CONFIGURATION-SPACE-BASED COLLISION-FREE PATH PLANNING FOR A 4-DOF SERIAL HARVESTING MANIPULATOR USING TWO-LEVEL HIERARCHICAL A* SEARCH

Haixin ZOU[1], Xiangjun ZOU[2], Mingxiang GUAN[3]*

Collision-free path planning presents challenges for fruit-harvesting robots operating in cluttered workspaces. This paper presents a configuration space (C-space) – based planning framework for a four-degree-of-freedom (4-DOF) serial harvesting manipulator that integrates analytical obstacle modelling with a two-level A (A-star) search strategy. The manipulator and typical harvesting obstacles (circular obstacles) are mapped into C-space to construct joint-level forbidden regions. Based on this representation, the planner first performs layer-wise two-dimensional (2D) A* searches on θ_3-sliced planes and then merges the resulting path points into a sparse three-dimensional (3D) graph for a second-stage A* search; an additional connectivity-restoration step is introduced to handle slices where no feasible in-plane path exists. Simulation results demonstrate that the proposed method can generate feasible 3D collision-free paths that are readily mapped back to joint-space commands. In comparisons with dense global 3D A* on a 91×91×91 joint-space grid, the proposed approach substantially reduces node expansions and planning time while producing paths with very similar lengths in the tested scenario. These results suggest that the proposed framework is a computationally efficient option for collision-free motion planning of 4-DOF harvesting manipulators in structured environments, with further validation in more complex and real-world settings left for future work.*

Keywords: collision-free path planning; harvesting robot; motion simulation; two-level A*; sparse 3D graph.

1. Introduction

Global horticultural sectors are increasingly constrained by severe labor shortages and rising production costs, necessitating a shift toward automated harvesting solutions. Consequently, the development of robotic manipulators capable of autonomous fruit picking has emerged as a critical research priority. A fundamental prerequisite for these systems is the ability to execute collision-free

[1] Lecturer, School of Information and Communication, Shenzhen University of Information Technology, China, e-mail: zouhaixin123@qq.com

[2] Prof., College of Engineering, South China Agricultural University, China, e-mail: xjzou1@163.com

[3*] Prof., School of Information and Communication, Shenzhen University of Information Technology, China, e-mail: guanmx@sziit.edu.cn

motion in unstructured environments, making robust path planning a primary technical barrier that must be overcome for successful field deployment[1-3].

To address motion planning in complex agricultural settings, researchers have investigated a diverse array of algorithmic strategies, generally classified into grid-based, potential field, sampling-based, discrete optimization, and hybrid approaches [4–6]. Grid-based methods discretize the workspace or configuration space (C-space) and search for feasible paths using graph-search algorithms such as A* and its variants, and have been frequently adopted in agricultural navigation and structured orchard/greenhouse settings due to their deterministic behavior and implementation simplicity [7,8]. Potential field methods guide motion using artificial attractive/repulsive fields and have been applied to harvesting manipulators for obstacle avoidance and energy-aware motion generation, although local minima can arise in dense canopies [9]. Sampling-based methods, such as Rapidly-exploring Random Tree (RRT) and its optimal variant RRT*, explore high-dimensional C-space via random sampling and have been applied to harvesting-manipulator planning in dragon-fruit harvesting and uncertain-environment studies [10,11]; related RRT/RRT* variants have also been reported for litchi-picking and tea-picking manipulators [12,13]. Discrete optimization methods—including genetic algorithms and ant colony optimization—have been used to address multi-objective planning problems such as efficiency, energy, and operation cost, and are particularly relevant when path planning is coupled with task allocation or multi-region scheduling [14,15]. Hybrid methods integrate multiple algorithms to improve performance, for example by combining global grid-based search with local dynamic obstacle avoidance, as in improved A*–fuzzy Dynamic Window Approach (DWA) schemes for greenhouse navigation [16]; in parallel, perception modules such as specialized localization of fruit or stem structures and geometric bounding models have been employed to support damage-free harvesting[17, 18].

Despite these advancements, achieving efficient path planning directly within the robot's high-dimensional C-space remains a significant computational bottleneck. Traditional dense 3D search methods, while rigorous, often incur prohibitive calculation costs when detailed obstacle mapping is required. There is a distinct need for planning frameworks that can accurately resolve C-space obstacles without succumbing to the "curse of dimensionality" inherent in full-resolution volumetric searches.

Addressing this challenge, this paper presents a hierarchical planning framework tailored for a 4-DOF serial harvesting manipulator. By analytically mapping workspace obstacles into C-space forbidden regions, we convert the complex 3D avoidance problem into a manageable graph search. The proposed method utilizes a unique two-stage strategy: first, layer-wise 2D A* searches are executed across discrete C-space slices to identify feasible transitions;

subsequently, these points are integrated into a sparse 3D graph for final trajectory generation. To ensure robustness, a connectivity-restoration mechanism is included to bridge gaps between slices. Validation via an extensive 91×91×91 grid simulation demonstrates that this approach significantly reduces node expansions and planning time compared to conventional global 3D A*, while maintaining equivalent path quality.

2. C-space modelling of a 4-DOF harvesting manipulator and typical obstacle primitives

2.1 C-space formulation of the harvesting manipulator

The harvesting robot considered in this work is a 4-DOF serial manipulator consisting of a base rotation and a three-revolute (3-R) planar arm. For C-space analysis, the base joint is treated as a coarse positioning degree of freedom that orients the manipulator toward the target fruit and operates largely independently of local collision avoidance. The C-space modelling in this section therefore focuses on the 3-DOF manipulator comprising Joints 1-3, in which each joint configuration $(\theta_1,\theta_2,\theta_3)$ is represented by a single point, and any physically feasible motion of the manipulator corresponds to a continuous curve connecting two points in this space.

