UDC 004.94

REDUNDANT ROBOTIC ARM PATH PLANNING USING RECURSIVE RANDOM INTERMEDIATE STATE ALGORITHM

Medvid A. Y. – Post-graduate student of the Department of Artificial Intelligence Systems, Lviv Polytechnic National University, Lviv, Ukraine.

Yakovyna V. S. – Dr. Sc., Professor of the Department of Artificial Intelligence Systems, Lviv Polytechnic National University, Lviv, Ukraine.

ABSTRACT

Context. Collision-free path planning in joint space for redundant robotic manipulators remains a challenging task due to the high-dimensional configuration space and dynamically changing environments. Existing methods often struggle to balance search time and path quality, which is crucial for real-time applications.

Objective. The aim of this study is to develop a new method to plan efficient, collision-free trajectories in real time for redundant robotic manipulators.

Method. A novel sampling-based algorithm for collision-free joint space path planning for redundant robotic manipulators presented in this study. The algorithm is called the Recursive Random Intermediate State (RRIS). The RRIS algorithm primarily works by generating a set of random intermediate states and iteratively selecting the optimal one based on the number of collisions along the discretized path. Furthermore, the paper proposes an axis-aligned bounding box generation strategy and an early exit strategy to improve algorithm speed. Finally, repeated calls of the algorithm are proposed to improve its reliability. The performance of the RRIS algorithm is evaluated through a set of comprehensive tests and compared with the popular RRT Connect algorithm implemented in Open Motion Planning Library.

Results. Experimental evaluations show that the RRIS algorithm under the test conditions produces collision-free paths with significantly shorter average lengths and reduces search time by approximately three times compared to the RRT Connect algorithm.

Conclusions. The proposed RRIS algorithm demonstrates a promising approach to real-time path planning for redundant robotic manipulators. By combining strategic intermediate state sampling with efficient collision evaluation and early termination mechanisms, the algorithm offers a robust alternative to known methods.

KEYWORDS: path planning, redundant robotic manipulator, collision avoidance.

ABBREVIATIONS

RRIS is a Recursive Random Intermediate State;

RRT is a Rapidly-Exploring Random Trees;

OMPL is an Open Motion Planning Library [17];

DOF is degrees of freedom.

NOMENCLATURE

J^i is the angle of the i-th joint of robotic arm;

S is a state of the robotic arm;

S_i is the i-th state of discretized path between arm states;

J_i^j is the angle of j-th joint of an i-th state of discretized path between arm states;

$|Path|$ is a norm of the path between two arm states;

S_S is a start arm state;

S_F is a final arm state;

S_I is an intermediate arm state;

S_M is a middle state between start and final;

d is a step of displacement of the middle state in the joint space for generating intermediate states;

Dir_i is the step of displacement direction being made for the i-th joint (can take values –1, 0, 1);

BB is a bounding box in joints space;

M_{BB} is a bounding box margin;

N_r is a number of random intermediate states generated;

$step$ is a discretization step in radians used to trace between two arm states;

$Collisions_{S->F}$ is a number of collisions on discretized path between start and final states;

$Collisions_{S->I}$ is a number of collisions on discretized path between start and intermediate states;

$Collisions_{I->F}$ is a number of collisions on discretized path between intermediate and final states;

$Path_{S->I}$ is a path between start and intermediate states;

$Path_{I->F}$ is a path between intermediate and final states;

$States_{Collides}$ is a list of states of discretized path between the start state and the final state which has collisions;

$Collisions_{Best}$ is a current minimal number of collisions on the discretised path from start state to intermediate and from intermediate state to final;

$Collisions_{EE}$ is a thershold number of collisions used to check early exit condition during selecting intermediate state;

$|delta|_{Max}$ is a maximum absolute difference between joint angles in two arm states;

$|delta|_{Manhattan}$ is a sum of absolute differences between joint angles in two arm states.

INTRODUCTION

Articulated robots are currently used for a variety of automation tasks. Industrial robotic arms with revolute (articulated) joints are widely employed for tasks such as palletizing, material handling, welding, quality inspection, picking and placing objects [1], and many others.

Despite the fact that robotic arms perform different types of tasks, in general, we can summarize the task for a robotic arm: to move to a certain place at a specific time without causing damage to surrounding objects or itself. From this arises the problem of planning a collision-free path for the robotic arm.

DOI 10.15588/1607-3274-2025-3-16

The control of the arm takes place by specifying rotation angles for each joint of the arm, that is, the control program specifies coordinates in the state space of the joints (or simply "joint space"). Meanwhile, the position of the manipulator's end point (the last link of the robotic arm, if counting from the base, also known as the "end effector") is determined in Cartesian space.

An articulated robotic arm requires at least 6 joints to achieve full 6-degree-of-freedom movement in Cartesian space [2]. But despite the fact that a "6-joint arm with 6 independent joints can specify any position and orientation of the manipulator" [3], robotic arms with a larger number of joints (such robotic arms are also called redundant) and degrees of freedom, respectively, are widely used in the industry. The reasons for this are the ability to avoid the problem of singularities in the robot's workspace and the solution to the problem of joint restrictions [3].

The large dimensionality of the state of the robotic arm and the complexity of the inverse kinematics problem (search for arm coordinates in joint space given coordinates in the Cartesian space), as well as the presence of many possible solutions to inverse kinematics due to the redundant joint, make path planning task difficult. There are plenty of parameters to optimize in path planning algorithms while the most important among them are path planning time and success rate. At present, there is no universal path planning algorithm for redundant robotic arms that would guarantee finding the optimal path, or guarantee finding any path if it exists.

