PATH PLANNING RESEARCH ON GRAPE PICKING ROBOTIC ARM BASED ON IMPROVED RRT ALGORITHM
/
基于改进 RRT 算法的葡萄采摘机械臂路径规划研究

Yifan HU[1,2)], Jianjun QIN[1,2*)], Luyang WANG[1,2*)], Xifu CHEN[1,2)], Yue ZHAO[1,2)]
[1)] Beijing University of Civil Engineering and Architecture, School of Mechanical-electronic and Vehicle Engineering, Beijing/China
[2)] Beijing Engineering Research Center for Building Safety Monitoring, Beijing / China;
Tel: +86 18855174840; E-mail: ginjianjun@bucea.edu.cn
DOI: https://doi.org/10.35633/inmateh-74-73

Keywords: *grape picking; improved RRT algorithm; robotic arm; path planning*

ABSTRACT
The robot's operation in a grape orchard environment is often disrupted by obstacles such as vines and leaves, resulting in low fruit picking efficiency. To achieve stable obstacle avoidance, an improved RRT algorithm based on global adaptive step size and target-biased sampling was developed. First, the kinematic equations of the grape-picking robotic arm were established using the PoE method, and both forward and inverse kinematics calculations were performed to determine the robot's workspace. Then, to address the issues of lack of target orientation and other shortcomings in the traditional RRT algorithm when planning collision-free paths, dynamic updating and global adaptive step size strategies were proposed. Simulation experiments conducted using MATLAB software demonstrated that our improved RRT algorithm, compared to the RRT, RRT_informed, and RRT_star algorithms, offered advantages in terms of lower planning time, fewer sampling points, and shorter path lengths in both 2D and 3D map scenarios. Finally, grape-picking experiments were conducted in both a laboratory setting and a real orchard. The results demonstrated that the average path planning time using the proposed algorithm was shorter compared to baseline algorithms, effectively validating the efficiency and practicality of the algorithm.

摘要
机器人在葡萄果园环境中作业会受到藤枝叶等障碍物的干扰，导致果实采摘效率低。为实现稳定避障，研究出一种基于全局自适应步长与目标偏置采样的改进型 RRT 算法。首先，通过 PoE 法建立了葡萄采摘机械臂运动学方程，进行了正、逆运动学计算，并计算出了机械臂工作空间。然后，针对传统 RRT 算法在规划无碰撞路径时缺乏目标导向性等问题，提出了动态更新和全局自适应步长策略，应用 MATLAB 软件进行了仿真实验，验证了我们改进后的 RRT 算法相对于 RRT 算法、RRT_informed 算法和 RRT_star 算法在二维和三维地图场景中，具有规划耗时低、采样点个数少以及路径长度短的优点。最后，分别在实验室和真实果园进行了葡萄采摘试验，路径规划的平均时间相较于基线算法更短，有效验证了算法的高效性。

INTRODUCTION
Robotic technology has rapidly advanced, providing robotic arms with significant application prospects in agricultural harvesting (*Zhang et al., 2022*). For harvesting robotic arms, the performance of the path planning algorithm directly impacts the efficiency and accuracy of the harvesting process (*Yang et al., 2023*). In harvesting environments, immature fruits, vegetables, and grapevines are often randomly distributed around ripe fruits (*Yu et al., 2022*), increasing the difficulty of harvesting operations. Therefore, efficiently planning a high-quality obstacle-avoidance path for harvesting robotic arms is critical.

Among path planning algorithms for robotic arms, the Rapidly-exploring Random Tree (RRT) algorithm (*LaValle S. et al., 1998*) demonstrated superior exploration capabilities in high-dimensional spaces compared to algorithms such as A* (*Hart P.E. et al., 1968*), Ant Colony Optimization (*Dorigo M. et al.*, *1996*), and Artificial Potential Field (*Khatib O. et al., 1986*), including RRT-Connect (*Kuffner J.J. et al., 2000*), RRT* (*Karaman S. et al., 2011*), Informed RRT* (*Gammell J.D. et al., 2014*), and RRT*-Smart (*Islam F. et al., 2012*). These variants significantly enhanced search efficiency and path quality, driving continuous progress in the field of path planning. In the realm of RRT algorithm improvements, *Gao et al. (2023)*, developed a path planning algorithm, BP-RRT*, based on backpropagation neural networks and an improved RRT* algorithm.

This algorithm introduced a distributed sampling method, transitioning from global search to local search in stages. A neural network model predicts the number of nodes required at each stage, improving the efficiency of path optimization. *Pohan et al., (2023)*, proposed a novel path re-planning method, RRT-ACS+RT, based on the Rapidly-exploring Random Tree Star (RRT-ACS) algorithm and Ant Colony System. This method incorporated a rule-template set based on the mobile robot in dynamic environmental scenarios during the path re-planning process. Through extensive experiments, the authors demonstrated that the proposed method outperforms other algorithms. *Shi et al., (2022),* introduced a dual-arm robotic obstacle-avoidance path planning method, GA_RRT, based on target probability bias and cost functions. During random state generation, the algorithm calculated cost functions and selected the point with the lowest cost as a sub-node. For collision detection, the primary arm performed obstacle-avoidance path planning against static obstacles, while the secondary arm considered both static and dynamic obstacles, treating the primary arm as a dynamic reference point. *Mohammed et al. (2020)* proposed an improved RRT*N algorithm, which used a probabilistic distribution strategy to generate new nodes. Nodes closer to the target had a higher generation probability, forming a more focused tree structure along the robot-to-target connection. Simulations and experiments verified the effectiveness and robustness of the RRTN algorithm, demonstrating its potential in complex environments. *Cao et al. (2023)* proposed an enhanced RRT algorithm combining goal bias strategies and the Artificial Potential Field method, achieving significant improvements in iteration counts, planning speed, path length, and path smoothness. *Jia et al. (2023)* developed a collision-free bidirectional RRT algorithm (CGB-RRT) and a flexible obstacle-avoidance path planning strategy based on the RRT algorithm, successfully addressing obstacle-avoidance challenges in complex environments. *Alam et al. (2023)* proposed the FC-RRT* algorithm for energy-efficient motion planning in industrial robots, especially for pick-and-place tasks. The algorithm optimized motion trajectories by generating nodes along predefined directions and calculating energy consumption using a circular-point approach. By applying the work-energy principle to the rotational axes of a 6DOF industrial robot, energy consumption was reduced by 1.6% to 16.5% compared to kinematic solutions and the traditional RRT* algorithm. In addition, the intelligent fuzzy adaptive RRTN path planning method (FA-RRTN) was proposed by Khattab et al. (2023), the waypoint simplification and smoothing RRT method (WSS-RRT) proposed by *Gültekin et al. (2023),* and the algorithm combining the metaheuristic Salp Swarm Algorithm (SSA) with the RRT algorithm (IRRT-SSA) proposed by *Muhsen et al. (2024)* had all provided valuable insights for this paper.

