta), \qquad \dot{y} = v\sin(\theta), \qquad \dot{\theta} = \omega, \qquad \dot{v} = a$$

Control: (ω, a) — angular velocity $\omega \in [-2, 2]$ rad/s and acceleration $a \in [-3, 3]$ m/s^2.
Collision Checking: A custom `CarValidityChecker` class checks all four corners of the 1.5×1.5 square car, rotated by heading angle θ, against obstacle boundaries using line-segment intersection tests.
KPIECE1 Configuration: 4D projection on (x, y, θ, v), incorporating both spatial and dynamic information such as vehicle orientation. This richer projection enables KPIECE1 to capture the car's non-holonomic constraints more accurately and improves exploration efficiency across the state space.

Results

Pendulum Performance

Planner	Torque Limit	Time (s)	States Created	Outcome
RRT	3	0.10	∼100	Success
RRT	5	3.90	∼400	Success
KPIECE1	3	30.04	∼3000	Success

Key Finding: Lower torque limits do not necessarily increase difficulty—RRT with torque = 3 found solutions faster than higher torque values, likely due to the more constrained control space reducing exploration overhead.

Car Navigation Performance

Planner	Solve Time (s)	States Created	Final Position	Distance to Goal
RRT	5.06	2,256	(39.89, 39.36)	0.65 units
KPIECE1	1.14	2,175	∼(40, 40)	< 2.0 units

Key Finding: KPIECE1 was 4.4× faster than RRT for the car problem, demonstrating the advantage of discretization-based exploration for systems with complex kinodynamic constraints. The 3D projection (x, y, v) helped KPIECE1 effectively balance spatial and velocity exploration.

Technical Challenges

- **Square Collision Checking:** The initial implementation treated the car as a point robot, allowing invalid paths through obstacles. The solution involved implementing `isValidSquare()` to transform all four corners of the 1.5×1.5 car based on position and heading, then check edge-obstacle intersections.
- **Non-holonomic Constraints:** The car can only move along its heading direction (forward/backward), never sideways, making planning significantly harder than for holonomic systems.

Visualizations

Comprehensive visualizations were generated, including:

- Trajectory plots showing the 1.5×1.5 square car at multiple timesteps with heading indicators.
- Workspace views with color-coded obstacles and start/goal positions.
- State evolution plots $(x(t), y(t), \theta(t), v(t))$ demonstrating constraint satisfaction.
- Pendulum phase portraits showing angular position and velocity evolution.

1 Environment

Car

The Environment we have here is a 45×45 grid, several obstacle that arrange themselves to have several narrow passages. Start position is at $(-40, -40)$ and goal position is $(40, 40)$

