Motion planning of free-floating space robots through multi-layer optimization using the RRT* algorithm

Liu, Ruipeng; Guo, Jian; Gill, Eberhard

DOI
10.1016/j.actaastro.2024.12.036

Publication date
2025

Document Version
Final published version

Published in
Acta Astronautica

Citation (APA)
Liu, R., Guo, J., & Gill, E. (2025). Motion planning of free-floating space robots through multi-layer optimization using the RRT* algorithm. *Acta Astronautica*, *228*, 940-956. https://doi.org/10.1016/j.actaastro.2024.12.036

Acta Astronautica 228 (2025) 940–956

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Research paper

Motion planning of free-floating space robots through multi-layer optimization using the RRT* algorithm

Ruipeng Liu, Jian Guo *, Eberhard Gill

Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, Delft, 2629 HS, The Netherlands

ARTICLE INFO

Keywords:
Space robots
Active debris removal
Motion planning
Manipulator

ABSTRACT

This paper introduces a motion planning method for capture of tumbling objects using a free-floating space robot. The proposed approach incorporates an improved Rapidly Exploring Random Tree Star (RRT*) algorithm enabling obstacle avoidance and generating desired trajectories for the robot's end-effectors. Additionally, a multi-layer optimization process and a greedy policy are proposed to achieve singularity avoidance, joint velocity, and acceleration optimization by leveraging the robot arm's joint energy distribution, torque, and manipulability. By adopting this motion planning strategy, the space robotic system demonstrates improved performance in obstacle and singularity avoidance, without the need for inverse Jacobian matrix calculations. Furthermore, the multi-layer optimization process enhances trajectory smoothness and reduces end-effector vibration through energy and torque optimization. This research contributes to advancing space robotic systems by enhancing the entire energy and torque consumption on motion planning to make the end-effector move smooth and reduce the vibration.

1. Introduction

With the increasing demand for space applications, the number of spacecraft has tremendously grown in the past decades. As a consequence, space debris, including malfunctioning satellites, upper stages, and many other objects, are posing threats to space operations. To address this threat, space robots, show great potential ability to solve this problem. Over the past two decades, a number of enabling technologies have been developed and several technology demonstration missions have been successfully completed to support this [1].

Before initiating the capture process, the space robot performs a fly-around observation of the debris to measure its rotational angular velocity, axis of rotation, and other relevant parameters. The challenges in this phase primarily arise from the tumbling motion of the debris, which requires precise trajectory planning. These challenges include accurately predicting the debris' motion based on observed data, avoiding obstacles during manipulator movement, ensuring a smooth and precise trajectory for the robotic arm, and minimizing end-effector jitter to achieve stable and reliable capture. Addressing these factors introduces significant complexity to the optimization process, and this paper focuses on tackling these specific issues.

After space robots perform close-range rendezvous with debris, the capturing process begins with four phases: observation and planning, manipulator movement, debris capture, and stabilization (pose capture). Before initiating the capture process, the space robot performs a fly-around observation of the debris to measure its rotational angular velocity, axis of rotation, gap point and other relevant parameters. In the second phase, the challenges include accurately predicting the debris' trajectory, avoiding obstacles, ensuring a smooth and precise trajectory for the robotic arm, and minimizing end-effector jitter to ensure stable and reliable capture, The primary focus of this paper is optimizing the motion of the robot arm to approach the target optimally. Key optimization problems include avoiding obstacles, ensuring a smooth trajectory, and minimizing end-effector. The optimization process involves energy and torque optimization, avoiding robot arm singularity, and ensuring obstacle clearance, presenting a complex multi-objective optimization problem. Space mission motion planning for robotic systems differs significantly from ground-based systems due to specific working conditions, introducing numerous challenges. Researchers have proposed various methodologies and strategies to address these issues.

The singularity working point of robot arms is a primary challenge that needs to be addressed. It often leads to the loss of one or more degrees of freedom (DOFs) in the robot's motion. Various methodologies have been proposed to tackle this issue. Jin et al. [2] introduced a method that combines Damped Least Squares (DLS) and feedback compensation to avoid singularities for high DOF space robots in the Cartesian space. Wang et al. [3] implemented a constrained particle

* Corresponding author.
E-mail address: j.guo@tudelft.nl (J. Guo).

https://doi.org/10.1016/j.actaastro.2024.12.036
Received 25 September 2024; Received in revised form 27 November 2024; Accepted 17 December 2024
Available online 25 December 2024

swarm optimization (PSO) scheme with adaptive inertia weight to overcome dynamics singularities in free-floating space robots, utilizing Bézier curves to simplify the calculations. Xu et al. [4] proposed the singularity separation plus damped reciprocal (SSPDR) method, which eliminates the need for Singular Value Decomposition (SVD), singularity value, and Jacobian matrix in the system. Yoshihiko et al. [5] introduced the singularity robust inverse (SR-inverse) as an alternative to the pseudoinverse of the Jacobian matrix for robot arm on the ground, providing an approximate solution when the inverse kinematic solution is not feasible at singular points.

The second challenge in a capturing mission is the problem of free-floating systems, where the base is not fixed. These systems are particularly relevant in space robotics, where both the manipulator and its base (e.g., a satellite) are subject to dynamic interactions. Research in this area addresses the complexities of planning trajectories and handling the coupled dynamics of the manipulator and its free-floating base, taking into account obstacles and differential constraints (nonholonomic and kinodynamic). Such studies are crucial for advancing autonomous robotic motion planning in space applications. Benevides et al. [6] developed a collision-avoidance path planner for a free-floating planar manipulator, utilizing a dynamically equivalent model and Rapidly-Exploring Random Trees framework, with simulations demonstrating the approach's effectiveness and suggesting promising directions for future work. Rybus et al. [7] applied the RRT algorithm to trajectory planning for free-floating satellite-manipulator systems, demonstrating its effectiveness in managing high-dimensional states and constraints through simulation. Serrantola et al. [8] using the RRT Control algorithm for trajectory planning of a Dual-Arm Planar Free-Floating Manipulator (FFSM) with static obstacle avoidance, showing promising results through simulations. Rybus et al. [9] presents a new path planning method for robotic arms on free-floating spacecraft, using spline functions in joint space and an active-set algorithm to generate a paths without collision. Zhang et al. [10] present a novel motion planning algorithm for free-floating space robots, utilizing RRTs to address challenges like nonholonomic systems and base attitude constraints. Yu et al. [11] present a coordinated path planning framework for a dual-arm free-floating space robot, using a novel coordinated RRT*-based approach to generate initial paths and quartic splines for smooth execution.

The third challenge for motion planning is the obstacle clearance problem. The potential collision with obstacles will cause the mission to fail and may create secondary debris. Qian et al. [12] proposed a collision prediction method including collision against the environment and the robot itself, and they improved Lazy Theta* algorithm with the ability to adjust itself to accommodate different planning environments. A robust decentralized control strategy is proposed by Liu, et al. [13] based on signal compensation and back-stepping, which ensures the system automatically avoids collision with good robust performance. S. Liu et al. [14] solve the collision-free motion planning problem for space manipulators by improving the artificial potential field method and combining the algorithms with the Generalized Jacobian Matrix with inverse kinematics. The nonholonomic redundancy is discussed in [15], and a path-planning scheme using Lyapunov functions in the hierarchy is proposed. Researchers also applied the multi-resolution potential-field-based iterations for an experiment based on PUMA 560 manipulator [16]. Besides, the optimization method is also an effective way for space robot collision avoidance. Based on nonlinear optimization the end-effort collision avoidance planning problem is solved in [17]. In [18] the robustness and also the actuation energy problem of the motion planning process are solved by the selected cost functions. In [19] the obstacle-moving problem is solved by applying the Chebyshev-pseudospectral method to the non-linear optimization problem. Huang et al. [20] performed the planning problem by utilizing the genetic approach to determine the trajectory in the joint space.

Another challenge of space robots is the energy optimization and the motion control problem. The approaching and the grasping quality of space robots is determined by the velocity and the acceleration of the end-effectors, These two elements correspond to the energy and the torque of robot arms. The purpose of minimizing the velocity is to reduce the contact force between end-effectors and targets and at the same time minimizing the acceleration is to reduce the vibrations of the end-effector. Umetani et al. [21] resolved the motion rate control problem for space manipulators by means of introducing the momentum conservation law in their method and deriving a new Jacobian matrix in generalized form. In [22] a method based on the trapezoidal velocity profile is proposed, this method takes the moving stability and the constraints into consideration simultaneously and provides full joint driving performances. Huang et al. [20] minimized the maximum jerk of the joint by applying the genetic algorithm to search the joint inter-knot space and determining the optimal trajectory. In another paper [23] considering the limited energy supply Huang et al. present a minimum-torque path-planning scheme for space robots, a genetic approach is used to optimize joint angle, velocity, acceleration, and torque constraints. In [24], Lei et al. analyzed the dynamic characteristics of free-floating space robots and proposed an optimal control algorithm by employing the driving torque in joint space as the objective function.

To meet the demands of active debris removal missions, which may require capturing of tumbling targets, autonomous motion planning for robot arm is a crucial component of these missions Moreover, during the approaching and capture process, safety is always one of the most significant factors. Thus, not only the trajectory planning problem but also obstacle avoidance, angular velocity, robot arm manipulability, and singularity must be taken into account at the same time. Therefore the main motivation of this paper is to obtain a motion planning methodology for space robots that can realize collision avoidance, velocity, and acceleration optimization.

The literature highlights several challenges in autonomous motion planning for free-floating space robots. While algorithms like RRT* offer asymptotic optimality, their computational complexity increases with iterations, limiting real-time applicability. Dynamic uncertainties, including variations in base attitude and nonholonomic constraints, pose ongoing challenges. Effective handling of singularities is critical for maintaining manipulator flexibility. Balancing objectives such as energy efficiency, torque minimization, and obstacle avoidance presents complex optimization trade-offs. Despite these challenges, advancements in collision avoidance and trajectory optimization strategies improve operational safety and efficiency.

The original contribution of this paper comprise of 3 parts: (1) Improved the RRT* algorithm is proposed to realize collision clearance and path searching, when compared with the original algorithm, the proposed algorithm greatly decreases the computation workload, but still maintains the same solution level as the original RRT*. (2) The multi-layer planning method was proposed to establish the connection between the kinetic and dynamic layers. In the kinetic layer, the method focus on arm manipulability to avoid singularity, while in the dynamic layer, it aims to achieve energy-optimal distribution for motion execution. (3) Proposed a greedy policy for the velocity and acceleration joint planning for the robot arm.

This paper is organized as follows. Section 2 presents the dynamics model of the space robotic system as the basis for the subsequent sections. Section 3 introduces the improved RRT*-based algorithm, three improvements based on the basic RRT* algorithm are developed to make this algorithm better for the planning model. Section 4 introduced the multi-layer optimization method and the greedy policy to deal with the energy and torque joint planning problem simultaneously. Finally, the simulation results are given in Section 5. Summary and conclusion are made in Section 6.

