Computers and Electronics in Agriculture 215 (2023) 108353

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

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

Workspace decomposition based path planning for fruit-picking robot in complex greenhouse environment

Binhao Chen [a], Liang Gong [a,b,*], Chenrui Yu [a], Xiaofeng Du [a], Jianhuan Chen [a], Shenghan Xie [a], Xinyi Le [c], Yanming Li [a], Chengliang Liu [a]

[a] School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
[b] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai 200240, China
[c] School of Electronic Information and Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

ARTICLE INFO

Keywords:
Fruit picking robot
Collision-free path planning
Local search
Discretized workspace
Robotic grasping

ABSTRACT

The working environment of fruit-picking robots is very complicated. Generally, there are a large number of obstacles such as branches, immature crops, etc. Besides, the agricultural scene is not as static. For sampling-based methods such as Probabilistic RoadMap (PRM) and Rapidly-exploring Random Trees (RRT), they suffer from inferior path quality in obstacle condensed environments. This paper proposes a local search path planning method in the feasible region of fruit-picking robots based on discrete workspace guidance and configuration space exploration correction. It is used to plan the motion of fruit-picking robots for avoiding obstacles and grasping in dynamic and complex agricultural environments. First, the workspace is discretized, which makes the location information of obstacles computable. Then, a connected region path is found to guide the robot to search in the local configuration space. This greatly improves the continuity of the generated paths in the configuration space. The local search is further used to correct the weights of discrete regions of the workspace to find better connected paths with higher planning speed. The experimental results show that the quality of paths planned by proposed method is generally better than that of RRT and RRT-CONNECT, and the planning speed is faster than that of Task-Space RRT (TSRRT). Proposed algorithm enables real-world picking at 10.5 piece/s and a recovery rate of 80.0 %. This method significantly improves the path quality in dynamic and complex agricultural scenarios.

1. Introduction

In agriculture, fruit-picking is a very laborious task that needs to be improved by applying robotic technology. The picking operation requires the robot to identify the target crops and environmental obstacles in a complex and dynamic environment. Then, plan a feasible and high-quality path that allows the robotic arm to quickly reach the desired position and execute the picking action. For multi-joint robots in industrial scenes and there are relatively mature methods to solve such problems, such as PRM (Kavraki et al., 1996), RRT (LaValle and Kuffner 1999), etc. However, in agricultural scenarios, these methods may not be directly applicable. Bac et al. surveyed the literature on autonomous picking robots and pointed out that more than half of the studies did not include motion planning (Bac et al., 2014). The problem is due to the complex and changing environment as compared to industrial scenarios. This results in poor quality paths generated by sampling-based planning methods with many redundant waypoints. To implement path planning in agricultural scenarios, a planning algorithm that can generate high-quality paths is urgently needed.

In collision-free path planning for grasping robots, a variety of algorithms have been proposed, including the artificial potential field and the A* algorithm. Khatib uses the artificial potential field to complete the path planning task of grasping robots (Khatib 1986). In (Van Henten et al., 2003), the A* algorithm was applied to cucumber picking. However, these methods can only solve the planning problem of $2-3$ degrees of freedom (DOF) robots. It is because the spatial properties of obstacles need to be accurately described, which increases the computational complexity exponentially with the increase in the degrees of freedom. Moreover, the actual complex agricultural scene was over-simplified, for example, the cucumber rattan was considered as two cylinders.

Since then, the emergence of algorithms such as RRT and PRM has

* Correspondence author at: School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China.
E-mail address: gongliang_mi@sjtu.edu.cn (L. Gong).

https://doi.org/10.1016/j.compag.2023.108353
Received 31 July 2022; Received in revised form 3 July 2023; Accepted 19 October 2023
Available online 30 November 2023

provided solutions to robot motion planning in industrial scenarios. Yang et al. used RRT to realize path planning for the hybrid manipulator with a high degree of freedom and complex structure (Yang et al., 2017). However, the RRT algorithm has a very slow convergence speed due to the use of a global uniform random sampling strategy.

In order to solve the planning problem in narrow areas, some researchers have proposed a class of methods called 'guided planning', which are similar to proposed method. Holleman proposes to use the central axis of the workspace in PRM to guide planning (Holleman and Kavraki, 2000). Jory Denny introduces the method of workspace medial axis guidance in RRT (Denny et al. 2014), which improves the robot's ability to explore in obstacle gaps. Later, he proposed an optimization scheme and introduced Dynamic Region to guide exploration (Denny et al. 2020). Vonásek searches for an auxiliary path before planning (Vonásek et al. 2009), and uses the auxiliary path to guide the exploration, which improves the path quality and planning speed. Then, he guided the exploration and improved the performance of the algorithm by finding multiple approximate solutions (Vonásek and Pěnička, 2019). Attali synthesizes previous experience and encapsulates many seemingly distinct prior works under the same framework for guided exploration (Attali et al. 2022).

Other researchers, such as Kinston et al. proposed three approaches to explore constrained manifolds, and presented the basic framework of sampling-based planning with manifold constraints (Kingston et al., 2019). Considering the picking process as a multi-link task, path planning involves the multi-modal problem. Hauser et al. discretely sampled the continuous space by constructing a transition tree in the continuous modal transition space to avoid exact computation in the continuous space (Hauser and Ng-Thow-Hing 2011). Plaku et al. proposed the SyCLop algorithm, which is the earliest layered collaborative algorithm for motion planning with dynamic constraints (Plaku et al., 2010). Kinston et al. used SyCLop for multi-modal planning and solved the modal switching problem (Kingston et al., 2020). However, these methods mainly focus on traditional planning problems and have not been used in practical scenarios, especially in complex and dynamic agricultural environments.

In agricultural scenarios, because the environment is dynamic and more complex than the industrial environment, it is difficult to obtain satisfactory planning results with sampling-based planning for robots with low degrees of freedom. The hierarchical collaborative algorithm, frequently used for path planning of ultra-high-degree-of-freedom robots, utilizes the precise obstacle information of the workspace. It generates discrete layers after discretizing the workspace and plans the working path of the robot in the discrete layers. A continuous layer is formed in the continuous configuration space, which is used to verify whether the working path planned by the discrete layer is feasible and adjust the path weight. Due to the use of accurate obstacle information, the quality of the path generated is significantly improved compared with the sampling-based planning methods. It is also more suitable for high-complexity and high-dynamic agricultural scenarios.

Before path planning, grasping in agricultural scenarios also needs to consider system calibration, the recognition of targets and obstacles, grasping strategies, grasping pose estimation and other issues. Yang et al. have developed an efficient Tool Center Point (TCP) calibration method based on inherited constraints of target object (Yang et al., 2021). Tao et al. proposed a method to identify apples, branches and leaves based on color and 3D features (Tao and Zhou 2017). For tomato grabbing, they analyzed the viscoelastic characteristic of tomatoes and derived an efficient method for tomato grabbing (Tao et al., 2017). Guo et al. proposed a state-of-art point cloud processing-based pre-grasp planning method, which was tested on a variety of fruits and obtained reliable pose estimation results (Guo et al., 2020).

In agricultural scenarios, robot motion planning methods in various fruit harvesting tasks have currently been considered. Schuetz et al. designed a global optimization algorithm to determine the optimal motion trajectory of the manipulator when it picks bell peppers in an assembly line to achieve higher picking efficiency (Schuetz et al., 2015). Cao et al. used RRT to complete lychee picking task and introduced the concept of target gravity into the RRT algorithm to speed up the path search (Cao et al., 2019). Ahlin et al. proposed the Void Space path planning for apple picking (Ahlin et al., 2017). However, this method only plans the end effector, ignoring the interaction between the manipulator itself and the environment. Ye et al. introduced the target gravity concept and adaptive coefficient adjustment into the Bi-RRT algorithm to make it applicable in high-dimensional environments (Ye et al., 2021). However, the average computation time for a path is 4.24 s, which greatly affects the picking efficiency. Furthermore, Wang et al. proposed a smooth trajectory planning method for fruit-picking robots, but they mainly focused on smoothing the picking trajectory and ignored the actual difficulty of avoiding obstacles with multi-degree-of-freedom manipulators in complex dynamic environments (Wang et al., 2022).