The object of study is the process of path planning for redundant robotic manipulators.

The subject of study is the reliability and execution speed of path planning algorithms.

The purpose of the work is to develop a new sampling-based algorithm for path planning for redundant robotic arms that selects an optimal intermediate point based on the number of collisions along the path passing through it.

1 PROBLEM STATEMENT

Before describing the developed algorithm, let's clarify the task it has to solve. Suppose we have a robotic arm operating in an environment that changes its state during the robot's operation. We need to plan the path of the robotic arm from one state in joint space to another state in joint space. The arm should avoid collisions with the surrounding environment along the found path.

The algorithm can use third-party tools for collision checking at a certain state of the arm. The path segment collision checking will be based on discretization of the path with a given step and checking all discrete states for collisions.

The state of the robotic arm is described by a vector of rotation angles for each joint, starting from the arm's fixation point. We can write it down in the next way:

$$S = \left\{ J^0, J^1, ..., J^{DOF-1} \right\} \quad (1)$$

The norm of the difference between two states will be defined as the maximum absolute difference in rotation angles for each joint of the robotic arm:

$$|S_1 - S_2| = MAX\left(J_1^0 - J_2^0, J_1^1 - J_2^1, ..., J_1^{DOF-1} - J_2^{DOF-1}\right) \quad (2)$$

The length of the path is the sum of the norms of the differences between all neighboring states on the path:

$$|Path| = \sum_{i=1}^{N-1} |S_{i+1} - S_i|. \quad (3)$$

We'd like to get the result of the planning as soon as possible, because we are building trajectories real-time. So, the main criteria for evaluating a planning algorithm are the success rate and the average time to build the path. An additional parameter is the length of the found path, so the shorter the path, the better is the solution.

2 REVIEW OF THE LITERATURE

The well-known path planning algorithms for a redundant robotic arm among others include the following algorithms:

– Probabilistic Roadmaps: it's a sampling-based method for path planning where random samples from the configuration space are used to create nodes, which are then connected to create a roadmap [4];

– Rapidly-Exploring Random Trees (RRT): this algorithm is particularly useful for high dimensional spaces and real-time applications [5];

– Artificial Potential Fields: this method treats the robot as a particle moving under the influence of artificial forces. The goal and obstacles generate attractive and repulsive forces, respectively [6];

– Deep Reinforcement Learning based approaches: recent works have proposed learning-based methods for path planning, which can effectively handle redundant manipulators [7].

Among others, worth noting one of the recent works where Khan et al. proposed a model-free kinematic tracking controller for redundant robotic manipulators using Zeroing Neural Networks (ZNN) and Beetle Antennae Search (BAS). The ZNNBAS algorithm avoids traditional Jacobian-based approaches by leveraging a meta-heuristic optimization method in continuous time, eliminating the need for precise kinematic modeling. Tested on a 7-DOF manipulator, it achieved real-time redundancy resolution with minimal tracking errors, demonstrating the potential of hybrid optimization techniques for real-time path planning [8].

The RRT algorithm is the most popular solution today. There are many variations of it. In particular, the following should be mentioned:

RRT-Connect: this variation of RRT makes aggressive attempts to connect the tree directly to the goal, leading to faster solutions [9];

Bidirectional RRT (Bi-RRT): in this method, two RRTs grow towards each other, one from the initial state and the other from the goal. This can be more efficient in some problem spaces [10, 18];

DOI 10.15588/1607-3274-2025-3-16

RRT* (RRT Star): This variant of RRT introduces the idea of an "optimal" path, gradually improving the path quality by selectively rewiring nodes in the tree to minimize total path cost [11].

The disadvantages of the RRT-based algorithms described above can include their lack of evaluation for the currently generated states in the tree. As a result, even if the algorithm has almost found a collision-free path (i.e., one of the tree vertices can reach the target state with minimal collision), it will not attempt to complete the path, but will continue to generate states randomly without additional changes [12].

Ganesan et al. propose Hybrid-RRT, a novel path-planning algorithm that combines uniform and non-uniform sampling to improve the performance of RRT*-based motion planning. The hybrid approach balances exploration and exploitation by dynamically selecting between uniform and goal-directed non-uniform sampling. Experimental results demonstrate that Hybrid-RRT* achieves faster convergence, higher success rates, and reduced node exploration compared to baseline algorithms, including RRT*, Informed RRT*, and RRT*-N. The method is particularly effective in complex environments, addressing limitations of both traditional uniform and non-uniform sampling strategies [13].

One of the algorithms that changes its behavior based on the current state assessment is Informed RRT* [14]. This is a further improvement to RRT*, it takes into account the best current path to guide the sampling process, leading to faster convergence towards an optimal solution. Despite the advantages of this approach, the idea of assessing a specific state is not widely used today.

3 MATERIALS AND METHODS

The main idea of the newly developed Recursive Random Intermediate State (RRIS) algorithm is that a set of random intermediate states is generated. Then we iterate through all of the states, and if a state has collisions, then we need to skip it. If a state has no collisions, then we need to calculate a penalty for this state. This penalty is based on the number of states that have collisions on the discretized path from initial state to intermediate state and from intermediate state to final state.

Among all intermediate states we choose the one with the lowest penalty. Then the task of finding a path from the initial state to the final state is reduced to the task of finding a path from the initial state to the intermediate state and from the intermediate state to the final state. So the algorithm calls itself recursively.

Algorithm will stop current recursion step execution and in one of two cases:

– we found intermediate state, which creates path without collisions;

– we checked all intermediate states and none of them creates a path that is better than a straight one. Which means that the number of states with collisions on discretized path doesn't get smaller on any checked intermediate state.

