*Machine Intelligence and Digital Applications*
*L. Trajković et al. (Eds.)*

*doi:10.3233/ATDE250696*

# A Sampling-Based Motion Planning Strategy for Robotic Manipulators in Highly Dynamic Workspaces

Cheng YAN[a], Jie ZHANG[a,1], Min LV[a], Jingyi GU[a] and Khalid YAHYA[b]
[a] *School of Automation, Nanjing University of Science and Technology Jiangsu, Nanjing 210094*
[b] *Istanbaul Gelisim University, Istanbul, Turkey.*

**Abstract.** With the increasing deployment of collaborative robots, motion planning in shared human-robot workspaces has gained growing theoretical and practical significance. This paper focuses on the development of a trajectory planning algorithm for robotic arms under environmental constraints. Aiming at the problems caused by traditional trajectory planning algorithms, such as slow convergence speed and the lack of the consideration of obstacles along the path of the robotic arm, a dynamic obstacle avoidance trajectory planning algorithm based on the configuration and kinematic model of the robotic arm is proposed, which enables the robotic arm to dynamically plan a smooth trajectory from the start node to the goal node while avoiding encountering obstacles in complex constrained environments. Based on the results of the Informed RRT* algorithm, the algorithm integrates single-step path optimization, path refinement and quintic polynomial trajectory planning process to realize the shortest path planning and smooth trajectory generation while avoiding encountering obstacles. Simulation results on a 6-degree-of-freedom robotic arm validate the effectiveness of the proposed algorithm in obstacle avoidance and trajectory planning within constrained environments in three-dimensional space. Compared to traditional algorithms, the proposed algorithm demonstrates faster convergence speed and more comprehensive obstacle avoidance capabilities for the whole robotic arm.

**Keywords.** Motion planning, inverse kinematics, motion constraint, robotic arms, trajectory planning

## 1. Introduction

The constrained environment refers to a static or dynamic workspace in which one or more obstacles impose restrictions on the movement of a robotic manipulator.In such complex operational environments,the presence of uncertainties—such as randomness and unpredictability arising from environmental changes—poses significant challenges to motion planning [1-4]. The core problem of motion planning in these environments can be formulated as follows:given a robot's kinematic parameters and incorporating real-time updates of environmental constraints,the goal is to generate a kinematic trajectory from an initial state to a target state.In practice,a robot's workspace often contains obstacles that obstruct task execution,and the essence of motion planning lies

[1] Corresponding Author: Jie Zhang, zhangjienjust@163.com

in devising a feasible trajectory from the initial pose to the goal pose while ensuring safe task execution in high-dimensional configuration spaces under dynamically changing, unstructured environments [5-7]. Among the numerous approaches, sampling-based algorithms [8-9] emerged as a prominent direction in global path planning within such constrained spaces.

In recent years, extensive research has been devoted to motion planning for robotic manipulators in constrained environments.To address the challenge of high computational complexity,Gammell et al. [10-11] proposed the Batch Informed Trees (BIT*) algorithm,which incorporates principles from graph theory and improves upon the traditional Informed RRT*. Specifically, BIT* generates a batch of samples within an elliptical sampling domain and performs tree pruning, thereby substantially increasing search efficiency compared to pruning individual nodes.

Building on these foundations, the Gammell research group proposed Adaptively Informed Trees (AIT*) [12] and Effort Informed Trees (EIT*) [13], both of which demonstrate superior adaptability and convergence speed when optimizing more complex cost functions.Subsequently,they introduced the Effort Informed Roadmaps (EIRM*) [14] algorithm,which employs asymmetric bidirectional search to identify previously validated edges in the roadmap. These known-valid segments are then prioritized to reduce computation and accelerate the discovery of viable paths connecting start and goal configurations. More recently, the Task and Motion Informed Trees (TMIT*) algorithm [15], also from Gammell's group, represents a task planning framework that combines near-optimal task scheduling with asymptotically optimal motion planning. By interleaving asymmetric forward and backward searches, TMIT* defers high-cost operations until necessary and performs efficient, informed search directly within the hybrid task-motion state space.