2. Dynamic model

A schematic diagram of a space robotic system is depicted in Fig. 1. The system consists of a base spacecraft equipped with one or more

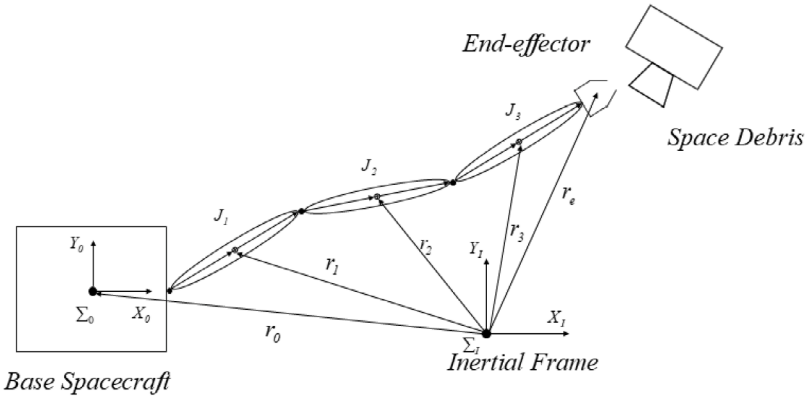


Fig. 1. Schematic diagram of space robotic system.

multi-jointed robotic arms. The robot arm studied in this paper, assumed as a 3-joint, 2 degrees of freedom (DOF) robot arm which are j_1 to $j3$, the center of the mass(COM) of the base is u_0, the attitude of the base is R_0 position of arm center of mass are $\vec{r_1}$, $\vec{r_2}$, $\vec{r_3}$, from joint 1 to joint 3, and all the joints being rotation joints, the angle of joint are q_1, q_2, q_3, attitude of arm are R_1, R_2, R_3. The coordinate reference system developed as $\sum_0$, centered at the base of the space robot arm, the position of the end-effector is r_e, all vectors and coordinates given in this paper are referring under $\sum_0$, $\sum_I$ is the inertial reference system. The position vector of the center of mass of each robot arm joint can be expressed as follows(i from 1–3, $i = 0$ is for the base):

$$r_i = \begin{bmatrix} r_x i, & r_y i \end{bmatrix}^T \tag{1}$$

And for the position of end effector,it can be express as:

$$r_e = T_2^3 T_1^2 T_0^1 r_0 \tag{2}$$

Where T_{i-1}^i is the transformation matrix from link $i-1$ to link i.

Due to the free floating system being enacted upon by any external forces, the position of the system's center of mass remains unchanged.

$$\sum_{i=0}^{n} m_i r_r = M r_g \tag{3}$$

Where m_i and r_i is the mass and position of the base spacecraft and links, (i from 0–3), m_0 and r_0 is for the base, m_{1-3} and r_{1-3} are for links.

$$\begin{bmatrix} v_e \\ w_e \end{bmatrix} = J_b \begin{bmatrix} v_0 \\ w_0 \end{bmatrix} + J_m \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} \tag{4}$$

J_b is the Jacobi matrix associated with the motion of the base, and J_m is the Jacobi matrix for the links, v_e and w_e are the velocity and angular velocity of the end effector, $v_0 = \begin{bmatrix} \dot{r_{x0}}, & \dot{r_{y0}} \end{bmatrix}^T$, is the velocity of the base spacecraft, $w_0 = w_{z0}$. The dynamic equation of a space robot system can be expressed as follows:

$$H\dot{u} + Cu = \tau \tag{5}$$

Where, H is a symmetric, positive-definite Generalized Inertia Matrix (GIM), C represents the Convective Inertia Matrix (CIM), and τ denotes the generalized forces (joint-space forces). When formulating the equations of motion, it is possible to explicitly delineate the respective influences of the base-link and the manipulator.

$$\begin{bmatrix} H_0 & H_{0m} \\ H_{0m}^T & H_m \end{bmatrix} \begin{bmatrix} \dot{u_0} \\ \dot{u_m} \end{bmatrix} + \begin{bmatrix} C_0 & C_{0m} \\ C_{0m}^T & C_m \end{bmatrix} \begin{bmatrix} u_0 \\ u_m \end{bmatrix} = \begin{bmatrix} \tau_0 \\ \tau_m \end{bmatrix} \tag{6}$$

Where u_0 is the base angular velocity and velocity, $u_0 = \begin{bmatrix} w_0 & \dot{r_0} \end{bmatrix}^T$, u_m is the angular velocity of joints, $u_m = \begin{bmatrix} \dot{q_1} & \dot{q_2} & \dot{q_3} \end{bmatrix}^T$, and τ_0 is the torque on the base and τ_m is the torque on each joints. It is needs to be note that, for the sake of simplicity, the simulation environment is a 2D environment, so the third dimension in this part of the model is set to 0.

3. Motion planning based on the improved RRT*-based algorithm

In this section, the motion planning optimization problem will be formulated. The sampling based motion planning algorithm is also introduced in this chapter to give a brief overview of the methodology used in this paper. Normally optimization is a process that minimizes one or more functions under one or more constraints. For the space robots motion planning problem, the optimization is usually under Eq. (4).

To optimize the trajectory of a robot arm, the vector v_e and w_e need to be calculated or acquired in the operation space, and then through the inverse of the matrix J_m^{-1}, the configuration and the angular velocity of the robot arm can be acquired in the joint space. One problem involved in this process is that the inverse of the Jacobian matrix is uninvertible if the number of joints are more than 3, due to the fact that the Jacobian matrix is a 6*3 matrix. In addition, we also need to solve the multiple solutions problem. However this problem can be avoided through the method we proposed in this paper, which does not need to calculate the inverse of the Jacobian matrix but optimizes the linear velocity in operation space.

3.1. Improvement of RRT*-based algorithm

In this section, we will introduce the improved RRT* algorithm. RRT* serves as both a data storage structure and an algorithm. It efficiently explores non-convex, high-dimensional spaces by randomly constructing a space-filling tree [25,26]. This algorithm is an asymptotically optimal algorithm that offers several advantages, including obstacle avoidance and the ability to address nonholonomic and kinematic constraints, more detailed information can be found in appendix. However, due to the computational complexity and limitations in search theory of this algorithm, we have made improvements to the algorithm, enhancing its efficiency in searching and avoiding obstacles.

3.1.1. Improved RRT* algorithm

RRT* is an asymptotically optimal algorithm suitable for motion planning problems, but it faces significant challenges: high computational workload, slow growth even without obstacles, and excessive computational demands far from the goal. To address these issues, we improved RRT* from the three aspects: a probabilistic sampling strategy, an obstacle avoidance strategy, and a variable step size growth strategy.

Probabilistic sampling strategy

Although researchers proposed many extensions of the basic RRT algorithms [25,27], most previous studies focus on dispersion and discrepancy of sampling, such as applying Halton sequence, linear congruential method, or other sampling strategies [26,28,29]. However, when considering the space robot motion planning problem, which needs to accommodate the context. Algorithm needs to focus more on the space highly relevant to the goal state, rather than searching the entire space. At the same time, we also need the algorithm to have the ability to search more space when meet obstacles, so the probabilistic sampling strategy is proposed.

Definition 1 (*Probabilistic Sampling*)**.** There is a triplet $\{x_{init}, x_{obs}, x_{goal}\}$, in the planning space, where x_{init} is initial state, x_{goal} is goal state, and x_{obs} are the obstacles. The algorithm explores the space by growing the branch randomly in the planning space(sampling). According to the status of the system, the sampling and growth strategy are designed as follows:

$$x_{new} = x_{sample}\frac{stepsize}{\rho(x_{sample}, arg\ min\ \rho(x_{sample}, x_i))} + arg\ min\ \rho(x_{sample}, x_i) \quad (7)$$

where, x_{new} is the new branch that will be added to the tree, x_{sample} is the sample point used to get the x_{new}.

where, $x_{sample} = \begin{cases} x_{goal} & if \quad > \varepsilon \\ random & if \quad < \varepsilon \end{cases}$ $p \in [0,1]$ subject to the uniform distribution, ε is a number set up the user used to make a trade-off between growth to the goal or exploring. ρ(A,B) is the Euclidean distance between A and B.

Obstacles avoidance strategy

The obstacles avoidance strategy is crucial for all sample-based planning problems. An advanced obstacles avoidance strategy needs to have low computational complexity, and low repetition times in the entire planning process. However, unlike analytic problems, those aspects pose weakness of the sample-based optimization algorithms. In convention methods, the algorithm often attempts many times in the same unfeasible direction. Thus wastes lots of computational resources and slow in obstacle avoidance problem. As a result, we modify the steering strategy to address the issue of high obstacles avoidance repetition times for the same obstacle. As the branch grows into an obstacles space, the steer function is activated.

Fast-growing strategy

The step size represents the distance that an algorithm traverses within the search space. The selection of an appropriate step size holds paramount importance in planning algorithms, as it significantly impacts both the convergence speed of the algorithm and the quality of the final solution. When the step size is excessively small, the algorithm may exhibit slow convergence, requiring a larger number of iterations to reach the optimal solution. Conversely, if the step size is overly large, the algorithm runs the risk of overlooking the optimal solution or failing to converge entirely.

In the RRT* algorithm this implies the distance for each growing. This character exerts an important role in motion planning because the same step size means the same time period if the object is moving at a constant velocity, which is convenient for discrete control on a robotic motion planning system.

However during our studies, we found that this character can also exert negative impacts. The constant stepsize, limited convergence speed, difficulty in balancing exploration and exploitation, and fixed step size typically relies on the characteristics and experience of the problem, which can vary among different problems and their search space structures.

Definition 2 (*Growing Function*)**.** As shown in Fig. 2, in order to overcome these effects and improve the algorithm with both of convergence speed and exploration, we propose an exploring and growth strategy. In each iteration, a parent node is equipped with three exploring nodes. These exploring nodes expand in the planning space much farther than normal branches, following the easy feasible checking, the algorithm will choose the farthest one as the new node. Otherwise, if the one or more exploring nodes failed in the easy feasible checking, there are obstacles around the parent node. In that case the algorithm will swap to the detailed exploring model. This model employs a smaller step size to explore the path around obstacles. As a result, which is designed as a triple:

$$\{x_{partenst}, x_{explore^i}, x_{new}\}, (i = 1,2,3) \quad (8)$$

where, $x_{explore^i}$ is the ith exploring node, $x_{explore^i} = x_{parents} + i \cdot stepsize, (i = 1,2,3)$, x_{new} is the new node will be add on the tree, where $x_{new} = x_{explore^i}$ if $x_{explore^i} \notin obstacles$.

Definition 3 (*Steer Function*)**.** As shown in Fig. 3 a branch will be pruned when it grows into the obstacle space. After pruned, the father node of this branch becomes the terminal node. From this terminal node, a new branch will grow according to the following strategy: First, the algorithm sets the goal direction as the direction of a new branch. It then grows this branch and checks if it enters the obstacle space. Not only is the end effector considered for obstacle avoidance, but each joint and link of the manipulator as well. If the branch is feasible (e.g., it does not intersect with any obstacles), it is added to the tree. Furthermore, the allowable end effector altitude range at each node in the tree is marked. However, if the branch is infeasible, a live update direction library is created, containing directions for new branches.

This process is referred to as the "first trying", as shown in Fig. 2 Subsequently, when the "first trying" fails, the algorithm will select a new direction for the branch in the next iteration from the live direction library. The live direction library is updated using a uniform distribution ranging from $[0-2\pi]$. If the chosen direction is not feasible, a group of elements will be removed from the library. The live direction library is defined as follows:

$$directionlib_i = U(0, 2\pi), (i = 1) \quad (9)$$

where, $directionlib_i$ is a N elements set with discrete uniform distribution from $[0 - 2\pi]$.

$$directionlib_{i+1} = directionlib_i - Dir_{i-distribution}, (i > 1) \quad (10)$$

where, $directionlib_{i+1}$ is a M elements discrete Gaussian distribution, which mean and variance, set by the user to control cut off zone in live update library.

$$Dir_{i-distribution}\ f(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x - dir_{i-1})^2}{2\sigma^2}) \quad (11)$$

where, dir_{i-1} is the mean of the Gaussian distribution, also the unfeasible direction in the last iteration.