Some researchers have studied planning based on vision. Mehta et al. achieved precise picking of citrus with 95 % confidence (Mehta and Burks 2014). Strawberry picking was achieved by Han et al. (Han et al., 2012) and Hayashi et al. (Hayashi et al. 2010). However, it is common in picking environments where leaves and stems occlude the target. Using vision-based methods in densely vegetated environments will fail to localize the target crops due to occlusion. Chiang et al. combined deep reinforcement learning with the classic RRT algorithm to provide a more efficient solution to the robotic arm planning problem (Chiang et al., 2019). James et al. used Q-Learning to implement reinforcement learning for robotic grasping in 3D simulation (James and Johns 2016). Lin, Guichao et al. also used reinforcement learning to plan picking paths for guava (Lin et al., 2021). Experiments in the simulation environment have certain scalability, but in the real environment (especially the picking environment) it is highly complex and dynamic, and cannot be fully learned. Robots are likely to fail to move when they encounter unexpected obstacles. Currently, deep reinforcement learning is rarely used in fruit picking robots.

In this paper, we propose a local search path planning method in complex agricultural dynamic scenarios. It uses the information of target crops and environmental obstacles identified for planning. The obtained target and obstacle information is first obtained through binocular target recognition, calibration and shape and position reconstruction. Then, the continuous configuration space is discretized into smaller local areas in the working space. The feasible region of a fruit-picking robot is obtained from discretized workspace guidance and configuration space exploration correction. When the exploration of a local space exceeds the time limit, the algorithm will automatically update the connectivity weights between regions. More high-quality and reliable paths are then generated.

In the next section, Section 2, our hierarchical planning algorithm will be presented in detail. In Section 3, the experimental results will be evaluated, including a comparison between proposed algorithm and traditional sampling-based motion planning algorithms, and the sensitivity analysis of the algorithm space decomposition granularity. Section 4 contains the conclusions and implications of this work.

2. Methodology

In the agricultural picking scene, the distribution of obstacles is often very dense and the fruits and the branches of the plants are stacked and staggered. In such a complex environment, the robot needs to avoid obstacles in the working space, and also needs to plan a continuous and smooth path in the configuration space, which is quite difficult. To provide sufficient flexibility, grasping robots often have high degrees of freedom. The dual-arm system even needs to cooperate in avoiding collisions. This becomes a major challenge in the planning of picking robots. Fig. 1 illustrates the visualization of the simulated scene. The Kinect-V2 camera is used to obtain RGB-D images of the scene, and the extended Mask-RCNN network is used to perform precise semantic

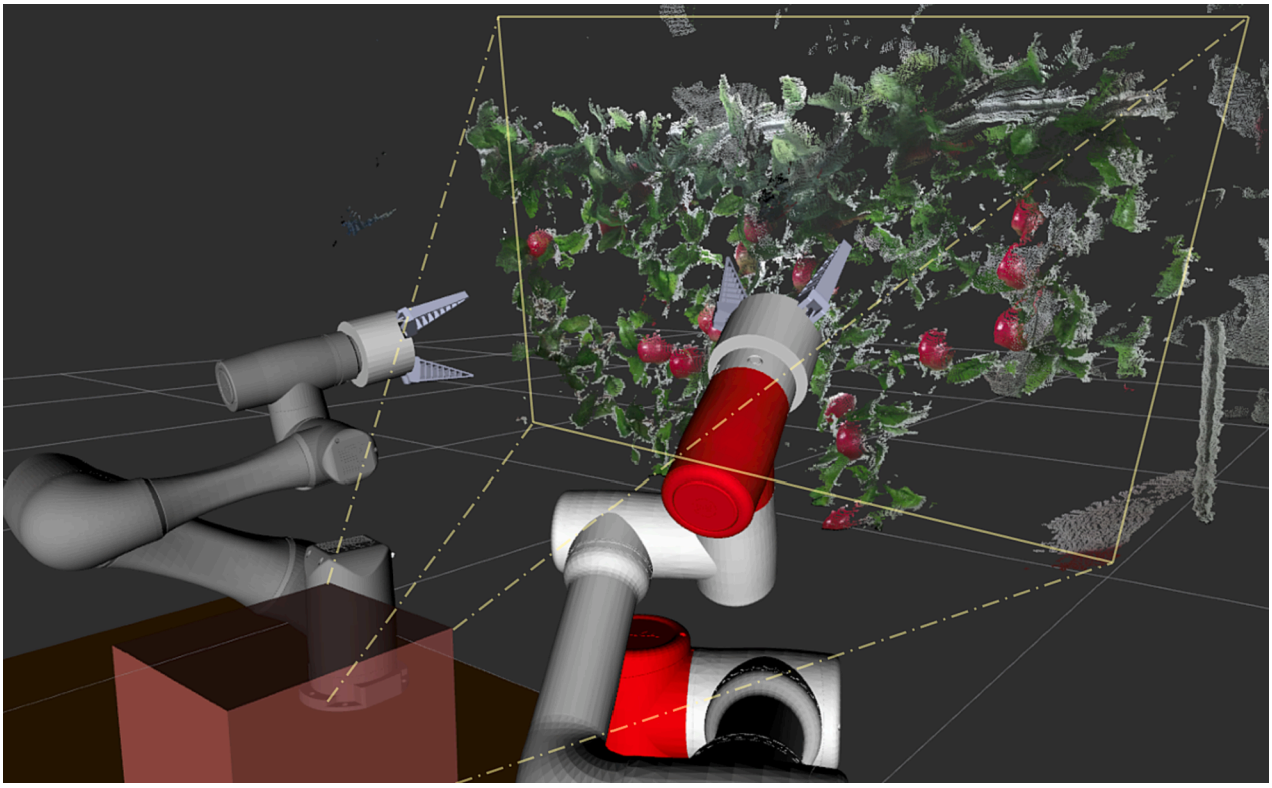


Fig. 1. An illustration of tomato picking scene.

recognition of the target fruit (Gong et al., 2022). The rest of the point cloud, including tree branches and leaf obstacles, are considered obstacles. Furthermore, the robot is also modeled in a simulation environment.

The planning method proposed in this paper is a probabilistically complete motion planning algorithm specially tailored for the multi-DOF flexible picking robotic arm platform as shown in Fig. 1. The basic flow of the algorithm is shown in Fig. 2. To facilitate a better understanding of the algorithmic concept, we will first clarify the definitions of robot workspace and configuration space before introducing the algorithm. The workspace refers to the set of spatial points that the robot end effector can reach during its motion, primarily characterized by the spatial pose of the robot's end effector. The configuration space refers to the collection of all possible configurations of the robot, where a robot configuration represents a complete representation of the positions of each point on the robot, primarily characterized by the positions of each degree of freedom of the robot. The idea is first to discretize the workspace into smaller local regions. Secondly, plan a channel in the workspace to connect the starting point to the target and avoid obstacles while searching in the configuration space. Thirdly, when the