These conditions may vary depending on the chosen strategy and we will return to this later.

In the end, if both parts of the path are successfully found, then we can build the whole path by merging the path from initial state to intermediate state and path from intermediate state to final state. And if we fail to find a safe path in at least one of the parts, then we fail to find a safe path.

The generation of intermediate states can depend on the specific implementation. In particular, such options can be used:

– select the step of displacement in the joint space based on the length of the direct path, and for each joint consider 3 displacement options: clockwise, counter-clockwise, or zero displacement. Then the intermediate state can be calculated using formula (4):

$$S_I = \left\{ J_M^0 + d \cdot Dir^0, J_M^1 + d \cdot Dir^1, \ldots, J_M^{DOF-1} + d \cdot Dir^{DOF-1} \right\} \quad (4)$$

Thus, we will have $N = 3^{DOF}$ intermediate states;

– build an axis-aligned bounding box around all colliding states on the straight path from initial to final state (as described in Algorithm 4), generate a fixed number of intermediate states randomly and uniformly within the bounding box (as described in Algorithm 5).

Intermediate states generation using a bounding box showed better results as can be seen in Table 3.

To compare paths that goes through different intermediate states, we can minimize the number of collisions in at least two ways:

1. Minimize the total number of collisions on two path parts. In this case, we assume that a smaller total number of collisions means that we will need to expend fewer efforts to avoid collisions in the subsequent steps.

2. Minimize the maximum number of collisions on two path parts. In this case, we believe that even if the number of collisions through the intermediate path increases, but they are both less than the maximum, then we will have to circumvent fewer in each of the two parts of the path, making it easier to bypass them.

Both approaches show good results and it's shown in Table 3.

Depending on the input data and the sequence of generated intermediate states, the algorithm may not get an intermediate state that would lead us to the goal without collisions. However, a sufficiently "good" state, a path through which contains a small number of collisions, may appear among the first. During the recursive descent we often can build a collision-free path through a "good" state quite quickly.

Therefore, instead of always iterating through the entire set of intermediate states, a check for quick exit from the iteration can be introduced. We propose the following condition for early exit: if both parts of the path through the intermediate state have fewer than half collisions of the direct path, then we choose this intermediate state and exit the iteration.

DOI 10.15588/1607-3274-2025-3-16

The use of the early exit strategy significantly improved the algorithm's speed, which is evident in the results section in Table 3.

The algorithm described above, despite its high speed, still has one major drawback. Due to the fact that the algorithm has no backtracking tool, in some cases it will get stuck in local minima. As shown in Table 3 the failure rate of an algorithm on a test set with a single run is about 90 percent. And the most common reason for an algorithm to fail is stucking in local minima.

In Table 3 can be seen that the failure rate of different variations of this algorithm is much higher than that of the RRTConnect algorithm from OMPL [17]. Therefore, we decided to add a simple way to escape from the local minima. Specifically – rerunning the algorithm a certain number of times until a path is found.

In the results section in Table 4 the testing results of the algorithm that initiates the search path up to 5 times in case of failures in previous steps are presented.

The pseudo code description of an optimal version (based on test results) of the RRIS algorithm is described below together with additional algorithms used by main algorithm. Intermediate states generated uniformly random in the axis-aligned bounding box. Intermediate states comparison is based on minimizing the maximum number of collisions on two path parts. And an early exit strategy applied.

Data: S_S, S_F, Algorithm options (M_{BB}, N_r, $step$)
Result: Path from S_S to S_F or failure
$Collisions_{S\to F} \leftarrow$ `CountCollisions`$(S_S, S_F, step, \infty)$
if $Collisions_{S\to F} = 0$ **then**
  return direct path (S_S, S_F) ; `/* Path is collision-free */`
end
$States_{Collides} \leftarrow$ `CollidingStatesList`$(S_S, S_F, step)$;
$BB \leftarrow$ `ComputeBoundingBox`$(States_{Collides}, M_{BB})$;
$States_I \leftarrow$ `GenerateRandomStates`(BB, N_r);
Sort $States_I$ by total path distance (ascending);
$S_{best} \leftarrow$ `NULL`;
$Collisions_{best} \leftarrow Collisions_{S\to F}$;
$Collisions_{EE} \leftarrow Collisions_{S\to F}/2$;
foreach $S_I \in States_I$ **do**
  if S_I *has collisions* **then**
    continue;
  end
  $Collisions_{S\to I} \leftarrow$ `CountCollisions`$(S_S, S_I, step, Collisions_{best})$;
  $Collisions_{I\to F} \leftarrow$ `CountCollisions`$(S_I, S_F, step, Collisions_{best})$;
  if $Collisions_{S\to I} < Collisions_{EE}$ *and* $Collisions_{I\to F} < Collisions_{EE}$ **then**
    $S_{best} \leftarrow S_I$;
    break;
  end
  if $Collisions_{S\to I} + Collisions_{I\to F} < Collisions_{best}$ **then**
    $Collisions_{best} \leftarrow Collisions_{S\to I} + Collisions_{I\to F}$;
    $S_{best} \leftarrow S_I$;
  end
end
if $S_{best} = NULL$ **then**
  return failure ; `/* No valid path found */`
end
$Path_{S\to I} \leftarrow$ `CollisionFreePathPlanning`$(S_S, S_{best}, M_{BB}, N_r, step)$;
$Path_{I\to F} \leftarrow$ `CollisionFreePathPlanning`$(S_{best}, S_F, M_{BB}, N_r, step)$;
if $Path_{S\to I}$ *and* $Path_{I\to F}$ *found* **then**
  return concatenated path $(Path_{S\to I}, Path_{I\to F})$;
end
return failure;

Algorithm 1 – Collision-Free Path Planning

To speed up the path planning additional parameter $Collisions_{\max}$ is passed to *CountCollisions* and *CollidingStatesList* methods. This parameter used to interrupt algorithm if collisions count exceeds the collisions limit.