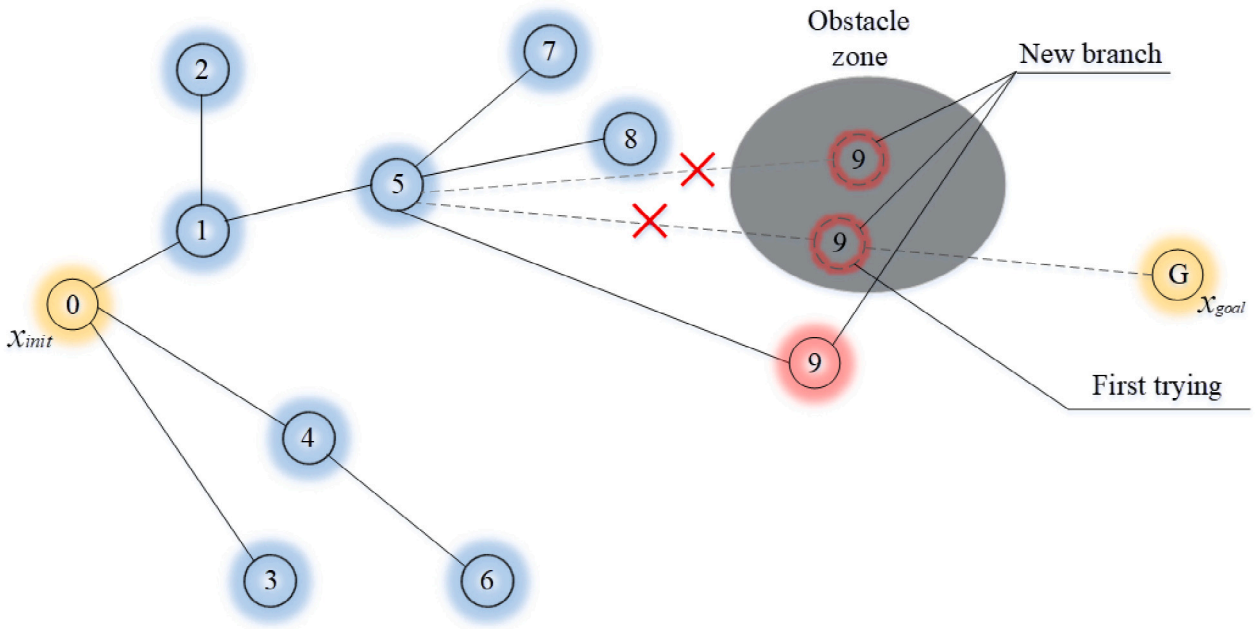


Fig. 2. First trying of improved RRT* algorithm.

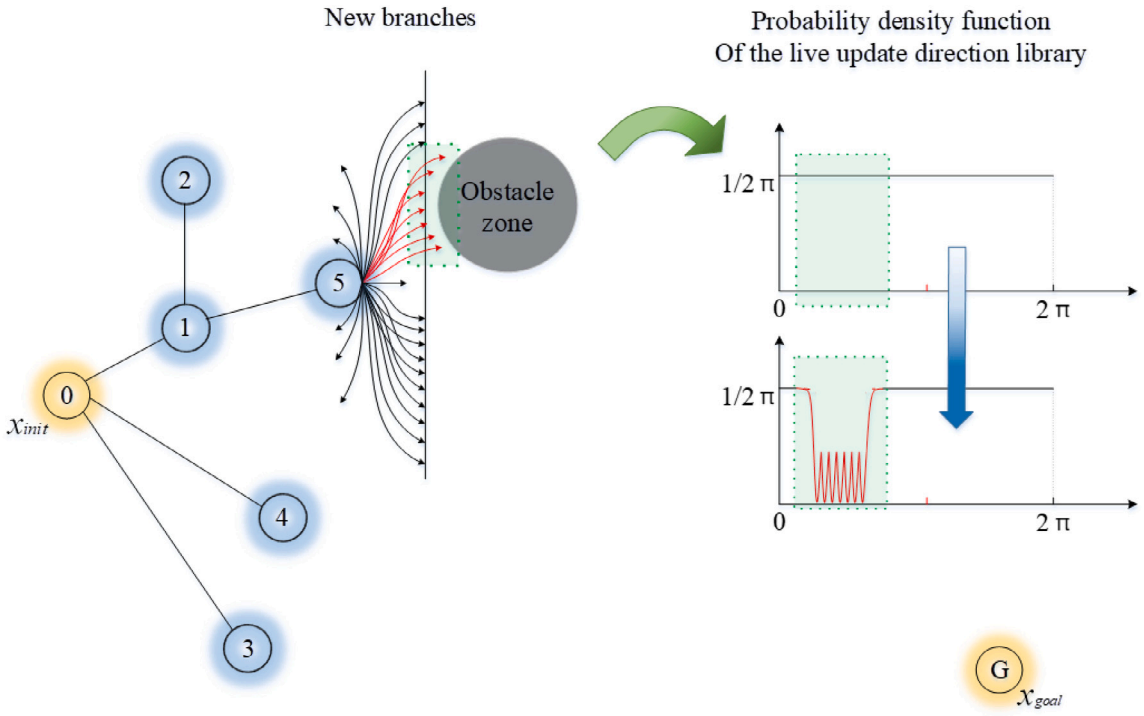


Fig. 3. Strategy Of steer function from live update direction library.

3.2. Simulation of improved RRT algorithm*

In this section, the simulation is conducted solely to better illustrate the advantages and disadvantages of the improved RRT* algorithm. Therefore, a standard test scenario is used for comparison instead of directly applying it to a space robotic arm, a comparative simulation is conducted among the classical RRT algorithm, classical RRT* algorithm, RRT-lazy, and RRT-connect algorithm.

In the first simulation, a virtual environment is set up. This virtual environment does not have units assigned to it. The units in this simulation range from 0 to 100 and can be mapped to a real environment based on users. The end-effector of the robot arm is initially positioned at (5,5) and commanded to grasp a target satellite located on the opposite side of the search space. Multiple obstacles are present in this

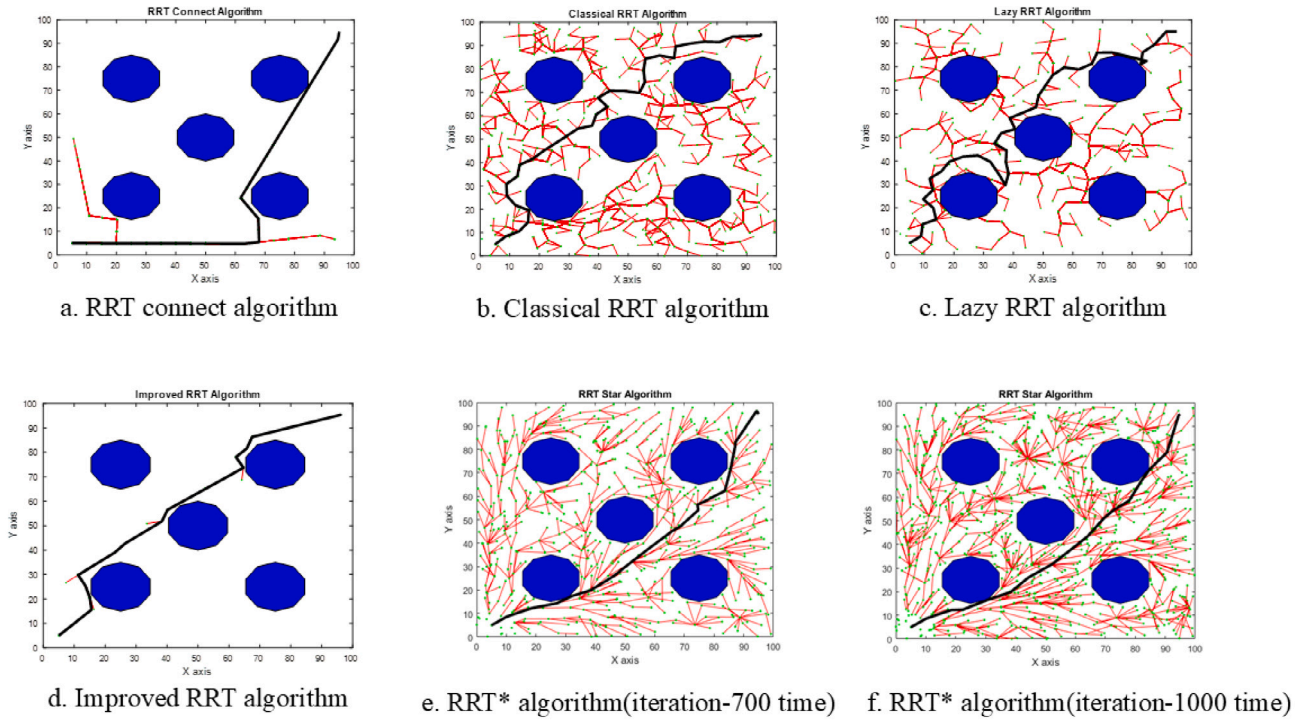


Fig. 4. (a-f) Comparison of different rrt algorithm with improved rrt algorithm.

area. The target coordinates are set to (95,95). The simulation results are presented in Fig. 4.

The numerical simulations reveal that among the five tested algorithms in Fig. 4, the RRT* algorithm excels in trajectory smoothness, distance optimization, and obstacle avoidance, thoroughly exploring the search space after 1000 iterations. Our improved RRT* algorithm ranks second, efficiently focusing on the path to the destination. RRT-Connect performs the worst in these aspects due to its bidirectional strategy.

In terms of computational efficiency, RRT-Connect is the best, followed by Lazy RRT, which only considers obstacles after reaching the goal. Our improved algorithm, more efficient than Lazy RRT, classic RRT, and RRT-Connect, modifies RRT*'s approach to enhance efficiency.

RRT*'s extensive space coverage results in superior solutions. While classical RRT and Lazy RRT explore broadly, their suboptimal node linkages yield less smooth trajectories. Our improved algorithm concentrates on the path between start and goal, outperforming the other three algorithms but not matching RRT*'s optimality.

Based on the numerical simulations shown in Fig. 5(a-d), we observe that while the RRT* algorithm produces the best trajectory, it has an extremely high computational cost, nearly 10 times that of the other algorithms. Our improved algorithm, though slightly higher in computational load than the others, delivers significantly better planning results, approaching those of RRT*. This is achieved by focusing computational efforts on relevant spaces during exploration and avoiding unnecessary computations in large, unimportant areas. Additionally, our algorithm excels at finding optimal collision avoidance paths when encountering obstacles.

The improved algorithm incorporates a steer function and a growing function, resulting in the fewest number of states in the planning result. It also shows better stability in mean value and data distribution compared to the other algorithms. This improved performance is due to a more efficient exploration strategy, distinguishing it from the classical RRT algorithm.

4. Multi-layer optimization process

In this section, we propose a motion planning optimization methodology for the space robot arm. Our goal is to generate a desired trajectory that enables the end-effector to safely and efficiently approach the capture position. To achieve this, we begin by generating a trajectory from a random initial position.

In order to optimize the approaching process to the target and minimize the interaction between the base satellite and target, the end-effector of the robot arm needs to be gradually approaching the grasp point with the minimum torque and kinetic energy.

The motion planning problem is not complicated if we only have one or two optimization indexes. However for a space robots capture mission, several problems have to be considered simultaneously, which are obstacle avoidance, redundant robotics system multiple solution, kinematics and dynamic problem, this makes the optimization problem complicated. In light of these objectives, the multi-layer optimization process is proposed.

4.1. Multi-layer optimization

As portrayed in Fig. 6, the multi-layer optimization process comprises 5 layers in total. The first layer is the tree-expanding layer. The rest layers are optimization layers, which include the obstacle clearance layer, kinematics layer, dynamic layer(energy and torque layer).

Tree expanding layer

The RRT* algorithm, introduced in the previous chapter, plays a crucial role in the multi-layer optimization system as the exploration algorithm employed to search the planning space between different indices. To support the multi-layer optimization process, we employ the enhanced RRT* optimization algorithm presented in the previous chapter. This algorithm is responsible for exploring the space within this layer and interacts with other layers by extending branches in specific directions.

In Eqs. (7) to (8) the RRT* tree grows in the tree extending layer, which is more like a base layer, growing without any restriction or requirements, just exploring the space. As portrayed in Fig. 7.

