Comparison of Sampling-based Planning Methods with Constraint for a 7-DoF Robot Arm

Aaron Guan
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
zhongg@andrew.cmu.edu

Yichen Xu
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
yichenxu@andrew.cmu.edu

Abstract

Robots with many degrees of freedom (DoF) have increasingly been employed to accomplish realistic tasks in domains such as logistics and home caretaking. Sampling-based motion planning algorithms are effective for these high-dimensional systems. However, incorporating task constraints into the planning process introduces significant challenges. This work examines the behavior of four different sampling-based motion planning algorithms with constraints for a 7-DoF robot manipulator in different environment with obstacles. The planning algorithms include Rapidly-exploring Random Trees (RRT), Constrained Bi-directional Rapidly-exploring Random Tree (CBiRRT), Probabilistic Roadmap (PRM), and Obstacle-based Probabilistic Roadmap (OBPRM). Finally, time of planning collision-free path, total number of nodes sampled in the configuration space, and trajectory cost are examined.

1 Introduction

Finding feasible motions for the robot arm with many degree of freedoms autonomously is essential for their operation. The sampling-based motion planning algorithms have become popular in obstacle avoidance motion planning due to the probability completeness, and they can easily plan a valid path for high degree of freedom systems. The most basic sampling-based algorithms are the Rapidly-exploring Random Trees (RRT) and the Probabilistic Roadmap (PRM). Additionally, the motion planning system must respect the constraints on a task in order to achieve success. For example, a common task in service applications is to move a water-filled cup with a manipulator, which requires the manipulator to move with a certain pose. In this project, we have implemented different sampling-based planning algorithms including RRT, PRM, Constrained Bi-directional Rapidly-exploring Random Tree (CBiRRT) [1], and Obstacle-based Probabilistic Roadmap (OBPRM) [2]. We also incorporated the end-effector pose constraint into the planning process and compared the behavior of these algorithms in different environment. Some static boxes are put in the environment as obstacles to check whether these planners can find collision-free path to the target in the narrow passage configuration space.

The robotic system used in this project is 7-DoF Franka Emika robot arm. Therefore, the workspace is the 3D Cartesian space which consists of both location and orientation of the end-effector. The configuration space is the angle of every joint on the robot arm which has 7 DoF. The task is to plan a collision-free path for the robot arm to move from an initial configuration to a goal configuration while satisfying the end-effector constraint of keeping upright. The planning time, total number of nodes sampled in the configuration space, and the path quality are examined for each planning algorithm.

The remainder of this paper is organized as follows. The second section describes the methodology we used in this work. Then, the experiment results and analysis are presented in the third section. Finally, the conclusion is given based on the comparison of these planning algorithms and several directions to improve the current work have been mentioned in the concluding section.

2 Methodology

In this section, the methodology of the four motion planning algorithms: RRT, CBiRRT, PRM, and OBPRM is described. Additionally, the projection sampling with constraint and the path smoothing method used in this work are also presented.

2.1 RRT

The RRT algorithm operates by growing one tree in the C-space of the robot (see Algorithm 1). During each iteration of the algorithm the tree moves toward a randomly-sampled configuration q_{sample} with a step size Δq_{step} using the ConstrainedExtend function. Meanwhile, the ConstraintProjection function is used to project q_{sample} to a constraint manifold, which will be described in section 2.5. In order to improve the search efficiency, the idea of goal-bias is combined with the naive RRT algorithm. The function terminates when the tree is making progress toward the new configuration q_{new}.

Algorithm 1 ConstrainedExtend

Require: T, q_{target}

while True **do**
  if $rand() < p_{target}$ **then**
    $q_{sample} \leftarrow q_{target}$
  else
    $q_{sample} \leftarrow$ SampleValidJoints()
  end if
  $q_{near} \leftarrow T$.Nearest(q_{sample})
  $q_{new} \leftarrow q_{near} + \min\left(\Delta q_{step}, \|q_{sample} - q_{near}\|_2\right) \frac{q_{sample} - q_{near}}{\|q_{sample} - q_{near}\|_2}$
  $q_{new} \leftarrow$ ConstraintProjection(q_{new})
  if InCollision(q_{new}) **then**
    continue
  end if
  T.AddVertex(q_{new})
  T.AddEdge(q_{near}, q_{new})
  if $\|q_{sample} - q_{near}\|_2 < \Delta q_{step}$ **then**
    return True, q_{new}
  end if
  return False, q_{new}
end while

2.2 CBiRRT

The CBiRRT, which builds on the standard RRT-Connect algorithm, operates by growing two trees in the C-space of the robot (see Algorithm 2). During each iteration of the algorithm one of the trees grows a branch toward a randomly-sampled configuration, while the other tree then grows a branch toward the configuration reached by the first tree. If the other tree reaches tat configuration, the trees have connected and a path has been found. If not the trees are swapped and the above process is repeated.