Data: $S_a, S_b, step, Collisions_{max}$
Result: Collisions count
return $|CollidingStatesList(S_a, S_b, step, Collisions_{max})|$;

Algorithm 2 – Count Collisions on Discretized Path

Data: $S_a, S_b, step, Collisions_{max}$
Result: Set of colliding states $States$
$States \leftarrow \emptyset$;
$Path \leftarrow LinearDiscretization(S_a, S_b, step)$;
foreach $S_i \in Path$ **do**
  if S_i *collides in PyBullet* **then**
    Add S_i to $States$;
  end
  if $|States| > Collisions_{max}$ **then**
    return $States$;
  end
end
return $States$;

Algorithm 3 – Colliding States List

Data: $States_{Collides}$, M_{BB}
Result: BB
foreach *joint* j *in the arm* **do**
  foreach *state* S_i *in* $States_{Collides}$ **do**
    $BB_{min}[j] \leftarrow \min(S_i[j] - M_{BB}, BB_{min}[j])$;
    $BB_{max}[j] \leftarrow \max(S_i[j] + M_{BB}, BB_{max}[j])$;
  end
  Clip $BB_{min}[j], BB_{max}[j]$ to joint limits;
end
return BB;

Algorithm 4 – Compute a Bounding Box in a Joint Space

Data: BB, N_r
Result: Set of random states $States$
$States \leftarrow \emptyset$;
for $i = 1$ *to* N_r **do**
  Generate S_i uniformly in BB;
  Add S_i to $States$;
end
return $States$;

Algorithm 5 – Generate Random States

Data: S_a, S_b, $step$
Result: List of discretized states $Path$
$Path \leftarrow [S_a]$;
$n \leftarrow \lceil |S_b - S_a|/step \rceil$;
for $i = 1$ *to* $n - 1$ **do**
  $S_i \leftarrow S_a + i \cdot \frac{S_b - S_a}{n}$;
  Append S_i to $Path$;
end
Append S_b to $Path$;
return $Path$;

Algorithm 6 – Linear Dicretization of Path Between Two States

Please note, that separate runs of an algorithm are absolutely independent, and the rerunning process is not included in algorithm description.

DOI 10.15588/1607-3274-2025-3-16

4 EXPERIMENTS

Before starting to describe the results of the algorithm's work let's clarify which auxiliary software products were used, for what hardware the test trajectories were constructed and what is the working space of the robotic arm.

The auxiliary software used for algorithm development includes:

– Bullet Collision Detection & Physics Library – a library for collision detection and physics simulation, used in the algorithm for collision search [15];

– software code by Somatic Holdings LTD, which allows for quick simulation and visualization of the motion planning results of the robotic arm in a working environment.

The test trajectories were constructed for the following robotic arm:

UFACTORY xArm 7 Robotic Arm – a 7-degree-of-freedom robotic arm with revolute joints [16].

Visual representation of a robot with robotic arm installed and test working environment shown in Fig. 1 and Fig. 2 and the working range for each joint can be seen in Table 1.

The parameters of OMPL's RRT-Connect algorithm used for comparison in testing process are:

– state space: a 7-dimensional RealVectorStateSpace, corresponding to the 7 degrees of freedom (DOF) of the robotic arm;

– joint limits: the search space is bounded using a margin of 120 degrees and limited with arm joint limits;

– collision checking resolution: set the portion of 0.05 of the state space's maximum extent (0.05 / space->getMaximumExtent());

– planner time limit: the algorithm attempts to find a solution within maximum 20.0 seconds, but interrupts as soon as any solution is found.

5 RESULTS

In Table 2 we represent four versions of the RRIS algorithm that are tested and compared. Base version of the algorithm is the one with generating states in the bounding box, intermediate states paths comparison minimizing the maximum number of collisions on two path parts and early exit strategy applied. In three other versions we checked how changing states generating strategy, states comparison method or early exit usage affects algorithm performance. So, the first algorithm version is a base algorithm, in the second version states generating strategy changed to middle state displacement, in the third version states comparison method changed to minimize the sum of collisions on path parts, in the fourth version early exit strategy disabled.

Also, RRIS based algorithm versions compared with RRTConnect algorithm (one of the most efficient nowadays), presented in OMPL.

Table 1 – Joint limits for xArm7 robotic arm [16]

Joint number	0	1	2	3	4	5	6
Minimum angle	–360°	–118°	–360°	–11°	–360°	–97°	–360°
Maximum angle	360°	120°	360°	225°	360°	180°	360°