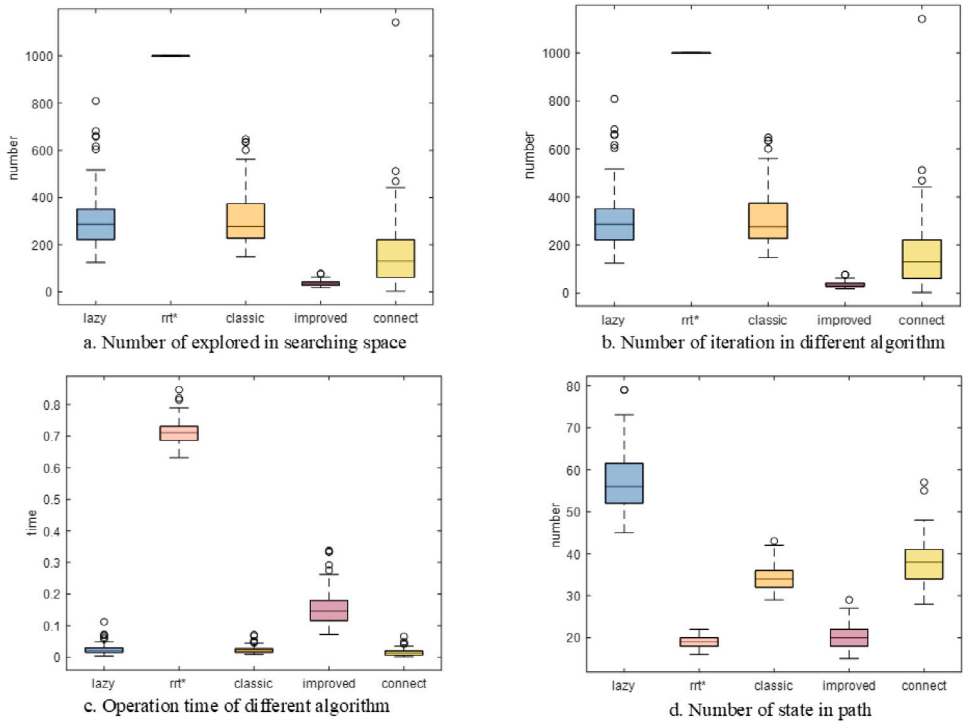


Fig. 5. (a-d) Algorithm operation statistics for different algorithm.

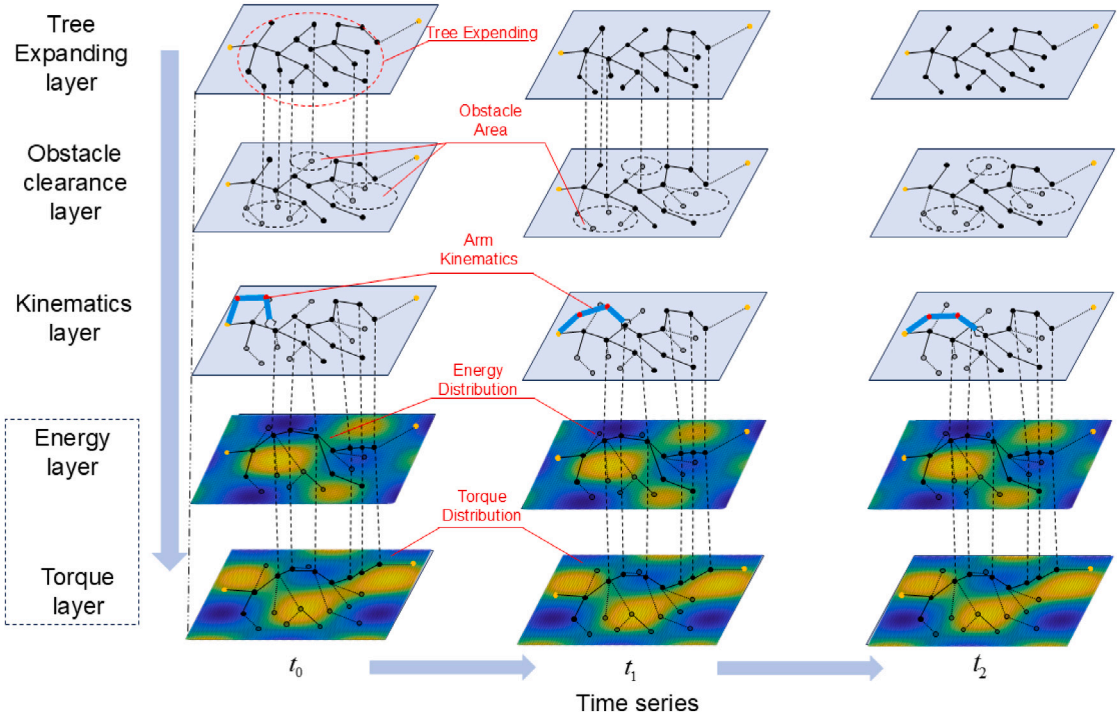


Fig. 6. Multi-layer optimization process.

Obstacles clearance layer

For all sample-based planning algorithms, obstacle avoidance is a major function, which enables the algorithms to solve the problem under constraints quickly, unlike RRT-lazy [30]. Avoiding obstacles alone is not enough for optimizing a space robot arm. Collision risk varies with distance, impacting energy consumption, manipulability, and safety. The end effector must stay safely away from obstacles without excessive detours. This method balances safety and optimization by using a repulsive force strategy instead of traditional collision checking. The repulsive force, defined below, ensures adequate obstacle clearance while maintaining efficient movement.

$$F = \frac{k}{exp(log_a^{dis/d_{safe}})} \tag{12}$$

Where d_{safe} is the minimum safety distance between the end effector and obstacles, dis is the distance between the end effector and

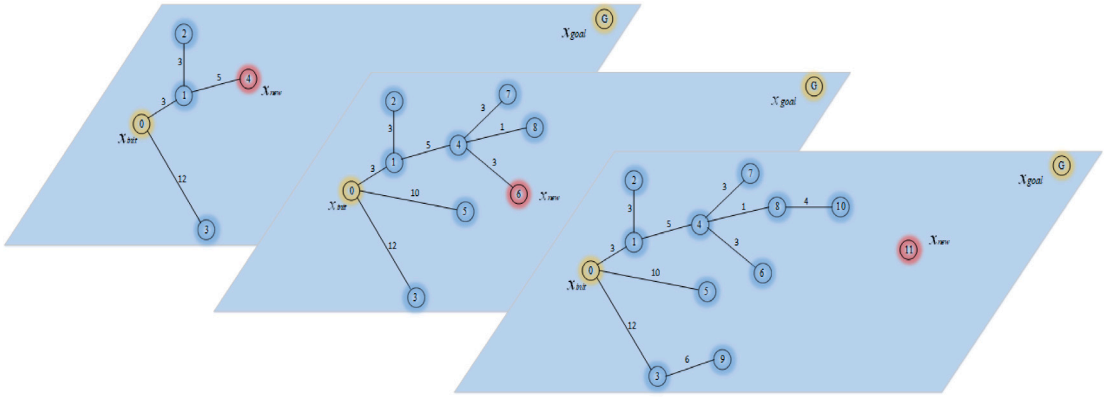


Fig. 7. Tree expanding layer.

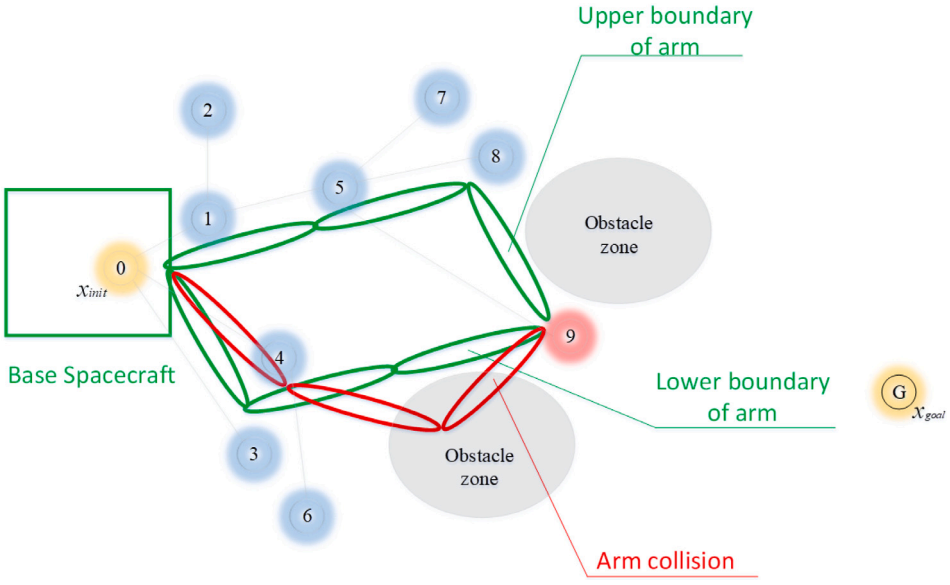


Fig. 8. Feasible solution zone search in the kinetic layer.

obstacles, and k is a user-set parameter to fine-tune the force F.

Kinetic layer

In the kinetic layer, the problem of multiple solutions will be addressed. In the kinetic layer, the problem of multiple solutions will be addressed. In the tree expanding layer, the trajectory of the end-effector is determined in the Cartesian space while obtaining the configuration and angular velocity of the robot arm in the joint space. In practical robotics systems, redundancy often leads to multiple solutions from the Cartesian space to the joint space. Two key considerations arise: Firstly, ensuring that neither the arm nor the joints come into contact with obstacles, not just the end-effector. Secondly, not only must the end-effector's position consider obstacle avoidance, but also the positions of each joint and link. As shown in Fig. 8, feasible configurations of the manipulator at each path point are calculated (the green configurations), and configurations that would result in collisions are eliminated (the red ones). Based on the usable configurations in the green configuration space, a feasible solution space is generated.

Energy layer

The initial two layers, the kinetic and obstacle clearance layers, focus on constraint satisfaction. In the following energy and torque layers, optimization problems related to joint velocity and acceleration are addressed. These dynamic layers correspond to the kinetic layer. Once the end-effector's position is determined in joint space, angular velocity and acceleration are calculated to optimize the robot arm's movement. The main objective here is to optimize the robot arm's angular velocity using energy considerations. Based on the position of the robot arm, the energy distribution is proposed as follows:

$$T = \sum_{i=1}^{3} \frac{1}{2} m_i v_i^T v_i + \frac{1}{2} w_i^T I_i w_i + \frac{1}{2} m_{base} v_{base}^T v_{base} + \frac{1}{2} w_{base}^T I_{base} w_{base} \tag{13}$$

where, T is the energy of the robotics system, m_i, v_i, w_i, I_i are the robot arm mass, linear velocity, angular velocity, inertia tensor of the ith robot arm respectively. According to Eq. (13), the energy can be calculated.