When addressing complex obstacles, existing studies still face challenges in maintaining high harvesting efficiency. To further improve efficiency, an enhanced RRT algorithm for harvesting robotic arms was proposed. By introducing dynamic update strategies for the sampling area and a global adaptive step size strategy, the algorithm's performance was significantly improved. The effectiveness of the enhanced RRT algorithm was validated in both simulated orchard environments and real-world orchard experiments.

MATERIALS AND METHODS

Background

Standardized vineyards are primarily categorized into two types: trellis systems and pergola systems. As shown in Figure 1, trellis systems are typically constructed with one or two rows of vertical posts aligned along the grapevine rows. The distance between posts is generally 30–60 cm, with the trellis height ranging from 150 to 180 cm. The width between trellis rows usually measures 150–300 cm. The grape cluster picking points are typically located at a height of 80–150 cm from the ground. Currently, the grape-picking robot developed is designed for vineyards using trellis systems. In the future, an electric telescopic mechanism will be incorporated to meet the harvesting requirements of vineyards utilizing pergola systems.

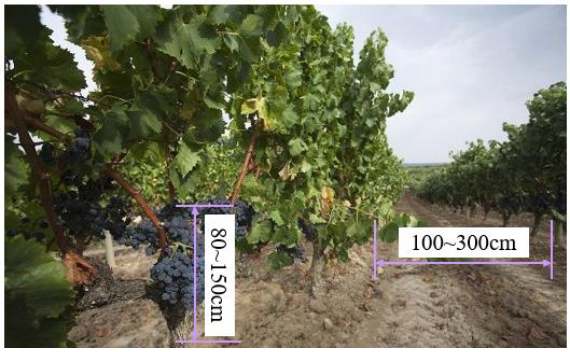


Fig. 1 - Standardized Vineyard - Trellis Structure

Forward kinematic analysis

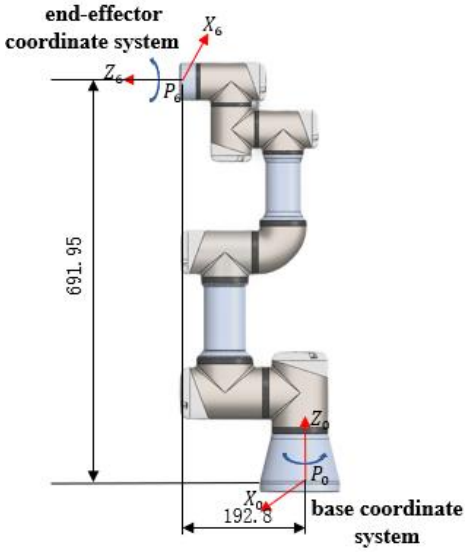
To meet the requirements of grape harvesting, this study employed the Universal Robots UR3 robotic arm. The UR3 is a collaborative robotic arm with a payload capacity of 3 kg, a self-weight of 11 kg, a reach of 500 mm, and a repeatability of 0.1 mm. These performance parameters adequately satisfy the demands for flexibility and precision in grape harvesting.

For mathematical modeling of the UR3 robotic arm, the Product of Exponentials (PoE) method was adopted. Compared to the traditional Denavit-Hartenberg (DH) parameter method, the PoE method eliminated the need for complex coordinate frame establishment and instead focuses on the properties of each joint of the robotic arm. This made the modeling process more intuitive and concise, particularly well-suited for the multi-degree-of-freedom characteristics of the UR3 robotic arm.

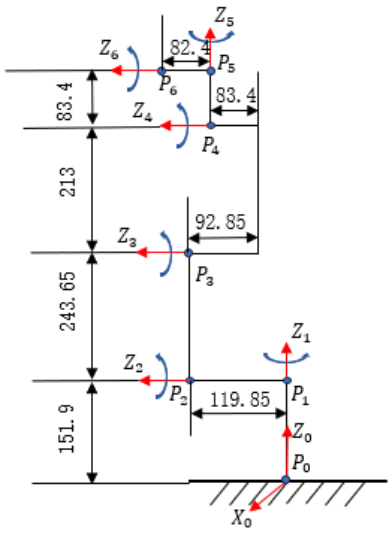
To model using the PoE method, it was necessary to determine the end-effector pose in its initial configuration, the screw axes of all joints relative to the base frame (S1, …, Sn), and the joint variables (θ1, …, θn). Fig 2(a) illustrates the initial state of the robotic arm, from which the end-effector pose matrix E can be derived, as shown in Equation (1).

$$E = \begin{bmatrix} -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & -192.8 \\ 0 & -1 & 0 & 691.95 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

The selection of the base coordinate system and end-effector coordinate system in Figure 2(a) follows the right-hand rule. Figure 2(b) illustrates the screw axes of each joint along with the points PPP on each axis, where the rotation direction of the screw axes is indicated by arrows. Since joints 1 to 6 are rotational joints, the coordinate values of any point on the joint axes are given in the base coordinate system. Typically, points with physical significance, such as the intersection of the joint and the link, are selected for this analysis.