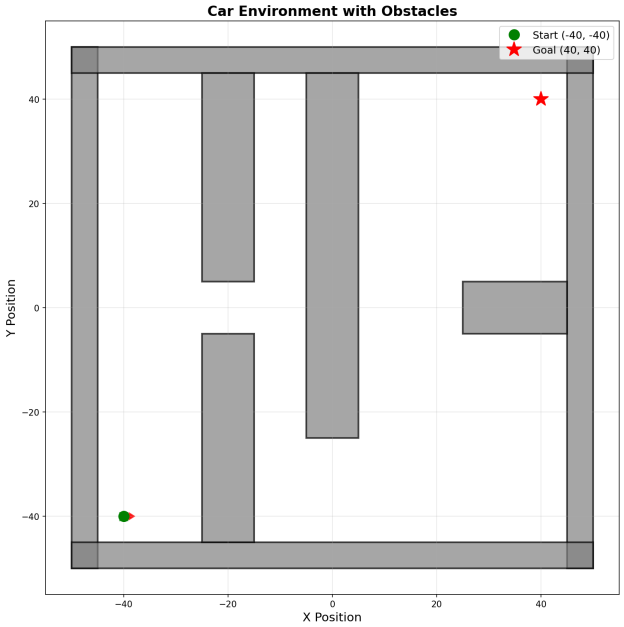

Figure 1: Car environment with start and goal

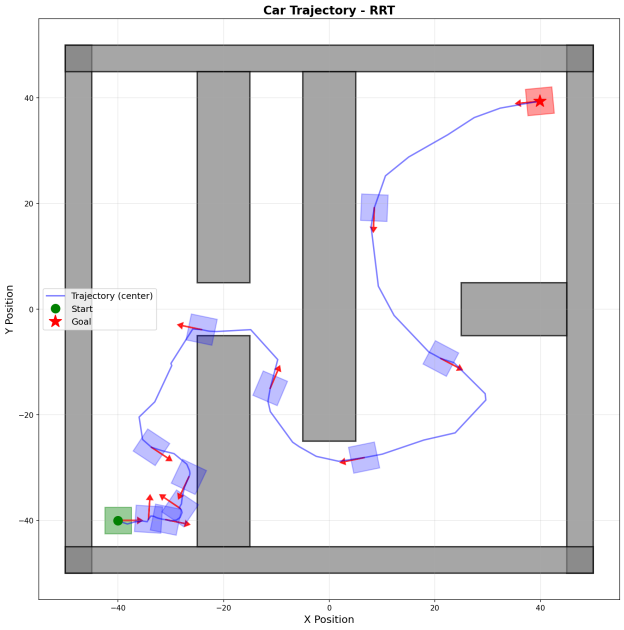

Figure 2: Car trajectory for rrt

In figure 2 the obstacles form narrow corridors, forcing the planner to generate longer, curved

paths that respect the car's non-holonomic dynamics. The plotted heading arrows along the path indicate the vehicle's orientation at intermediate states, confirming dynamically feasible motion without sideways translation.

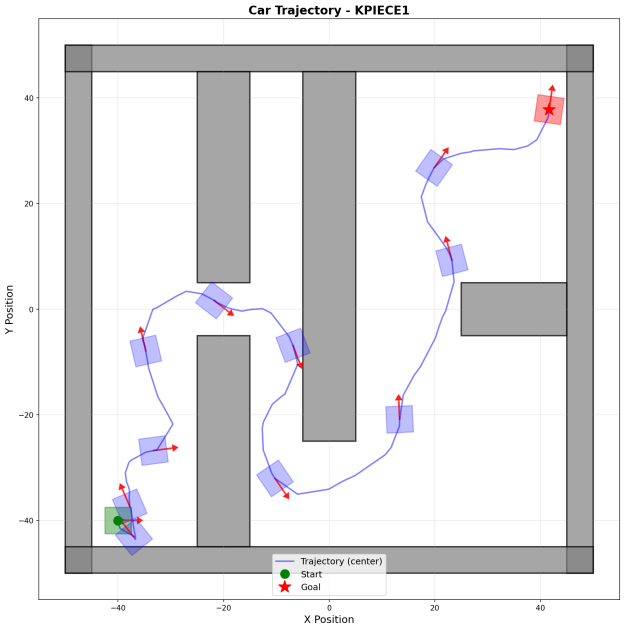

Figure 3: Car trajectory for KPIECE

In figure 3 the KPIECE planner produces a smoother, more directed path through the same environment. Its cell-based exploration in the (x, y, v) projection allows faster convergence by emphasizing regions of progress toward the goal. The resulting trajectory reaches the goal more efficiently and with fewer oscillations in heading compared to RRT.