As shown in Fig. 9, around every new node of the tree, the optimization process is based on the distribution of the energy, which is a kind of local optimization. Four steps are included in this optimization process: *Sample*, *Pickup*, *Replace*, *Replace*. When the iteration begins, *Sample* is the first step, here, normally a circular region with radius, is defined around a new position of the end-effector. In this circular region several nodes are randomly sampled as backup nodes. *Replace* is the second step, according to the gradient of the energy field where nodes are located. Every node offsets a certain distance. The offset function is set as:

$$\vec{D}_{offset} = \frac{|k_i|}{|k_{max}|} \vec{k}_i S \tag{14}$$

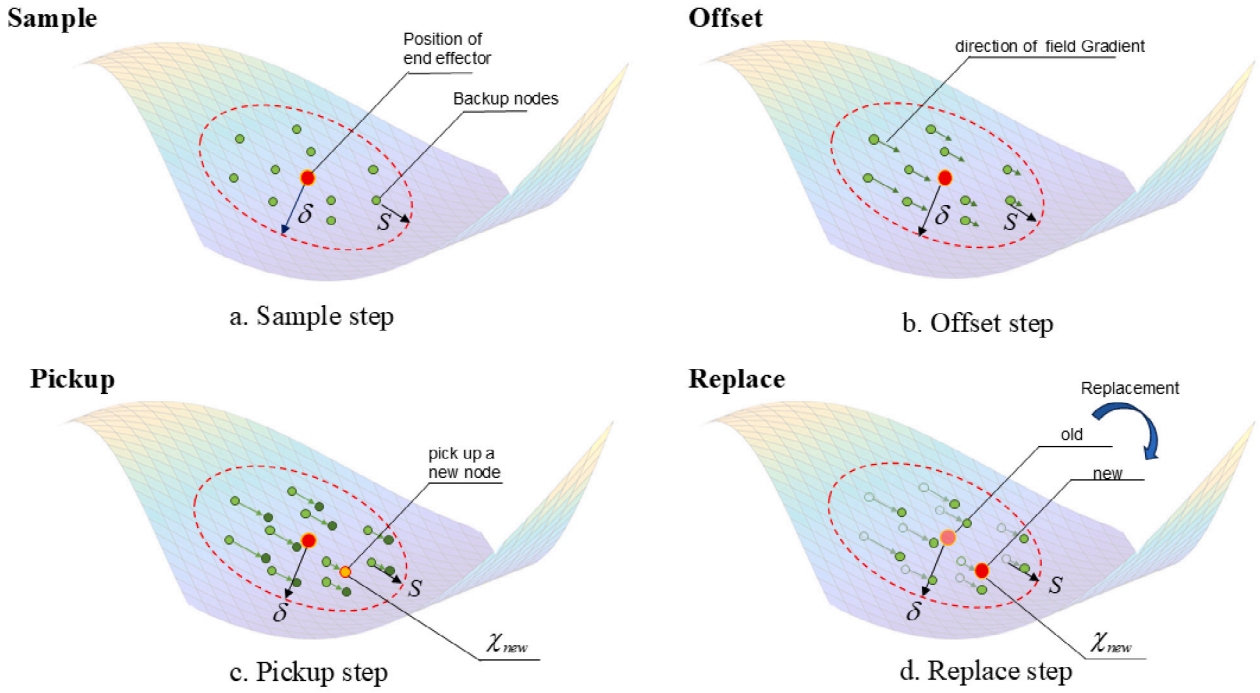


Fig. 9. (a-d) The energy distribution around a new RRT node.

Where k_i is the gradient of the energy distribution at i-the node, k_{max} is the maximum gradient in this region, S is the distance from the node to the edge of this square alone the direction of the gradient of this node. *Pickup* is the third step, in case of the algorithm stuck in to the local optimal. In this step, a node with the lowest energy is selected with a probability of "p" Alternatively, with a probability of "1-p", a node is randomly chosen from the *Replace* nodes, which is χ_{new}. This approach aims to diversify the search and increase the algorithm's chances of escaping local optima. *Replace* is the last step of this process which marks χ_{new} as a new node. This iteration will repeat a certain number of times set by the user or iterate until the energy of χ_{new} converged.

Energy and Torque layer with greedy policy

In the energy layer, the angular velocity, in the form of energy, is optimized, but the torque of each joint is also needed to optimize the angular acceleration. To address this, the torque layer is proposed, which is similar to the energy layer. According to Eq. (6) the distribution of torque of joint can be calculated. The torque layer also employs the four-mode iterative optimization process. However, unlike the energy layer, velocity information is required at each step prior to acceleration optimization. Therefore, a proposed approach called the Greedy Policy Energy & Torque is utilized. As shown in Fig. 10, in each iteration, when waypoint 1 is optimized from the energy layer, the velocity of this waypoint is generated which is $\dot{\theta}_1$. Then, the acceleration for this waypoint $\ddot{\theta}_1$ can be optimized as well, the difference between velocity and acceleration optimization is that the velocity of the next waypoint is needed for the torque optimization.

According to the Energy & Torque greedy Policy the optimal the acceleration ($\ddot{\theta}_1$) is computed, and via this acceleration and the velocity of the next waypoint ($\dot{\theta}_2$), the acceleration of waypoint 2 ($\ddot{\theta}_2$) is computed. In this optimization process, only the minimum value of velocity and acceleration is as follows.

$$\dot{q_{i+1}} = arg \max_{\dot{q}_{i+1} \in \dot{q}_{feas}} T_{i+1}(q_{i+1}, \dot{q}_{i+1}) - T_i(q_i, \dot{q}_i) \tag{15}$$

$$\ddot{q_{i+1}} = arg \max_{\ddot{q}_{i+1} \in \ddot{q}_{feas}} \tau_{i+1}(q_{i+1}, \dot{q}_{i+1}, \ddot{q}_{i+1}) \tag{16}$$

where the $\dot{q}_{feas}$ and $\ddot{q}_{feas}$ indicate the feasible solution space for angular velocity and acceleration, $\dot{q}_{i+1}$ and $\ddot{q}_{i+1}$ are the joint angular velocity and acceleration, T_i is the energy distribution off the ith waypoint, τ_{i+1} is the torque distribution in the ith waypoint. This process is repeated from the initial waypoint to the final one. During this repetition, the angular velocity and acceleration are optimized, ultimately converging to an optimal value within this region.

4.2. Simulation of multi-layer optimization

In this section, we will validate the process of a base-free-floating space manipulator capturing a fixed space debris through simulations. The simulation is divided into 2 parts:

(1) Multi-Layer Optimization Process.

- The tree expending & obstacle clearance layer.
- The kinetic layer.
- The Dynamic layer.

(2) Manipulator's Motion Before and After Applying Multi-Layer Optimization.

4.2.1. Multi-layer optimization process

The tree expending & obstacle clearance layer

The provided simulation results illustrate the use of the improved RRT* algorithm for trajectory planning of a robotic arm's end-effector in Cartesian space, corresponding to the tree expansion layers of the multi-layer algorithm.

Fig. 11 illustrates the RRT* algorithm's iterative path optimization. In the initial iteration (a), the tree is sparse with many non-optimal paths avoiding red obstacles. The mid-iteration (b) shows a denser tree with more paths navigating around obstacles toward the goal. By the late iteration (c), the tree expansion is detailed, with many paths avoiding obstacles and some connecting to the goal. In the final iteration (d), the tree is densely expanded, with multiple green paths reaching the goal, and the algorithm identifies the shortest path. This process ensures the robotic arm's end-effector finds an efficient, obstacle-free trajectory.

The kinetic layer

In the kinetic layer, the following factors are primarily considered.

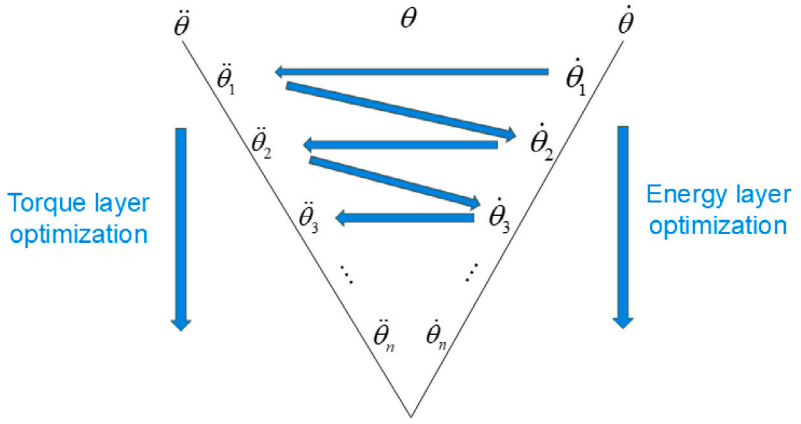


Fig. 10. Energy & Torque greedy policy.

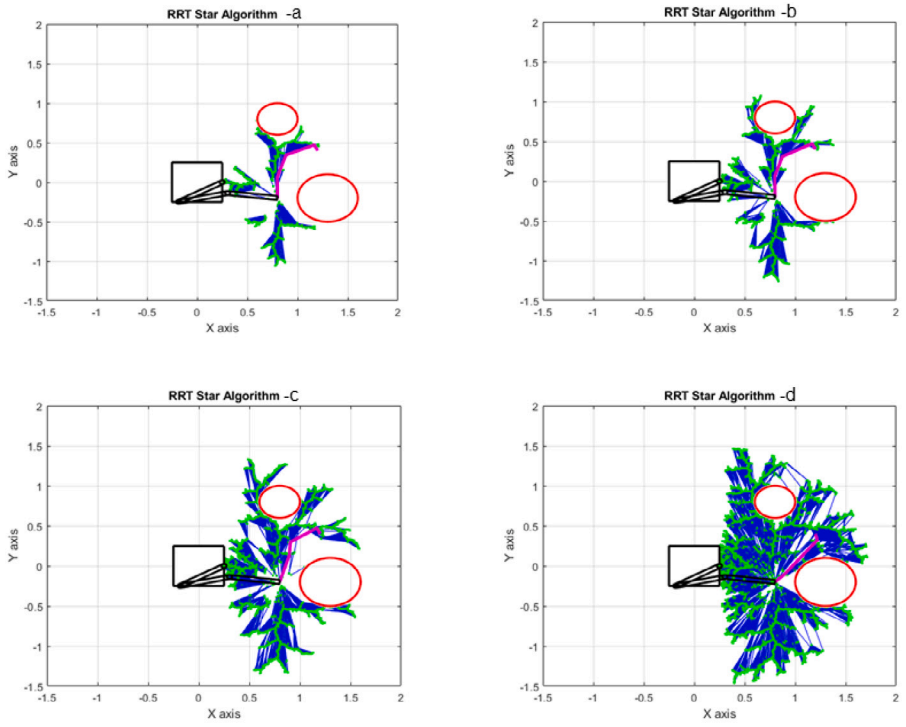


Fig. 11. (a-d) Tree expanding of improved RRT* algorithm.

(1) Flexibility of the Robotic Arm:

The ability to allow more poses at specific points without the arm contacting any obstacles. This indicates higher flexibility and is visually represented by the color gradient in the simulation, with warmer colors indicating greater pose variability.

(2) Manipulability:

Based on the manipulability of the robotic arm, calculated using the Jacobian matrix as described in Eq. (4). This metric evaluates how effectively the arm can move and exert forces in different directions, which is crucial for optimizing the trajectory planning. By integrating these factors into the kinetic layer, the algorithm ensures that the planned paths not only avoid obstacles but also maximize the arm's operational efficiency and effectiveness in complex environments.

In Fig. 12, we can observe that the end-effector of the robotic arm can have different attitudes at the same position. Depending on the allowable range of attitudes (flexibility), the flexibility weight is assigned to nodes during the growth of RRT* branches. The warmer colors indicate higher flexibility, which translates to a higher sampling weight under equivalent conditions.

Also in the kinetic layer, from Fig. 13, another function is the delete the node from the tree with low flexibility(magenta) which at the end of the tree or the back side of the keep out zone, also the node which entered the keep out zone(red).

In Fig. 14, illustrates the distribution of the robotic arm's manipulability within Cartesian space, based on the Jacobian matrix. Since the end-effector can assume various orientations at each point, the manipulability varies accordingly. The variations in manipulability distribution across different points highlight the impact of orientation on the arm's maneuverability at each location.

Fig. 14 depicts the manipulability distribution at each point with different orientations, ranging from 0° to 360°, using box plots. These box plots show the statistical distribution of the robot arm's manipulability, including the mean, maximum, minimum, and interquartile range (25%–75%). Key features highlighted in the Figure include singularity points (blue circles), where values are close to zero and must be avoided.

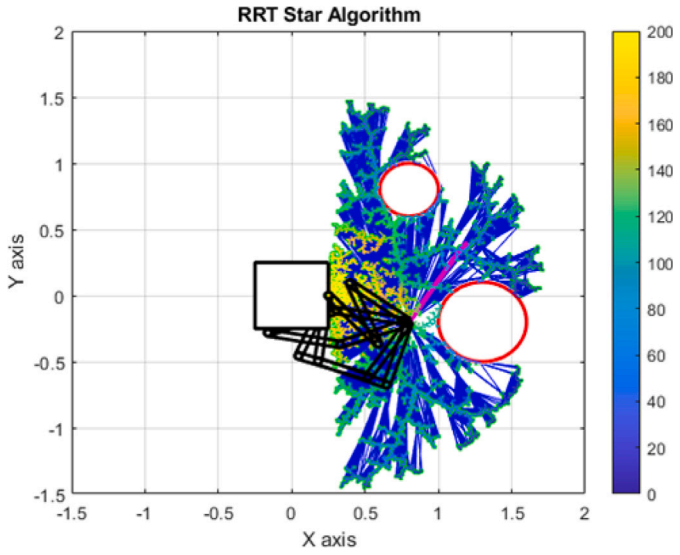


Fig. 12. The flexibility of the robot arm in the same position (allowing the end effector to have more attitude).