In this paper, we propose an optimization strategy based on the Informed RRT* framework to reduce the motion path length of robotic manipulators. First, a single-step path refinement approach is introduced to eliminate redundant path segments in the initially generated trajectory. Then,new nodes are generated along the planned path using smaller step sizes, and redundant nodes in the motion process are pruned,further minimizing the end-effector displacement. Finally, comparative experiments are conducted to evaluate the motion planning performance of the proposed algorithm against existing state-of-the-art methods.

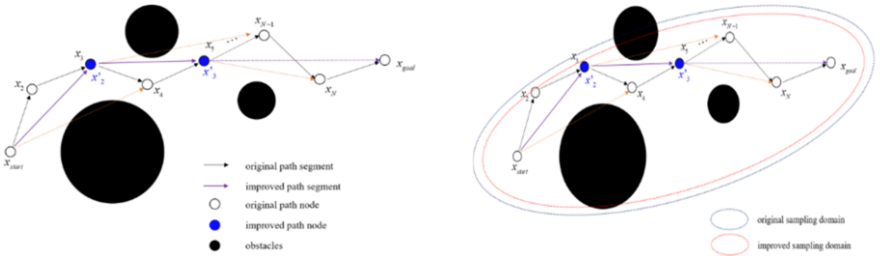
## 2. Single-Step Path Optimization

Through the Informed RRT* algorithm, a feasible path for the end-effector of a robotic arm can be obtained by setting the initial and goal joint angles for each joint of the robotic arm. However, the convergence speed of the Informed RRT* algorithm is related to the distance between the start and goal nodes as well as the length of the feasible path [16-17]. To accelerate the algorithm's convergence speed, redundant path nodes can be removed to shorten the path length by revisiting the path nodes on the current path. During the above process, the optimization goal is to minimize the total distance of the feasible path of the end-effector of the robotic arm, as defined in Equation1.In Equation, the number of the path nodes on the current path,including the start and goal nodes, is N+1,and the node $(x_i, y_i, z_i)$ represents the spatial coordinates of the end-effector of the robotic arm at the i-th path node on the path. These coordinates are derived from the

forward kinematics of the robotic arm through the joint angle information at the i-th path node.

$$c = min \sum_{i=1}^{N} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \quad (1)$$

The process of the single-step path optimization is as follows:




(a) Process (b) Sampling domain.

**Figure 1.** Schematic diagram of single-step path optimization process.

First of all, the start node is set as the departure node,and collision detection is performed between the robotic arm and the obstacles for each remaining node in the current path. For the i-th path node,the joint angle of the robotic arm at this situation is denoted as $q_i$, and the spatial coordinate of the end-effector of the robotic arm $(x_i, y_i, z_i)$ is obtained through calculating the forward kinematics with the knowledge of joint angles. That is to say, the collision detection is performed between the robotic arm and obstacles by checking the distance between the obstacles and each joint and link of the robotic arm for collisions. If there is no collision occur not only between the joints, links and the obstacles, but during the motion between the two adjacent path nodes, the next path node is selected and its collision status is checked. If a collision is detected at a particular path node, then the information of the joint angles and end-effector's spatial coordinate of the previous path node are added to the new path. By calculating the difference between the new path node and the spatial coordinate of the end-effector of the robotic arm at the previous path node in the new path, the spatial distance between the two path nodes is obtained, and the total path length of the new path is updated. Afterwards, the previous path node in the optimized path is set as the departure node, and the process mentioned above continues until all nodes in the original path have been traversed and the optimized path is obtained. Through this process, redundant nodes in the original path are removed, and a shorter path is found. The process is shown in Figure 1. To simplify, the end-effector of the robotic arm is modeled as a node. In Figure 1(a), the process of removing redundant nodes from the current path to generate a new path is clearly illustrated. The black hollow circles and arrows represent the original path nodes and the original path, respectively, while the blue solid circles and purple arrows represent the improved path nodes and the new path, respectively. As shown in Figure 1(a), the node $x_{start}$, $x_3$, and $x_3$, $x_5$ can be directly connected through a straight line without collision, thus reducing the total path length. Figure 1(b) compares the size of the elliptical sampling domain before and after the single-step path optimization. The dark blue ellipse represents the sampling domain corresponding to the original path, and the red ellipse represents the sampling domain corresponding to the optimized path. This shows that the single-step path optimization reduces the path length and shrinks the elliptical sampling domain, thereby accelerating the convergence speed of the algorithm.