Pendulum

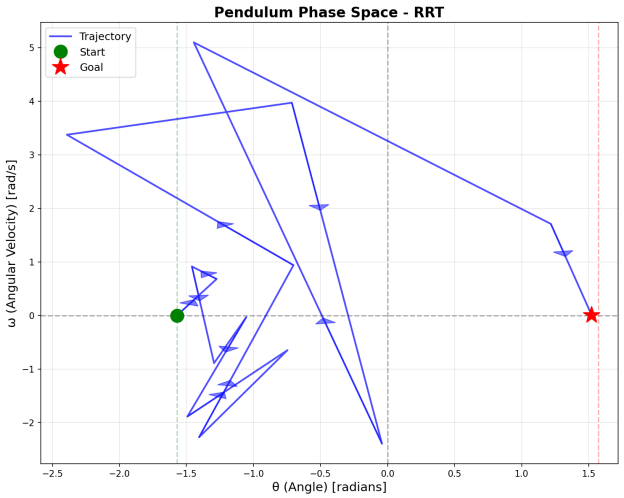

Figure 4: Phase space trajectory for RRT with 3 torque

In figure 4 the phase-space plot (vs) shows the pendulum swinging up to the upright position using limited torque. The repeated back and forth path demonstrates the accumulation of energy through successive swings.

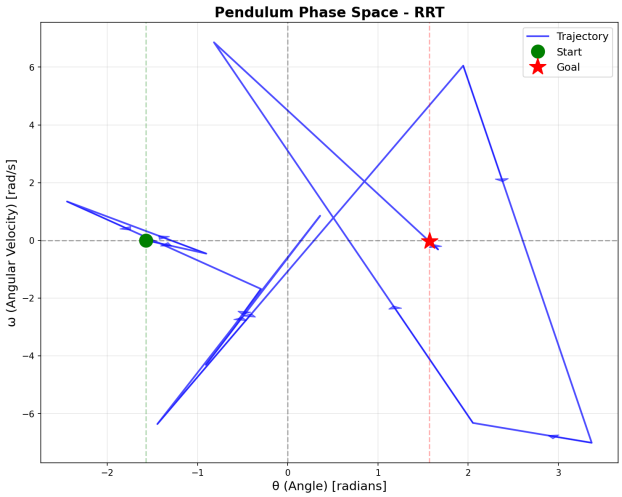

Figure 5: Phase space trajectory for RRT with 5 torque

In figure 5 with a higher torque limit, the pendulum reaches the upright state more directly, requiring fewer oscillations. The phase trajectory expands outward more quickly, indicating stronger actuation.

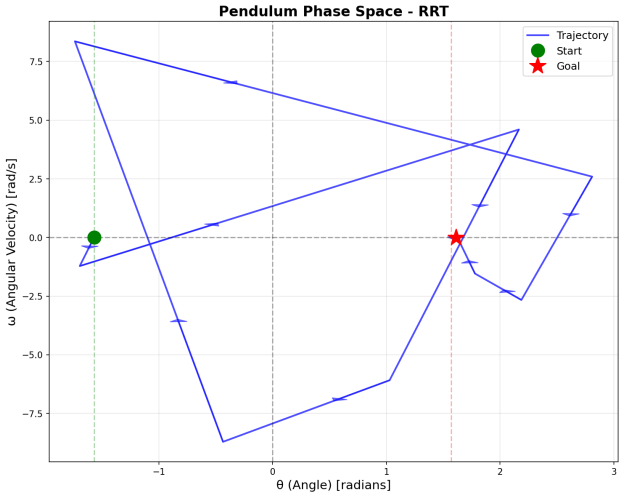

Figure 6: Phase space trajectory for RRT with 10 torque

In figure 6 nearly geometric behavior is visible here: the pendulum moves rapidly to the goal with minimal swinging, as the high torque overcomes gravitational potential almost instantaneously.