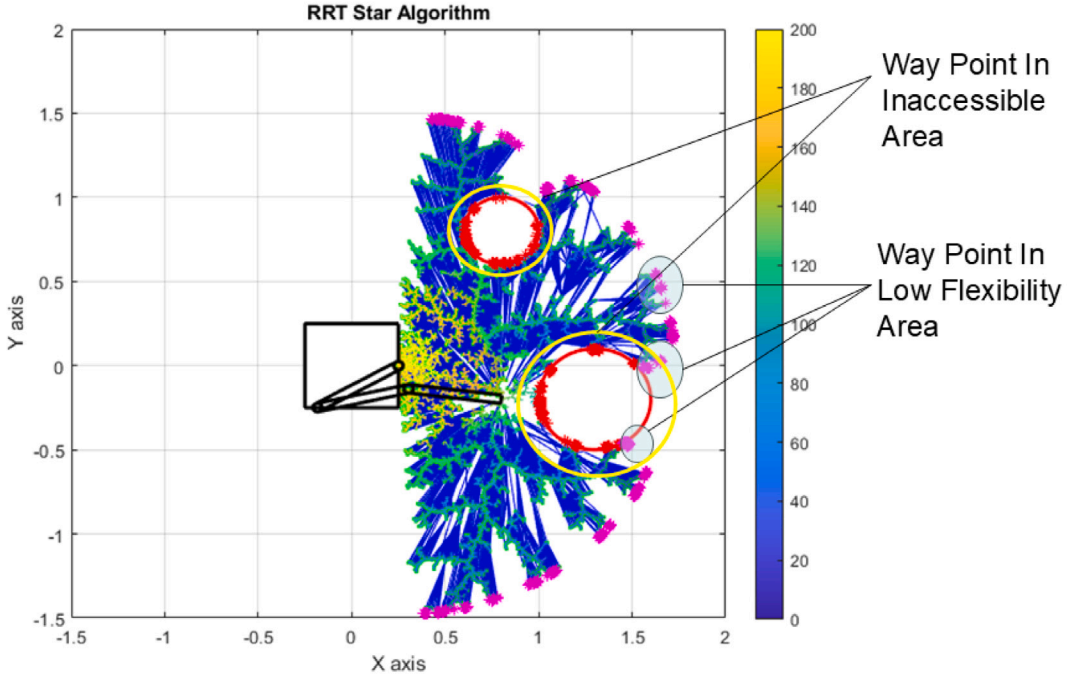


Fig. 13. Eliminate nodes in inaccessible areas(red) and nodes with poor flexibility (magenta).

The path planning of the robotic arm in the kinetic layer takes into account manipulability while simultaneously achieving singularity avoidance.

The Fig. 15, illustrates the trajectory planning results after considering manipulability and flexibility in the kinetic layer. After this layer, the planned path shows some differences. The robotic arm avoids singularity while navigating through regions of high manipulability, ultimately reaching the target. This demonstrates that by integrating both flexibility and manipulability into the kinetic layer, the arm's trajectory becomes more efficient and effective(More detailed simulations and quantitative calculations will be presented in the next chapter).

The Dynamic layer

In the Dynamic layer, we primarily consider the energy distribution of the robotic arm. Based on Eq. (13), lower energy implies that the robotic arm can achieve the desired end-effector position and velocity with lower joint angular velocities and accelerations. This optimization ensures that the arm reaches its target efficiently while minimizing energy consumption. The optimization of the mechanical arm's speed and acceleration has further reduced the impact on the spatial robot's body during motion. This enhancement is crucial for ensuring the overall stability and efficiency of the robotic system.

Similar to the distribution in Fig. 14, Fig. 16 illustrates the energy distribution of the robotic arm's end-effector at various points. At each point, the end-effector can assume different poses, and for each pose, it can have different directions of velocity. The position and pose affect

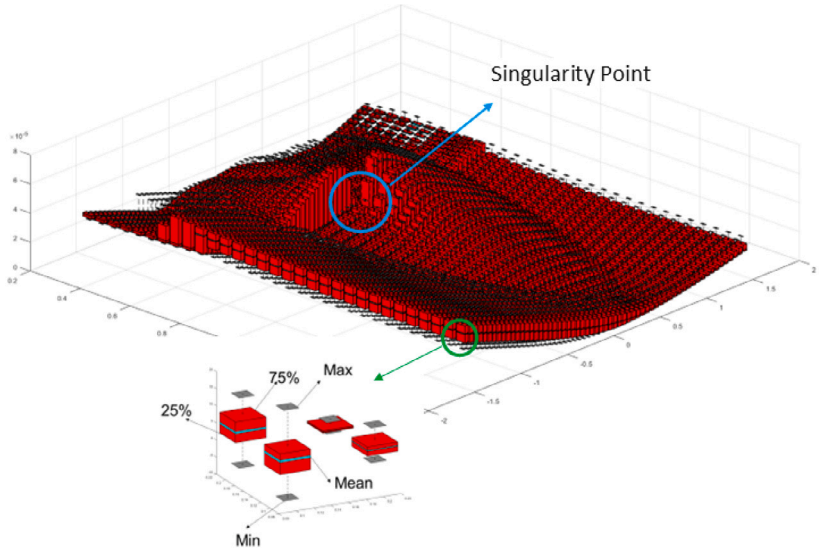


Fig. 14. Manipulation distribution of the robot arm (based on the Jacobian matrix).

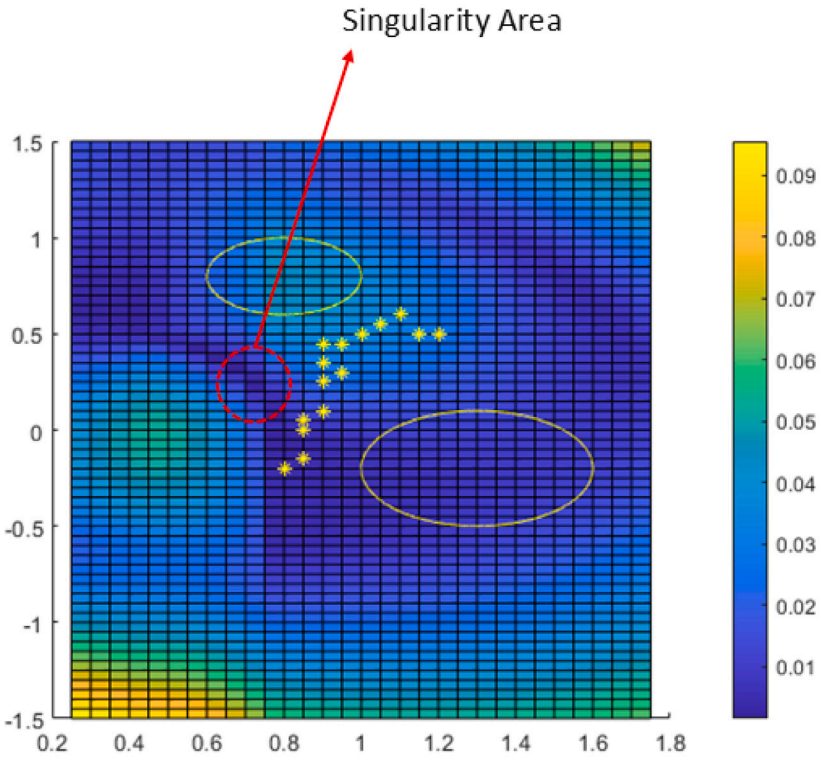


Fig. 15. Planning result based on the kinetic layer.

the link inertias, while the velocity direction influences the angular and linear velocities. These factors lead to variations in the energy distribution of the robotic arm.

The Fig. 16 shows the energy distribution based on different poses and velocity directions at each point. As a result, the planning results derived from this distribution allow the robotic arm to achieve the desired positions and velocities with lower joint angular velocities, as well as reduced end-effector cartesian space linear and angular velocities. Consequently, this reduces the required torques, making the arm's movements smoother and more gentle.

We can observe in Fig. 17 that the energy distribution of the robotic arm also demonstrates the singularity avoidance. Compared to the results in the kinetic layer, there are slight differences to meet the energy optimization requirements. This ensures that the arm's movements remain efficient while avoiding singularities and optimizing for lower energy consumption.

5. Numerical simulation

In this section, numerical simulations are set up to verify the performance of the motion planning results using the multi-layer optimization method. The space robotic system comprises a base spacecraft and a 3-DOF kinematically redundant manipulator. Fig. 18 illustrates the robot arm, which consists of 3 joints, the yellow rectangle is a debris that

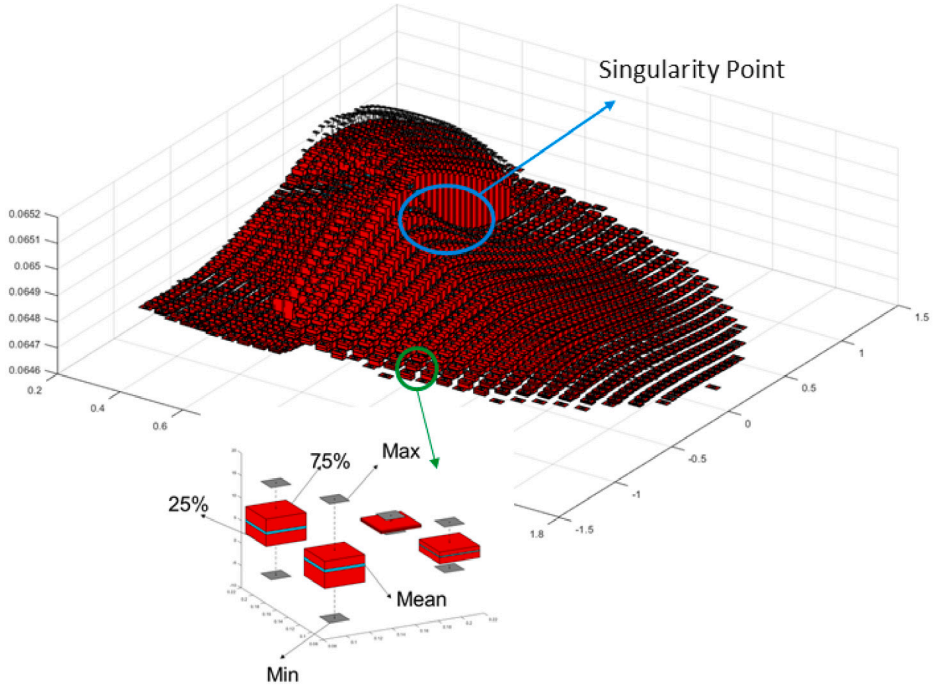


Fig. 16. Distribution of robotic arm energy in the operable space.

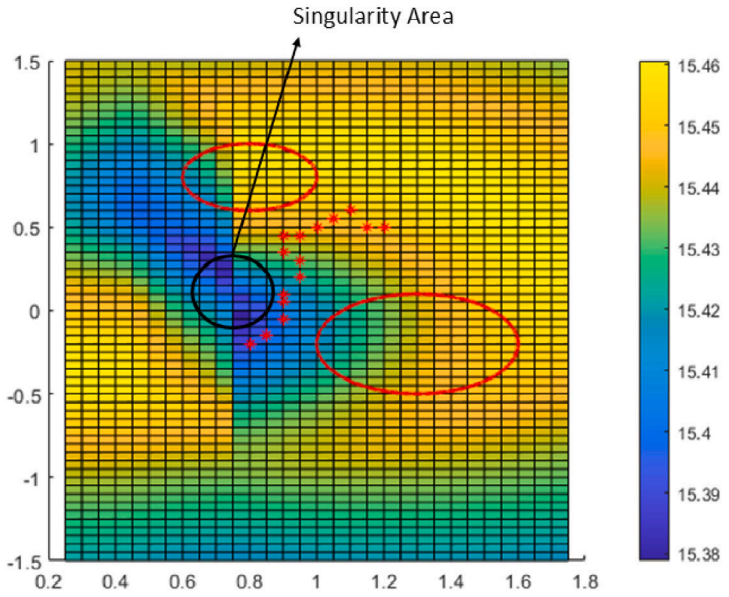


Fig. 17. Planning result based on the dynamic layer.