（a）Initial state of the robotic arm



（b）Screw axes of each joint and the points P on each axis

Fig. 2 - PoE modeling method

Table1

Parameter table of joints 1 to 6

n	w_n	P_n	v_n
1	(0, 0,1)	(0 ,0 ,151.9)	(0 ,0 ,0)
2	(0,-1,0)	(0 ,-119.85,151.9)	(151.9,0 ,0)
3	(0,-1,0)	(0,-119.85,395.55)	(395.55,0,0)
4	(0,-1,0)	(0 ,-110.4,608.55)	(608.55,0,0)
5	(0 ,0, 1)	(0 ,-110.4,691.95)	(-110.4,0,0)
6	(0,-1,0)	(0 ,-192.8,691.95)	(691.95,0,0)

As shown in Table 1, the parameter table for the PoE method modeling is provided, $\omega_n \in R^3$ and $v_n \in R^3$ represents the angular velocity and linear velocity of joint n. $\hat{s}_n$ denotes the unit vector along the positive direction of the joint axis, which represents the rotational axis of joint n relative to the base coordinate system. The velocity expression consists of two parts: the left part corresponds to the linear velocity caused by rotation, and the right part represents the linear velocity along the rotation axis. $[S_n]$ is the skew-symmetric matrix, calculated from S_n. As shown in Equation (2), $[S_n]$ is presented in Equation (3), and the homogeneous transformation matrix is given in Equation (4). Once T_6^0 is calculated, the forward kinematic analysis of the robotic arm is completed. The solution to the forward kinematics aids in the subsequent determination of the robotic arm's workspace.

$$S_n = \begin{bmatrix} w_n \\ v_n \end{bmatrix} = \begin{bmatrix} \hat{s}_n \\ -\hat{s}_n \times P_n \end{bmatrix} \tag{2}$$

$$[S_n] = \begin{bmatrix} [w_n] & v_n \\ 0 & 0 \end{bmatrix} \tag{3}$$

$$T_6^0 = e^{[S_1]\theta_1} \ldots e^{[S_{n-1}]\theta_{n-1}} e^{[S_n]\theta_n} M \tag{4}$$

Inverse Kinematic Analysis

The inverse kinematics analysis of the robotic arm requires the known position and orientation of the robot's end-effector relative to the base coordinate system, in order to determine the six joint angles of the robotic arm. Assuming that the pose matrix of the robot's end-effector relative to the base coordinate system is given by Equation (5), the six joint angles of the robot can be derived from matrix calculations, as shown in Equation (6). The first three columns of the matrix in equation (5) represent the rotational relationship between the original coordinate system and the new coordinate system, while the last column of the first three rows indicates the translation of the origin of the new coordinate system relative to the original one. The final row ensures the matrix is in homogeneous form, enabling the unified representation of both rotation and translation transformations. There are a total of 8 solutions, with $\theta_1, \theta_2, \theta_4$ and θ_6 having unique solutions, and θ_3 and θ_5 each having two solutions. In Equation (6), s_i, c_i represents $\sin\theta_i, \cos\theta_i$, c_{ij}, s_{ij} represents $\cos(\theta_i + \theta_j), \sin(\theta_i + \theta_j)$, c_{ijk}, s_{ijk} represents $\cos(\theta_i + \theta_j + \theta_k), \sin(\theta_i + \theta_j + \theta_k)$, a_i denotes the link lengths, and d_i represents the link distances. The value of m, n, s, t is given in Equation (7). The aforementioned inverse kinematics calculations lay the foundation for the subsequent trajectory interpolation.

$$T_6^0 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$\begin{cases} \theta_1 = A\tan 2(m,n) - \\ A\tan 2(-d_2 - d_4, \\ \pm\sqrt{m^2 + n^2 - (d_2 + d_4)^2}) \\ \theta_2 = A\tan 2(s_2, c_2) \\ \theta_3 = \pm\arccos(\dfrac{r_{14}^2 + r_{34}^2 - a_3^2 - a_2^2}{2a_2a_3}) \\ \theta_4 = A\tan 2(s_{234}, c_{234}) - \theta_2 - \theta_3 \\ \theta_5 = \pm\arccos(s_1a_x - c_1a_y) \\ \theta_6 = A\tan 2(s,t) - A\tan 2(-s_5, 0) \end{cases} \tag{6}$$

$$\begin{cases} m = p_y - a_y d_6 \\ n = p_x - a_x d_6 \\ s = c_1 n_y - s_1 n_x \\ t = c_1 o_y - s_1 o_x \end{cases} \quad (7)$$

Design of the Robot Hardware

This study developed an experimental platform for the grape-picking robot. As shown in Figure 3, the 3D model of the main equipment primarily consists of a wheeled chassis and a robotic arm. The robotic arm is equipped with a self-designed end effector, two RealSense D435i depth cameras, and a 16-line LiDAR (model: Robosense RS-Helios-16P). The upper computer for the robotic arm is a laptop with 32GB of RAM and an RTX 3060 GPU, while the upper computer for the chassis is a Jetson Xavier NX with 64GB of memory.