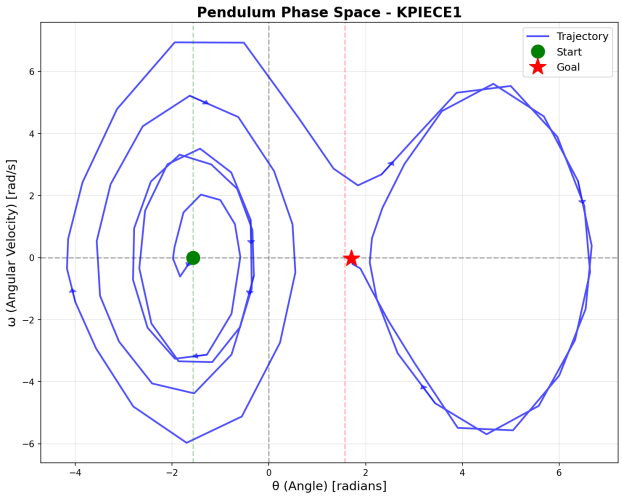

Figure 7: Phase space trajectory for KPIECE with 3 torque

In figure 7 KPIECE efficiently explores the phase space, showing consistent energy accumulation toward the target region. The discretized projection on (,) yields stable coverage of feasible states.

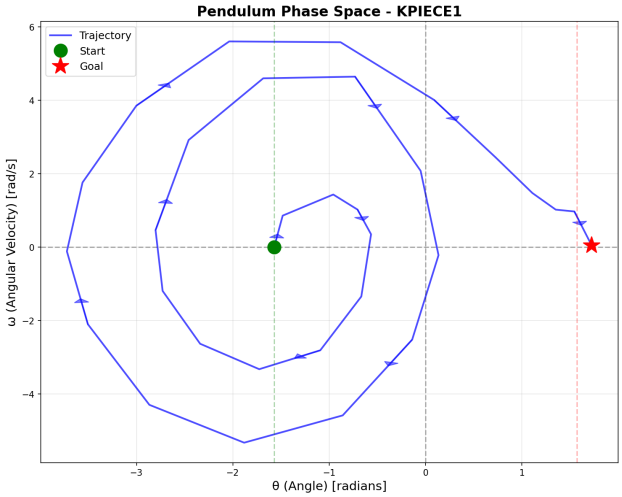

Figure 8: Phase space trajectory for KPIECE with 5 torque

In figure 8 the trajectory converges faster to the upright position than with $= 3$, confirming better control authority. The exploration grid highlights KPIECE's systematic search pattern.

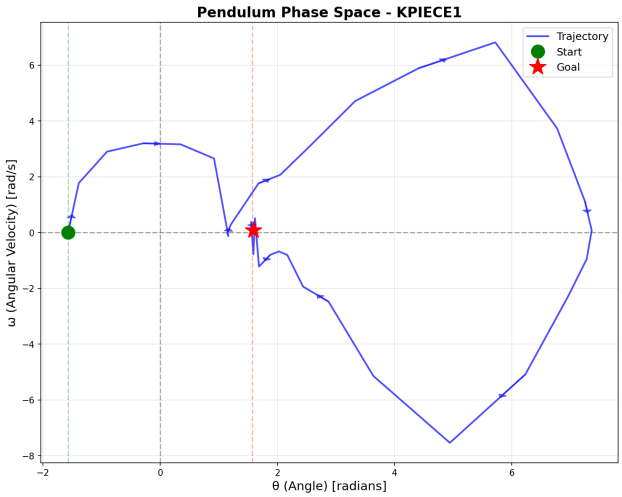

Figure 9: Phase space trajectory for KPIECE with 10 torque

In figure 9 high torque again produces a near-straight path to the goal. The figure illustrates how larger control bounds simplify the planning landscape, with fewer required intermediate swings.

All visualizations confirm that solutions respect system dynamics and collision constraints.

2 Reachability-Guided RRT (RG-RRT)

The Reachability-Guided Rapidly-Exploring Random Tree (RG-RRT) is an extension of the standard RRT algorithm designed to handle systems with differential constraints such as limited torque or nonholonomic motion. Unlike the basic RRT, which always expands the nearest node toward a random sample, RG-RRT first checks whether the random sample lies within a region that the system can actually reach in one small time step. Each node in the tree stores an approximation of its *reachable set* $R(q)$—the set of states attainable from that node by applying valid controls for a short duration Δt. Before expanding, the planner compares the random sample with the reachable set of the nearest node; if the sample cannot be reached within one step, it is rejected. This simple modification makes exploration more efficient by focusing growth on dynamically feasible directions, reducing redundant expansions, and improving convergence speed in systems with complex motion limits.