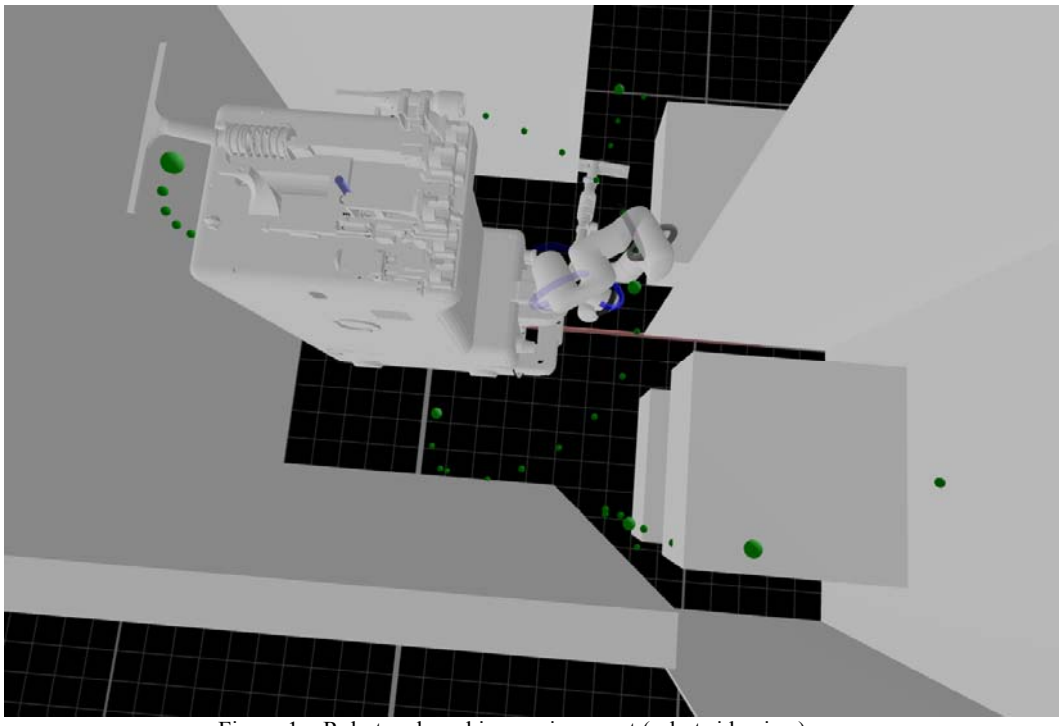

Figure 1 – Robot and working environment (robot side view)

DOI 10.15588/1607-3274-2025-3-16

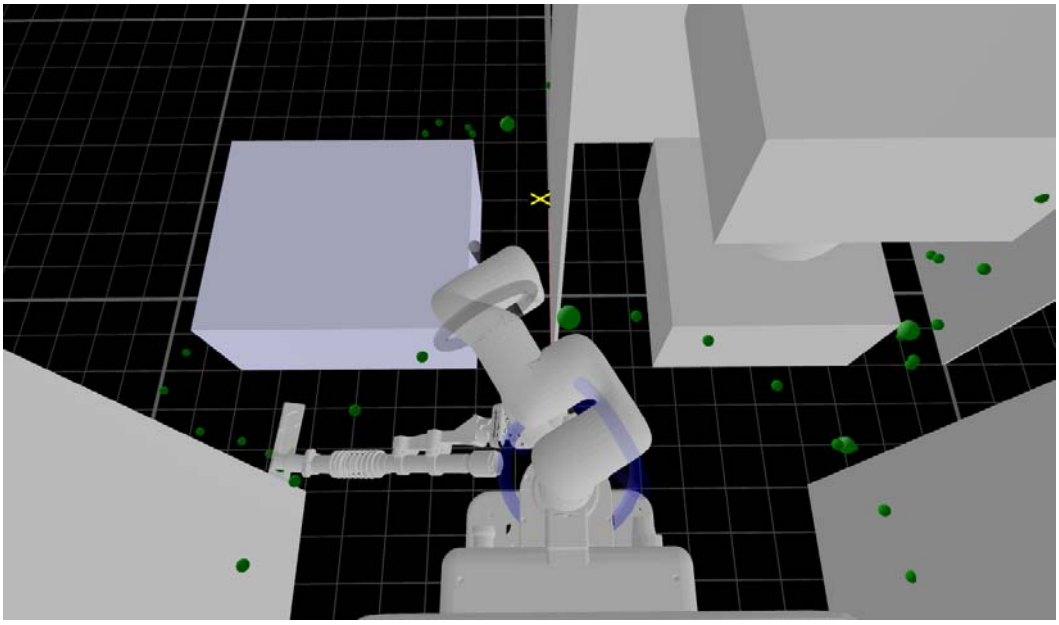

Figure 2 – Robot and working environment (robot back view)

Test set containing 104 pairs of states. It is guaranteed under experimental conditions that there is always a path without collisions between these pairs of states. Only 4 test cases have no collision on a straight path, requiring pathfinding in 96% of cases. There are three different types of tools installed to the arm wrist: sprayer, tip and vacuum. Vacuum (on Fig. 1 and Fig. 2) is much larger than two others and the wide majority of failures occurred with this tool.

Test cases contain various difficult situations like: arm should go from one side of the wall to another side, arm should move from one side of the robot to another side, arm should move between two boxes, and a lot of other complicated variations.

All the algorithm versions take 0.05 radians as a collision check step. The bounding box margin is set to 60 degrees for algorithm version 1, 3, and 4. Number of states generated on each recursive function call is 1000 for 1, 3, 4 versions and $3^7 = 2187$ for version number 2. Step for middle state displacement for version 2 calculates by formula (5).

$$d = 0.1 \cdot |S_F - S_S|_{\text{Max}} + 0.1 \cdot |S_F - S_S|_{Manhat\,\text{tan}}. \quad (5)$$

Here the constant 0.1 selected experimentally and can be configured for other robotic arms and work environments.