In the C-space formulation, the geometry of the manipulator is implicitly encoded by transforming workspace obstacles into C-space obstacles, while the robot itself is represented as a point and obstacles are mapped to forbidden regions in C-space. Any configuration lying inside these forbidden regions corresponds to at least one collision between the manipulator and the environment, whereas configurations outside them represent collision-free motions that are admissible for execution, as illustrated in Fig. 1. This is achieved by expanding the obstacles to account for the finite link widths and joint geometries, so that the union of all C-space obstacles delineates the full set of infeasible joint states, and the remaining collision-free region provides a compact joint-space domain for global and local path-planning algorithms.

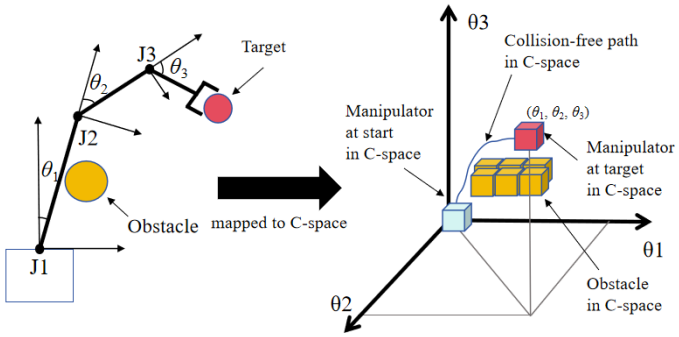

Fig. 1. C-space representation of a 3-DOF harvesting manipulator with obstacles

2.2 C-space modelling for circular obstacles

In the harvesting plane, unripe fruits and nearby branches are approximated as circular obstacles to obtain a simple yet representative model of the environment. Each manipulator link is approximated by a rectangle of length L and width $2w$, and each obstacle is modelled as a circle with center **C** = (x, y) and radius r. For collision checking, the link width is absorbed into the obstacle by defining an expanded radius R=w+r, so that the link can be treated as a line segment of length Z, interacting with an enlarged circle.

A single joint, such as Joint 1, is used here to illustrate the modelling procedure, as shown in Fig. 2. In the planar frame of Fig. 2, the base joint is located at the origin **O**, the first link rotates about **O**, and the obstacle center **C** lies at distance **OC** = $\rho=\sqrt{x^2+y^2}$ and angle $\beta=\arctan(\frac{x}{y})$. When the obstacle is sufficiently close, the segment **OC**, **BC** and **OB** form a right triangle, from which two collision angles θ_1 and $\theta_{1'}$, are obtained analytically. The interval between these angles defines the forbidden rotation range of Joint 1 for that obstacle.

The different geometric configurations between the link and the circular obstacle can be classified into three cases.

(1) Case 1: when $\rho \le \sqrt{L_1^2+R^2}$, where R=w_1+r, the triangle **OBC** is always right-angled, as illustrated in Fig. 2. In this case, the collision angles θ_1 and $\theta_{1'}$ of Joint 1 can be obtained by formula (1),

$$\begin{cases} \alpha_1 = \arcsin(\frac{R}{\rho}) \\ \theta_1 = \arctan(\frac{x}{y}) - \alpha_1 \\ \theta_{1'} = \arctan(\frac{x}{y}) + \alpha_1 \end{cases} \tag{1}$$

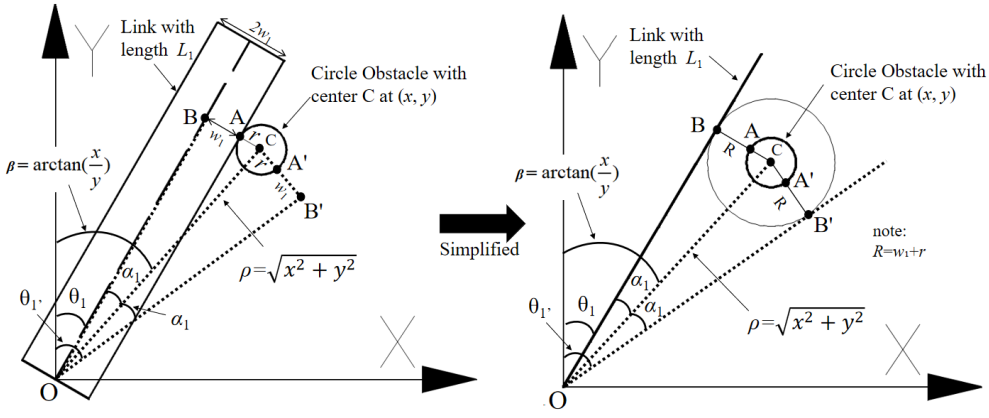

Fig. 2. Case 1: Right-triangle configuration of OBC

（2）Case 2: when $\sqrt{L_1^2+R^2}<\rho<\sqrt{L_1^2+w_1^2}+r$, as shown in Fig. 3, the triangle **OBC** becomes obtuse. In the exact model, the auxiliary angle α_1 can be derived from triangle **AOC** using the cosine law, where **AO**=$\sqrt{L_1^2+w_1^2}$, **OC**=ρ and **AC**=r. Because $L_1 \gg w_1$ in practical harvesting manipulators, the geometry can be further simplified by approximating **OA**≈**OB**=L_1, **BC**≈R=w_1+r, and $\alpha_1 \approx \alpha'_1$, so that an approximate angle α'_1 can be computed directly from triangle **BOC** using the cosine law. The simplified construction is illustrated on the right-hand side of Fig. 3. In this case, the collision angles θ_1 and $\theta_{1'}$ of Joint 1 can be obtained by formula (2):

$$\begin{cases} \alpha_1 = \arccos\left(\dfrac{L_1^2+\rho^2-R^2}{2L_1\rho}\right) \\ \theta_1 = \arctan\left(\dfrac{x}{y}\right)-\alpha_1 \\ \theta_{1'} = \arctan\left(\dfrac{x}{y}\right)+\alpha_1 \end{cases} \tag{2}$$