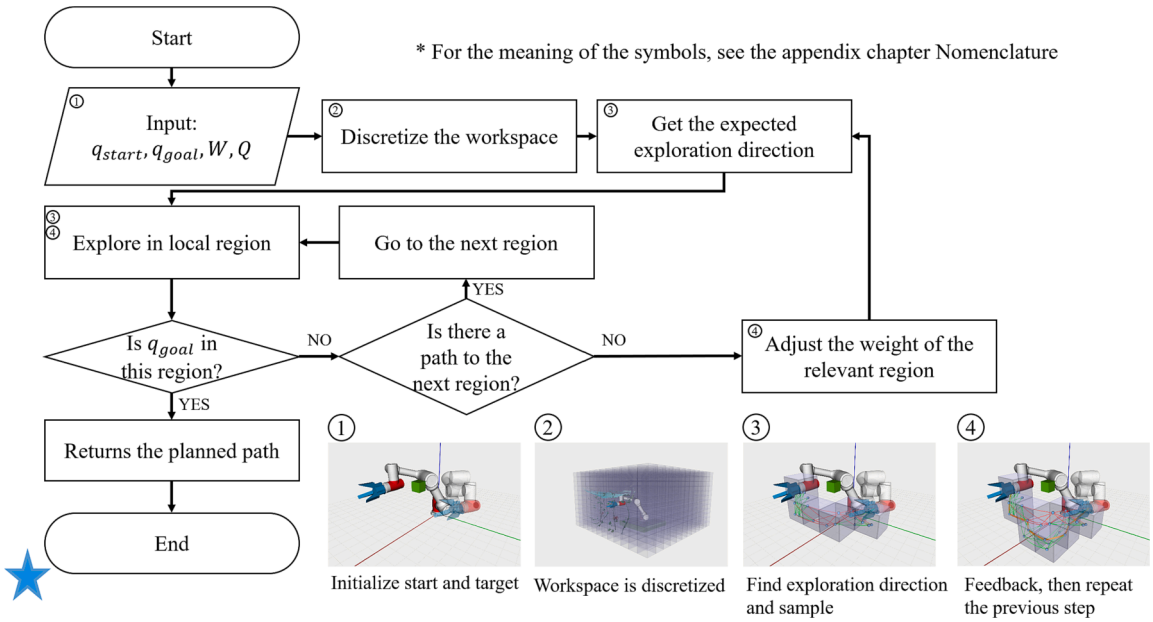


Fig. 2. The flow chart of the proposed algorithm.

configuration space exploration time in the current region is too long, the algorithm automatically updates the connectivity weights between regions. Finally, the algorithm uses the new weights to plan new paths in the workspace. In summary, the search direction is guided in the continuous configuration space from the discrete workspace, while the progress of the search in the continuous configuration space is fed back to the discrete layer. This enables the algorithm to recalculate the most promising new directions in the workspace.

The pseudocode is provided in Algorithm 1, which primarily aims to illustrate the algorithmic concept of the path planning method proposed in this paper. The algorithm follows a two-step approach: first, performing path planning in the discretized workspace to identify block-based paths within the workspace; and second, exploring the configuration space within each workspace region to determine robot configuration motion paths that connect the paths within the workspace. The steps carried out in the workspace and configuration space are highlighted in the pseudocode using blue and yellow highlights, respectively. The workspace is first discretized into several regions R_i, which are stored in the set D, and then the roadmap is initialized with q_{start} as the root node. The main loop of the algorithm first plans the expected exploration order $\left[R_i^{(j)}\right]_{j=1}^{k}$. The variable i denotes the number of the region in the discretization regions, the superscript j represents the number of the region in order, and the usage of superscript k and subscript j outside the square brackets is similar to the superscript and subscript of the series. This expected exploration order connects the starting configuration and the target configuration corresponding to the regions R_{start} and R_{goal} in the workspace and does not include the region where the obstacles are located. At the same time, the order is the shortest path in the current set of regions D. Subsequently, $\{R_i^{(j)}\}$ is randomly sampled based on the workspace properties, and then adds the sampling points to the roadmap (V, E). Next, it checks whether the vertices in V can be connected, and updates the edge set E of the roadmap. When a feasible solution is found, the algorithm returns this solution. If the coverage and connectivity of each region in the expected exploration order are not improved within limited times, it will be fed back to the workspace to correct the weight of the corresponding region. After the weights are corrected, an expected exploration order is recalculated with a certain probability p_2, and then configuration space sampling and roadmap updating are restarted. These procedures repeat until a feasible solution is found or the algorithm times out. Both p_1 and p_2 in Algorithm 1 are hyperparameters, where p_1 is a parameter used to influence random exploration and directed exploration, and p_2 is the probability used to terminate meaningless exploration.

Algorithm 1 Path Planning Algorithm

Input: Starting point q_{start}, target point q_{goal}, workspace $\mathscr{W}$, configuration space $\mathscr{C}$, maximum computation time $t_{max} > 0$, maximun sampling times n_{max}, coverage threshold c_1, connectivity threshold c_2, hyperparameters p_1, p_2.
Output: A feasible solution or null.

$\mathscr{D} \leftarrow DC(\mathscr{W})$; // Decomposing the workspace
RM.$Init(q_{start})$; // Initialize the roadmap
while $t < t_{max}$ **do**
  $a_1 \leftarrow$ Random(0,1); // Generate random numbers in the range (0,1)
  if $a_1 < p_1$ **then**
    $\left[\mathscr{R}_i^{(j)}\right]_{j=1}^{k} \leftarrow GetExplorationOrder()$; // Generate the desired exploration order randomly
  else
    $\mathscr{D}.updateweights()$; // Update the weights of discrete regions
    $\left[\mathscr{R}_i^{(j)}\right]_{j=1}^{k} \leftarrow GetExplorationOrder(\mathscr{D})$; // Generate the desired exploration order from $\mathscr{D}$
  end if
  for $i \in \{1, 2, \cdots n_{max}\}$ **do**
    $q \leftarrow WS\left(\left[\mathscr{R}_i^{(j)}\right]_{j=1}^{k}\right)$ // The workspace-guided sampling algorithms is detailed in Algorithm 2

(*continued on next column*)

(*continued*)

Algorithm 1 Path Planning Algorithm

  RM.$update(q)$;
  if RM.$ispath\left(q_{start}, q_{goal}\right)$ **then**
    return RM.$path\left(q_{start}, q_{goal}\right)$ // Check if there is a feasible path connecting q_{start} and q_{goal}
  end if
  if $cov(i) - cov(i-1) \le c_1$ and $con(i) - con(i-1) \le c_2$ **then** // Check the coverage and connectivity
    $a_2 \leftarrow$ Random(0,1); // Generate random numbers in the range (0,1)
    if $a_2 < p_2$ **then**
      Break; // Jump to step 4
    end if
  end if
  end for
end while
return NULL

2.1. Workspace

Workspace processing includes two main parts: the decomposition of the workspace and the calculation of the expected exploration order. Decomposition deals with how to discretize the workspace and how to express various elements after discretization. The computation of the expected exploration order concerns how to build a chain of connected regions linking the origin and target points from the discrete workspace and avoiding obstacles, and how to set and apply the region weights.

Let the workspace W be the three-dimensional Euclidean space, i.e., $W \subset \mathbb{R}^3$, containing the robot and environmental obstacles. The workspace is discretized into n regions R_i, that is, $W = R_1 \cup R_2 \cup \cdots \cup R_n$. Besides, for $\forall R_i, R_j \subset W$, only the boundaries of R_i and R_j overlap. Define the set of the edges E between the R_i and its physically adjacent region R_j, that is, $(R_i, R_j) \in E$. Furthermore, define function LocateRegion:$W \rightarrow \{R_1, \cdots, R_n\}$. Therefore, the discrete regions contain R_{start} and R_{goal} where the starting configuration q_{start} and the target configuration q_{goal} are located can also be determined.

Many methods can be applied to the decomposition of the workspace. Here, a simple size consistent grid is used and the workspace is divided uniformly. The decomposition produces the set of regions $\{R_i\}$ and R_{start}, R_{goal}. At the same time, the projection function LocateRegion and the edge set E that need to be maintained are also generated.