2.3 PRM

The PRM uses randomization to construct a graph of representative paths in C-space whose vertices correspond to collision-free configurations and in which two vertices are connected by an edge. First, a random configuration projected on the constraint manifold is created. Then, it is connected to the k

Algorithm 2 CBiRRT(Q_s, Q_g)

T_a.Init(Q_s); T_b.Init(Q_g)

while TimeRemaining() **do**
  $q_{sample} \leftarrow$ SampleValidJoints()
  $q^a_{near} \leftarrow T_a$.Nearest($q_{sample}$)
  $q^a_{reached} \leftarrow$ ConstrainedExtend($T_a, q^a_{near}, q_{sample}$)
  $q^b_{near} \leftarrow T_b$.Nearest($q^a_{reached}$)
  $q^b_{reached} \leftarrow$ ConstrainedExtend($T_b, q^b_{near}, q^a_{reached}$)
  if $q^a_{reached} = q^b_{reached}$ **then**
    $P \leftarrow$ ExtractPath($T_a, q^a_{reached}, T_b, q^b_{reached}$)
    return SmoothPath(P)
  else
    Swap(T_a, T_b)
  end if
end while
return NULL

nearest neighbors less than a predetermined distance. Configurations and connections are added to the graph until the number of nodes in graph reaches the limitation. During query stage, the start and goal configurations are connected to the graph. Then, the naive A* searching algorithm is implemented to find the shortest path along the edges in the graph.

2.4 OBPRM

According to [2], the major difference between OBPRM and PRM is that the OBPRM would sample the nodes on or near the obstacles to better find feasible path in narrow configuration space. If the randomly-sampled configuration q_{sample} is in collision, then another configuration is re-sampled in the vicinity of the collision to find safe configuration near obstacle. Other than the sampling strategy, the other parts of pre-process and query stage are the same as the PRM.

2.5 End-Effector Pose Constraint

In order to meet pose constraints, we employ a gradient-descent projection method based on the Jacobian-pseudo inverse which is same as that used in [1]. The GetJacobian function returns the Jacobian of the manipulator with the rotational part of the Jacobian in the RPY convention.

Algorithm 3 ConstraintProjection

Require: T, q_{target}

while True **do**
  $\Delta x \leftarrow$ DisplacementFromConstraint(q_s, C)
  if $\|\Delta x\| < \epsilon$ **then return** q_s
  end if
  $\mathbf{J} \leftarrow$ GetJacobian(q_s)
  $\Delta q_{error} \leftarrow \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\Delta x$
  $q_s \leftarrow (q_s - \Delta q_{error})$
  if $|q_s - qs^{old}| > 2\Delta q_{step}$ **or** OutsideJointLimit(q_s) **then**
    return NULL
  end if
end while

2.6 Path Smoothing

The smoothPath functions repeatedly pick two random points from the generated path and check if there's a shortcut. Since constrainedExtainded function is used for each check, its guaranteed that the constraints will be satisfied along the line.

Algorithm 4 SmoothPath(P)

while TimeRemaining() **do**
  $T_{shortcut} \leftarrow \{\}$
  $i \leftarrow$ RandomInt(1, $P.size - 1$)
  $j \leftarrow$ RandomInt(i, $P.size$)
  $q_{reached} \leftarrow$ ConstrainedExtend($T_{shortcut}, P_i, P_j$)
  if $q_{reached} = P.j$ **and** Length($T_{shortcut}$) < Length($P_i \cdots P_j$) **then**
    $P \leftarrow [P_1 \cdots P_i, T_{shortcut}, P_{j+1} \cdots P.size]$
  end if
end while
return P

3 Experiment Analysis

We designed three problems for the robot arm, in which it needs to move from a start position to a given goal position while avoiding obstacles. For problem 1, the robot starts at the left of a vertical obstacle, and its goal is on the other side of the obstacle. For problem 2, the robot arm starts under a table-like obstacle, and it is supposed to move its way up to the top of the table. Problem 3 is similar to problem 2, except that the sizes of the obstacles and safe zone are bigger, simulating a narrow passage condition. On the high level, problem 1 is the easiest, while problem 3 is the most challenging one.