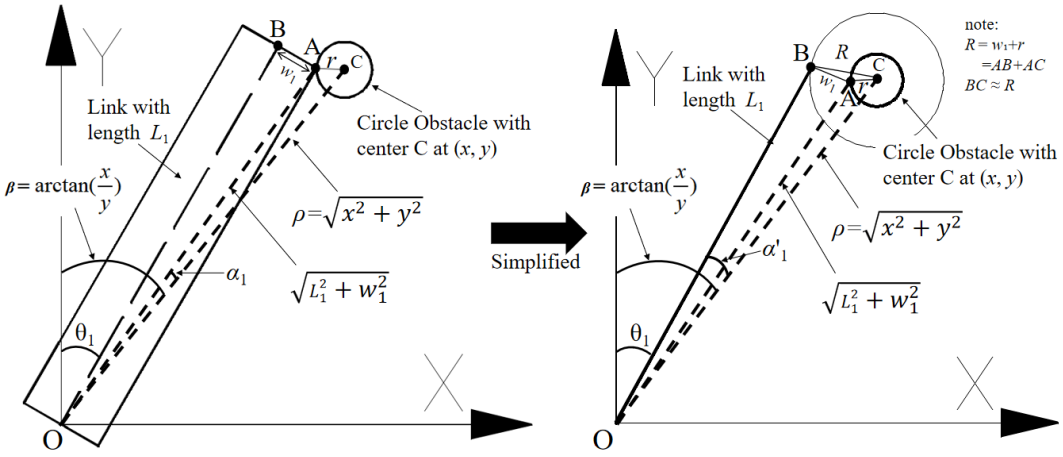

Fig. 3. Case 2: Obtuse-triangle configuration of OBC

（3）case 3: when $\rho=\sqrt{L_1^2+w_1^2}+r$, the triangle **AOC** degenerates into a straight line, as shown in Fig. 4, and only a single collision angle exists; this collision angle can be computed directly from the coordinates of the obstacle center **C**: $\theta_1=\theta_{1'}=\arctan\left(\dfrac{x}{y}\right)$; when $\rho>\sqrt{L_1^2+w_1^2}+r$, it indicates that no collision occurs at all, and the link of the manipulator can move freely within the workspace.

For the case of two consecutive links, Link 1 and Link 2 (see Fig. 5), a coordinate frame *XOY* is attached to Joint 1 and a second frame *X'O'Y'* is attached to Joint 2. The obstacle center has coordinates (*x*,*y*) in the *XOY* frame and (*x'*,*y'*) in

the $X'O'Y'$ frame. By sweeping Joint 1 over different angles θ_1, the pose of Link 2 changes accordingly, and the obstacle position expressed in the $X'O'Y'$ frame can be updated via a straightforward rigid-body transformation between the two frames. Once the obstacle is expressed in the local frame of Joint 2, its relative position falls into one of the geometric cases discussed above, so that the corresponding collision angles θ_2 and $\theta_{2'}$ for Joint 2 can be computed using the same formulae as for Joint 1.

The same idea extends directly to manipulators with more degrees of freedom. For each additional joint, an appropriate local coordinate frame is attached at the joint, and the obstacle center is transformed into that frame using the forward kinematics of the preceding joints. The resulting relative position is then classified into Cases 1-3, and the associated collision angles for the current joint are obtained using the previously derived analytical expressions..

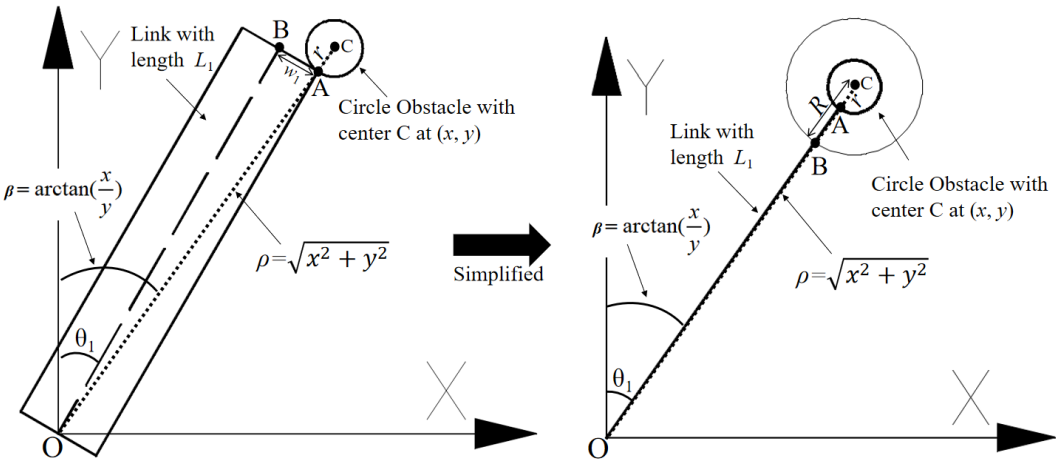

Fig. 4. Case 3: Tangential configuration between the link and the circular obstacle