Table 1
DH parameters of the space robot arm.

Link Offsets $[m]d_i$	Joint Angles $[rad]\theta_i$	Length Lengths $[m]a_i$	Twist Angles $[rad]\alpha_i$
$d_1 = 0$	$\theta_1 = 0$	$a_1 = 0.5$	$\alpha_1 = 0$
$d_2 = 0$	$\theta_2 = 0$	$a_2 = 0.5$	$\alpha_2 = 0$
$d_3 = 0$	$\theta_3 = 0$	$a_3 = 0.5$	$\alpha_3 = 0$

need to be captured. The DH parameter table is also provided below (see Table 1).

5.1. Manipulator's motion before and after applying multi-layer optimization

This section verifies the optimization results obtained from both the kinematic and dynamic layers through simulation. It examines the mechanical arm's joint angles, angular velocities, and accelerations, as well as the motion of the spatial robot's base, and the torque requirements, before and after optimization.

Based on same end-effector trajectories, velocities, and angular velocity requirements, this simulation is designed to verify the differences in joint angular velocity, angular acceleration, and torque when applying the proposed optimization method. The results demonstrate that the optimization significantly improves the smoothness of joint

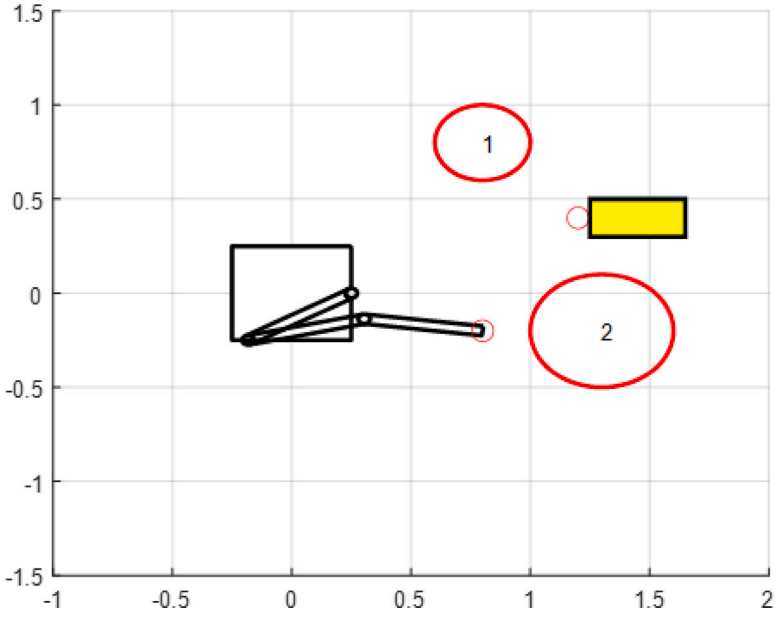


Fig. 18. Space robot arm used in the simulation.

movements while reducing angular velocity, angular acceleration, and torque compared to pre-optimization values. Additionally, disturbances in the base are diminished, with both the magnitude and frequency of fluctuations reduced. Moreover, building upon these, there is a considerable reduction in the mechanical arm's torque requirements.

Based on the simulation results in Figs. 19–21, we can observe that the multi-layer optimization significantly improves the performance of the robotic arm. The angular velocity and acceleration of the joints have noticeably decreased, as evidenced by the comparison between the "Before" and "After" graphs. Additionally, the torque applied to the joints has also been reduced, reflecting lower energy consumption and improved control precision and smoothness. Furthermore, the amplitude of vibrations has been considerably diminished, indicating a smoother and more stable operation of the robotic arm after optimization.

(a). Angular Velocity:

In Figs. 19, the "Before" and "After" graphs show a significant reduction in angular velocity for all joints. The blue lines represent the initial state, while the red lines indicate the optimized state. This reduction suggests that the robotic arm's movements have become more controlled and less abrupt, contributing to overall stability.

(b). Angular Acceleration:

In Figs. 20, similar improvements are seen in the angular acceleration graphs. The optimized state shows a marked decrease in peaks and overall acceleration values. This indicates a reduction in sudden changes in velocity, which can enhance the precision and smoothness of the arm's movements.

(c). Torque:

In Figs. 21, the torque graphs reveal a substantial decrease in the torque applied to each joint after optimization. This not only means reduced mechanical stress and potential wear on the joints but also reflects lower energy consumption and improved control precision and smoothness.

It is worth noting that after optimization, although the velocity and acceleration of joint 2 have increased, its torque has still decreased. This is mainly due to the systematic design of the optimization strategy. By adjusting joint angles and velocities, the optimization method improves the force distribution within the mechanical structure, thereby reducing torque. Even with an increase in acceleration, the balancing effect of the dynamic optimization layer ensures a reduction in the torque of joint 2.

From Fig. 22, it is evident that the optimized joint movements significantly reduce the angular velocity and linear velocity of the base spacecraft. Specifically, the first graph shows the variation of the angular velocity of the base spacecraft over time, with the blue curve representing the pre-optimization state and the red curve representing the post-optimization state. The comparison indicates that the amplitude of the angular velocity decreases significantly after optimization, especially within the first 0 to 50 ms. This suggests that the optimized joint movements effectively reduce the attitude disturbances of the base spacecraft. The second and third graphs display the changes in the linear velocity of the base spacecraft. The comparison between the pre-optimization (blue curve) and post-optimization (red curve) states shows that the amplitude of the linear velocity fluctuations is smaller after optimization. This further demonstrates the effectiveness of the optimized joint movements. The reduction in linear velocity implies that the position disturbances of the base spacecraft are also correspondingly reduced. These results indicate that the optimization of the joint movements effectively controls both the attitude and position disturbances of the base spacecraft. By optimizing the joint movements, not only are the fluctuations in the angular and linear velocities of the base spacecraft reduced, but the disturbances affecting the spacecraft are also significantly decreased, enhancing the overall stability and precision of the spacecraft.

Fig. 23 illustrates the motion planning results with and without obstacle avoidance strategies, and further analyzes the impact of the lack of obstacle avoidance on joint movement. In this part, we utilized the SPART[31] toolkit to setup the simulation scenario. Sub-figure (a) shows the results of applying a multi-layer path planning algorithm. The manipulator starts from the initial position (lower-left corner) and successfully avoids the obstacle areas marked as "1" and "2" along the planned path. During this process, due to the conservation of linear and angular momentum, the motion of the manipulator causes a corresponding drift of the base. Despite this base drift, the manipulator is able to accurately reach the target position (upper-right corner). This result demonstrates that the multi-layer path planning algorithm effectively combines obstacle avoidance requirements with motion control, achieving safe and reliable path planning under the dynamic conditions of a free-floating base.

In contrast, sub-figure (b) shows the motion planning results without obstacle avoidance strategies. Although the end-effector's trajectory manages to avoid the obstacle areas to some extent, the planning

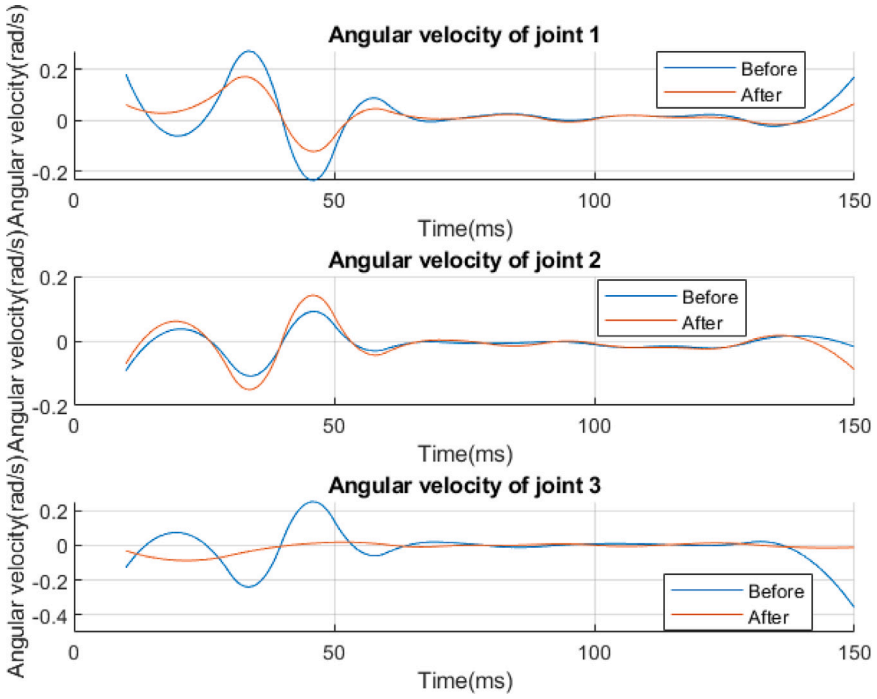


Fig. 19. Angular velocity before and after multi-layer optimization.

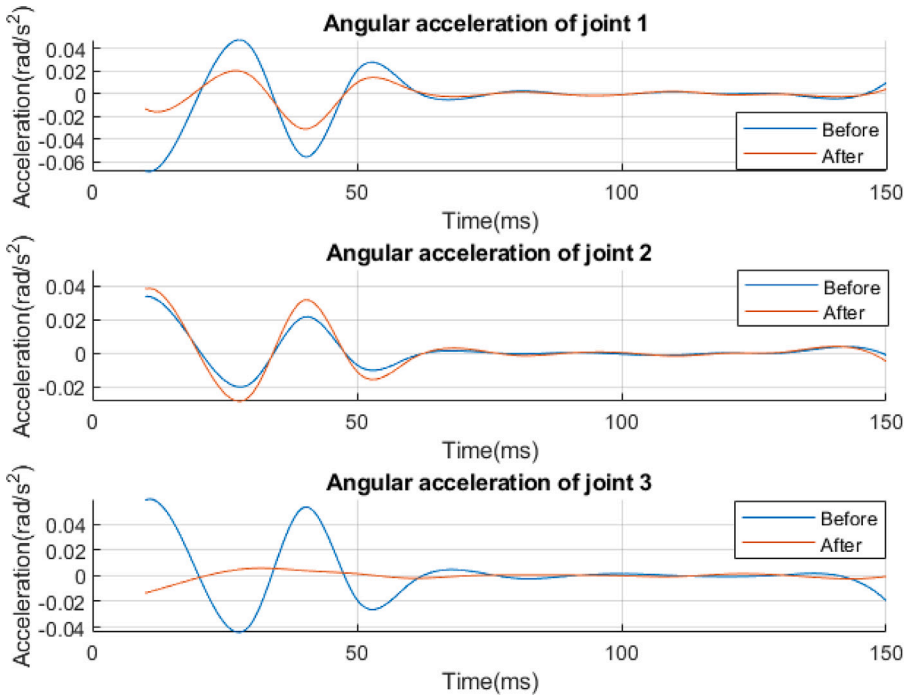


Fig. 20. Angular acceleration before and after multi-layer optimization.

process does not account for joint-level obstacle avoidance. The path is generated using randomly sampled joint configurations, where the constraints are applied only to the position of the end-effector, without considering the overall posture or joint safety of the manipulator. Consequently, certain joints of the manipulator may pass through the obstacle areas "1" and "2", leading to potential collision risks. Moreover, the lack of optimization in the random joint configurations further undermines the safety and stability of the planned path. The comparison between sub-figures (a) and (b) highlights the significant advantages of the multi-layer path planning algorithm. It not only ensures obstacle avoidance for the end-effector but also takes into account the overall posture and joint motion of the manipulator, achieving precise and safe path planning under dynamic base conditions.