The building of a chained region connecting R_{start} and R_{goal} in the workspace and the setting of their corresponding weights are further considered. In each planning cycle, the optimal exploration order is determined that connects the starting point and the waypoints from the set of regions. We use Dijkstra's shortest path search algorithm to find the current expected exploration order. In exploring the expected order, the weight of the edge is an extremely important indicator. The following four indicators are defined to establish the weight model:

(1) COV (R_k)– evaluates the exploration coverage of the region R_k by the sampling-based motion planner, that is, to count the coverage of the vertices in the current roadmap to the region R_k. This is a rough but fast evaluation method. The workspace is divided into fine meshes and then counts how many subdivided meshes in R_k contain vertices in the roadmap.
(2) FREEVOL (R_k)– evaluates the difficulty of exploring the region R_k. After the workspace is decomposed, the configuration space is randomly sampled and then the validity of each sampled point is checked independently. For example, whether there is a collision and/or the robotic arm is reachable, etc. Finally, the sampling points in the configuration space are mapped to the workspace, and the ratio of the valid sampling points in the region R_k to all the sampling points in R_k is taken as the result of FREEVOL (R_k).

(3) CONN (R_k)− evaluates the connectivity of points in R_k with points in other regions, which is measured by calculating the number of times R_k is directly connected to points in other regions in the current expected exploration order after begin sampled by the sampling-based motion planner.
(4) SEL (R_k)− counts the frequency of points in R_k used by the algorithm when it builds the roadmap. Therefore, this weight will cause the sampling-based motion planner to prioritize regions that are less sampled and less tried.

Based on such a weighting model, the region weight can reflect the effectiveness of the connected regions in the planning. Before planning, the initial weights of the regions are only determined by FREEVOL (R_k). That is, only the influence of obstacles on the feasible probability in the region is considered. After exploring in the configuration space, the weights of the connected regions are updated. This enables the workspace to guide the exploration in the configuration space and also accept the feedback from the configuration space. Hence, a feasible solution with higher quality can be more efficiently obtained.

2.2. Configuration space

In the configuration space, there are two main processes, that is, workspace-guided sampling and the weight feedback of discrete regions in the workspace. They make use of the results of configuration space sampling to feedback and update the weights of discrete regions and correct the expected exploration order.

2.2.1. Workspace-guided sampling

Traditional sampling-based motion planning algorithms have the following shortcomings: 1. Short paths in the configuration space are not necessarily short paths in the workspace, which leads to the unintuitive and unpredictable motion trajectories of the robotic arm. 2. Environmental obstacles are usually defined in the workspace, while sampling-based motion planning algorithms focus on configuration space exploration and cannot make full use of obstacle information. In order to solve these problems, workspace-guided sampling is proposed instead of sampling directly in the configuration space. The pseudocode is given in Algorithm 2.

Algorithm 2 Work space Guided Sampling Algorithm

Input: Upper bounds h_1, h_2, h_3 and lower bounds l_1, l_2, l_3 in the three dimensions of the workspace $\mathscr{W}$, Collection of sampled points $\mathscr{C}^*$, distance threshold α.
Output: valid sampling point or NULL

$a_1, a_2, a_3 \leftarrow$ Random(0,1); // Generate random numbers in the range (0, 1)
for $i \in \{1, 2, 3\}$ **do**
  $a_i \leftarrow (h_i - l_i)a_i + l_i$;
end for
$h \leftarrow$ RandomSO3(); // SO(3) group uniform random sampling
$\mathscr{C} \leftarrow IK(h)$; // Inverse kinematics solution
$\mathscr{C}_{valid} \leftarrow \varnothing$;
for $q \in \mathscr{C}$ **do**
  if $isValid(q)$ **then**
    Add($\mathscr{C}_{valid}$,q); // Check if the configuration is valid
  end if
end for
$q_{nearest} \leftarrow FindNearestInW(\mathscr{C}_{text}, a_1, a_2, a_3)$; // Find the closest point in workspace
$q \leftarrow FindNearestInQ(\mathscr{C}_{valid}, q_{nearest})$; // Find the closest point in configuration space
if $\|q_{nearest} - q\|_2 < \alpha$ **then**
  return NULL
end if
return q

The algorithm first samples in the workspace. To constrain all degrees of freedom of the robot, sampling on the workspace is in fact sampling the $SE(3)$ group. The picking robot's workspace $W \subset \mathbb{R}^3$, and W is usually bounded in three dimensions, so position sampling is a bounded uniform sampling in each dimension. After the sampling points are obtained, the analytical solution can be obtained through the inverse kinematics of the manipulator, that is, the projection of the sampling points in the configuration space. Since the inverse kinematics of the 6-DOF manipulator has multiple solutions, it is necessary to select only one solution. First, the algorithm needs to check that all solutions are valid, i.e., whether collisions will occur. Secondly, the algorithm selects the sampled point that is closest to the sampled point in position and selects the solution with the closest distance to the sampled point in the configuration space as the new sampling point. In addition, in order to prevent the new sampling point from still being far away from the existing sampling point in the configuration space, it is also necessary to set the threshold of the maximum transformation joint angle.

2.2.2. Configuration space exploration correction

The principle of configuration space feedback is introduced to correct the weights of discretized regions in the workspace. The configuration space uses a workspace-guided sampling approach to search within the configuration space of the manipulator to find feasible solutions. In each path planning loop, the corresponding exploration paths are generated after sampling by the workspace-guided sampling stage. In some cycles, it may be difficult for the robot to further explore within the narrow obstacle passage, and the configuration is likely to be invalid. The configuration space will feedback this condition into the weights of the neighboring regions and increase them significantly. After many unsuccessful attempts, the exploration information of the configuration space is used to update the weights of each discrete region. Then, the new expected exploration order is determined using the shortest path algorithm and the configuration space is re-explored.

Next, we consider the continuous layer feedback process in detail and start with an example of a 2D simulation environment. As shown in Fig. 3, the example robot is in the initial state and its end effector is located at the starting point indicated by the asterisk. The green area in the figure represents the branch obstacles scattered in the environment and the target grasped by the robotic arm is shown as the small tomato in the figure. According to the workspace-guided sampling algorithm, the initially expected exploration direction connecting the start and end points is found, which is shown by the blue connected area.

Fig. 4 is the result of the robot's expected exploration order after receiving feedback from the configuration space exploration. It shows that the robot is unable to reach the target along the expected exploration sequence obtained by the first iteration. The reason is that in the configuration space exploration process, the robot cannot find the subsequent feasible configuration under the path after it explores and travels to the position shown in the figure on the left. Hence, the weight of this region has to be significantly increased. Then, using the shortest path algorithm, a new expected exploration order is calculated that

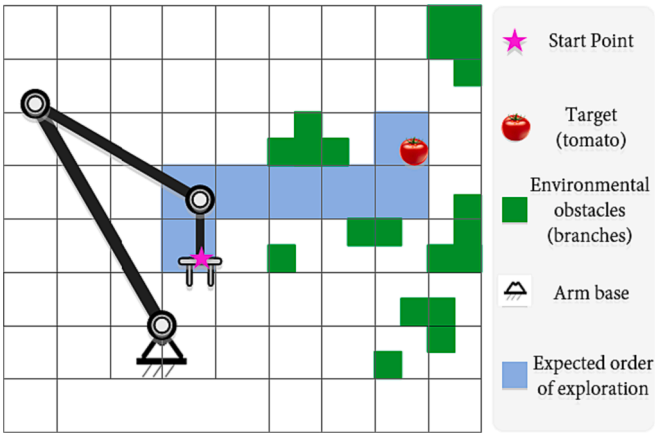


Fig. 3. Schematic diagram of the 2-D simulation environment.