For the task of grape picking, a specific end effector was designed. The end effector consists of an electric parallel two-finger gripper and electric scissors. The electric parallel two-finger gripper is the Z-EFG-20P model from HuiLing Technology, which offers an adjustable gripping force ranging from 30 N to 80 N. The electric scissors are self-designed, and their working principle involves a stepper motor driving a lead screw to control the opening and closing of the scissors. Additionally, a limit switch is used to fix the maximum and minimum opening angles of the scissors. The scissors blades extend 2 cm beyond the electric gripper, ensuring that grape clusters can be effectively cut. Both the electric gripper and electric scissors can be controlled via I/O, and after connecting them to the control cabinet of the UR3 robotic arm, the end tool can be controlled via the robotic arm teach pendant or the ROS2 in the upper-level computer. The 3D model of the end effector is shown in Figure 3. The distance from the TCP to the end of the flange is 176.2 mm. This distance, referred to as the TCP offset, is used when calculating the workspace of the robotic arm.

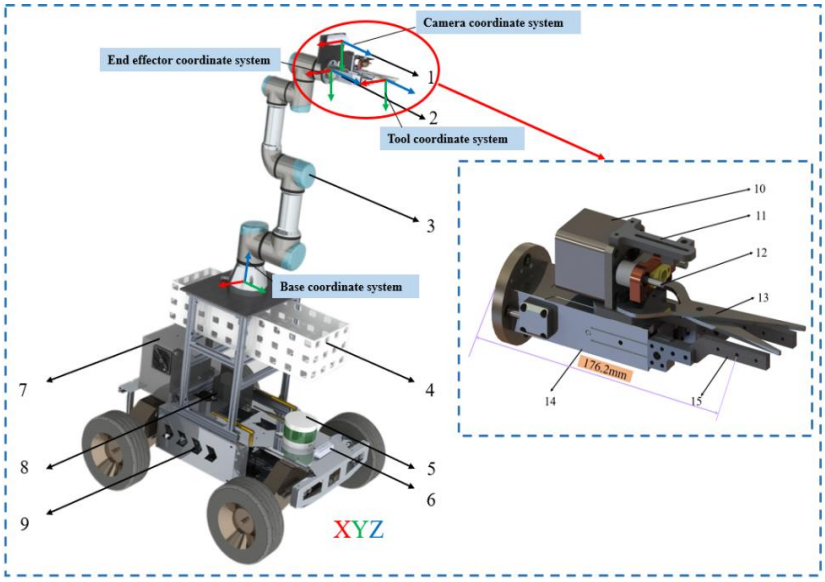


Fig. 3 - 3D Model of the Grape Picking Robot.

1. RGB-D Camera; 2. End Effector; 3. UR3 Robotic Arm; 4. Storage Basket; 5. LiDAR; 6. RGB-D Camera; 7. Robotic Arm Control Cabinet; 8. Chassis Host Computer; 9. Chassis; 10. Stepper motor; 11. Limit switch; 12. Lead screw; 13. Picking scissors;14. Electric parallel gripper; 15. Gripper extension section

Workspace analysis

The workspace refers to the set of all spatial points that the robot's end-effector can reach during the grape-picking process. Its shape and range are critical factors influencing the robot's operational performance. After completing the forward and inverse kinematics analysis of the robotic arm, the Monte Carlo method was employed in MATLAB to calculate the workspace of the UR3 robotic arm. The Monte Carlo method randomly generates multiple end-effector position coordinates using the forward kinematics equations (Equation 4), and then incorporates the TCP offset to obtain the workspace of the grape-picking robot (including the tool), as shown in Figure 4. The workspace of the UR3 robotic arm approximates an ellipsoid, with the range in the X and Y directions being approximately -600 mm to 600 mm, and the range in the Z direction being approximately -500 mm to 800 mm. When planning paths for grape-picking tasks using an improved RRT algorithm, the randomly sampled points should be confined within the workspace of the robotic arm.

(a) Three-dimensional workspace *(b) XOY Planar projection* *(c) XOZ Planar projection* *(d) YOZ Planar projection*

Fig. 4 - Workspace of the grape-picking robot

RRT algorithm improvement and simulation experiment

Improvement Strategies for RRT Algorithm

The RRT algorithm is a sampling-based path planning algorithm that uses the starting point as the root node, increases the number of nodes through random sampling and piling, and connects the nodes to generate a random tree. Nodes that do not satisfy the constraint requirements are discarded during the generation of subsequent nodes. When the random tree contains the goal point or enters the goal area, a route connecting the start point to the endpoint exists.

This paper proposes two improvement strategies for the traditional RRT algorithm: the sampling region dynamic update strategy and the global adaptive step size strategy, aimed at enhancing the algorithm's performance. First, the initial improvement strategy involved dynamically updating the sampling region. Traditional RRT algorithms often exhibited a lack of goal orientation when planning collision-free paths, sometimes growing in the reverse direction. This phenomenon primarily arose from random sampling in unsuitable regions, resulting in numerous ineffective branches. To address this shortcoming, a dynamic update mechanism for the sampling region was designed. The key to this strategy was to gradually reduce and optimize the sampling space as the random tree grows, particularly shifting focus to the region near the target point. This approach encouraged the random tree to grow toward the target point, defining positive growth as the expansion of random tree nodes toward the area between the current node and the target point. This helped continuously reduce the distance to the target point and ensured effective tree expansion. Conversely, if the growth was oriented toward the starting point, it is considered to have departed from the target point, thereby increasing the distance from it. This type of growth was defined as reverse growth and regarded it as a non-ideal form of expansion. Sampling only within the valid region may cause the algorithm to get stuck in a local optimum. Therefore, if the current node fails to expand after 30 iterations, it is considered invalid and removed from the random tree.