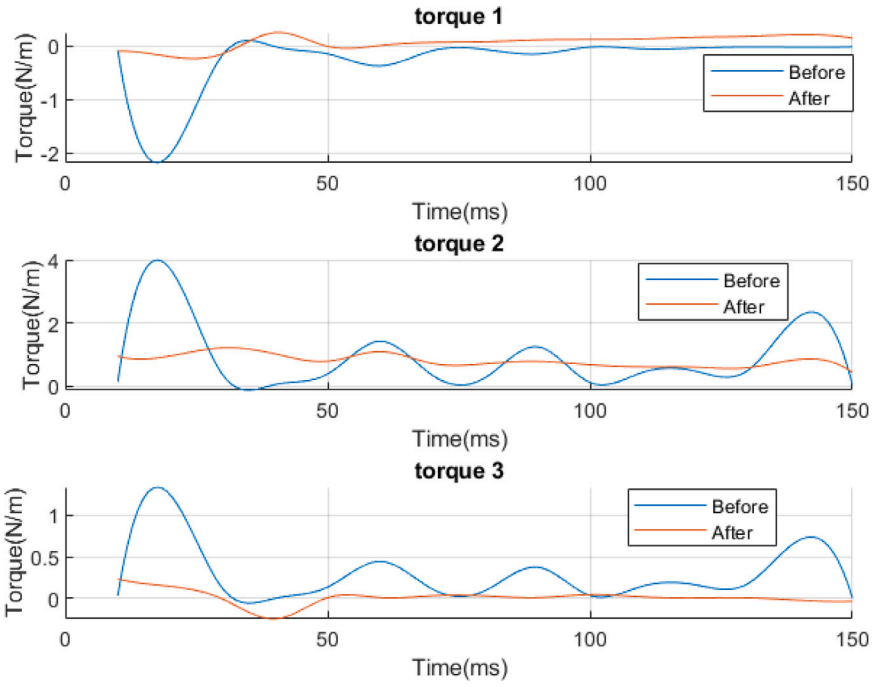


Fig. 21. Torque requirements on joins before and after multi-layer optimization.

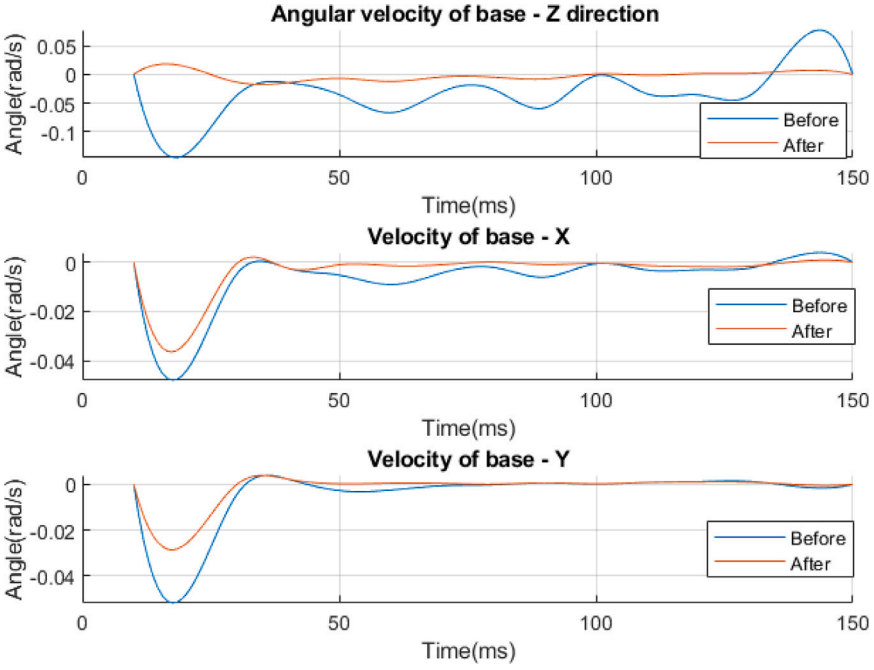
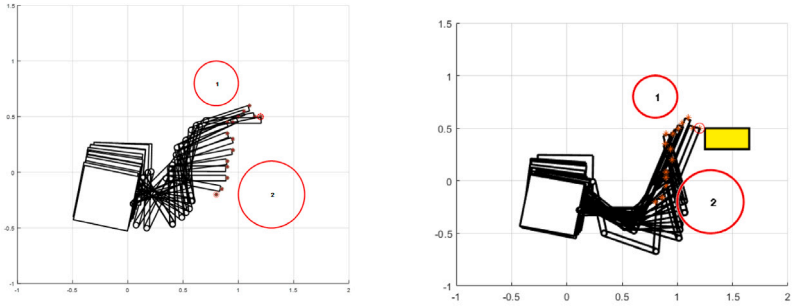


Fig. 22. Disturbance to the base, base angular velocity, linear velocity.

6. Conclusion

In this study, we presented a motion planning method for active debris removal using a free-floating space robot. The proposed approach use an improved version of the RRT* algorithm, enabling obstacle avoidance and generating desired trajectories for the robot's end-effectors. The proposed multi-layer optimization process and a greedy policy contributed to achieving singularity avoidance, joint velocity, and acceleration optimization by utilizing the robot arm's joint energy distribution, torque, and manipulability. The adoption of this motion planning strategy has demonstrated improvements in trajectory smoothness, reduced end-effector vibration, lower energy consumption. The improved RRT* algorithm effectively ensures the robot's trajectory avoids obstacles, while multi-layer optimization eliminates trajectory fluctuations, resulting in smoother paths. The reduction in angular velocity and acceleration peaks has led to more controlled and precise movements, improving the robot's stability. These contributions enhance the motion planning process, making the trajectory smoother



(a) Multi-Layer result (b) Without collision avoidance

Fig. 23. Planning results demonstration.

and reducing torque consumption. Furthermore, this paper lays a solid foundation for our future research on the removal of tumbling debris, which providing a framework for further research.

CRediT authorship contribution statement

Ruipeng Liu: Writing – original draft, Validation, Methodology, Formal analysis, Data curation. **Jian Guo:** Writing – review & editing, Supervision, Resources, Project administration, Investigation, Conceptualization. **Eberhard Gill:** Writing – review & editing, Supervision, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Ellery, Tutorial review on space manipulators for space debris mitigation, Robotics 8 (2) (2019) 34.

[2] R. Jin, P. Rocco, Y. Geng, Cartesian trajectory planning of space robots using a multi-objective optimization, Aerosp. Sci. Technol. 108 (2021) 106360.

[3] M. Wang, J. Luo, J. Yuan, U. Walter, Coordinated trajectory planning of dual-arm space robot using constrained particle swarm optimization, Acta Astronaut. 146 (2018) 259–272.

[4] W. Xu, B. Liang, Y. Xu, Practical approaches to handle the singularities of a wrist-partitioned space manipulator, Acta Astronaut. 68 (1-2) (2011) 269–300.

[5] Y. Nakamura, H. Hanafusa, Inverse kinematic solutions with singularity robustness for robot manipulator control, 108, (3) 1986, pp. 163–171,

[6] J.R. Benevides, V. Grassi, Autonomous path planning of free-floating manipulators using RRT-based algorithms, in: 12th Latin American Robotics Symposium and 3rd Brazilian Symposium on Robotics, LARS-SBR, IEEE, 2015, pp. 139–144.

[7] T. Rybus, K. Seweryn, Application of rapidly-exploring random trees (RRT) algorithm for trajectory planning of free-floating space manipulator, in: 2015 10th International Workshop on Robot Motion and Control, RoMoCo, IEEE, 2015, pp. 91–96.

[8] W.G. Serrantola, V. Grassi, Trajectory planning for a dual-arm planar free-floating manipulator using RRTControl, in: 19th International Conference on Advanced Robotics, ICAR, IEEE, 2019, pp. 394–399.

[9] T. Rybus, M. Wojtunik, F.L. Basmadji, Optimal collision-free path planning of a free-floating space robot using spline-based trajectories, Acta Astronaut. 190 (2022) 395–408.

[10] H. Zhang, Z. Zhu, Sampling-based motion planning for free-floating space robot without inverse kinematics, Appl. Sci. 10 (24) (2020) 9137.

[11] M. Yu, J. Luo, M. Wang, D. Gao, Spline-RRT*: Coordinated motion planning of dual-arm space robot, IFAC-PapersOnLine 53 (2) (2020) 9820–9825.

[12] Y. Qian, J. Yuan, W. Wan, Improved trajectory planning method for space robot-system with collision prediction, J. Intell. Robot. Syst. 99 (2) (2020) 289–302.

[13] Y. Liu, C. Yu, J. Sheng, T. Zhang, Self-collision avoidance trajectory planning and robust control of a dual-arm space robot, Int. J. Control Autom. Syst. 16 (6) (2018) 2896–2905.

[14] S. Liu, Q. Zhang, D. Zhou, Obstacle avoidance path planning of space manipulator based on improved artificial potential field method, J. Inst. Eng. (India) C 95 (2014) 31–39.

[15] R. Mukherjee, Y. Nakamura, Nonholonomic redundancy of space robots and its utilization via hierarchical liapunov functions, in: 1991 American Control Conference, IEEE, 1991, pp. 1491–1496.

[16] J.B. Balaram, H.W. Stone, Automated assembly in the jpl telerobot testbed, in: Intelligent Robotic Systems for Space Exploration, Springer, 1992, pp. 297–342.

[17] R. Lampariello, Motion planning for the on-orbit grasping of a non-cooperative target satellite with collision avoidance, i-SAIRAS 2010 (2010).

[18] R. Lampariello, On grasping a tumbling debris object with a free-flying robot, IFAC Proc. Vol. 46 (19) (2013) 161–166.

[19] K.W. Lee, H. Kojima, P.M. Trivailo, Applications of Optimal Trajectory Planning and Invariant Manifold Based Control for Robotic Systems in Space, IntechOpen London, 2011.

[20] H. Panfeng, X. Yangsheng, L. Bin, Global minimum-jerk trajectory planning of space manipulator, Int. J. Control Autom. Syst. 4 (4) (2006) 405–413.

[21] Y. Umetani, K. Yoshida, Resolved Motion Rate Control of Space Manipulators with Generalized Jacobian Matrix (Ph.D. thesis), Tohoku University, 1989.

[22] Z. Bin, Joint-space trajectory planning for robots under multiple constraints, J. Mech. Eng. 47 (21) (2011) 1–6.

[23] Y.T. Zhao, B. Zheng, H.L. Ma, A new method of 6-DOF serial robot's trajectory planning under multi-constraints, Appl. Mech. Mater. 602 (2014) 1352–1357.

[24] Q.-L. Hu, Y.-Z. Wang, Z. Shi, Minimum torque trajectory planning algorithm for free-floating space robot, J. Harbin Inst. Tech. 43 (11) (2011) 20–24.

[25] S.M. LaValle, From dynamic programming to RRTs: Algorithmic design of feasible trajectories, in: Control Problems in Robotics, Springer, 2003, pp. 19–37.

[26] S.M. LaValle, J.J. Kuffner Jr., Randomized kinodynamic planning, Int. J. Robot. Res. 20 (5) (2001) 378–400.

[27] C. Zhou, M. Jin, Y. Liu, H. Liu, Motion planning for redundant free-floating space robot with local optimization of reaction torque and joint torque simultaneously, in: 2016 IEEE International Conference on Robotics and Biomimetics, Robio, IEEE, 2016, pp. 344–349.

[28] J.J. Craig, Introduction to Robotics, Pearson Educacion, 2006.

[29] S.M. LaValle, Rapidly-Exploring Random Trees: A new Tool for Path Planning, Research Report 9811, SAGE Publications, 1998.

[30] K.G. Yavari, Mohammadreza, M. Mehrandezh, Lazy steering RRT*: An optimal constrained kinodynamic neural network based planner with no in-exploration steering, in: 19th International Conference on Advanced Robotics, ICAR, IEEE, 2019.

[31] J. Virgili-Llop, et al., SPART: an open-source modeling and control toolkit for mobile-base robotic multibody systems with kinematic tree topologies, 2017.