An Empirical Study of Sampling-Based Motion Planning Algorithms for Robotic Manipulation

Aryaman Rao[1], Kush Patel[1]

Abstract— **Sampling-based motion planners are widely used for high-dimensional robotic manipulation in cluttered environments. This work evaluates several common planners, including RRT, RRT-Connect, RRT*, BiRRT*, Informed RRT*, Lazy PRM, and PRM*, on a 6-DOF PR2 arm in a simulated tabletop setting with obstacles. Each planner is tested over multiple trials using fixed start and goal configurations to account for stochasticity. Performance is compared using metrics covering computational efficiency and trajectory quality, including planning time, node expansions, path length, waypoint count, end-effector travel distance, and joint-space discontinuity. Shortcut-based smoothing is applied to all successful plans to assess trajectory refinement. The results highlight trade-offs between speed, optimality, and smoothness, providing practical guidance for planner selection in cluttered manipulation tasks. The complete source code and implementation details can be accessed here: `https://github.com/aryamanr26/rrt-planning-suite`.**

I. INTRODUCTION

Motion planning is a fundamental problem in robotics, particularly for robotic manipulators operating in constrained and cluttered environments. A robot must compute a collision-free trajectory that moves its joints from a start configuration to a desired goal while respecting kinematic limits and avoiding obstacles. This problem becomes increasingly challenging as the dimensionality of the system grows and as the environment becomes more complex, making classical grid-based or deterministic planning methods computationally infeasible.

Sampling-based motion planning algorithms have emerged as a practical solution to these challenges. Methods such as Rapidly-exploring Random Trees (RRT) [1] and Probabilistic Roadmaps (PRM) [2] avoid explicit construction of the full configuration space and instead rely on random sampling to incrementally explore feasible regions. These algorithms are particularly well-suited for high-degree-of-freedom robotic systems, where exact representations of the configuration space are prohibitively expensive. As a result, sampling-based planners have been widely adopted in real-world robotic systems for manipulation, navigation, and autonomous operation.

Robotic manipulation in tabletop environments is a representative and practically important application domain for motion planning. Tasks such as pick-and-place, object rearrangement, and tool use require a robot to plan precise arm motions while avoiding collisions with tables, objects, and the robot's own body. These scenarios are common in industrial automation, service robotics, and assistive robotics, where safety and reliability are critical. Planning in such environments requires not only finding feasible paths, but also producing trajectories that are efficient, smooth, and suitable for execution on real hardware. Despite their widespread use, sampling-based planners exhibit different trade-offs in terms of planning time, solution quality, and robustness due to their stochastic nature and algorithmic design choices. Variants such as RRT* [3], BiRRT* [4], and Informed RRT* [5] aim to improve path optimality, while PRM-based approaches emphasize reuse and global connectivity of the configuration space. In practice, selecting an appropriate planner for a given manipulation task requires understanding these trade-offs under realistic conditions.

In this project, we perform a systematic comparison of several sampling-based motion planning algorithms using a simulated PR2 [6] robot operating in a cluttered tabletop environment containing a table and objects placed on its surface 1. We evaluate planners across multiple metrics that capture both computational efficiency and trajectory quality, including planning time, node expansion, path length, and end-effector motion characteristics. By conducting repeated trials under a fixed experimental setup, this work aims to provide empirical insights into the strengths and limitations of different sampling-based planners for high-dimensional robotic manipulation tasks.

[1]All four authors are from University of Michigan, Ann Arbor. aryamanr@umich.edu, kushkp@umich.edu

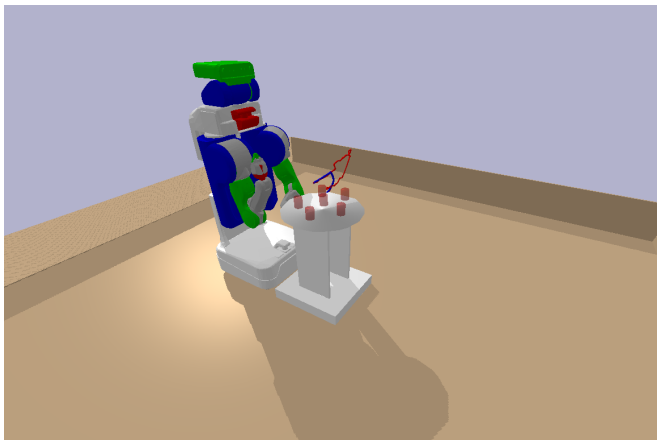

Fig. 1. **Visualization of the PR2 robot environment**. The PR2's 6-DOF arm operates in a cluttered tabletop setting containing obstacles. The *red line* shows the raw trajectory produced by the motion planning algorithm, while the *blue line* represents the corresponding smoothed trajectory after post-processing.