The second improvement strategy involved global adaptive step size adjustment. Traditional RRT algorithms typically utilize a predetermined fixed step size for tree expansion, failing to dynamically adjust this parameter based on the information gathered during the search process. This could result in improper selection of the step size, either too large or too small, adversely affecting the algorithm's convergence speed and the quality of the paths, particularly in narrow or obstacle-intensive regions, where exploration efficiency might be significantly diminished. The improved RRT algorithm considers the relationship between environmental complexity and obstacle characteristics by calculating factors such as the ratio of obstacle area to total area, the average distance between obstacles, and the reciprocal of the number of obstacles. In addition, different weighting coefficients were introduced to comprehensively determine the environmental complexity metric w_1. This allowed the algorithm to self-calculate an adaptive initial step value to more effectively adapt to environmental changes.

$$step = step_{start} = \left[N_{size} * w_1 \right] \tag{8}$$

Then expand the new node q_{new} in steps, as shown in Equation (9):

$$q_{new} = step * \frac{|q_{nearest} - q_{rand}|}{\| q_{nearest} - q_{rand} \|} + q_{nearest} \tag{9}$$

In Equation (8) N_{size} was the map size, and $[N_{size}*w_1]$ denotes $N_{size}*w_1$ rounded down to the calculation results, during the random expansion of the random tree, the strategy continuously acquired environmental information, and the improved RRT algorithm could adaptively adjust the step size in the region with more obstacles, as shown in Equation (10):

$$step = step * w_2 \tag{10}$$

In Equation (10), w_2 was the weight of step size reduction, and w_2 was selected according to the number of obstacles, in general, w_2 was taken as 0.7 for simple environment, and w_2 was taken as 0.3 for complex environment, and the initial step size would be restored after the extended tree covered the complex obstacle region, in summary the ability to realize the global adaptive step size strategy effectively increased the global search capability of the improved RRT algorithm. Below is the pseudocode for the improved RRT algorithm.

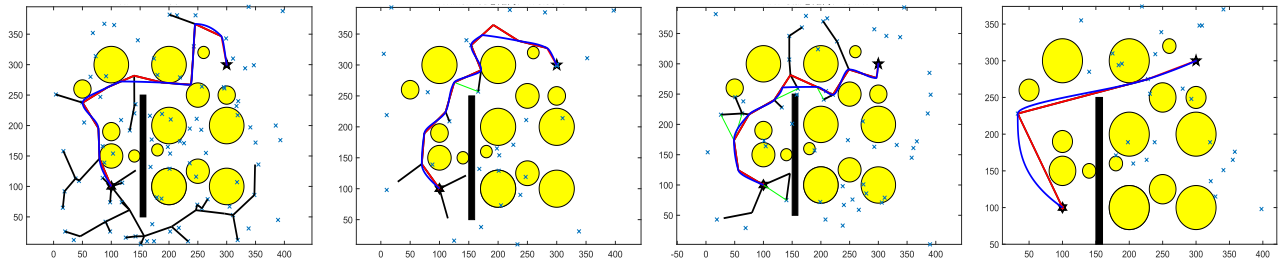
Algorithm:improved_rrt

Input: *q_start*, *q_goal*, M (map)
Output: Path from *q_start* to *q_goal* (if found)
Initialize random tree with root *q_start*
Set initial valid region near *q_goal*, invalid region elsewhere
while (random tree has not reached *q_goal*):
    Update Sampling Region
    if random tree grows towards *q_goal*:
        Shrink valid region toward target
    else if reverse growth:
        Expand valid region and remove invalid branches
    Sample *q_rand* within valid region
    Find nearest node *q_nearest*
    Calculate step size:
    if environment is complex:
        step = step * w_2
    else:
        step = step * w_1
    Steer towards *q_rand*:
    q_new = Steer(*q_nearest*, *q_rand*, step)
    Check for collision:
    if ObstacleFree(*q_nearest*, *q_new*):
        Add *q_new* to random tree
        if *q_new* is near *q_goal*, return path
if no path found after max iterations, return failure

Two-dimensional scene simulation

In this study, the traditional RRT algorithm, RRT_star algorithm, RRT_informed algorithm, and the improved RRT algorithm were used to plan paths on a two-dimensional map. Parameters such as the number of obstacles, the positions of obstacles, and the random seed were modified in MATLAB to test each algorithm 60 times, in order to validate the performance of the improved RRT algorithm for path planning in two-dimensional environments. The path-planning capabilities of the four algorithms were evaluated based on three indicators: planning time, number of sampling points, and path length. If an algorithm took less time, required fewer sampling points, and produced a shorter path, it was considered to have better path-planning performance. In the experiments, the two-dimensional map had dimensions of 450 cm by 450 cm, with the start point for path planning at (100 cm, 100 cm) and the endpoint at (300 cm, 300 cm). The fixed step size for the traditional RRT, RRT_informed, and RRT_star algorithms was set to 30 mm.

As shown in Figure 5, the hexagonal star shape represents the starting point of the path planning, and the pentagonal star shape represents the endpoint. The blue × symbols indicate randomly generated sample points, while the yellow circles and black rectangles represent obstacles. If a sample point is randomly generated inside an obstacle, it is not used during the path planning process. The black lines represent the connections between the sample points during the exploration of the random tree, the red line indicates the path before smoothing, and the blue line represents the path after smoothing.



(a) RRT algorithm *(b) RRT_informed algorithm* *(c) RRT_star algorithm* *(d) improved RRT algorithm*

Fig. 5 - Comparison of four RRT algorithms in two-dimensional maps

Table 2

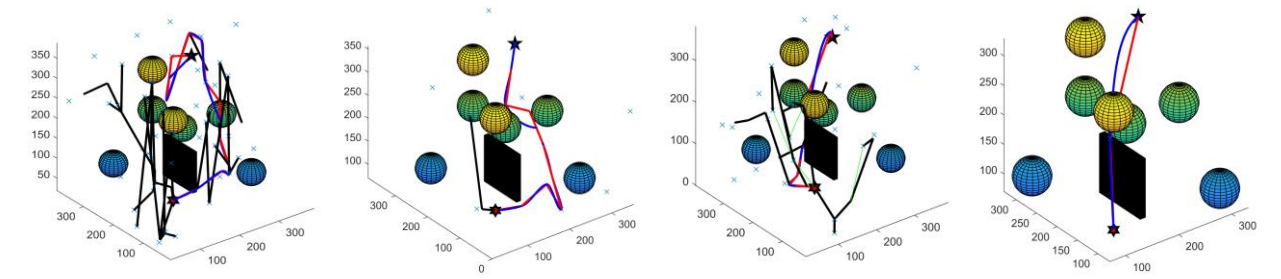
Comparison of indicators of four RRT algorithms for two-dimensional scenarios