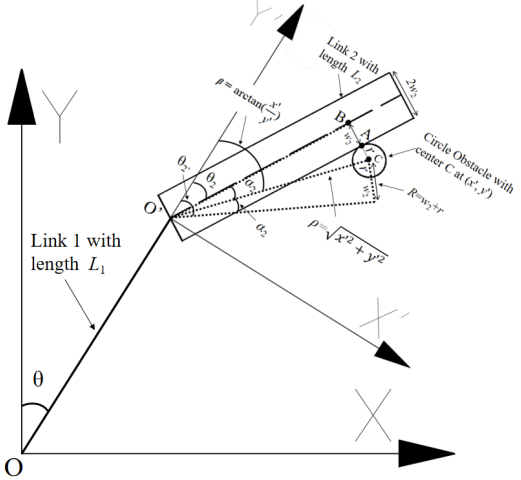

Fig. 5. simplified model of a circle obstacle for two links

3 Two-level A* path planning via layered 2D slicing and sparse 3D graphs

The A* search algorithm is built around the evaluation function $f(n) = g(n) + h(n)$, where $g(n)$ denotes the accumulated cost from the start node to the current node n, and $h(n)$ is a heuristic estimate of the remaining cost to the goal. When the heuristic $h(n)$ is admissible, i.e., it never overestimates the true remaining cost, A* is guaranteed to find an optimal path on a given graph. In a regular 3D grid, each voxel can have up to 26 neighbors in its 3×3×3 neighborhood (analogous to a Rubik's cube with all cells except the center), which leads to a rapid growth of both branching factor and state space. Directly running A* on a dense 3D voxel grid therefore incurs prohibitive computational and memory costs for practical applications.

To reduce the complexity of 3D path planning, this work proposes a two-level A* framework based on "layered 2D slices and a sparse 3D graph". The key idea is to first compute optimal in-plane paths on a set of 2D slices, and then run a global A* search on a sparse 3D graph constructed from the union of these paths. In this way, the search state space is reduced from voxel-level grid nodes to a much smaller set of path-level nodes, significantly lowering the computational burden while preserving optimality on the sparse graph.

In the workspace frame, as shown in Fig. 6 for example, a spherical obstacle is placed at the center with its center at (30,30,15) and diameter 15. The start and goal positions are (5,10,5) and (55,50,25), respectively. Along the Z-axis, a set of horizontal planes $z = z_k$ is taken from 5 to 25 with a step size $\Delta z = 1$, yielding 21 slices. Intersecting the 3D sphere with each plane produces a stack of 2D occupancy grids. On each slice, the start and goal are projected to that plane, and an independent grid-based A* search is performed. The cost g(n) is defined as the accumulated Euclidean step length (unit cost for horizontal/vertical moves and $\sqrt{2}$ for diagonal moves), while the heuristic $h(n)$ is chosen as the Euclidean distance to the projected goal, ensuring that $h(n)$ never overestimates the true in-plane shortest-path cost. This yields an optimal 2D path P_k at each height z_k. A subset of these layer-wise A* results for six representative planes is shown in Fig. 7.

The second stage constructs a sparse 3D graph from these paths. All 3D nodes along the paths $\{P_k\}$, together with the start and goal, are merged and deduplicated to form a sparse node set V, with $|V| \ll N^3$. For each node in V, all 26-neighborhood offsets are checked, and an undirected edge is added whenever another node in V is found within this neighborhood. The edge weight is defined as the 3D Euclidean distance between the two nodes, resulting in a 26-connected sparse graph $G = (V, E)$.

In the final stage, A* is executed once more on the sparse 3D graph G. The cost term $g(n)$ is the accumulated sum of edge weights along the current path,

and the heuristic $h(n)$ is the Euclidean distance from node n to the 3D goal position. Since both the edge costs and the heuristic are based on Euclidean distance, and $h(n)$ does not overestimate the minimal remaining cost on G, A* is guaranteed to find the minimum-cost 3D geometric path on this sparse graph. The resulting 3D path is illustrated in Fig. 6.