II. IMPLEMENTATION

A. Sampling-Based Motion Planning Overview

All planners operate in the PR2's joint configuration space and compute collision-free trajectories for the 6-DOF left arm between fixed start and goal configurations. Sampling-based methods avoid explicit construction of the configuration space by incrementally exploring feasible regions through random sampling and local collision-checked connections, enabling efficient planning in high-dimensional, cluttered tabletop environments.

Each planner differs in how samples are generated, how connections are formed, and whether path quality is explicitly optimized. The following subsections describe the algorithms implemented and evaluated in this work.

B. Rapidly-Exploring Random Tree (RRT)

The Rapidly-Exploring Random Tree (RRT) 1 algorithm incrementally builds a tree rooted at the start configuration by repeatedly sampling random configurations in the joint space. For each sample, the nearest node in the existing tree is identified using a distance metric in configuration space. The tree is then extended toward the sample by a fixed step size, provided the motion is collision-free.

RRT is designed to rapidly explore large, high-dimensional spaces and is effective at finding feasible paths quickly. However, it does not attempt to minimize path cost, often resulting in suboptimal trajectories with unnecessary detours and poor smoothness.

Algorithm 1 Rapidly-Exploring Random Tree (RRT)

Require: $q_{\text{start}}, q_{\text{goal}}, \mathcal{Q}, \delta, N$
Ensure: Collision-free path or **FAIL**

$T \leftarrow \{q_{\text{start}}\}$
for $i = 1$ to N **do**
  $q_{\text{rand}} \leftarrow \textsc{Sample}(\mathcal{Q})$
  $q_{\text{near}} \leftarrow \textsc{Nearest}(T, q_{\text{rand}})$
  $q_{\text{new}} \leftarrow \textsc{Steer}(q_{\text{near}}, q_{\text{rand}}, \delta)$
  if $\textsc{CollisionFree}(q_{\text{near}}, q_{\text{new}})$ **then**
    $\textsc{Add}(T, q_{\text{new}})$
    if $\textsc{Distance}(q_{\text{new}}, q_{\text{goal}}) < \delta$ **then**
      return $\textsc{ExtractPath}(T)$
    end if
  end if
end for
return FAIL

1) RRT-Connect: RRT-Connect extends the basic RRT framework by growing two trees simultaneously: one rooted at the start configuration and the other at the goal configuration. At each iteration, one tree attempts to extend toward a random sample, while the other aggressively attempts to connect to the newly added node. This bidirectional strategy significantly improves convergence speed compared to single-tree RRT, especially in environments where a direct connection between the start and goal is feasible. Like RRT, RRT-Connect focuses on rapid feasibility rather than path optimality.

2) RRT:* RRT* introduces asymptotic optimality to the RRT framework. In addition to extending the tree toward random samples, RRT* performs a rewiring step in which nearby nodes are reconnected if doing so reduces the overall path cost. The path cost is typically defined as the cumulative distance in configuration space. As the number of samples increases, the solution produced by RRT* converges to the optimal path. This improvement in path quality comes at the cost of increased computational overhead due to neighborhood searches and rewiring operations.

3) Bidirectional RRT (BiRRT*):* BiRRT* combines bidirectional search with the asymptotic optimality properties of RRT*. Two RRT* trees are grown from the start and goal configurations, respectively, and are periodically connected and rewired to reduce overall path cost. By leveraging bidirectional exploration, BiRRT* often converges to high-quality solutions faster than standard RRT*. However, the added complexity increases computational cost and makes performance sensitive to parameter like step size and rewiring radius.