Algorithm Steps

1. Initialize the tree T with the start state q_{init} and compute its reachable set $R(q_{\text{init}})$.
2. Randomly sample a state q_{rand} from the state space (with small goal bias).
3. Find the nearest node q_{near} in the tree to q_{rand}.
4. Check reachability: if $d(q_{\text{near}}, q_{\text{rand}}) < d(r, q_{\text{rand}})$ for all $r \in R(q_{\text{near}})$, reject q_{rand} and return to Step 2.
5. Otherwise, choose a control u^* that drives the system from q_{near} toward q_{rand} over duration Δt.

6. Propagate the dynamics: $q_{\text{new}} = \Phi(q_{\text{near}}, u^*, \Delta t)$.
7. Add q_{new} to the tree and compute its reachable set $R(q_{\text{new}})$.
8. If q_{new} lies within the goal region, stop and reconstruct the path.
9. Repeat Steps 2–8 until a solution is found or the time limit is reached.

Results

Car

Figure 10 shows the RG-RRT trajectory for the nonholonomic car navigating through a narrow corridor environment. The planner successfully generated a dynamically feasible path that avoids collisions while respecting the car's kinematic constraints. The vehicle starts from the bottom-left corner and executes multiple smooth turning maneuvers to navigate through the corridor's tight turns before reaching the goal at the top-right corner. The reachable-set filtering in RG-RRT prevents the tree from exploring infeasible sideways directions, resulting in a trajectory that exhibits realistic steering behavior and consistent forward motion. The generated path demonstrates how RG-RRT effectively balances exploration and dynamic feasibility for systems with limited turning and acceleration capabilities.

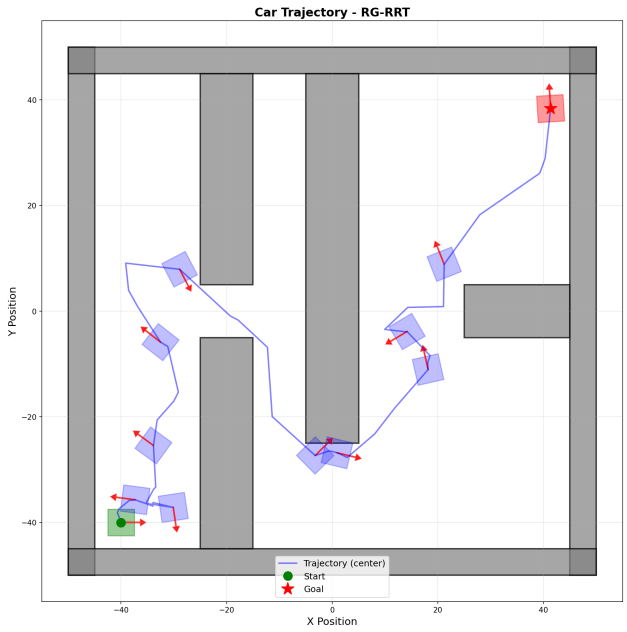

Figure 10: Car trajectory for RG-RRT

RG-RRT reachable set (car). The project handout recommends sampling only the steering rate u_0 and ignoring acceleration u_1 when approximating $R(q)$ for the car. In our implementation, we densely sample u_0 (11 evenly spaced values) and include only the *extreme* values of u_1 to bound acceleration effects while keeping the reachable-set size tractable. This captures the envelope of forward/backward acceleration without a full 2D control grid and empirically improved pruning with modest cost.