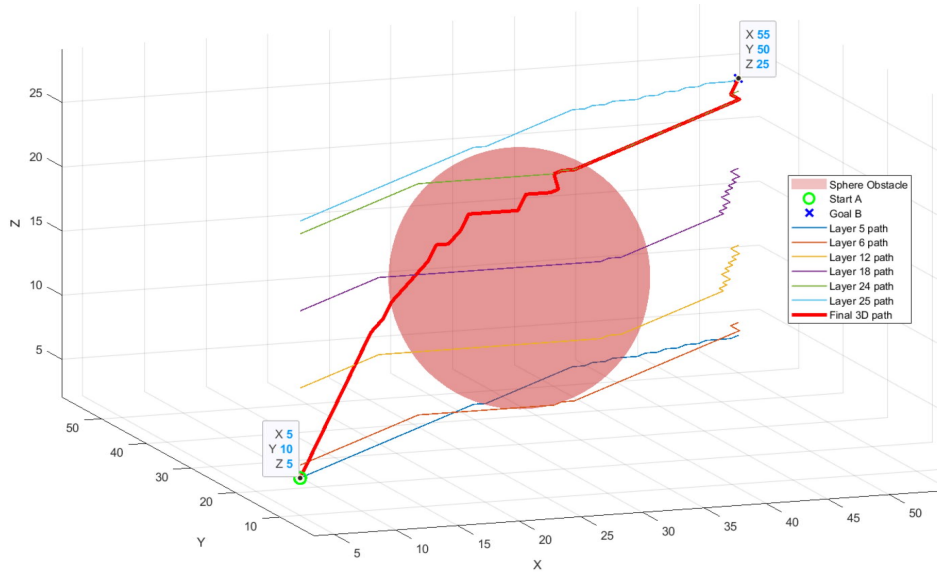

Fig. 6. A* path in 3D with a sphere obstacle

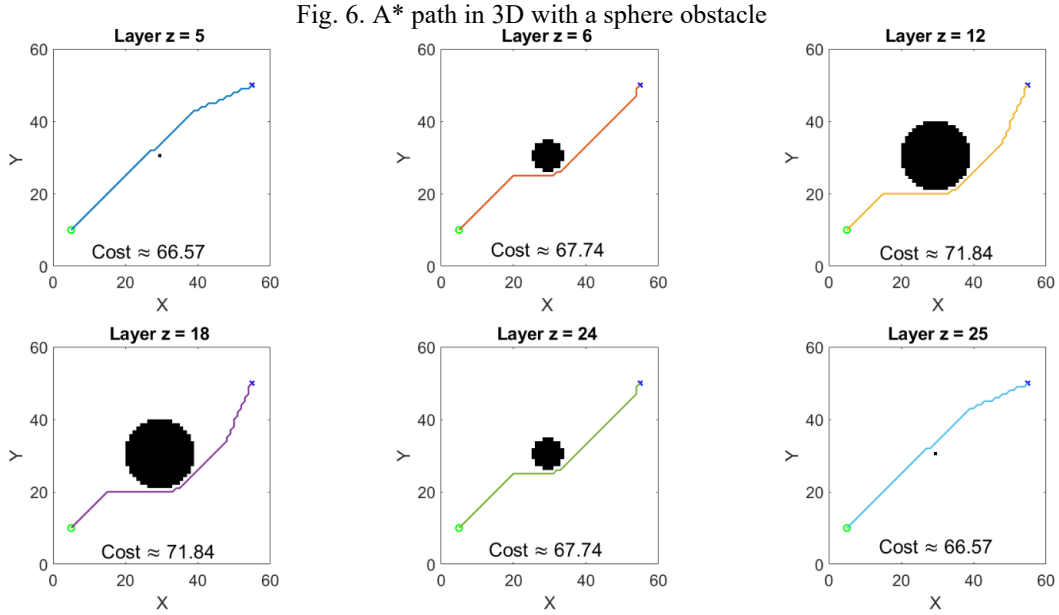

Fig. 7. layer-wise A* path (when z = 5, 6, 12, 18, 24, 25)

Compared with applying A* directly on the full 3D grid, the proposed two-level method reduces the search space from the voxel-level node set of

size $O(N^3)$ to the sparse node set V composed of layered path points, with the number of edges reduced from approximately $26N^3$ to $O(26|V|)$. As $|V|$ is typically orders of magnitude smaller than N^3, this leads to a substantial reduction in both computational time and memory usage, while retaining optimality with respect to the constructed sparse 3D graph.

4 Simulation results

Simulation setup. Simulations are conducted on the manipulator described in Section 2.1 with a single circular obstacle to validate the proposed C-space modelling and two-level A* planning pipeline. The geometric and kinematic parameters are summarized in Table 1, and the robot-obstacle arrangement in the 3D Cartesian workspace is illustrated in Fig. 8. Based on the circular-obstacle formulation in Section 2.2, the corresponding C-space obstacle distribution is computed and visualized in Fig. 9.

Layer-wise planning results. The proposed planner is evaluated by first performing layer-wise 2D A* searches on each slice plane, following the procedure in Section 3. The slice results reveal a feasibility gap across layers: as shown in Fig. 10, no valid 2D path is found for any slice when $\theta_3 \in [0°, 44°]$, while feasible paths emerge only when $\theta_3 \geq 45°$. To illustrate this effect, Fig. 10 presents representative slices at θ_3=0°, 22°, 44°, 45°, 66°, and 70°, where the first three slices contain no valid paths, whereas collision-free paths are obtained and plotted on the latter three slices. This behavior primarily arises because the first-stage search is performed within a fixed 2D slice (i.e., at a fixed θ_3), and on those slices with smaller θ_3, the forbidden region induced by the obstacle in C-space fully blocks the connectivity between the projected start and goal configurations; only when $\theta_3 \geq 45°$ does a connected free-space corridor become available. Consequently, for smaller θ_3, the layered stage cannot provide usable path nodes for establishing global 3D connectivity.

Connectivity handling. Because the start slice cannot connect to any slice with feasible layer-wise paths, directly constructing and searching a 3D graph from the raw layer-wise outputs cannot produce a start-to-goal solution in this scenario. To address this, the first waypoint on the earliest feasible slice ($\theta_3 = 45°$) is propagated backward to preceding infeasible slices, and the propagated paths are truncated to retain only the pre-contact (collision-free) waypoint segments before the obstacle boundary (Fig. 11). This processing yields a sparse waypoint set spanning from the start toward the feasible layers, enabling the subsequent 3D search to operate on a connected sparse representation .