Types of algorithms	Time consumption/s	Sampling points/each	Path length/cm
RRT	13.05	93	547.48
RRT_star	2.62	54	454.48
RRT_informed	2.93	45	475.96
Improved RRT	0.77	29	421.28

The results comparing the planning effects and indicators of the improved RRT algorithm presented in this paper with those of the other three RRT algorithms on two-dimensional maps were shown in Fig 5 and Table 2. Figure 5 shows that the traditional RRT algorithm generates numerous invalid nodes during path planning, resulting in directionless planned paths. The RRT_informed and RRT_star algorithms generate fewer nodes; however, the planned paths are more tortuous. Nevertheless, our proposed improved RRT algorithm not only significantly reduces the number of invalid sampling points but also efficiently plans smooth paths. Table 2 indicates that our improved RRT algorithm, compared to the traditional RRT algorithm, RRT_informed algorithm, and RRT_star algorithm, resulted in a 94.1%, 70.61%, and 73.72% decrease in planning elapsed time, a 68.82%, 46.3%, and 35.56% reduction in the number of sampling points, and a 23.78%, 7.31%, and 11.49% decrease in path length, respectively. In summary, compared to the other three RRT algorithms, the performance of our improved RRT algorithm has improved.

Three-dimensional scene simulation

The path planning in the three-dimensional scene of this study is similar to that in the two-dimensional scene. The length, width, and height of the three-dimensional map are 450 cm, with the starting point at (100 cm, 100 cm, 100 cm) and the endpoint at (300 cm, 300 cm, 300 cm). In the three-dimensional simulation scene, obstacles have been modified to colorful spheres and rectangular cuboids, while the other settings remain similar to those in the two-dimensional simulation scene. Similarly, several parameters were modified and the improved RRT algorithm was tested along with the three baseline algorithms in MATLAB, conducting 60 trials. The number of sampling points and the time recorded are the average values from the 60 trials, as shown in the Table 3.



(a) RRT algorithm *(b) RRT_informed algorithm* *(c) RRT_star algorithm* *(d) improved RRT algorithm*

Fig. 6 - Comparison of four RRT algorithms in three-dimensional maps

Table 3

Comparison of indicators of four RRT algorithms for two-dimensional scenarios

Types of algorithms	Time consumption/s	Sampling points/each	Path length/cm
RRT	5.88	50	844.80
RRT_star	3.57	28	586.07
RRT_informed	0.74	12	555.36
Improved RRT	0.24	2	365.88

The results of the comparison of the planning effects and indicators between the improved RRT algorithm in this paper and the remaining three RRT algorithms on three-dimensional maps were shown in Figure 6 and in Table 3. Figure 6 and Table 3 indicate that the improved RRT algorithm presented in this paper, compared to the traditional RRT algorithm, RRT_informed algorithm, and RRT_star algorithm, results in a 95.92%, 93.28%, and 67.57% decrease in planning elapsed time, a 96%, 92.9%, and 83.33% reduction in the number of sampling points, and a 58.65%, 34.5%, and 34.12% decrease in path lengths, respectively. In summary, our improved RRT algorithm demonstrated strong planning capability on three-dimensional maps.

RESULTS

Planning system design

The simulation in MATLAB was conducted to validate the performance of the improved RRT algorithm. In real-world experiments, the MoveIt2 software in ROS2 and the OMPL library were primarily utilized to implement the RRT algorithm for planning and control. The real-world experiments were mainly divided into three parts: robotic arm path planning, target detection and localization, and the coordination between the chassis and the robotic arm. Figure 7(a) showed the hardware design and communication architecture of the grape-picking robot.

Path planning using the RRT and its variant algorithms involved five main processes: initializing the tree, random sampling, tree extension, path checking, and path optimization. In the real-world experiments, the host computer of the robot was connected to the UR3 robotic arm via an Ethernet cable, and the arm could be controlled using the MoveIt2 software. Various RRT algorithms were encapsulated in the OMPL library, which provided interfaces for path planning. The planned path required optimization, and MoveIt2 offered libraries to achieve trajectory smoothing and interpolation. Cubic spline interpolation was also used for trajectory optimization, and comparative results showed that it produced similarly smooth paths as those generated by the built-in MoveIt2 libraries. The ‘ros2_control’ framework had parsed planned trajectory into specific joint commands, which were then executed via hardware interfaces. Meanwhile, the execution status of the robotic arm was fed back to MoveIt2 through relevant ROS2 topics to enable real-time status monitoring and subsequent planning adjustments. Since multiple points were generated during trajectory interpolation, the inverse kinematics of the robotic arm was required to calculate the joint angles for these points (Equation (6)), allowing smooth movement along the generated trajectory. In most cases, path planning for the robotic arm was conducted in the joint space, but MoveIt2 also provides interfaces for planning based on the TCP (Tool Center Point) if required. The planning process was shown in Figure 7(b).

As shown in Fig 7(c), the target detection and localization process employed the YOLOv9 algorithm to identify grapes and their picking points. Once a picking point was detected by the RealSense D435i camera, the camera API was called to obtain the center position of the bounding box of the grape, along with its coordinates (x, y, z) in the camera coordinate system. Through hand-eye calibration and coordinate transformation of the robotic arm, the coordinates could be converted from the camera coordinate system to the end-effector coordinate system of the robotic arm. The transformed coordinates were then sent to MoveIt2 for path planning. The relevant coordinate systems of the robotic arm and the camera were shown in Figure 3.