Table 2 – Tested algorithms versions

Version Num	States Generating	Comparison method	Early Exit
1	Bounding Box	Minimize Max	Yes
2	Middle State Displacement	Minimize Max	Yes
3	Bounding Box	Minimize Max	Yes
4	Bounding Box	Minimize Max	No

Table 3 – Testing algorithm versions results

Algorithm Version	Version 1	Version 2	Version 3	Version 4	RRTConnect from OMPL
Found Paths	92/104	92/104	91/104	94/104	104/104
Average straight path length* (radians)	3.474	3.605	3.508	3.603	3.731
Average path without collisions length* (radians)	5.750	5.857	5.652	5.756	44.167
Average search time (s)	0.555	1.341	0.582	1.825	1.109
Collision check count	4235.07	10164.01	4204.25	14421.64	10408.2

* – average straight path length and average path without collisions length calculated only for test cases where path was found

Table 4 – Comparing RRIS with repetitive calls and RRTConnect

Algorithm	RRIS algorithm with 5 attempts	RRTConnect from OMPL
Found Paths	104/104	104/104
Average straight path length (radians)	3.731	3.731
Average path without collisions length (radians)	7.427	44.167
Average search time (s)	0.36	1.109
Collision check count	2958.808	10408.2

DOI 10.15588/1607-3274-2025-3-16

Table 3 presents the summarized results of the performance of different versions of the algorithm.

We use the RRT Connect algorithm (OMPL implementation) to compare with the described algorithm. We tested RRT Connect with margins 60, 90, 120, 150 degrees and selected 120 degrees as it shows the best results with this margin value.

As can be seen in Table 3 RRIS algorithm with single run has a success rate of 87.5% – 90.4% depending on algorithm version.

In Table 4 we present the results of the RRIS algorithm that initiates the search path up to 5 times in case of failures in previous steps and compares it to the RRTConnect from OMPL. We are using version 1 (see Table 2) algorithm but reducing the number of generated states to 500. And we call it repeatedly until a path is found (but no more than 5 times).

As shown in Table 4 multi-run RRIS algorithm has 100% success rate as well as RRTConnect from OMPL, but it has 3.08 times smaller average path search time and 3.52 times smaller collisions check count. Also, multiple algorithm runs allowed to decrease the number of generated states from 1000 to 500, which decreased average search time from 0.555s to 0.36s.

6 DISCUSSION

As shown in Table 3 generating random states in the bounding box gives us better results than displacing the middle state. Probably, the reason for this may be better flexibility of this type of solution. It could generate states close or far from initial and final states and find the best option in most cases faster.

Also, results presented in Table 3 shows that early exit strategy has a great impact on algorithm productivity. It means that ideas described in section 3 are correct.

On the other hand, we can't see much difference between minimizing maximum collisions count and minimizing the sum of collisions count strategies. One strategy works better in one part of test cases and the other strategy works better for the other part.

Calling the algorithm multiple times significantly improved its reliability as shown in Table 4. The issue of local minima is significantly reduced now. Also, this allowed for a reduction in the number of generated states in a single iteration without degrading the algorithm's performance.

Compared to the RRTConnect algorithm implemented in the OMPL library, the algorithm proposed in this paper not only has better performance but also constructs a shorter path on average (as shown in Table 4). The OMPL library has an integrated path improvement system that works very well, but still the initial result path of the algorithm proposed in this paper is on average 5.947 times shorter.

However, the comparison between the performance of the RRIS algorithm and RRT-Connect depends significantly on the specific parameter settings of RRT-Connect. A deeper investigation is required to make a better comparison between these algorithms.

DOI 10.15588/1607-3274-2025-3-16

The core idea of the algorithm – to select an intermediate state based on collisions count criteria shows its effectiveness. Despite the fact that the test dataset contained many trajectory scenarios that were challenging to search for, still algorithms managed to find a path in these situations.

CONCLUSIONS

A new motion planning sampling-based algorithm was developed for solving the problem of collision-free path planning for redundant robotic manipulators in joint space in real-time mode. The algorithm is based on the principle of selecting an optimal intermediate point based on the number of collisions along the discretized path that passes from the initial to the final point through the intermediate point.

A strategy for generating intermediate points within an axis-aligned bounding box was proposed for this algorithm. Additionally, an early exit strategy was proposed to improve the algorithm's speed.

The algorithm demonstrated high efficiency. An implementation of this algorithm with iterated calls managed to find a path in test cases 3.08 times faster than the RRTConnect algorithm implemented in OMPL under the testing conditions. Also, the length of original paths found by algorithm is on average 5.947 times shorter than paths found by RRTConnect in the presented tests set.

The scientific novelty of obtained results is a newly developed sampling-based algorithm called the Recursive Random Intermediate State (RRIS) algorithm. This algorithm is able to plan the path in a dynamic environment in real time. Besides, we propose an axis-aligned bounding box generation strategy and an early exit strategy to improve algorithm speed.

The practical significance of this study lies in the development of the Recursive Random Intermediate State algorithm, which enables real-time path planning for redundant robotic arms.

Prospects for further research include enhancing the RRIS algorithm by incorporating machine learning techniques for adaptive intermediate state selection.

ACKNOWLEDGEMENTS

The authors would like to express their deep gratitude to Somatic Holdings LTD, whose codebase greatly facilitated the development of the algorithm presented in this paper.

REFERENCES

1. Data Center Solutions, IOT, and PC Innovation [Electronic resource]. Mode of access: https://www.intel.com/content/www/us/en/robotics/robotic-arm.html (date of access: 15 June 2023).
2. Pennestri E., Cavacece M., Vita L. On the Computation of Degrees-of-Freedom: A Didactic Perspective [Electronic resource], *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference: proceedings of the conference, [Long Beach, California, USA]*, September 24–28, 2005, Mode of access: https://doi.org/10.1115/DETC2005-84109, pp. 1733–1741.