3D search outcome. A* is then executed on the constructed sparse representation to obtain a 3D path (Fig. 12). Mapping the resulting 3D path back to C-space provides a joint-angle trajectory (θ_1, θ_2, θ_3) that drives the manipulator

through collision-free motion toward the target, as demonstrated by the simulated execution in Fig. 13. This result supports that the proposed two-level framework can still produce a valid avoidance trajectory even when a subset of layers contains no feasible in-plane path, provided that inter-layer connectivity is explicitly restored.

Table 1

Simulation Parameters of the Manipulator

Category	Symbol / Name	Value	Unit	Description
Link geometry	$[L_1, L_2, L_3]$	[1.0, 0.8, 0.4]	m	Length of Link 1/2/3
Link width	$[w_1, w_2, w_3]$	[0.05, 0.04, 0.03]	m	Half width of link 1/2/3
Obstacle	obstacle_xy= [x,y]	[1.0, 1.3]	m	Obstacle center coordinate
Obstacle	obstacle_r	0.15	m	Obstacle radius
Joint limits	$\theta_{\min}$ to $\theta_{\max}$	0-90	deg (°)	Bound of joint angle grid
Joint sampling	$\Delta\theta$	1	deg (°)	Joint angle grid step
Start config	A_deg = $[\theta_1,\theta_2,\theta_3]$	[0,0,0]	deg (°)	Start joint configuration
Goal config	B_deg = $[\theta_1,\theta_2,\theta_3]$	[45,60,70]	deg (°)	Goal joint configuration

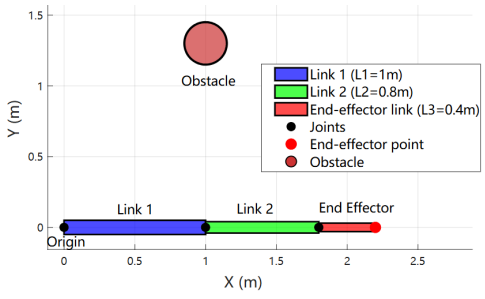

Fig. 8. Manipulator and obstacle in Cartesian space

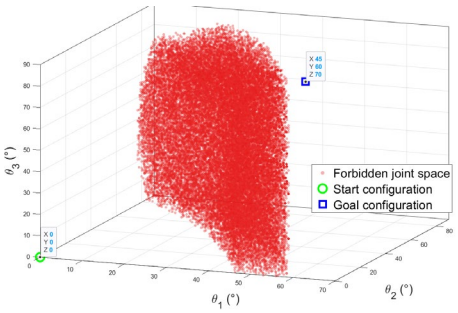

Fig. 9.Manipulator and obstacle in C-space

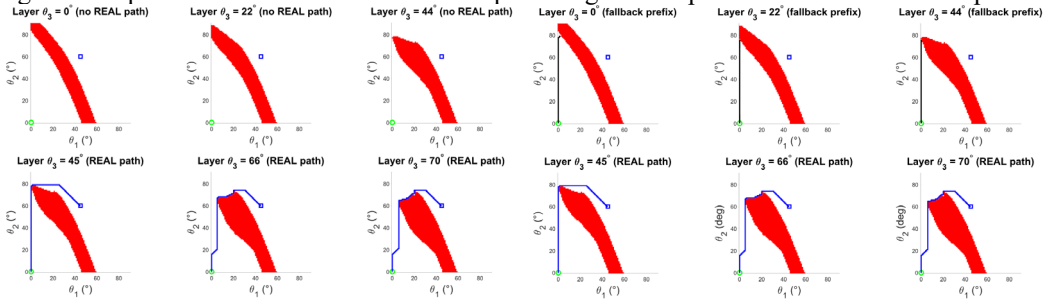

Fig. 10. 2D A* paths on slice planes

Fig. 11. Processed slice paths for sparse 3D A*

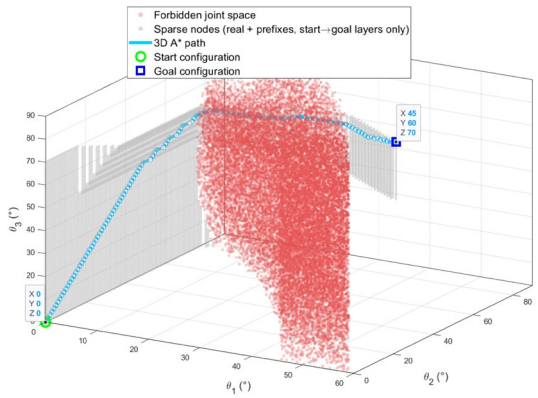

Fig. 12. 3D path from the sparse A* search

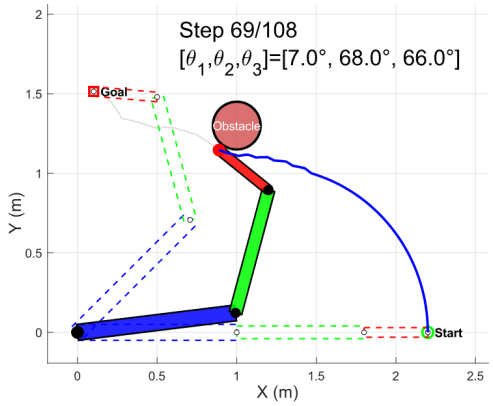

Fig. 13. Joint-space execution of the planned 3D path

Comparison with dense 3D A*. Under the same simulation setup, a dense global A* baseline is implemented on the same 91×91×91 joint-space grid using 26-neighborhood connectivity and Euclidean edge costs. The planned paths are compared in C-space in Fig. 14 and in the Cartesian workspace in Fig. 15. Visually, the two methods produce very similar paths when approaching and bypassing the obstacle, while the dense A* trajectory appears slightly smoother in regions far from the obstacle.Quantitative results are summarized in Fig. 16: dense A* expands approximately 1.5×10^5 nodes, whereas the proposed sparse-graph 3D A* expands about 6.6×10^3 nodes, reducing the search scale by more than one order of magnitude. The planning time decreases from 23.5s to about 2.6s (≈ 9× speed-up), while the A* cost remains nearly unchanged (145.21 vs 145.80, <0.5% relative error) and the number of waypoints is similar (107 vs 108).