The coordination between the chassis and the robot's host computer was achieved through a CAN bus. After the robotic arm completed the picking task for all recognizable grape clusters at the current location, the robot's host computer sent a signal to the chassis host computer to move the chassis forward by 15 cm. Once the chassis reached the new position, the robotic arm resumed the picking task, ensuring an efficient coordinated workflow, the process was detailed in Figure 7(b).

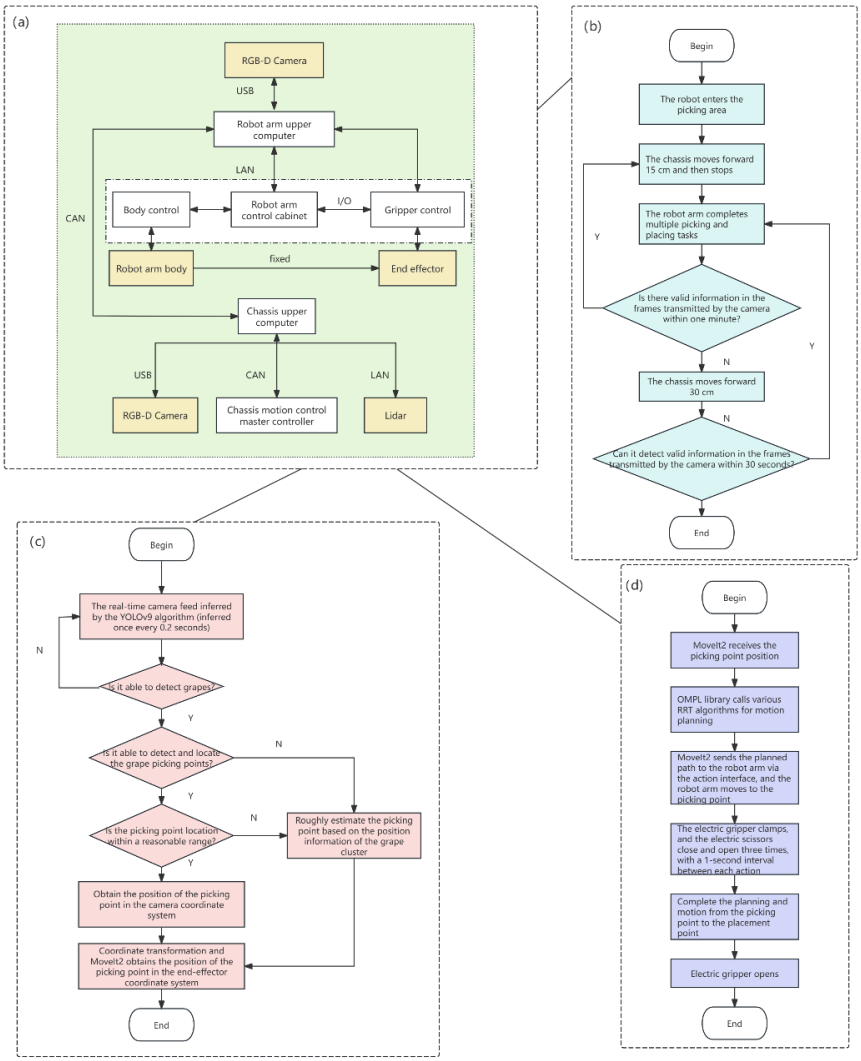


Fig. 7 - Hardware system of the grape picking robot and the process design of each module

(a)Picking Robot hardware design; (b) Chassis planning process;
(c) Object detection and localization process; (d) Robotic arm planning and control process

Indoor experiment

The indoor picking experiment was conducted in a laboratory setting in September 2024, as shown in Figure 8. At the start of the experiment, 15 artificial grape bunches, each with an average weight of 200 g, were placed at varying heights on the vine. During the experiment, the RGB-D camera was first used to identify the grape bunches and locate the picking points. Then, the OMPL library in MoveIt2 invoked the specified algorithms for path planning. After planning, the robotic arm moved at a speed of 1 m/s to the vicinity of the nearest grape bunch for harvesting. Upon completion of the picking task, the upper computer planned a new path to place the grapes into the storage basket. As shown in Fig 8, the process of the picking experiment is illustrated. The three-fingered gripper used here was insufficient in gripping force, so in subsequent real-world experiments, it was replaced with a specialized gripper (Figure 3). For the indoor experiments, path planning was conducted within the joint space, and the use of the end-effector did not affect the validation of the planning algorithm.

(a) recognizing grapes *(b) grabbing grapes* *(c) picking grapes* *(d) collecting grapes*

Fig. 8 - Picking process in the laboratory

The RRT algorithm, RRT_informed algorithm, RRT_star algorithm, and our improved algorithm were utilized for path planning. Each algorithm was tested 16 times, with 8 tests for picking and collecting a single grape bunch, and the remaining 8 for picking and collecting multiple bunches (averaging 4–5 bunches per experiment). The average time for each planning algorithm was recorded in every test. As shown in Table 4, our algorithm demonstrated improved planning efficiency compared to the other three baseline algorithms.

Table 4

Average Planning Time of Four Algorithms in Indoor Experiments

Type of Algorithms	Single-bunch grape picking experiment		Multi-bunch grape picking experiment
	Average Planning Time from Initial Position to Picking Point /s	Planning Time from Picking Point to Storage Basket /s	Total Average Planning Time for the Entire Process /s
RRT	0.42	0.58	5.44
RRT_informed	0.34	0.41	4.03
RRT_star	0.39	0.44	4.12
Ours	**0.29**	**0.40**	**3.87**

Outdoor experiment

In October 2024, a real-environment picking test was conducted at a grape picking garden in Fangshan District, Beijing, which covers an area of approximately 10 acres and uses a double-row trellis system, as shown in Fig 9. Four algorithms were used for the planning experiments, with each algorithm being tested 10 times for multi-bunch grape picking (averaging 6-7 bunches per test). The average planning time was calculated for each algorithm. As shown in Table 5, our improved algorithm still outperformed the other algorithms in terms of planning efficiency in the real-world scenario. However, due to the presence of obstacles in the real environment, the average path planning time for the robotic arm was longer compared to the indoor experiments.