3. Ashitava G. Resolution of Redundancy in Robots and in a Human Arm, *Mechanism and Machine Theory*, 2018, Vol. 125, pp. 126–136.
4. Kavraki L. E., Svestka P., Latombe J.-C., Overmars M. H. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces, *IEEE Transactions on Robotics and Automation*, 1996, Vol. 12, No. 4, pp. 566–580. doi: 10.1109/70.508439.
5. LaValle S. M. Rapidly-Exploring Random Trees: A New Tool for Path Planning, *The Annual Research Report*, 1998.
6. Khatib O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, *The International Journal of Robotics Research*, 1986, Vol. 5(1), pp. 90–98. doi:10.1177/027836498600500106.
7. Lillicrap T. P., Hunt J. J., Pritzel A., Heess N., Erez T., Tassa Y., Wierstra D. Continuous Control with Deep Reinforcement Learning, *arXiv preprint arXiv:1509.02971*, 2015.
8. Khan A. T., Cao X., Li Z., Li S. Evolutionary Computation Based Real-Time Robot Arm Path-Planning Using Beetle Antennae Search, *EAI Endorsed Transactions on AI and Robotics*, 2022, Vol. 1, P. e3.
9. Kuffner J. J., LaValle S. M. RRT-Connect: An Efficient Approach to Single-Query Path Planning, *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. San Francisco, CA, USA, 2000, Vol. 2, pp. 995–1001, doi: 10.1109/ROBOT.2000.844730.
10. LaValle S. M., Kuffner J. J. Randomized Kinodynamic Planning, *The International Journal of Robotics Research,* 2001, Vol. 20(5), pp. 378–400. doi:10.1177/02783640122067453.
11. Karaman S., Frazzoli E. Sampling-Based Algorithms for Optimal Motion Planning, *International Symposium on Robotics Research*, 2011, May, Vol. 71, pp. 65–70.
12. Kang J.-G., Lim D.-W., Choi Y.-S., Jang W.-J., Jung J.-W. Improved RRT-Connect Algorithm Based on Triangular Inequality for Robot Path Planning, *Sensors*, 2021, Vol. 21, P. 333. https://doi.org/10.3390/s21020333.
13. Ganesan S., Ramalingam B., Mohan R. E. A Hybrid Sampling-Based RRT Path Planning Algorithm for Autonomous Mobile Robot Navigation, *Expert Systems with Applications*, 2024, Vol. 233, P. 125206. https://doi.org/10.1016/j.eswa.2024.125206.
14. Gammell J. D., Srinivasa S. S., Barfoot T. D. Informed RRT*: Optimal Sampling-Based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic, *arXiv preprint arXiv:1404.2334*, 2014.
15. Bullet Collision Detection & Physics Library [Electronic resource]. Mode of access: https://pybullet.org/Bullet/BulletFull/index.html (date of access: 15 June 2023).
16. The Difference Between UFACTORY xArm5, UFACTORY xArm6 and UFACTORY xArm7 [Electronic resource], *UFACTORY*. Mode of access: http://help.ufactory.cc/en/articles/4491842-the-difference-between-ufactory-xarm5-ufactory-xarm6-and-ufactory-xarm7 (date of access: 15 June 2023).
17. Open Motion Planning Library, Version 1.4.2 [Electronic resource]. OMPL Development Team. Mode of access: http://ompl.kavrakilab.org (date of access: 8 August 2023).
18. Xin P., Wang X., Liu X., Y. Wang, Z. Zhai, X. Ma Improved Bidirectional RRT* Algorithm for Robot Path Planning, *Sensors*, 2023, Vol. 23, No. 2, P. 1041. https://doi.org/10.3390/s23021041.

Received 07.04.2025.
Accepted 30.06.2025.

УДК 004.94

ПЛАНУВАННЯ ШЛЯХУ ДЛЯ НАДЛИШКОВИХ РОБОРУК З ВИКОРИСТАННЯМ АЛГОРИТМУ РЕКУРСИВНОГО ВИПАДКОВОГО ПРОМІЖНОГО СТАНУ

Медвідь А. Я. – аспірант кафедри Систем Штучного Інтелекту, Національний університет «Львівська політехніка», Львів, Україна.

Яковина В. С. – д-р техн. наук, професор кафедри Систем Штучного Інтелекту, Національний університет «Львівська політехніка», Львів, Україна.

АНОТАЦІЯ

Актуальність. Планування шляху без зіткнень в просторі суглобів для надлишкових роборук (роботизованих маніпуляторів) залишається складною задачею через високу вимірність конфігураційного простору і динамічну зміну середовища. Існуючі методи планування часто стикаються з труднощами у балансуванні між часом пошуку та якістю траєкторії, що є критично важливим для застосувань у режимі реального часу.

Мета роботи – розробка нового методу планування траєкторій без зіткнень в режимі реального часу для роборук з надлишковими суглобами.

Метод. У цьому дослідженні представлений новий алгоритм планування шляху без зіткнень у просторі суглобів для надлишкових роборук, що працює на основі генерації випадкових станів. Алгоритм отримав назву Рекурсивного Випадкового Проміжного Стану (РВПС). Принцип роботи алгоритму полягає у генерації набору випадкових проміжних станів із подальшим ітеративним вибором оптимального на основі кількості зіткнень уздовж дискретизованої траєкторії. Крім того, у статті пропонується стратегія побудови обмежувального прямокутного паралелепіпеда (bounding box) та стратегія раннього виходу для підвищення швидкості роботи алгоритму. Нарешті, для підвищення надійності пропонується повторне викликання алгоритму. Ефективність алгоритму РВПС оцінюється шляхом проведення комплексних тестів та порівнюється з популярним алгоритмом RRT Connect, реалізованим у бібліотеці Open Motion Planning Library.