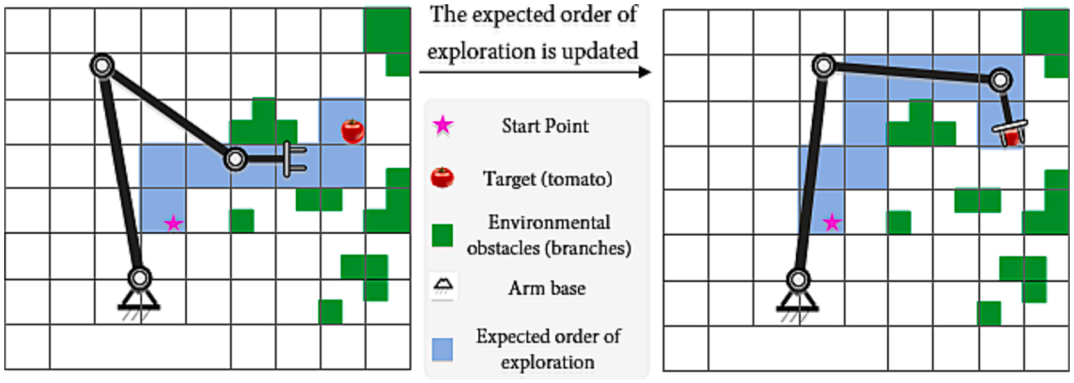


Fig. 4. Change of the expected exploration order after continuous layers are fed back.

bypasses the narrow passage in the middle to reach the goal point from above. Therefore, proposed planning algorithm has the advantage of first trying the closest obstacle avoidance route in the workspace, and then automatically switching to other routes when the search cannot continue on that route. Since the selection of the new exploration order is closely related to the exploration in the previous configuration space, the workspace guidance and configuration space sampling achieve a collaborative interaction and a coupled improvement.

3. Materials and experiments

3.1. Experimental equipment and scenarios

In order to test the feasibility of the algorithm, the experiments are conducted on the dual-arm tomato picking robot, which is shown in Fig. 5. The camera is hidden between the bases of the two robotic arms and fixed on the picking platform together with the robotic arms. The robot consists of two main subsystems: the rail-lifting chassis system and the double-arm picking system. The track-lifting chassis subsystem enables the dual-arm robot to move along the track. It can also lift the entire dual-arm picking subsystem to expand its longitudinal working surface. The double-arm picking subsystem is mainly composed of the double-arm robot, a sensing module, an end gripper, an engineering control computer, and a power supply. Our robotic arms adopt JAKA Zu 7 machinery produced by Shanghai JAKA Robot Technology Co., Ltd. (https://www.jakarobotics.com).

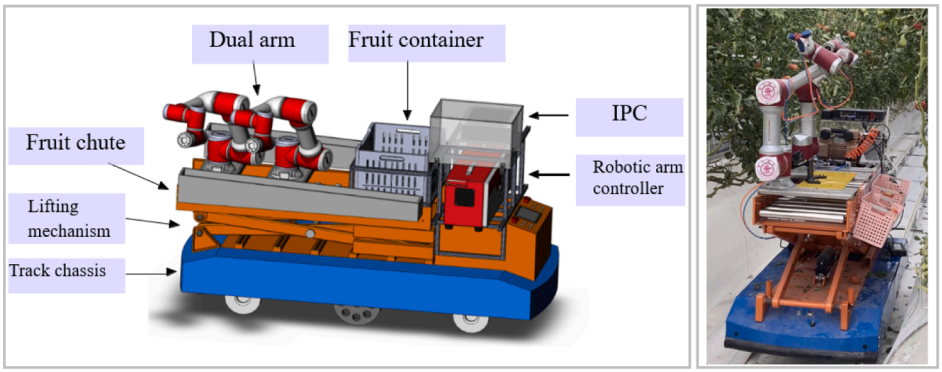
The actual picking experiment was carried out in a modern soilless culture greenhouse. The tomato plants grow upward mainly in the way of ambrosia, up to 3 m high, and the fruit can bear 12 layers. The spacing between tomato plants is 1.2–1.7 m. One row of tomatoes bears fruit in all directions and the interval between each row is about 1 m. The ground is erected with tracks with a spacing of about 0.5 m for electric lift cars to pass.

Besides, in order to objectively evaluate the efficiency and quality of the proposed planning algorithm, a simulated picking scenario is set up as shown in Fig. 6, which requires the manipulator to quickly approach the specified target. The robotic system is asked to plan a collision avoidance trajectory, under a fast-approaching action, from the initial configuration to the specified configuration. In the experiment, a time-out limit of 2 s is set, that is, the planning algorithm will be forced to end after 2 s. At each planning, the position of each obstacle block changes randomly within a region. Specifically, the position of each obstacle block is randomly sampled and generated in a region with a neighborhood of 0.1 m radius. These neighborhoods are based on the possible locations of obstacles in actual scenarios. The density of obstacles is also determined according to the volume ratio of obstacles in actual scenarios. The experimental scenario artificially ensures that each planning problem has a feasible solution.

3.2. Performance analysis

Experiments were conducted to compare the results of proposed algorithm and the RRT, RRT-CONNECT, TSRRT (Shkolnik and Tedrake 2009), and KPIECE (Sucan and Kavraki 2012) algorithms. Among them, RRT-CONNECT is a bidirectional connection version of RRT. TSRRT uses workspace sampling like proposed method, but without workspace discretization and configuration space exploration corrections. The inverse kinematics solutions in the experiment are all solved by the analytical method. The path obtained by each algorithm uses a typical fast path smoothing algorithm (Geraerts and Overmars 2007) to shorten the path as much as possible.

The above algorithms, including the planning algorithm proposed in



a) 3D model b) Photo of Picking Robot

Fig. 5. The dual-arm tomato harvesting robot.

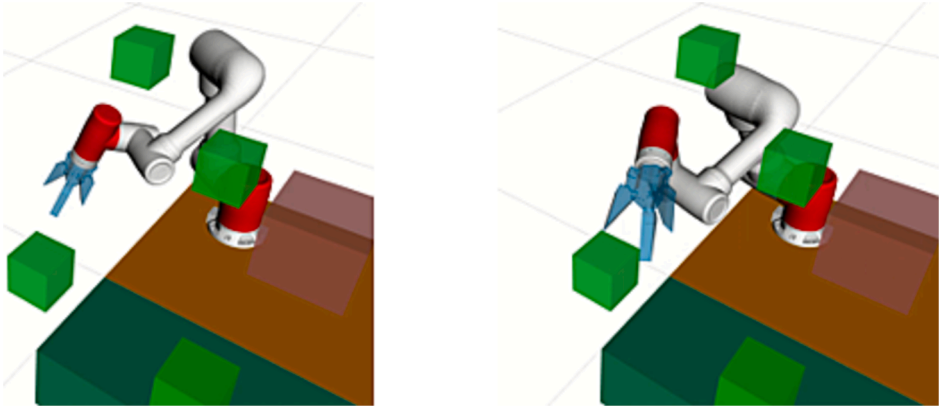


Fig. 6. Experimental scene model for planning algorithm performance analysis.

this paper, are based on the open motion planning library OMPL (Sucan et al., 2012), and ROS and MoveIt are also used in simulation experiments and specific implementations.

The experimental scene in this section is built according to Fig. 6 in Section 3.1, and its settings are the same as described in the introduction of Fig. 6.

The experiment evaluates the computation time and path quality of the first feasible path returned by these algorithms. The path quality is measured with the Cartesian distance of the end effector to the target in the workspace. Each algorithm is repeated 50 times, the results of the computation time are shown in Fig. 7 and the results of path quality are shown in Fig. 8.

It can be seen from the experimental results that almost all algorithms can produce feasible paths within the specified time. Among them, RRT and RRT-CONNECT can produce a path within 0.5 s, but their path quality is relatively poor, which is reflected in the long distance traveled by the end effector. In contrast, both TSRRT and proposed algorithm perform better in path quality. This is because both TSRRT and proposed algorithm use a workspace-guided-based approach, which can potentially find paths with shorter distances in the workspace. However, when sampling in the configuration space, it is easy to produce points that are far away in the workspace, which makes the motion trajectory of the robotic arm redundant and complex. The experiments also show that the planned paths' quality of proposed method is much higher than that of RRT and RRT-CONNECT. In practice, due to the difficulty of accurately estimating the position of obstacles, it is difficult to precisely control the movement of the picking robot manipulator, and redundant movements have potential collision risks. Therefore, RRT and RRT-CONNECT with poor path quality are not applicable in complex and dynamic agricultural environments. In comparison with TSRRT, the planning speed of TSRRT is significantly slower than that of proposed method, and it is only comparable to proposed method in terms of path quality.