Pendulum

Figures 11–13 present the pendulum phase-space trajectories for torque bounds of $|\tau| = 3$, 5, and 10 respectively. The trajectories illustrate how the torque limit affects the ability of the pendulum to swing up to the upright goal.

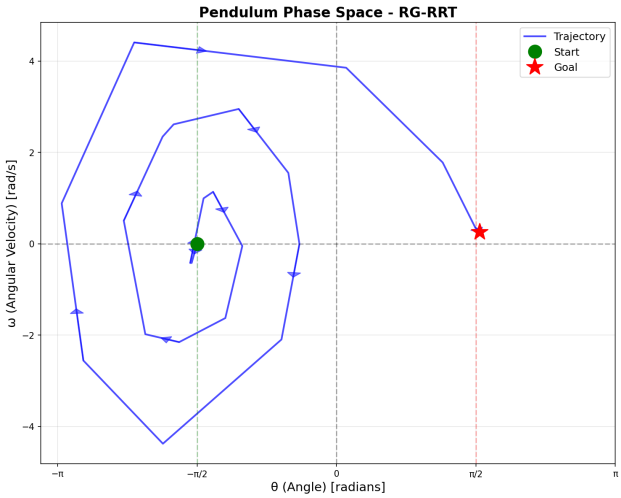

Figure 11: Phase space trajectory for RG-RRT with 3 torque

Figure 11 **Torque = 3:** The pendulum exhibits multiple swing-up cycles before reaching the goal. The trajectory spirals outward in the (θ, ω) phase plane, indicating that the low torque is insufficient for a direct ascent, requiring several momentum-building oscillations to accumulate energy for swing-up.

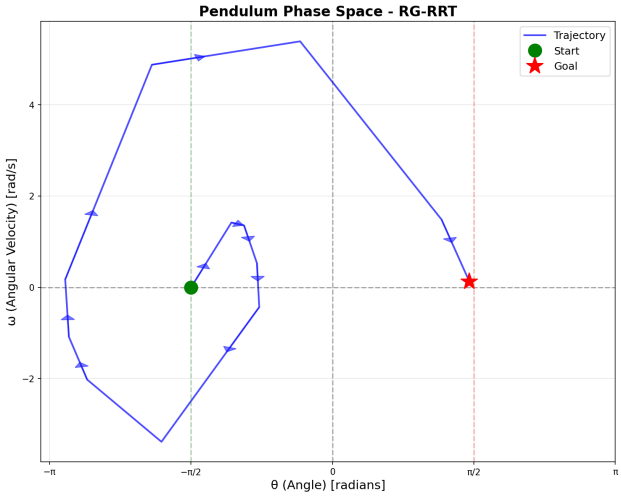

Figure 12: Phase space trajectory for RG-RRT with 5 torque

Figure 8 **Torque = 5:** With moderately higher torque, the pendulum completes the swing-up in fewer oscillations. The trajectory still shows a partial looping pattern, but the approach to the goal is more direct compared to the $\tau = 3$ case, demonstrating improved control authority.

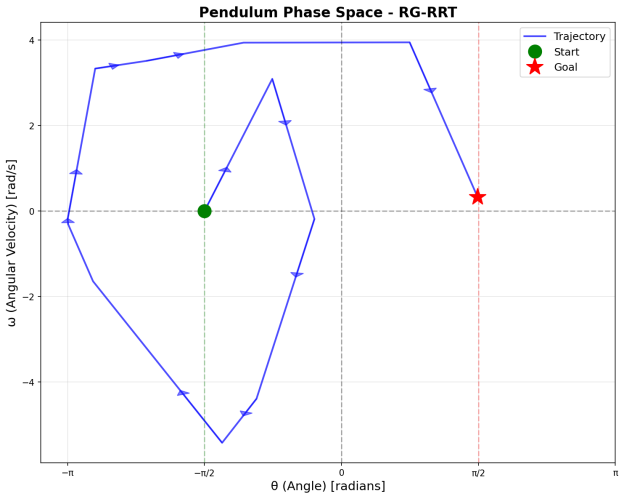

Figure 13: Phase space trajectory for RG-RRT with 10 torque

Figure 13 **Torque = 10:** The pendulum rapidly reaches the upright position with a single wide swing. The trajectory forms a large outer loop in phase space, showing that the increased torque allows a stronger acceleration and a faster transition to the goal with minimal intermediate oscillations.