**Algorithm 1:** Single-Step Path Refinement.

Require: robotic arm model robot, original path path
$q_{start} \leftarrow$ getNode(***path***,1);
$x_{start} \leftarrow$ ***robot***.fkine($q_{start}$);
$newPath \leftarrow \{q_{start}\}$;
$pathCost \leftarrow 0$;
$num \leftarrow$ length(***path***);
for $i = 2 : num$ do
$q_{new} \leftarrow$ getNode(***path***, $i-1$);
$q_{goal} \leftarrow$ getNode(***path***, $i$);
$x_{new} \leftarrow$ ***robot***.fkine($q_{new}$);
if checkCollision($q_{start}, q_{goal}$) then
$newPath \leftarrow newPath \cup \{q_{new}\}$;
$pathCost \leftarrow pathCost + \text{dis}(x_{start}, x_{new})$;
$q_{start} \leftarrow q_{new}$;
$x_{start} \leftarrow x_{new}$;
end if
if $i == num$ then
$newPath \leftarrow newPath \cup \{q_{goal}\}$;
$x_{goal} \leftarrow$ ***robot***.fkine($q_{goal}$);
$pathCost \leftarrow pathCost + \text{dis}(x_{start}, x_{goal})$;
end if
end for
return *newPath*, *pathCost*

## 3. Path Refinement

The process of single-step path optimization can effectively reduce the path length and accelerate the convergence speed of the algorithm. However, in single-step path optimization, the optimization targeted at the path nodes obtained through original search process [18-19]. Since the single-step path optimization only starts from the existing path nodes, it may still result in some path redundancy, leaving room for further optimization. After completing the single-step path optimization, the path refinement process is executed to further refine the resulting path, thereby shortening the path length and accelerating the algorithm's convergence speed.

The process of path refinement is as follows: For the path obtained through single-step path optimization, first of all, a series of path nodes are obtained between adjacent path nodes by sampling path nodes along the segment between their corresponding spatial coordinates with a fixed step size, and then named the new set of path nodes as V. Then, starting from the start node, inverse kinematics calculations are performed for the

remaining nodes in the set V to obtain the corresponding joint angles of the robotic arm. Based on the calculated joint angles, a collision check is performed to determine whether any collisions occur between the robotic arm's joints, links, and the motion process between two consecutive path nodes and each obstacle at the current situation. If no collisions are detected between both the robotic arm and links and obstacles, the next path node in set V is selected for a collision check. If a collision occurs at a certain path node, the previous path node of the collided node is added to the new path, and the length of the refined path is calculated. Then,in the next step, the added path node is set as the new start node, and collision checks are performed for all subsequent node in set V until all nodes are traversed, resulting in a refined path. Through this process, a shorter path can be further found from the path obtained through single-step optimization.

**Algorithm 2:** path refinement.

**Require:** robotic arm model **robot**, original path **path**, fixed step size **stepSize**

$q_{start}$ ← getNode(path,1);
$x_{start}$ ← robot.fkine($q_{start}$);
newPath ← {$q_{start}$};
V ← {$x_{start}$};
pathCost ← 0;
num ← length(path);
newNum ← 1;
for i = 2:num do
  $x_{old}$ = robot.fkine(getNode(path,i-1);
  $x_{now}$ = robot.fkine(getNode(path,i);
  pathDis ← dis($x_{old}$, $x_{now}$);
  count ← fix(pathDis / stepSize);
  while count > 0 do
    newNum ← newNum + 1;
    dir ←$x_{now}$ - $x_{old}$;
    unitDir ← dir / norm(dir);
    grove ← stepSize * unitDir;
    $x_{new}$ ← createNode(V, grove, newNum);
    V ← V ∪ {$x_{new}$};
    count ← count - 1;
  end while
  newNum ← newNum + 1;
  V ← V ∪ {robot.fkine(getNode(path,i)};
end for
Num ← length(V);
for i = 2:Num do
  $q_{new}$ ← robot.ikine(getNode(V,i-1));
  $q_{goal}$ ← robot.ikine(getNode(V,i));
  $x_{new}$ ← getNode(V,i-1);
  if checkCollision($q_{start}$, $q_{goal}$) do
    newPath ← newPath ∪ {$q_{new}$};
    pathCost ← pathCost + dis($x_{start}$, $x_{new}$);
    $q_{start}$ ← $q_{new}$;
    $x_{start}$ ← $x_{new}$;
  end if
  if i == num do
    newPath ← newPath ∪ {$q_{goal}$};
    $x_{goal}$ ← getNode(V,i);
    pathCost ← pathCost + dis($x_{start}$, $x_{goal}$);
  end if
end for
return newPath, pathCost