Результати. Експериментальні дослідження показують, що алгоритм РВПС за умов тестування забезпечує траєкторії без зіткнень зі значно коротшою середньою довжиною та скорочує час пошуку приблизно у три рази порівняно з алгоритмом RRT Connect.

Висновки. Запропонований алгоритм РВПС демонструє перспективний підхід до планування траєкторій у режимі реального часу для надлишкових роботизованих маніпуляторів. Поєднуючи стратегічну вибірку проміжних станів із ефективною оцінкою зіткнень та механізмами раннього завершення, алгоритм пропонує надійну альтернативу відомим методам.

КЛЮЧОВІ СЛОВА: планування шляху, надлишковий роботизований маніпулятор, уникнення зіткнень.

DOI 10.15588/1607-3274-2025-3-16

ЛІТЕРАТУРА

1. Data Center Solutions, IOT, and PC Innovation [Електронний ресурс]. – Режим доступу: https://www.intel.com/content/www/us/en/robotics/robotic-arm.html (дата звернення: 15 червня 2023).
2. Pennestri E. On the Computation of Degrees-of-Freedom: A Didactic Perspective [Електронний ресурс] / E. Pennestri, M. Cavacece, L. Vita // ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference: proceedings of the conference, [Long Beach, California, USA], September 24–28, 2005. – Режим доступу: https://doi.org/10.1115/DETC2005-84109. – С. 1733–1741.
3. Ashitava G. Resolution of Redundancy in Robots and in a Human Arm / G. Ashitaya // Mechanism and Machine Theory. – 2018. – Том 125. – С. 126–136.
4. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces / [L. E. Kavraki, P. Svestka, J.-C. Latombe, M. H. Overmars] // IEEE Transactions on Robotics and Automation. – 1996. – Том 12, № 4. – С. 566–580. doi: 10.1109/70.508439.
5. LaValle S. M. Rapidly-Exploring Random Trees: A New Tool for Path Planning / S. M. LaValle // The Annual Research Report. – 1998.
6. Khatib O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots / O. Khatib // The International Journal of Robotics Research. – 1986. – Том 5(1). – С. 90–98. doi:10.1177/027836498600500106.
7. Lillicrap T. P. Continuous Control with Deep Reinforcement Learning / [T. P. Lillicrap, J. J. Hunt, A. Pritzel et al.] // arXiv preprint arXiv:1509.02971. – 2015.
8. Evolutionary Computation Based Real-Time Robot Arm Path-Planning Using Beetle Antennae Search / [A. T. Khan, X. Cao, Z. Li, S. Li] // EAI Endorsed Transactions on AI and Robotics. – 2022. – Том 1. – С. e3.
9. Kuffner J. J. RRT-Connect: An Efficient Approach to Single-Query Path Planning / J. J. Kuffner, S. M. LaValle // Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 2000. – Том 2. – С. 995–1001. DOI: 10.1109/ROBOT.2000.844730.
10. LaValle S. M. Randomized Kinodynamic Planning / S. M. LaValle, J. J. Kuffner // The International Journal of Robotics Research. – 2001. – Том 20(5). – С. 378–400. doi:10.1177/02783640122067453.
11. Karaman S. Sampling-Based Algorithms for Optimal Motion Planning / S. Karaman, E. Frazzoli // International Symposium on Robotics Research. – 2011, May. – Том 71. – С. 65–70.
12. Kang J.-G. Improved RRT-Connect Algorithm Based on Triangular Inequality for Robot Path Planning / [J.-G. Kang, D.-W. Lim, Y.-S. Choi et al.] // Sensors. – 2021. – Том 21. – С. 333. https://doi.org/10.3390/s21020333.
13. Ganesan S. A Hybrid Sampling-Based RRT Path Planning Algorithm for Autonomous Mobile Robot Navigation / S. Ganesan, B. Ramalingam, R. E. Mohan // Expert Systems with Applications. – 2024. – Том 233. – С. 125206. https://doi.org/10.1016/j.eswa.2024.125206.
14. Gammell J. D. Informed RRT*: Optimal Sampling-Based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic / J. D. Gammell, S. S. Srinivasa, T. D. Barfoot // arXiv preprint arXiv:1404.2334. – 2014.
15. Bullet Collision Detection & Physics Library [Електронний ресурс]. – Режим доступу: https://pybullet.org/Bullet/BulletFull/index.html (дата звернення: 15 червня 2023).
16. The Difference Between UFACTORY xArm5, UFACTORY xArm6 and UFACTORY xArm7 [Електронний ресурс] / UFACTORY. – Режим доступу: http://help.ufactory.cc/en/articles/4491842-the-difference-between-ufactory-xarm5-ufactory-xarm6-and-ufactory-xarm7 (дата звернення: 15 червня 2023).
17. Open Motion Planning Library, Version 1.4.2 [Електронний ресурс] / OMPL Development Team. – Режим доступу: http://ompl.kavrakilab.org (дата звернення: 8 серпня 2023).
18. Improved Bidirectional RRT* Algorithm for Robot Path Planning / [P. Xin, X. Wang, X. Liu et al.] // Sensors. – 2023. – Том 23, № 2. – С. 1041. https://doi.org/10.3390/s23021041.

DOI 10.15588/1607-3274-2025-3-16