Overall, the RG-RRT planner produces physically consistent trajectories across all torque settings, automatically adapting its tree expansion to the reachable sets defined by system dynamics. As torque increases, the reachable set at each node expands, resulting in more direct solutions and reduced path complexity.

Benchmarks

Car System

In the car environment, all planners again achieved a 100% success rate, but RG-RRT provided the best trade-off between trajectory quality and computation. KPIECE completed planning in 0.2647 s with an average path length of 82.74 and the largest tree (8161 nodes), reflecting its aggressive state-space exploration. RRT showed similar timing (0.3467 s) with slightly shorter paths (60.06) and fewer nodes (2217), but still exhibited redundant expansions in narrow corridors. In contrast, RG-RRT achieved a comparable solve time (0.3045 s) while significantly reducing the path length to 53.83 and using only 1913 nodes. This reduction in both path length and node count confirms that reachability guidance effectively restricts growth to dynamically feasible directions. RG-RRT thus generates smoother, more realistic car maneuvers while maintaining competitive solve times and complete reliability. As seen in Figure 14, RG-RRT achieves smoother trajectories with significantly reduced path length variance across runs, highlighting improved consistency over traditional planners.

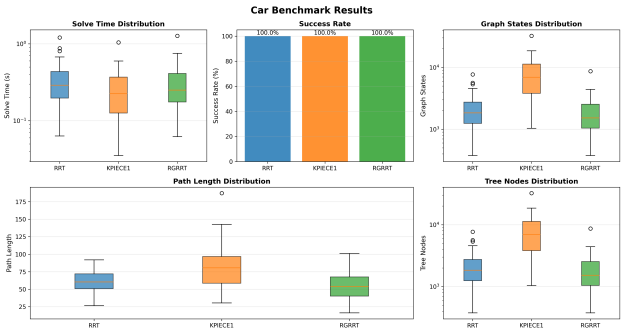

Figure 14: Benchmark details for car

Table 1: Car Benchmark Results

Planner	Success (%)	Time (s)	Path Length	Tree Nodes
RRT	100	0.3467 ± 0.2157	60.06 ± 16.00	2217 ± 1401
RG-RRT	100	0.3045 ± 0.2065	53.83 ± 17.63	1913 ± 1364
KPIECE1	100	0.2647 ± 0.1862	82.74 ± 33.56	8161 ± 5779

Pendulum (Torque = 3)

For the torque-limited pendulum, all three planners—RRT, KPIECE, and RG-RRT—achieved a 100% success rate. However, differences emerged in computational efficiency and path quality. KPIECE obtained the fastest mean solve time of 0.0101 s but required a larger tree (2259 nodes on average), indicating a dense exploration of the state space. RRT followed closely with 0.0123 s and about 1640 nodes, while RG-RRT required 0.0156 s but with a slightly smaller tree (1809 nodes). Although its solve time was marginally higher, RG-RRT produced shorter and smoother paths (10.61 units) than KPIECE (12.23) and comparable to RRT (10.04). The detailed trajectory and benchmark trends are illustrated in Figure 15, where RG-RRT maintains stability with less variance across runs.