The process is shown in Figure 2. To simplify, only the nodes corresponding to the end-effector of the robotic arm are shown in the figure. In Figure 2, yellow solid circles represent the path nodes generated with a fixed step size along the path obtained from single-step optimization, while green solid circles and red arrows represent the path nodes and the path obtained through path refinement, respectively. As shown in Figure 2, new path nodes can further eliminate path redundancy on top of the path obtained through single-step optimization process, thus shortening the total path length.

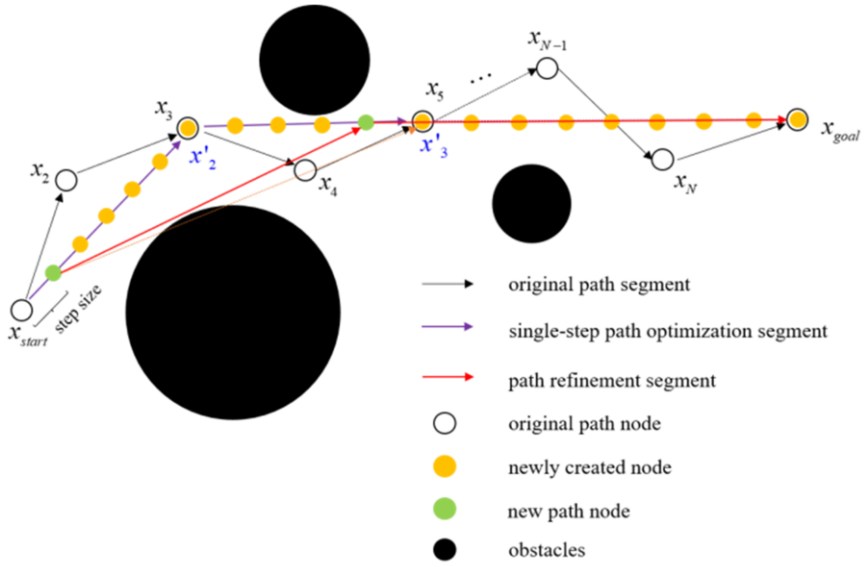



**Figure 2.** Schematic diagram of path refinement process.

## 4. Experiment Results

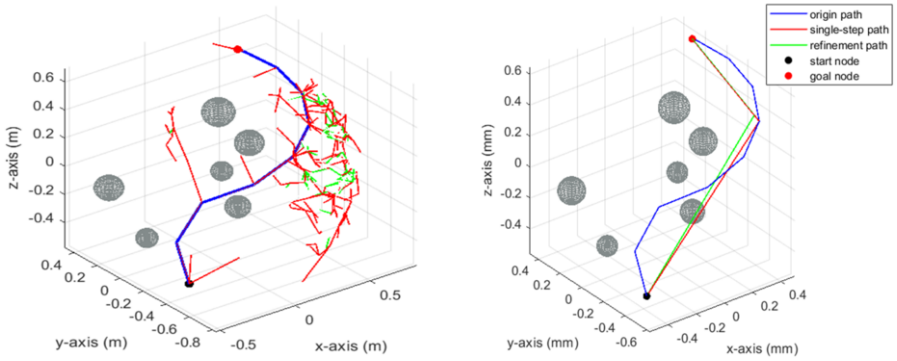
Experiments in this paper is based on Matlab R2020b.