Algorithm 2 Probabilistic Roadmap (PRM)

Require: $\mathcal{Q}, q_{\text{start}}, q_{\text{goal}}, k, N$
Ensure: Collision-free path or **FAIL**

$G \leftarrow \emptyset$
for $i = 1$ to N **do**
  $q \leftarrow \text{SAMPLE}(\mathcal{Q})$
  if $\text{COLLISIONFREE}(q)$ **then**
    $\text{ADDNODE}(G, q)$
    $\text{CONNECT}(G, q, k)$
  end if
end for
$\text{ADDNODE}(G, q_{\text{start}})$
$\text{ADDNODE}(G, q_{\text{goal}})$
$\text{CONNECT}(G, q_{\text{start}}, k)$
$\text{CONNECT}(G, q_{\text{goal}}, k)$
if $\text{PATHEXISTS}(G, q_{\text{start}}, q_{\text{goal}})$ **then**
  return $\text{SHORTESTPATH}(G)$
else
  return FAIL
end if

4) Informed RRT:* Informed RRT* improves sampling efficiency by restricting the sampling domain once an initial solution has been found. Instead of sampling uniformly over the entire configuration space, new samples are drawn from a heuristic-informed subset that is guaranteed to contain solutions better than the current best path. This focused sampling strategy reduces wasted exploration and accelerates convergence toward shorter paths, making Informed RRT* well-suited for scenarios where solution quality is important under limited computation time.

C. Probabilistic Roadmap Methods

Probabilistic Roadmap (PRM) 2 methods follow a two-phase approach consisting of a preprocessing phase and a query phase. During preprocessing, random collision-free configurations are sampled and connected to nearby neighbors to form a roadmap graph. During the query phase, the start and goal configurations are connected to the roadmap, and a graph search is performed to find a feasible path.

1) Lazy PRM: Lazy PRM [7] postpones collision checking of edges until a candidate path is found during the query phase. This reduces upfront computational cost and allows rapid generation of candidate solutions, at the risk of invalidating edges during validation.

2) PRM:* PRM* [3] extends the PRM framework by providing asymptotic optimality guarantees. As the number of samples increases, the solution path converges to the optimal path. This improvement in solution quality comes at the cost of denser graph connectivity and increased collision checking.

D. Performance Metrics

To enable a systematic and fair comparison of sampling-based motion planning algorithms, each planner is evaluated using a set of quantitative performance metrics that capture both computational efficiency and trajectory quality. Due to the stochastic nature of sampling-based planners, all metrics are averaged over multiple successful runs for each algorithm.

1) Planning Time and Node Expansion: Planning time measures the wall-clock time required for a planner to compute a feasible path from the start to the goal configuration. This metric reflects the computational efficiency of each algorithm and is particularly important for time-sensitive applications. In addition, the number of nodes expanded during planning is recorded as a measure of the search effort required to find a solution.

2) Raw Path Length: The raw path length corresponds to the total length of the joint-space trajectory produced directly by the planner before any post-processing. It is computed as the sum of Euclidean distances between consecutive configurations along the path. This metric reflects the inherent solution quality of each planning algorithm.

3) Raw Waypoint Count: The raw waypoint count is defined as the number of configurations in the initial solution path returned by the planner. A high number of waypoints often indicates a more irregular or fragmented path, which can negatively affect execution and downstream smoothing performance.

4) Path Smoothing and Improvement Percentage: To improve trajectory quality, shortcut-based path smoothing is applied to all successful plans. The smoothed path length is computed using the same joint-space distance metric as the raw path length. The improvement percentage quantifies the relative reduction in path length achieved through smoothing and is defined as

$$\text{Improvement} = \left(1 - \frac{L_{\text{smooth}}}{L_{\text{raw}}}\right) \times 100\%, \qquad (1)$$

where L_{raw} and L_{smooth} denote the raw and smoothed path lengths, respectively. This metric highlights how much redundancy exists in the original plan and how effectively smoothing can improve trajectory quality.