Table 5

Average Planning Time of Four Algorithms in Outdoor Experiments

Types of Algorithm	Average Planning Time / s
RRT	8.53
RRT_informed	6.98
RRT_star	7.11
Ours	6.61

(a) recognizing grapes *(b) grabbing grapes* *(c) picking grapes* *(d) collecting grapes*

Fig. 9 - Picking process in real orchard

CONCLUSIONS

(1) Compared to the traditional RRT algorithm, the improved RRT algorithm presented in this paper incorporates a dynamic sampling region update strategy and a global adaptive step size strategy. Simulation experiments conducted using MATLAB software demonstrated that the improved RRT algorithm required less planning time, utilized fewer sampling nodes, and generated shorter planning paths, thereby highlighting its superiority.

(2) Grape picking experiments were conducted in both laboratory and real orchard environments. In the laboratory setting, the average planning time for consecutive pickings was approximately 3.87 seconds, while in the real orchard, the planning time was 6.61 seconds. This increase in planning time in the real orchard environment is due to branches, leaves, and other obstacles that complicate the robotic arm's path planning. Overall, the path planning time using the improved RRT algorithm was shorter compared to the other three baseline algorithms.

ACKNOWLEDGEMENT

This paper was funded by the Special Funds Program for Basic Research Operating Expenses of Universities under Beijing Municipality (Grant No.X20060), Research Fund Project of Beijing Building Safety Monitoring Engineering Technology Research Center (Grant No.BJC2020K012) and Research on Intelligent Motion Control Design of Quadruped Robot (Grant No.PG2024139).

REFERENCES

[1] Alam, M. M., Nishi, T., Liu, Z., et al. (2023). A Novel Sampling-Based Optimal Motion Planning Algorithm for Energy-Efficient Robotic Pick and Place. *Energies*, 16(19), 6910.

[2] Cao, M., Zhou, X., & Ju, Y. (2023). Robot motion planning based on improved RRT algorithm and RBF neural network sliding. *IEEE Access*, 11, 121295-121305.

[3] Dorigo, M. (1996). The Any System Optimization by a colony of cooperating agents. IEEE Trans. *System, Man & Cybernetics-Part B*, 26(1), 1-13.

[4] Gammell, J. D., Srinivasa, S. S., & Barfoot, T. D. (2014). Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2997-3004.

[5] Gao, Q., Yuan, Q., Sun, Y., & Xu, L. (2023). Path planning algorithm of robot arm based on improved RRT* and BP neural network algorithm. *Journal of King Saud University-Computer and Information Sciences*, 35(8), 101650.

[6] Gültekin, Ayhan, Samet Diri, Yaşar Becerikli. (2023). Simplified and Smoothed Rapidly-Exploring Random Tree Algorithm for Robot Path Planning *Tehnički vjesnik.* 30.3 (2023): 891-898.

[7] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100-107.

[8] Islam, F., Nasir, J., Malik, U., Ayaz, Y, Hasan, O. (2012). RRT∗-smart: Rapid convergence implementation of RRT∗ towards optimal solution. *2012 IEEE International Conference on Mechatronics and Automation*, 1651-1656.

[9] Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7), 846-894.

[10] Khattab, O., Yasser, A., Jaradat, M., Romdhane, L. (2023). Intelligent Adaptive RRT* Path Planning Algorithm for Mobile Robots. *Advances in Science and Engineering Technology International Conferences (ASET). IEEE*. DOI: 10.1109/ASET56582.2023.10180740

[11] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1), 90-98.

[12] Kuffner, J. J., & LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. *Symposia Proceedings*, 2, 995-1001.

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

[14] Mashayekhi, R., Idris, M.Y.I., Anisi, M.H., Ahmedy, I., Ihsan, A. (2020). Informed RRT*-connect: An asymptotically optimal single-query path planning method. *IEEE Access*, 8, 19842-19852.

[15] Mohammed, H., Romdhane, L., & Jaradat, M. A. (2021). RRT* N: An efficient approach to path planning in 3D for Static and Dynamic Environments. *Advanced Robotics*, 35(3-4), 168-180.

[16] Muhsen, Dena Kadhim, Firas Abdulrazzaq Raheem, and Ahmed T. Sadiq. (2024). Improved rapidly exploring random tree using salp swarm algorithm. *Journal of Intelligent Systems*. 33.1: 20230219.

[17] Shi, W., Wang, K., Zhao, C., & Tian, M. (2022). Obstacle avoidance path planning for the dual-arm robot based on an improved RRT algorithm. *Applied Sciences*, 12(8), 4087.

[18] Yu, F., Zhou, C., & Yang, X. (2022). Design and Testing of Tomato Picking Robot for Daylight Greenhouse (日光温室番茄采摘机器人设计与试验). *Transactions of the Chinese Society of Agricultural Machinery*, 53(1), 41-49.

[19] Zhang, W., Zhang, B., & Gong, Y. (2022). Fruit and vegetable picking robotic arm research status and prospects (果蔬采摘机器人机械臂研究现状与展望). *Journal of Chinese Agricultural Mechanization*, 43(9), 232-237, 244.

[20] Zhang, Y., Wen, Y., & Tu, H. (2023). A method for ship route planning fusing the ant colony algorithm and the A* search algorithm. *IEEE Access*, 11, 15109-15118.