In this paper, spheres are used to simulate obstacles. Six spheres of different sizes are placed at different locations to represent obstacles with varying sizes randomly distributed in the space. The center coordinates of the spheres are: [0.1, -0.2, -0.2], [-0.3, 0.4, -0.2], [0.2, 0.1, 0.3], [-0.2, 0.2, -0.5], [0.3, 0.2, -0.2], and [0.1, -0.3, 0.3], respectively, with all coordinates measured in meters. The radii of the spheres are 0.07m, 0.08m, 0.09m, 0.06m, 0.06m, and 0.08m, respectively. The robot arm's Denavit-Hartenberg (D-H) parameters are given in Table1. The initial joint configuration of the robotic arm is [-2.0944, -0.8727, -0.7854, 0.2618, 0.3142, 0.6458] and the goal configuration is [0.7854, 0.5236, -0.5236, 0.1396, 0.2094, 1.3963], with all values in radians. The initial sampling domain is a six-dimensional cube within a six-dimensional space of length $\pi$. The threshold radius for the goal node is set to 0.15m, with a 0.2 probability for sampling the goal node as a sample node. The initial step size during the path generation of the original algorithm is set to 0.5m. During the path refinement process, the step size is fixed at 0.05m, and the total number of samples in the improved algorithm is 300.

Table 1. D-H parameters for the robotic arm.

| Joint | $\theta$ | d | a | $\alpha$ |
|---|---|---|---|---|
| 1 | 1.5708 | 0 | 0 | 1.5708 |
| 2 | 0 | 0 | 0.4318 | 0 |
| 3 | -1.5708 | 0.15005 | 0.0203 | -1.5708 |
| 4 | 0 | 0.4318 | 0 | 1.5708 |
| 5 | 0 | 0 | 0 | -1.5708 |
| 6 | 0 | 0 | 0 | 0 |

In a single experiment, an initial blue path represents the path obtained by using the Informed RRT* algorithm, while the red segments represent the edges of the tree, and the spheres represent the obstacles, as shown in Figure 3(a). Based on this initial path, an ellipsoidal sampling domain can be derived, within which random sampling is performed to obtain sampling nodes, resulting in the final blue path. This path is then used for the subsequent single-step path optimization process, where the green segments represent the edges of the tree formed during this sampling process. After obtaining the initial blue path, the single-step path optimization is performed to obtain the optimized path, and based on this, a path refinement process is carried out. The final result shows the initial path (blue line), the path after single-step optimization (green line), and the path after refinement (red line), as shown in Figure 3(b).




(a) Search process of the algorithm (b) Paths for algorithms

**Figure 3.** Schematic diagram of experiment results for improved Informed RRT* algorithm.

In this paper, a total of 20 experiments were conducted using both the original Informed RRT* algorithm and the improved Informed RRT* algorithm. The path lengths obtained after applying the original Informed RRT* algorithm, the single-step path optimization process, and the path refinement process were recorded. In each experiment, the path lengths obtained using the three algorithms are shown in Figure 4. The average path lengths obtained using Alg1 and Alg2 are shorter than that of the original algorithm, demonstrating the effectiveness of the proposed algorithm.

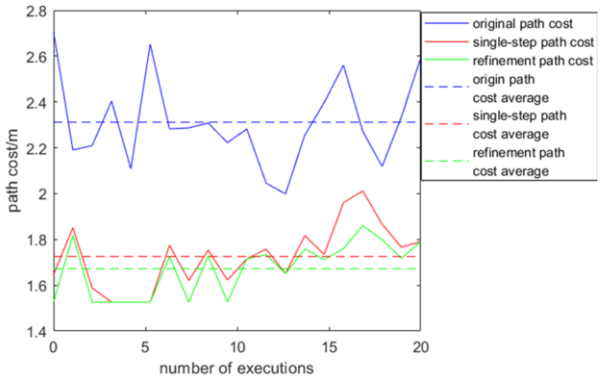
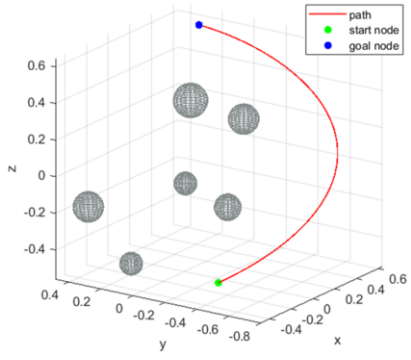