3.3. Sensitivity analysis of spatial decomposition granularity

The size of the workspace decomposition granularity will affect how proposed algorithm guides the search through the configuration space. The experiments are carried out in the planning scenario shown in Fig. 6. During the experiment, the workspace is decomposed with different granularities, and the three dimensions of the workspace are decomposed according to different granularities. In the experiment, the lower limit of decomposition granularity is 2. Because if the lower limit is 1, it is equivalent to no decomposition. The upper limit of decomposition granularity is 12. The reason is that the size of the regions after 13 equal divisions cannot pass the end of the robotic arm. For each decomposition granularity, the planning algorithm is executed 50 times to produce the feasible path. The resultant computation durations for all plans are shown in Fig. 9.

The experimental results show that when the decomposition granular size is small (from 2 to 6), the computation durations are decreasing. However, when the number of decompositions is larger than 6, the computation durations are increasing. This means that proposed algorithm relies on a suitable workspace decomposition granularity. Besides, it can be found that when the decomposition size is in the range of 3–6, the computation duration does not change much. This shows that

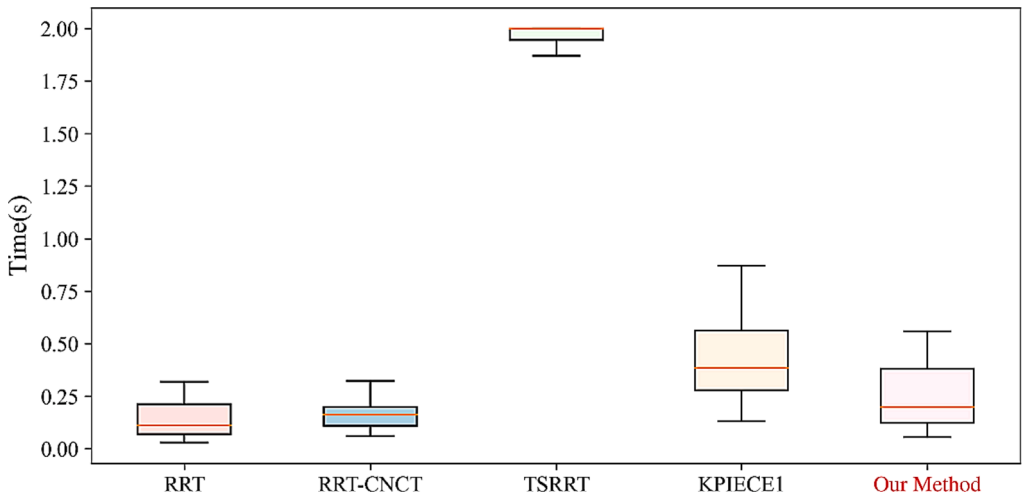


Fig. 7. Comparison of planning speed of different planning algorithms.

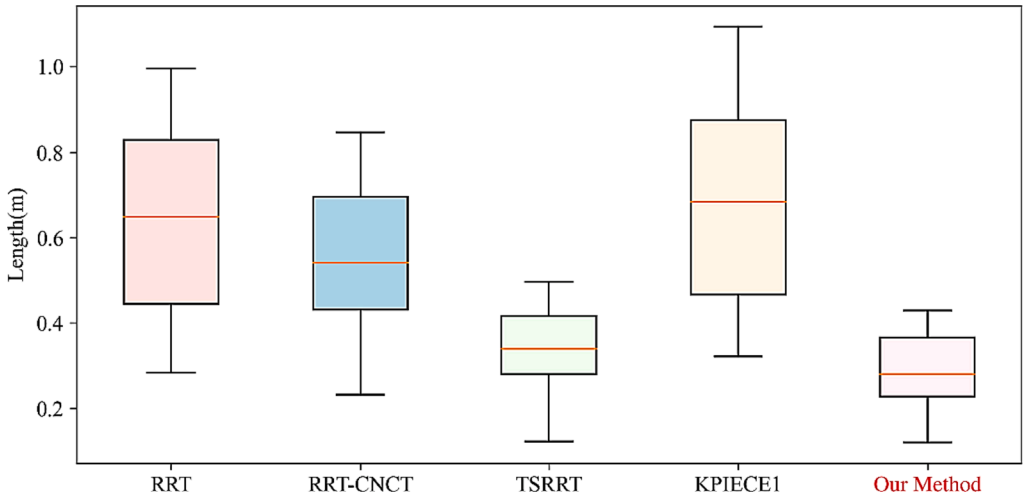


Fig. 8. Comparison of path quality of different planning algorithms.

Fig. 9. Relationship between workspace decomposition granularity and planning speed.

proposed algorithm is not very sensitive to the decomposition granularity when it is within a suitable interval.

3.4. Tomato grabbing experiment

Before conducting the grasping experiment in the greenhouse, we used the branch model and the tomato model to build a simulated grasping scene in the laboratory and used proposed algorithm to plan the path and control the robot. Fig. 10 shows the flow of a successful crawling process.

To quantify the picking results, the following metrics are defined to evaluate the performance of picking robots:

(1) Harvesting rate: The ratio of the number of harvested tomatoes to the number of fruits that need to be picked within the workspace of the robot.
(2) Average time: The total time for the picking robot to perform picking perception, planning, decision-making and carrying out the picking movement divided by the number of picked fruits.

In the laboratory environment, due to the high level of customization in the picking environment, we rearranged the vines and fruits multiple times to create multiple distinct working scenarios for experimentation. In each working scenario, we conducted experiments using different algorithms and performed statistical analysis based on the aforementioned metrics. The results are summarized in Table 1. A total of 8 different scenarios were used in this experiment, with the number of fruits to be picked ranging from 4 to 5 in each scenario.

From the results, our algorithm exhibits a high harvesting rate, which is attributed to the planning in our method within the workspace, ensuring a favorable grasping pose. Additionally, compared to other algorithms, we have a significant advantage in terms of average time. During the experiment, it was observed that the main source of the performance gap, when compared to the RRT and RRT-CNCT algorithms, lies in the higher quality of the paths generated by our algorithm. The paths generated by RRT and RRT-CNCT tend to be more convoluted, resulting in a substantial waste of time during arm movement. When compared to the TSRRT algorithm, TSRRT exhibits excessively long planning time before grasping, with an average delay of approximately 2 s before initiating the grasping action, thus leading to a significant time inefficiency.

After that, we carried out the grasping experiment in the greenhouse, and optimized the robotic arm control and grasping trajectory planning. Fig. 11 shows a successful grabbing process in a greenhouse. During the experiment, the arms work synchronously. When no fruit can be grasped on one grasping plane, the trolley will step through a certain distance to the next grasping plane.

The above indicators during the harvesting process were counted,

Fig. 10. The tomato model grasping process in the laboratory.

Table 1
Summary table of laboratory harvesting experiment.

Algorithms	The number of fruits to be picked	The number of fruits picked	Harvesting rate	Total time	Average time
Our method	36 pieces	32 pieces	88.9 %	303.9 s	9.5 s/piece
RRT	36 pieces	29 pieces	80.6 %	532.4 s	18.4 s/piece
RRT-CNCT	36 pieces	30 pieces	83.3 %	468.2 s	15.6 s/piece
TSRRT	36 pieces	30 pieces	83.3 %	372.9 s	12.4 s/piece

and the results are summarized in Table 2. From the results, the harvesting rate of robots applying this algorithm reaches 80 %, and the harvesting efficiency reaches 10.5 s/piece, which can basically complete automatic picking task in complex scenes.