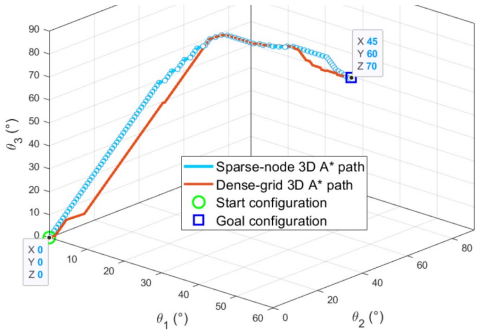

Fig. 14. 3D path from the sparse A* search

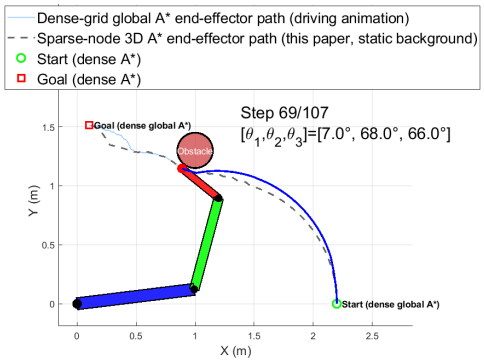

Fig. 15. Workspace execution of the planned 3D path

```
Running sparse-node 3D A* (this paper)...
Sparse 3D A*: SUCCESS
  #points            = 108
  A* reported cost   = 145.800739
  Expanded nodes     = 6605
  Time               = 2.555572 s
```

```
Running dense-grid 3D A* (global A*) on 91x91x91 grid...
Dense 3D A*: SUCCESS
  #points            = 107
  A* reported cost   = 145.214952
  Expanded nodes     = 154647
  Time               = 23.503668 s
```

Fig. 16. Quantitative comparison between dense 3D A and the proposed sparse 3D A*

5 Discussion

This study proposes a collision-free path-planning framework for a 4-DOF serial harvesting manipulator by combining analytical C-space modelling with a two-level A* strategy based on layered 2D slicing and a sparse 3D graph. Simulation results show that the planner can generate feasible 3D collision-free paths and map them to executable joint-angle trajectories, even when some 2D slices (e.g., no valid 2D path for $\theta_3 \in [0°, 44°]$ admit no in-plane solution. The slice-wise feasibility gap motivates the proposed inter-layer connectivity handling, which restores connectivity by propagating and truncating waypoints from the first feasible slice, ensuring the sparse 3D graph remains searchable from the start.

Compared with dense global 3D A* on a 91×91×91 grid, the sparse-graph planner drastically reduces node expansions and planning time while maintaining nearly the same A* cost and waypoint count. Qualitatively, both methods behave similarly near the obstacle, while the dense baseline appears slightly smoother in obstacle-free regions due to finer discretization. These results indicate that constructing a sparse 3D graph from informative 2D slice solutions can preserve path quality in typical obstacle-avoidance scenarios while significantly compressing the search space, offering a practical trade-off between efficiency and solution quality for 4-DOF harvesting arms in structured environments.

The framework is designed for 4-DOF harvesting manipulators in orchard or vineyard settings, where the end-effector must reach a target fruit cluster inside a dense canopy. Typical obstacles include trellis wires, stems/branches, and neighboring fruits, which create narrow corridors and require planning in C-space. Given a detected target pose and an approximate obstacle map, the method maps obstacles into C-space forbidden regions and computes a collision-free joint-space trajectory via the two-level A* planner, enabling safe navigation through cluttered vegetation under constrained clearances.

Several limitations remain. First, since the final 3D search is performed on a sparse graph constructed from layer-wise solutions (rather than the full continuous C-space), the resulting trajectory is guaranteed optimal only with respect to that constructed graph and may be suboptimal in global length or smoothness in continuous space. Second, the current validation is simulation-based and focuses on static, simplified obstacle models, so real-world factors such as sensing uncertainty, unmodelled branch compliance, and dynamic disturbances are not captured; future work should therefore include more complex environments and, ultimately, real-orchard evaluations, together with post-processing (e.g., smoothing/optimization) to further improve trajectory quality.

6 Conclusion

This paper presented a collision-free motion-planning framework for a 4-DOF serial harvesting manipulator by integrating analytical C-space obstacle modelling with a hierarchical A* planning pipeline. The planner constructs a compact search domain via layer-wise 2D C-space searches and then performs a final A* search on a connected sparse 3D graph, explicitly restoring inter-layer connectivity when some 2D slices admit no feasible in-plane path.