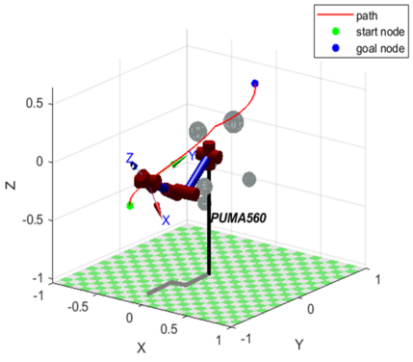

**Figure 4.** Schematic diagram of path lengths for multiple experiments.

The optimized path after trajectory optimization and the schematic diagram of the robotic arm moving along the optimized path are shown in Figure 5. The optimized path smooths the robotic arm's movement trajectory, making it more consistent with the actual operation of the robotic arm, ensuring stable motion of the robotic arm. Furthermore, Figure 5 can be observed that when the joints of the robotic arm are in the closest proximity to the obstacles, no collision occurs between the robotic arm and the obstacles. This indicates that the trajectory optimization process not only smooths the path but also prevents collisions along the optimized path. During this process, the variations in angles, angular velocities, and angular accelerations of the robotic arm are shown in Figure 6. As seen in the figure, the trajectory optimization process ensures the smoothness of the robotic arm's motion.

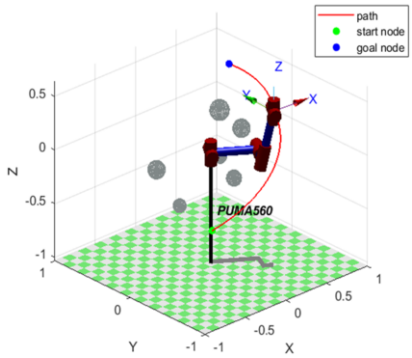
After completing 20 experiments, the path lengths obtained through the original Informed RRT* algorithm and the improved algorithms were recorded. The average path lengths obtained through the original Informed RRT* algorithm, the single-step path optimization process, and the path refinement process were 2.3115m, 1.7253m, and 1.6726m, respectively. Compared to the path length obtained through the original Informed RRT* algorithm, the path lengths of the single-step path optimization process and the path refinement process were reduced by 25.36% and 27.64%, respectively. Compared to the path length obtained through the single-step path optimization process, the path length obtained through the path refinement process was reduced by 3.05%. Therefore, the single-step path optimization process can effectively reduce the path length, and the path refinement process also significantly reduces the path length in the single-step path optimization process.



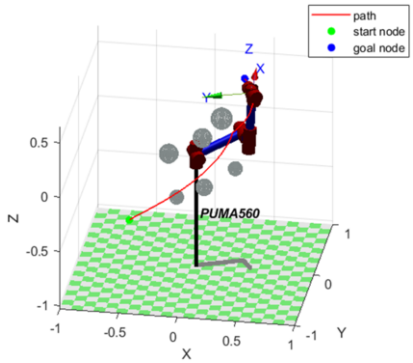

(a) Smoothed path




(b) The state when the robotic arm is closest to the obstacle at the beginning of the path




(c) The state when the robotic arm is closest to the obstacle in the middle of the path




(d) The state when the robotic arm is closest to the obstacle at the end of the path