However, there are still some fruits that cannot be harvested successfully. Summarize the failure cases in the picking experiment, and the main failure reasons are as follows:

(1) Due to factors such as lighting, occlusion, and random errors of the camera itself, the positioning of fruits is deviated, and the success rate of picking is reduced.
(2) Due to the complex occlusion in the environment and the limitation of the working space of the robotic arm, it is difficult to plan a feasible path for some fruits, resulting in a decrease in the picking rate.

4. Conclusions

In this work, a local search path planning method was developed in the feasible region of fruit-picking robots based on discrete workspace guidance and configuration space exploration correction. This method makes full use of the target crop information and obstacle information to explore a small local region by discretizing the dynamic and complex agricultural work environment. This makes planning faster and the robot has a greater probability of having continuous motions in the configuration space. At the same time, the algorithm discretizes the workspace and calculates the channel in the workspace with the most exploration potential to guide the expansion direction of the search tree in the configuration space. This results in significant improvements in the quality of the path generated by the proposed sampling planning method. Moreover, this method has a high planning speed, making it possible to solve the robot grasping path planning problem in the complex dynamic agricultural scenario. The experimental results show that in comparison with RRT, RRT-CONNECT, and other algorithms, proposed algorithm can achieve the same planning speed as most of the existing techniques and has a larger lead in path quality than those algorithms compared.

After simulation and experimental testing, our algorithm has demonstrated higher harvesting success rate and shorter average picking time compared to some classical algorithms, providing significant advantages in agricultural harvesting operations. In practical greenhouse work scenarios, the picking robot can efficiently perform continuous tomato harvesting tasks and can be deployed for production use under specific conditions. However, further improvements are needed. For instance, in our experiments, artificial defoliation was performed before grasping. If there are still leaves in the experimental scene, it will have a certain impact on the algorithm. Besides, proposed algorithm does not classify and identify the scene, but plans any scene from the beginning. So, we also desirable to enable the proposed algorithm to make use of scene knowledge from previous planning.

CRediT authorship contribution statement

Binhao Chen: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Visualization. **Liang Gong:** Conceptualization, Methodology, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Chenrui Yu:** Software, Validation, Visualization, Methodology. **Xiaofeng Du:** Software, Validation, Investigation. **Jianhuan**

Fig. 11. Robotic tomato picking in the greenhouse.

Table 2
Summary table of field harvesting experiment.

The number of fruits to be picked	The number of fruits picked	Harvesting rate	Picking success rate	Total time	Average time
35 pieces	28 pieces	80 %	81.8 %	294 s	10.5 s/ piece

Chen: Software, Validation, Writing – original draft. **Shenghan Xie:** Software, Validation, Investigation. **Xinyi Le:** Validation, Writing – review & editing. **Yanming Li:** Validation, Writing – review & editing. **Chengliang Liu:** Resources, Supervision, Writing – review & editing.

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.

Data availability

Data will be made available on request.

Acknowledgments

The authors would like to acknowledge the financial support provided by the National Natural Science Foundation of China (NSFC) under Grant No. 52175024.

Appendix A. Proof of probabilistic completeness of proposed algorithm

An algorithm is probabilistically complete, that is, when there is at least one solution the algorithm can find them after sufficient iterations. In order to prove this, it is necessary to prove that the algorithm is complete in the static sampling process and in the dynamic path planning process.

Denote $Q_p = \{q \in Q | \text{ feasible forward kinematics of } q \text{ is } p\}$ as the set includes all feasible inverse kinematics solutions of point p in the workspace. Denote $P_S(p)$ as the probability of which the sampler S returns to point p in the workspace after an attempt. Denote $P_s^n(p)$ as the probability that the sampler S has returned to point p in the workspace after n attempts. Denote $P_S(q|p)$ as the probability that the sampler S return configuration q under the premise of returning to the point p after an attempt. Denote $P_S^n(p,q)$ as the probability that the sampler S has returned to configuration q under the premise of returning to the point p after n attempts. Denote $\sigma : [0, L]$ as a path connecting the starting point and the target, where $\sigma(0)$ refers to the starting point, $\sigma(L)$ refers to the target, and L represents the distance between the starting point and the target along the path σ. Denote $\dot{U}(p,\delta)$ as the decentered neighborhood of point p with radius δ.

Based on the properties of proposed algorithm, we envisioned the basic definition that the sampler would have. It should be noted that the following definitions are all set in a square discrete workspace region, and can obviously be generalized to the entire workspace.

Since the sampling points in the workspace are randomly sampled within the three-dimensional upper and lower bounds of the workspace, we can have.

Definition 1.
$(\forall p \in R_i,\ p \text{ is not in the obstacle area})\ [P_S(p) > 0]$

The sampler is redundancy-robust, which is embodied in workspace-guided-based algorithms. We include all feasible inverse kinematic solutions into the set Q_{valid} for investigation, from which we can have.

Definition 2.
$\left(\forall p \in R_i,\ \forall q \in Q_p\right)\left[P_S(q|p) > 0\right]$

Then we proceed to the proof. First, we need to prove that proposed algorithm is complete in a static sampling process.

Proposition 1. *Sampling in the workspace is complete, that is, when* $n\to\infty$, $P_s^n(p)\to 1$.

Proof. For any point p in the discretized workspace region that does not overlap with the obstacle, the probability that it is sampled is $P_S(p)$, so the probability that it is not sampled is $1-P_S(p)$. After n attempts, the probability that the point p has not been sampled is $(1-P_S(p))^n$, it is easy to see that the probability that the point p has been sampled in the n attempts is $1-(1-P_S(p))^n$. According to Definition 1, we know that $[1-P_S(p)]\in(0,1)$, so we can have

$$\lim_{n\to\infty}\left[P_s^n(p)\right]=\lim_{n\to\infty}[1-(1-P_S(p))^n]=1 \tag{1}$$

Proposition 2. *The sampling in the configuration space is complete, that is, when* $n\to\infty$, $P_S^n(p,q)\to 1$.

Proof. For any feasible inverse kinematic solution q for any point p in the discretized workspace region, the probability that it is sampled is $P_S(p)\bullet P_S(q|p)$, so the probability that it is not sampled is $1-P_S(p)\bullet P_S(q|p)$. After n attempts, the probability that the configuration q has not been sampled is $(1-P_S(p)\bullet P_S(q|p))^n$, then the probability that the configuration q has been sampled in the n attempts is $1-(1-P_S(p)\bullet P_S(q|p))^n$. According to Definition1 and 2, we know $[1-P_S(p)\bullet P_S(q|p)]\in(0,1)$, so we can have

$$\lim_{n\to\infty}\left[P_S^n(p,q)\right]=\lim_{n\to\infty}[1-(1-P_S(p)\bullet P_S(q|p))^n]=1 \tag{2}$$

Then, we need to prove that proposed algorithm is complete in the dynamic path planning process.

Proposition 3. *In a discretized workspace region, chain connection between the starting point and the target is possible if there is at least one feasible connection path* $\sigma:[0,L]$. *Moreover, there is a constant* ε, *the distance from all points on the path to the workspace boundary and obstacles does not exceed* ε, *then the algorithm can find a feasible path.*

Proof. Suppose there are m feasible paths, denoted as $\sigma^i:[0,L], i=1,2,\cdots m$. Discretize sampling on the m feasible paths are denoted as

$$(p,q)_k^i=\begin{cases}\sigma^i(k\varepsilon), k=0,1,2,\cdots K-1\\ \sigma^i(L), \qquad k=K\end{cases} \tag{3}$$

where $K=\lfloor L/\varepsilon\rfloor.+1$. For any point on these paths except the target, $\dot{U}\left(p_j^i,\varepsilon\right)$ must be in the workspace and contain the point $(p,q)_{j+1}^i$. According to the conclusions of Propositions 1 and 2, after enough attempts, the algorithm will inevitably sample the point $(p,q)_{j+1}^i$. The algorithm will verify whether the connection between $(p,q)_j^i$ and $(p,q)_{j+1}^i$ is feasible. Since these points are connected by feasible paths σ^i, this local feasible path will be found and added to the roadmap. Beginning from the starting point, a search is carried out according to this procedure and the algorithm can find all feasible paths $\sigma:[0,L]$ connecting the starting point and the target.