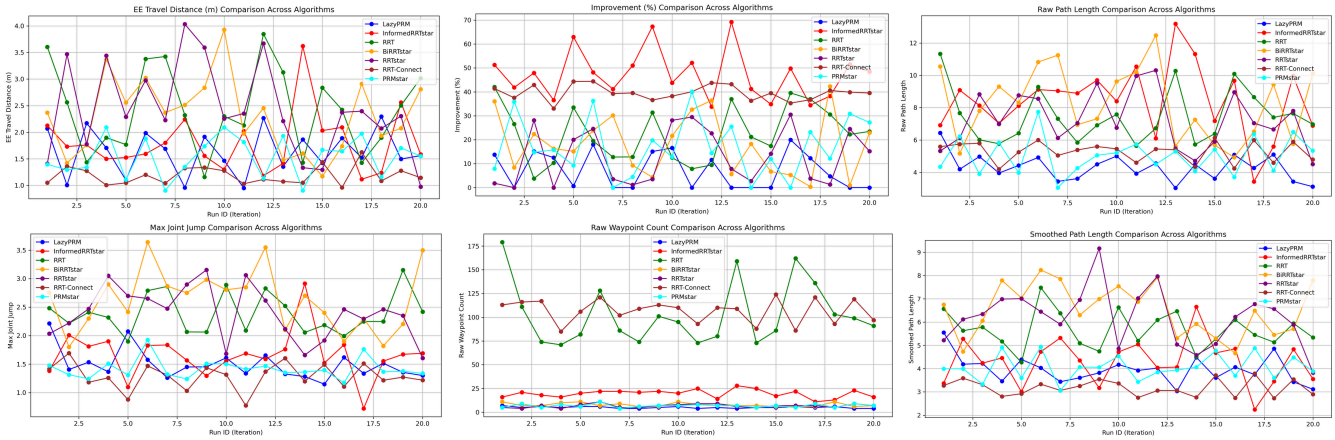

Fig. 2. Comparison of six trajectory quality metrics across all planners, averaged over 20 independent trials. Subplots report joint-space and task-space performance measures for each algorithm.

5) End-Effector Travel Distance: Task-space behavior is evaluated by computing the end-effector travel distance along the smoothed trajectory. End-effector positions are obtained using forward kinematics, and the total Cartesian distance traveled is computed. Lower values indicate more efficient and direct manipulation, while higher values reflect detours or overly conservative motion in cluttered environments.

6) Maximum Joint Jump: The maximum joint jump measures the largest change between consecutive joint configurations in the smoothed trajectory. Large jumps indicate reduced smoothness and may lead to undesirable motion or execution difficulty on real hardware.

E. Path Smoothing and Post-Processing

Shortcut-based path smoothing is applied to all successful plans to improve execution quality by replacing path segments with shorter, collision-free shortcuts. This post-processing step reduces path length, waypoint count, and joint-space discontinuities, resulting in smoother and more executable trajectories.

Several design decisions were made to ensure a fair and informative evaluation of the planners. Shortcut-based path smoothing was applied uniformly to all algorithms to reduce redundant motion and improve execution feasibility, as raw sampling-based solutions often contain unnecessary detours and irregularities. The selected performance metrics capture complementary aspects of planner behavior, including computational efficiency, joint-space smoothness, and task-space motion quality, which are all critical for manipulation tasks in cluttered environments. Due to the inherent randomness of sampling-based planners, each algorithm was executed over multiple independent trials and results were averaged to obtain representative performance trends and reduce sensitivity to outlier runs.

III. RESULTS AND DISCUSSION

All planners were evaluated on a simulated tabletop manipulation task using the PR2 robot's 6-DOF left arm. The environment contained static obstacles placed on a table, and all experiments used identical fixed start and goal configurations. Collision checking, planner parameters, and shortcut-based path smoothing were kept consistent across planners to ensure a fair comparison. Due to the stochastic nature of sampling-based methods, each planner was executed over 20 independent trials 2, and all reported metrics represent averages across successful runs.

In terms of planning time, Informed RRT* required the longest computation time, averaging approximately 200–250 s, reflecting the additional overhead of informed sampling and optimization. PRM* required moderate planning time, averaging around 30–50 s, while all remaining planners completed planning in under 30 s on average, indicating faster feasibility-focused behavior. These results show a clear trade-off between planning speed and solution refinement.

For task-space performance, RRT, BiRRT*, and RRT* exhibit the highest end-effector travel distances, indicating less direct motion in task space compared to other planners. Overall, RRT-based planners produce larger task-space trajectories than PRM-based methods. After shortcut-based smoothing, Informed RRT* and RRT-Connect show the largest relative reductions in end-effector travel distance, revealing significant redundancy in their raw paths, while PRM-based planners exhibit smaller improvements due to more direct initial solutions.