In simulation, the proposed planner consistently produced feasible 3D collision-free paths and demonstrated markedly improved planning efficiency compared with a dense global 3D A* baseline on a $91\times91\times91$ grid. Specifically, it reduced the number of expanded nodes by approximately 95.6% (from 1.5×10^5 to 6.6×10^3) and planning time by roughly 89% (from 23.5s to 2.6s), representing a nearly 9-fold speed-up. Despite this significant reduction in computational cost, the method maintains path quality comparable to the global optimum, with a deviation in the reported A* cost (which corresponds directly to the total geometric path length of the planned trajectory) of less than 0.5% (145.21 vs. 145.80) and a nearly identical waypoint count (107 vs 108). These results demonstrate that the proposed sparse-graph approach achieves a favorable trade-off between planning efficiency and solution quality for 4-DOF harvesting manipulators.

Future work will extend the validation to cluttered and dynamic environments with sensing uncertainty, refine the sparse 3D graph and incorporate post-planning smoothing/optimization to further improve trajectory quality, and deploy the planner on real hardware for field trials in orchards to assess robustness, scalability, and practical utility in large-scale robotic harvesting applications.

Acknowledgments

This work was supported by the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2022B1515120002), the Project of Shenzhen Science and Technology Innovation Committee (Grant No. KJZD20240903103300002), the Project of Shenzhen Science and Technology Innovation Committee (Grant No. KCXFZ20240903094011015), and other Project (Grant No. PT2024E007; 2024HT009，KJ2025C005).

REFERENCES

[1]. *A. Prasad, K. Chaudhary, B. Sharma, et al.,* “Robot path planning and motion control: a systematic review”, in Engineered Science, vol. 33, 2024.

[2]. *S. Chakraborty, D. Elangovan, P. L. Govindarajan, et al.*, "A comprehensive review of path planning for agricultural ground robots", in Sustainability, vol. 14, no. 15, 2022.
[3]. *L. Yang, P. Li, S. Qian, et al.*, "Path planning technique for mobile robots: A review", in Machines, vol. 11, no. 10, 2023.
[4]. *H. Zhao, J. Zhao, X. Duan, et al.*, "Research on collision avoidance control of multi-arm medical robot based on C-Space", in IEEE Access, vol. 8, 2020, pp. 93219–93229.
[5]. *S. K. Sahoo, and B. B. Choudhury*, "A review of methodologies for path planning and optimization of mobile robots", in Journal of Process Management and New Technologies, vol. 11, no. 1-2, 2023, pp. 122–140.
[6]. *T. Jin, and X. Han,* "Robotic arms in precision agriculture: A comprehensive review of the technologies, applications, challenges, and future prospects", in Computers and Electronics in Agriculture, vol. 221, 2024.
[7]. *J. E. Sinebe, I.P. Okokpujie, O. Folorunso, et al.,* "Modelling and simulation of a path-planning mobile robot for sustainable agricultural applications", in ARPN Journal of Engineering and Applied Sciences, vol. 19, no. 13, Dec. 2024, pp. 1389–1399.
[8]. *B. Chen, L. Gong, C. Yu, et al.*, "Workspace decomposition based path planning for fruit-picking robot in complex greenhouse environment", in Computers and Electronics in Agriculture, vol. 215, 2023.
[9]. *L. Luo, H. Wen, Q. Lu, et al.*, "Collision-Free Path-Planning for Six-DOF Serial Harvesting Robot Based on Energy Optimal and Artificial Potential Field", in Complexity, vol. 2018, no. 1, 2018.
[10]. *C. Fang, J. Wang, F. Yuan, et al.*, "Path Planning for Dragon-Fruit-Harvesting Robotic Arm Based on XN-RRT* Algorithm", in Sensors, vol. 25, no. 9, 2025.
[11]. *X. Li, T. Chen, R. Han, et al.*, "Research on path planning of robotic arm with improved RRT algorithm in uncertain environment for harvesting", in Journal of Chinese Agricultural Mechanization, vol. 45, no. 4, 2024, pp. 193–198.
[12]. *X. Cao, X. Zou, C. Jia, et al.,* "RRT-based path planning for an intelligent litchi-picking manipulator", in Computers and Electronics in Agriculture, vol. 156, 2019, pp. 105–118.
[13]. *X. Li, J. Yang, X. Wang, L. Fu, et al.,* "Adaptive Step RRT*-Based Method for Path Planning of Tea-Picking Robotic Arm", in Sensors, vol. 24, no. 23, 2024.
[14]. *L. Yang, P. Li, T. Wang, et al.*, "Multi-area collision-free path planning and efficient task scheduling optimization for autonomous agricultural robots", in Scientific Reports, vol. 14, no. 1, 2024.
[15]. *A. A. Bakhtiari, H. Navid, J. Mehri, et al.,* "Optimal route planning of agricultural field operations using ant colony optimization", in Agricultural Engineering International: CIGR Journal, vol. 13, no. 4, 2011.
[16]. *Y. Wang, C. Fu, and R. Huang, et al*., "Path planning for mobile robots in greenhouse orchards based on improved A* and fuzzy DWA algorithms", in Computers and Electronics in Agriculture, vol. 227, Dec. 2024.
[17]. *C. Xiong, J. Xiong, Z. Yang, et al*., "Path planning method for citrus picking manipulator based on deep reinforcement learning", in Journal of South China Agricultural University, vol. 44, no. 3, 2023, pp. 473–483.
[18]. *Q. Zhang, F. Liu, X. Jiang, et al.*, "Motion planning method and experiments of tomato bunch harvesting manipulator", in Transactions of the Chinese Society of Agricultural Engineering (Transactions of the CSAE), vol. 37, no. 9, 2021, pp. 149–156.