Appendix B. Nomenclature

W	Workspace
Q	Configuration space
$\mathcal{R}$	Discretized workspace regions
D	Set of discretized workspace regions
V	Set of points in the road map
E	Set of edges in the road map
p	Point in the workspace
q	Point in the configuration space
t	Time
h	Upper bounds
l	Lower bounds

Appendix C. Supplementary material

Supplementary data to this article can be found online at https://doi.org/10.1016/j.compag.2023.108353.

References

Ahlin, K.J., Hu A.-P. and Sadegh N., 2017. Apple Picking Using Dual Robot Arms Operating Within an Unknown Tree. ASABE Ann. Int. Meeting. St. Joseph, MI, ASABE: 1.

Attali, A., S. Ashur, I. B. Love, C. McBeth, J. Motes, D. Uwacu, Morales M. and Amato N. M., 2022. Evaluating Guiding Spaces for Motion Planning. arXiv preprint arXiv: 2210.08640.

Bac, C.W., van Henten, E.J., Hemming, J., Edan, Y., 2014. Harvesting Robots for High-value Crops: State-of-the-art Review and Challenges Ahead. J. Field Rob. 31 (6), 888–911.

Cao, X., Zou, X., Jia, C., Chen, M., Zeng, Z., 2019. RRT-based path planning for an intelligent litchi-picking manipulator. Comput. Electron. Agric. 156, 105–118.

Chiang, H.T.L., Hsu, J., Fiser, M., Tapia, L., Faust, A., 2019. RL-RRT: Kinodynamic Motion Planning via Learning Reachability Estimators From RL Policies. IEEE Rob. Autom. Lett. 4 (4), 4298–4305.

Denny, J., R. Sandström, Bregger A. and N. M. Amato., 2020. Dynamic Region-biased Rapidly-exploring Random Trees. Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics. K. Goldberg, P. Abbeel, K. Bekris and L. Miller. Cham, Springer International Publishing: 640-655.

Denny, J., Greco, E., Thomas, S., Amato, N.M., 2014. MARRT: Medial Axis biased rapidly-exploring random trees. 2014 IEEE International Conference on Robotics and Automation (ICRA).

Geraerts, R., Overmars, M.H., 2007. Creating High-quality Paths for Motion Planning. Int. J. Robot. Res. 26 (8), 845–863.

Gong, L., Wang, W., Wang, T., Liu, C., 2022. Robotic harvesting of the occluded fruits with a precise shape and position reconstruction approach. J. Field Rob. 39 (1), 69–84.
Guo, N., Zhang, B., Zhou, J., Zhan, K., Lai, S., 2020. Pose estimation and adaptable grasp configuration with point cloud registration and geometry understanding for fruit grasp planning. Comput. Electron. Agric. 179, 105818.
Han, K.S., Kim, S.C., Lee, Y.B., Kim, S.C., Im, D.H., Choi, H.K., Hwang, H., 2012. Strawberry Harvesting Robot for Bench-type Cultivation. J. Biosyst. Eng. 37 (1), 65–74.
Hauser, K., Ng-Thow-Hing, V., 2011. Randomized multi-modal motion planning for a humanoid robot manipulation task. Int. J. Roboti. Res. 30 (6), 678–698.
Holleman, C. and Kavraki L. E., 2000. A framework for using the workspace medial axis in PRM planners. Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065).
James, S. and Johns E., 2016. 3D Simulation for Robot Arm Control with Deep Q-Learning. arXiv:1609.03759.
Kavraki, L.E., Svestka, P., Latombe, J., Overmars, M.H., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Rob Autom 12 (4), 566–580.
Khatib, O., 1986. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. Int. J. Robot. Res. 5 (1), 90–98.
Kingston, Z., Moll, M., Kavraki, L.E., 2019. Exploring implicit spaces for constrained sampling-based planning. Int. J. Robot. Res. 38 (10–11), 1151–1178.
Kingston, Z., Wells, A.M., Moll, M., Kavraki, L.E., 2020. Informing Multi-Modal Planning with Synergistic Discrete Leads. 2020 IEEE International Conference on Robotics and Automation (ICRA).
LaValle, S. M. and Kuffner J. J., 1999. Randomized kinodynamic planning. Proceedings. IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C).
Lin, G., Zhu, L., Li, J., Zou, X., Tang, Y., 2021. Collision-free path planning for a guava-harvesting robot based on recurrent deep reinforcement learning. Comput. Electron. Agric. 188, 106350.
Mehta, S.S., Burks, T.F., 2014. Vision-based control of robotic manipulator for citrus harvesting. Comput. Electron. Agric. 102, 146–158.
Plaku, E., Kavraki, L.E., Vardi, M.Y., 2010. Motion Planning With Dynamics by a Synergistic Combination of Layers of Planning. IEEE Trans. Rob. 26 (3), 469–482.
Schuetz, C., Baur, J., Pfaff, J., Buschmann, T., Ulbrich, H., 2015. Evaluation of a direct optimization method for trajectory planning of a 9-DOF redundant fruit-picking manipulator. 2015 IEEE International Conference on Robotics and Automation (ICRA).
Shkolnik, A., Tedrake, R., 2009. "Path planning in 1000+ dimensions using a task-space Voronoi bias. " 2009 IEEE International Conference on Robotics and Automation.
Sucan, I.A., Kavraki, L.E., 2012. A Sampling-Based Tree Planner for Systems With Complex Dynamics. IEEE Trans. Rob. 28 (1), 116–131.
Sucan, I.A., Moll, M., Kavraki, L.E., 2012. The Open Motion Planning Library. IEEE Rob. Autom. Mag. 19 (4), 72–82.
Tao, Y., Zhou, J., 2017. Automatic apple recognition based on the fusion of color and 3D feature for robotic fruit picking. Comput. Electron. Agric. 142, 388–396.
Tao, Y., Zhou, J., Wang, M., Zhang, N., Meng, Y., 2017. An optimum strategy for robotic tomato grasping based on real-time viscoelastic parameters estimation. Int. J. Adv. Rob. Syst. 14 (4).
Van Henten, E.J., Hemming, J., Van Tuijl, B.A.J., Kornet, J.G., Bontsema, J., 2003. Collision-free Motion Planning for a Cucumber Picking Robot. Biosyst. Eng. 86 (2), 135–144.
Vonásek, V., Pěnička, R., 2019. Sampling-based motion planning of 3D solid objects guided by multiple approximate solutions. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).
Vonásek, V., Faigl, J., Krajník, T., Přeučil, L., 2009. RRT-path – A Guided Rapidly Exploring Random Tree. Springer, London, London.
Wang, H., Zhao, Q., Li, H., Zhao, R., 2022. Polynomial-based smooth trajectory planning for fruit-picking robot manipulator. Inform. Process. Agric. 9 (1), 112–122.
Yang, Z., Gong, L., Liu, C., 2021. Efficient TCP Calibration Method for Vision Guided Robots Based on Inherent Constraints of Target Object. IEEE Access 9, 8902–8911.
Yang, H., Li, L., Gao, Z., 2017. Obstacle avoidance path planning of hybrid harvesting manipulator based on joint configuration space. Trans. Chin. Soc. Agric. Eng. 33 (4), 55–62.
Ye, L., Duan, J., Yang, Z., Zou, X., Chen, M., Zhang, S., 2021. Collision-free motion planning for the litchi-picking robot. Comput. Electron. Agric. 185, 106151.