**Figure 5.** Schematic diagram of path optimization and robotic arm motion.

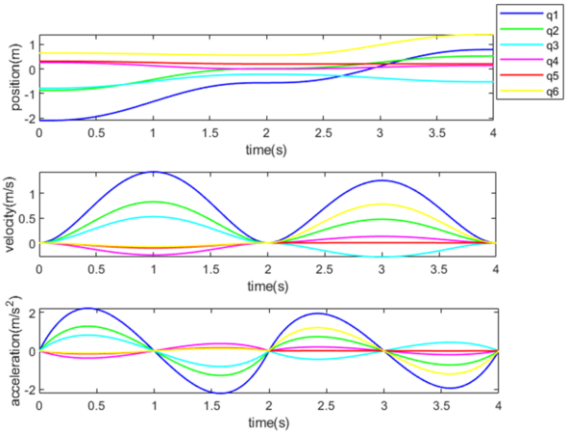



**Figure 6.** Schematic diagram of variations in joint status of the robotic arm.

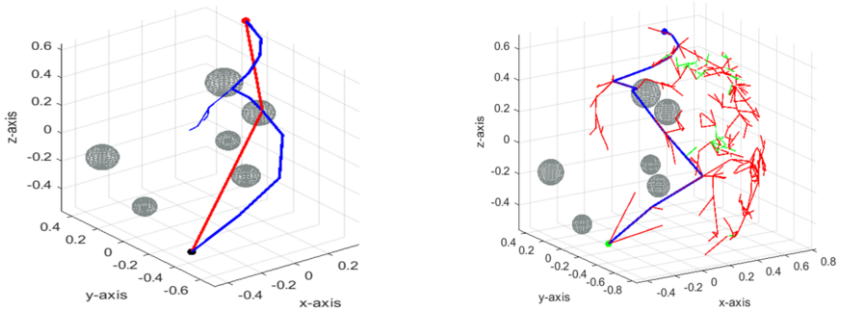
To validate the effectiveness of the proposed algorithm, obstacle avoidance planning for a robotic arm was implemented using the RRT, RRT-Smooth, Informed-RRT* algorithm together with the proposed algorithm under identical experimental

environments and conditions. A total of 50 experiments were performed for each algorithm to compare their performance, and the results are presented in Table2. The comparison among different algorithms was conducted based on the following performance metrics: mean initial path length ($C_{i_{mean}}$), best initial path length ($C_{i_{best}}$), mean initial path time ($T_i$), mean final path length ($C_{f_{mean}}$), best final path length ($C_{f_{best}}$), and mean termination time ($T_f$).

For the RRT and RRT-Smooth algorithms, execution terminates immediately after obtaining an initial feasible path; hence, the last three metrics are not applicable to these algorithms. In contrast, both the Informed-RRT* algorithm and the proposed algorithm continue execution beyond the initial feasible path generation until reaching a maximum iteration count of 300. The execution results of RRT and RRT-Smooth are illustrated in Figure 7(a), where the thick blue solid line represents the execution outcome of RRT, and the thick red solid line corresponds to that of RRT-Smooth. The execution result of the Informed-RRT* algorithm is depicted in Figure 7(b), where the thick blue solid line represents the final obtained path.

**Table 2.** Comparison of algorithm performance.

| Algorithm | $C_{i_{mean}}(m)$ | $C_{i_{best}}(m)$ | $T_i(s)$ | $C_{f_{mean}}(m)$ | $C_{f_{best}}(m)$ | $T_f(s)$ |
|---|---|---|---|---|---|---|
| RRT | 2.1961 | 1.926 | 1.2846 | / | / | / |
| RRT-Smooth | 1.7331 | 1.57 | 1.3234 | / | / | / |
| Informed-RRT* | 2.225 | 1.9574 | 5.8776 | 2.2062 | 1.9446 | 406.6682 |
| Single-step path optimization | 1.7286 | 1.5911 | 5.8895 | 1.7201 | 1.5775 | 406.6788 |
| path refinement | 1.6567 | 1.5264 | 10.6079 | 1.645 | 1.5264 | 410.9429 |




(a) RRT and RRT-Smooth (b) Informed RRT*

**Figure 7.** Schematic diagram of results of different algorithms.

For this series of algorithms, the average search time and average length of the initial path largely depend on the positions of randomly generated nodes and the sampling region. As a result, the initially feasible paths obtained through the search process are not optimal solutions.

In the RRT algorithm, nodes are randomly generated across the entire sampling space without further optimization, leading to significant path redundancy. However, this algorithm enables the robotic arm to quickly obtain a feasible path.

The RRT-Smooth algorithm improves upon the path generated by the RRT algorithm by reducing path redundancy to some extent. This allows it to obtain a path with less redundancy compared to RRT while maintaining a relatively high speed.

The Informed-RRT* algorithm, after obtaining an initial feasible path, constructs an elliptical region based on the current path and continues sampling within this region, and then continuously adjust the range of the elliptical region based on the newly generated

path. This reduces the region for random node generation. Additionally, the algorithm optimizes the path by comparing the newly generated path with the original path, effectively reducing path redundancy. However, due to the randomness of new node generation and the size of the elliptical region, the path convergence speed is relatively slow.

The Single-Step path optimization algorithm builds upon the Informed-RRT* algorithm by searching among existing nodes of the path and removing redundant nodes, thus obtaining a shorter path.