A similar trend is observed for raw path length. Lazy PRM and PRM* generate the shortest paths, followed by Informed RRT* and BiRRT*, reflecting the global

Algorithm	Planning Time (s)	Raw Waypoints	Raw Path Length	Smoothed Waypoints	Smoothed Path Length	Improvement (%)	EE Travel Distance (m)	Max Joint Jump
RRT-Connect	1.37	106.60	5.27	5.40	**3.18**	39.61	**1.19**	**1.29**
RRT	**1.17**	103.15	7.48	4.30	5.66	22.55	2.42	2.39
RRT*	1.28	6.75	7.27	**4.15**	6.20	13.16	2.42	2.39
BiRRT*	1.52	7.90	8.25	**4.15**	6.54	17.96	2.32	2.63
InformedRRT*	203.55	19.60	8.41	4.55	4.30	**47.28**	1.80	1.66
LazyPRM	1.41	**5.10**	**4.30**	4.30	3.96	7.02	1.63	1.49
PRM*	30.37	6.85	5.03	4.50	4.06	17.11	1.53	1.42

Fig. 3. Average performance metrics for all motion planning algorithms evaluated over multiple runs.

connectivity of roadmap-based planners. In contrast, tree-based planners favor rapid exploration, which often results in longer and less efficient raw trajectories.

Joint-space metrics further highlight planner differences. RRT-Connect achieves the lowest maximum joint-space discontinuity, indicating smoother transitions between consecutive configurations. PRM-based planners follow closely, while other RRT-based planners exhibit larger joint jumps due to aggressive tree expansion and less constrained local connections. RRT and RRT-Connect also produce the highest raw waypoint counts, a consequence of incremental tree growth that introduces many intermediate configurations along the solution path.

After smoothing, RRT* and BiRRT* result in the longest trajectory lengths, PRM-based planners yield slightly shorter paths, and RRT-Connect produces the shortest smoothed trajectories. Overall, PRM-based planners consistently produce shorter and smoother solutions, while RRT-based planners prioritize rapid exploration and benefit more from post-processing, particularly Informed RRT* and RRT-Connect.

IV. CONCLUSION

This work presented a comparative evaluation of several widely used sampling-based motion planning algorithms on a 6-DOF PR2 arm operating in a cluttered tabletop environment. The planners were assessed using both joint-space and task-space metrics, along with the effect of shortcut-based path smoothing, to capture differences in planning efficiency, trajectory quality, and motion behavior.

The results highlight clear trade-offs between planner families. PRM-based methods consistently produced shorter, smoother, and more direct trajectories, while RRT-based planners favored rapid exploration at the expense of path efficiency. Among RRT variants, RRT-Connect achieved the smoothest joint-space transitions and the most compact smoothed paths, whereas optimal and informed variants benefited most from post-processing due to redundancy in their initial solutions. Overall, the findings emphasize that planner selection should be guided by task requirements, environmental complexity, and tolerance for post-processing, providing practical insight for manipulation planning in cluttered settings.

V. FUTURE WORKS

Future work will extend this evaluation to more complex manipulation tasks involving dynamic obstacles, tighter kinematic constraints, and end-effector orientation requirements. Additional smoothing and optimization techniques, such as time-parameterized and torque-aware trajectory optimization, could be explored to further improve execution quality. Finally, benchmarking the planners on physical hardware and across different robot platforms would provide deeper insight into real-world performance and robustness

VI. ACKNOWLEDGMENT

We sincerely thank Prof. Dmitry Berenson for his guidance, insightful lectures, and continuous support throughout this project. We also thank our course GSI for valuable feedback and debugging assistance. Finally, we acknowledge the University of Michigan for providing the resources and academic environment that enabled this work, as well as the open-source tools used in our experiments

REFERENCES

[1] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” Computer Science Department, Iowa State University, Tech. Rep. TR 98-11, 1998, Available at http://msl.cs.uiuc.edu/ lavalle/papers/Lav98c.pdf.

[2] L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996. DOI: 10.1109/70.508439.

[3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. DOI: 10.1177/0278364911406761.

[4] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, Tech. Rep. MIT-CSAIL-TR-2013-021, Aug. 2013, Available at https://dspace.mit.edu/handle/1721.1/79884.

[5] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 2997–3004. DOI: 10.1109/IROS.2014.6942976.

[6] W. Garage, *Pr2: Personal robot 2*, Willow Garage, Inc. (now maintained by community via ROS and Clearpath Robotics), Platform description and manual available at http://www.ros.org/wiki/Robots/PR2, 2010–2014.

[7] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings 2000 ICRA. IEEE International Conference on Robotics and Automation*, 2000, pp. 521–528. DOI: 10.1109/ROBOT.2000.844107.