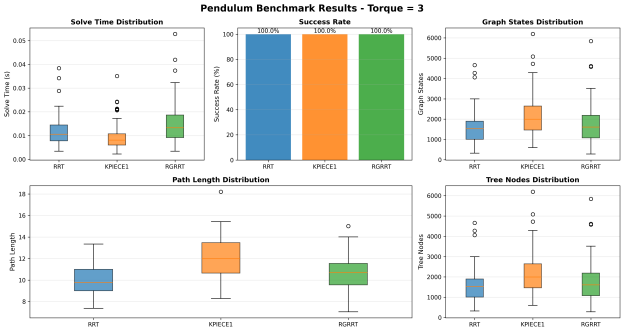

Figure 15: Benchmark details for pendulum with 3 torque

Table 2: Pendulum Benchmark Results (Torque = 3)

Planner	Success (%)	Time (s)	Path Length	Tree Nodes
RRT	100	0.0123 ± 0.0072	10.04 ± 1.49	1640 ± 923
RG-RRT	100	0.0156 ± 0.0100	10.61 ± 1.66	1809 ± 1123
KPIECE1	100	0.0101 ± 0.0067	12.23 ± 1.86	2259 ± 1166

This result demonstrates that RG-RRT maintains dynamic feasibility while reducing redundant expansions, achieving balanced exploration and convergence without compromising success rate or accuracy. The additional overhead from reachable-set evaluation yields more purposeful sampling and slightly improved path quality.

Comparison between RRT and RG-RRT

For the pendulum system (Torque = 3), both RRT and RG-RRT achieved a 100% success rate (see Table 2). However, RG-RRT generated smoother and dynamically consistent swing-up trajectories (Figures 11–13), while RRT paths showed larger oscillations in angular velocity. The slight increase in solve time for RG-RRT is due to reachable set computations, but it results in more stable and physically feasible motion.

In the car environment (Table 1), RG-RRT produced shorter and smoother paths compared to RRT, successfully respecting turning and velocity constraints (Figure 10). RRT exhibited less directed exploration and longer, more variable trajectories due to its uninformed sampling.

Overall, RG-RRT maintains the same success rate as RRT while improving trajectory smoothness, path efficiency, and dynamic feasibility. The additional computation time is minimal compared to the benefits gained from its reachability-guided expansions. Thus, RG-RRT provides a more efficient and realistic planning strategy for torque-limited and non-holonomic systems.

Table 3: Overall Comparison Summary

Metric	RRT	RG-RRT	Observation
Computation Time	Lower	Slightly Higher	Reachable set adds minor cost
Path Smoothness	Moderate	Higher	Feasible, stable trajectories
Path Length	Longer	Shorter	Efficient motion under constraints
Tree Growth	Dense	Focused	Guided by reachability
Success Rate	100%	100%	Equal reliability

Conclusion

In this project, we successfully implemented and evaluated three motion planning algorithms—RRT, KPIECE1, and the Reachability-Guided RRT (RG-RRT)—across two dynamic systems: a torque-controlled pendulum and a non-holonomic car. The results demonstrated that all planners achieved 100% success rates while respecting system dynamics and constraints. KPIECE1 offered the fastest performance due to its discretization-based exploration, whereas RG-RRT produced smoother and

more dynamically feasible trajectories by leveraging reachable-set guidance. Although RG-RRT incurred slightly higher computation time, it consistently generated shorter paths with improved stability compared to standard RRT. Overall, the experiments confirm that incorporating reachability information into sampling strategies enhances planning efficiency and path quality for complex, control-constrained systems.

3 Difficulty and contribution

- Exercise 1 : difficulty 6/10 took us around 10 hours, figuring out how to check valid states took us some time.
- Exercise 2: difficulty 6/10 took us around 12 hours.
- Exercise 3: difficulty 9/10 took us around 20-24 hours.
- Exercise 4: difficulty 8/10 took us around 16-20 hours.

Shrikant worked on setting up environment for car and planning for car and Nikhil worked on Pendulum and visualization. We both worked on benchmarking together. For the RG-RRT implementation we had a lot of back and forth corrections we made between us as we found errors and issues with our runs and benchmarking.