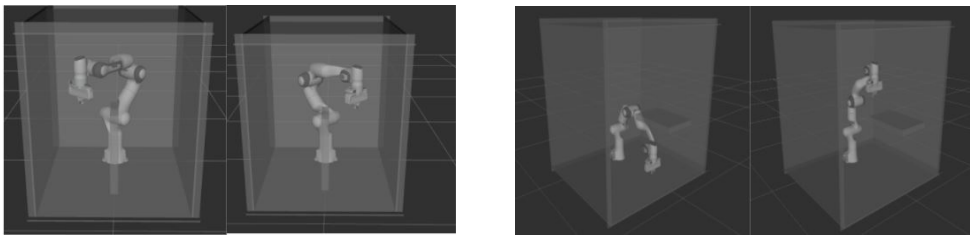

Figure 1: Problem 1 (left) and Problem 2 (right)

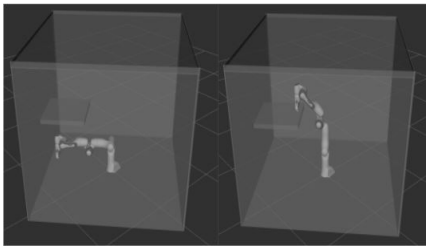

Figure 2: Problem 3

We run over all four algorithms on each problem and record the time it took, the number of nodes expanded and the total path quality listed in the table 1 below, and we would like to highlight three findings by analyzing these results.

Firstly, the shorter runtime, number of nodes and path qualities of CBiRRT, in comparison to RRT, demonstrate CBiRRT's finer performance over RRT under the condition of up-right end-effector constraint, which agrees with the performance comparison between standard RRT-Connect with RRT. In fact, RRT was not even able to find a solution within the node limitation for scene 3, which is the most challenging scene.

	Scene 1 (Vertical Obstacle)			Scene 2 (Small Table)			Scene 3 (Big Table)		
	Time (s)	# of Nodes	Path Distance	Time (s)	# of Nodes	Path Distance	Time (s)	# of Nodes	Path Distance
RRT	59.95	1181	3.89	165.12	2441	4.32	N/A	>100000	N/A
CBiRRT	16.05	405	2.03	38.04	650	3.42	74.34	361	6.24
PRM	103.51 (62118)	100000	2.38	92.62 (99485)	150000	3.35	76.96 (98593)	150000	4.56
OBPRM	136.17 (26512)	20000	2.22	60.37 (65079)	150000	3.15	94.19 (54778)	150000	3.79

Table 1: Comparison of planning algorithms in different scenes

Secondly, although RRT/CBiRRT tends to generate longer path compared to PRM/OBPRM, RRT/CBiRRT has the advantage of less parameters of tuning and zero pre-processing time. As a result, such algorithms require less memory and processing power and are able to take dynamic obstacles into account. On the other hand, although having a enormous pre-processing time (>15hr), the PRM family has the advantage that its graphs can be re-used for subsequent queries and is able to utilize searching algorithms that generate optimal solutions such as A*.

Lastly, comparing with PRM, OBPRM generates solutions with fewer nodes and smaller path qualities, which correlates closely to our expectations, as OBPRM tends to generate nodes in the vicinity of obstacles.

4 Conclusion

This paper simulates and compares several sampling-based path planning techniques for a 7 DOF robot arm. Several metrics have been defined and evaluated for each technique in order to compare and contrast each of them for different scenarios. As future work, it is expected to continue testing these algorithms with more challenging environments (additional obstacles) and with additional constraints (restricting one of the joint angle).

5 Links

Demo Video: https://www.youtube.com/watch?v=d0d5chidEpo
Code: https://github.com/aaronzguan/Sampling_based_Planning_for_Robot_Arm

References

[1] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *2009 IEEE International Conference on Robotics and Automation*, pp. 625–632, 2009.

[2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Obprm: an obstacle-based prm for 3d workspaces," 1998.