The path refinement algorithm further subdivides the path based on the optimized results, by eliminating redundant segments from the subdivided path, thus reducing path redundancy at the cost of additional computation time.

## References

[1] L. Jia, Y. Huang, T. Chen, Y. Guo, Y. Yin, and J. Chen, "Mda+rrt: A general approach for resolving the problem of angle constraint for hyper-redundant manipulator," Expert Systems with Applications, vol.193, p.116379, 2022.

[2] X. Yin, P. Cai, K. Zhao, Y. Zhang, Q. Zhou, and D. Yao, "Dynamic path planning of agv based on kinematical constraint a* algorithm and following dwa fusion algorithms," Sensors, vol.23, no.8, 2023.

[3] K. Nonoyama, Z. Liu, T. Fujiwara, M. M. Alam, and T. Nishi, "Energy-efficient robot configuration and motion planning using genetic algorithm and particle swarm optimization," Energies, vol.15, no.6, 2022.

[4] L. Jiang, S. Liu, Y. Cui, and H. Jiang, "Path planning for robotic manipulator in complex multi-obstacle environment based on improved rrt," IEEE/ASME Transactions on Mechatronics, vol.27, no.6, pp.4774-4785, 2022.

[5] X. Zhou, X. Wang, Z. Xie, F. Li, and X. Gu, "Online obstacle avoidance path planning and application for arc welding robot," Robotics and Computer-Integrated Manufacturing, vol.78, p.102413, 2022.

[6] L. Jia, Y. Huang, T. Chen, Y. Guo, Y. Yin, and J. Chen, "Mda+rrt: A general approach for resolving the problem of angle constraint for hyper-redundant manipulator," Expert Systems with Applications, vol.193, p.116379, 2022.

[7] A. Sinha and N. Chakraborty, "Geometric search-based inverse kinematics of 7-dof redundant manipulator with multiple joint offsets,"in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp.5592-5598.

[8] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE transactions on Robotics and Automation, vol.12, no.4, pp.566-580, 1996.

[9] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Research Report 9811, 1998.

[10] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (bit*):Informed asymptotically optimal anytime search," The International Journal of Robotics Research, vol.39, no.5, pp.543-567, 2020.

[11] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in 2015 IEEE international conference on robotics and automation (ICRA). IEEE, 2015, pp.3067-3074.

[12] M. P. Strub and J. D. Gammell, "Adaptively informed trees (ait*): Fast asymptotically optimal path planning through adaptive heuristics," in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp.3191-3198.

[13] M. P. Strub and J. D. Gammell. Adaptively informed trees (AIT*) and effort informed trees (EIT*): Asymmetric bidirectional sampling-based path planning. The International Journal of Robotics Research, 2022, 41(4): 390-417.

[14] V. N. Hartmann, M. P. Strub, M. Toussaint, and J. D. Gammell, "Effort informed roadmaps (eirm*): Efficient asymptotically optimal multiquery planning by actively reusing validation effort," in The International Symposium of Robotics Research. Springer, 2022, pp.555-571.

[15] W. Thomason, M. P. Strub, and J. D. Gammell, "Task and motion informed trees (tmit*): Almost-surely asymptotically optimal integrated task and motion planning," IEEE Robotics and Automation Letters, vol.7, no.4, pp.11 370-11 377, 2022.

[16] X. Cheng, J. Zhou, Z. Zhou, X. Zhao, J. Gao, and T. Qiao, "An improved rrt-connect path planning algorithm of robotic arm for automatic sampling of exhaust emission detection in industry 4.0," Journal of Industrial Information Integration, vol.33, p.100436, 2023.
[17] B. H. Meng, I. S. Godage, and I. Kanj, "Rrt*-based path planning for continuum arms," IEEE Robotics and Automation Letters, vol.7, no.3, pp.6830-6837, 2022.
[18] M. P. Strub and J. D. Gammell, "Adaptively informed trees (ait*) and effort informed trees (eit*): Asymmetric bidirectional sampling-based path planning," The International Journal of Robotics Research, vol.41, no.4, pp.390-417, 2022.
[19] W. Thomason, M. Strub, and J. Gammell, "Task and motion informed trees (tmit*): Almost-surely asymptotically optimal integrated task and motion planning," IEEE Robotics and Automation Letters, vol.7, pp.11370-11377, 102022.