WGIT*: Workspace-Guided Informed Tree for Motion Planning in Restricted Environments

Zhixing Zhang, Yanjie Chen, Feng Han, Junwei Fan, Hongshan Yu, Hui Zhang, Yaonan Wang

***Abstract*—The motion planning of robots faces formidable challenges in restricted environments, particularly in the aspects of rapidly searching feasible solutions and converging towards optimal solutions. This paper proposes Workspace-guided Informed Tree (WGIT*) to improve planning efficiency and ensure high-quality solutions in restricted environments. Specifically, WGIT* preprocesses the workspace by constructing a hierarchical structure to obtain critical restricted regions and connectivity information sequentially. The refined workspace information guides the sampling and exploration of WGIT*, increasing the sample density in restricted areas and prioritizing the search tree exploration in promising directions, respectively. Furthermore, WGIT* utilizes gradually enriched configuration space information as feedback to rectify the guidance from the workspace and balance the information of the two spaces, which leads to efficient convergence toward the optimal solution. The theoretical analysis highlights the valuable properties of the proposed WGIT*. Finally, a series of simulations and experiments verify the ability of WGIT* to quickly find initial solutions and converge towards optimal solutions.**

***Index Terms*—Motion planning, workspace-guided sampling, workspace-guided exploration, restricted environments**

I. Introduction

Motion planning is an essential component of robotics that enables robots to navigate and perform tasks in various environments. Representative planners commonly utilized in practical applications include sampling-based planners (SBPs), such as rapidly-exploring random trees (RRT) [1], probabilistic roadmaps (PRM) [2], and their variants [3], [4]. Although SBPs are efficient and widely applicable, restricted configuration space (C-space) severely compromises the planning efficiency of SBPs [5], as narrow regions in the C-space create bottlenecks for the sampling and exploration [6]. Given that various robotic applications involve environments with restricted C-space, such as transportation [7], healthcare [8], and manufacturing [9], it is imperative to mitigate the impact of restricted environments on SBPs.

In light of that the constraints on planning in restricted environments mainly depend on the distribution of obstacles in the C-space, adapting the sampling and expansion strategies based on the C-space information is a widely adopted approach to enhance the efficiency of SBPs in restricted environments. RRV [10] employed principal component analysis (PCA) to investigate the restricted regions in C-space and adapted various exploration strategies to enhance the expansion speed of the search tree within the restricted areas. LGM-BRRT* [11] preprocessed the C-space to identify narrow regions and generated local trees within these regions to enable efficient traversal of restricted areas. Besides, informed search is an effective strategy for dealing with restricted environment problems. By constructing an informed set to shrink the planning space, the informed search can concentrate the search on critical restricted regions, enhancing the sampling density and reducing redundant expansions. Informed RRT* [12] incorporated the informed set into RRT* to accelerate the convergence of RRT* toward a high-quality solution in restricted environments. BIT* [13] employed both the informed set and batch sampling strategies to further enhance the speed of informed search and then facilitate the integration with various sampling strategies. The study in [14] incorporated a biased sampling strategy for restricted regions based on BIT*, thereby improving the planning efficiency of BIT* in restricted environments. Although the algorithms mentioned above effectively enhance the ability of SBPs to handle problems in constrained environments, the improvement may be compromised by redundant sampling and exploration. Such redundancy is attributed primarily to a lack of utilization of workspace information, which is essential for guiding SBPs to focus the searches on critical areas in C-space to find solutions efficiently.

Considering that workspace information is easily obtainable and can help reduce redundant searches in C-space, many approaches have attempted to enhance the performance of SBPs in dealing with complex and restricted environments by directly planning in the workspace. For example, TS-RRT [15] dealt with planning problems in a high-dimensional C-space

(Corresponding author: Yanjie Chen. chenyanjiehnu@gmail.com)

This work was supported in part by the National Key R & D Program of China under Grant (No. 2021ZD0114503), National Natural Science Foundation of China (No. 62273098, 62027810, 61971071, 62133005), the Major Research plan of the National Natural Science Foundation of China (Grant No. 92148204), the Royal Society, UK (under the Newton International Fellowship Scheme, No. NIF\R1\221089), the Hunan Leading Talent of Technological Innovation under Grant (No. 2022RC3063), the Hunan Science Fund for Distinguished Young Scholars under Grant (No. 2021JJ10025), the Hunan Key Research and Development Program under Grants (No. 2021GK4011, 2022GK2011), the Changsha Science and Technology Major Project under Grant (No. kh2003026), the Joint Open Foundation of state Key Laboratory of Robotics under Grant (No. 2021-KF-22-17), and the China University Industry-University-Research Innovation Fund under Grant (No. 2020HYA06006).

Zhixing Zhang, Feng Han, and Junwei Fan are with the School of Mechanical Engineering and Automation, Fuzhou University, Fuzhou 350108, China (e-mail: 210227164@fzu.edu.cn; 220220032@fzu.edu.cn; fanjunwei0122@163.com).

Yanjie Chen is with School of Mechanical Engineering and Automation, Fuzhou University, China, 350108, Department of Computer Science, Aberystwyth University, Aberystwyth, UK, SY23 3DB, and the National Engineering Research Center of Robot Visual Perception and Control Technology, Hunan University, China, 410082 (e-mail: chenyanjiehnu@gmail.com).

Hongshan Yu, and Yaonan Wang is with the College of Electrical and Information Engineering, Hunan University, Changsha 410082, China(e-mail: yuhongshancn@163.com, yaonan@hnu.edu.cn).

Hui Zhang is with the School of Robotics, Hunan University, Changsha 410082, China (e-mail: zhanghui1983@hnu.edu.cn).

by planning in a low-dimensional workspace. However, due to the lack of attention to the connection between the workspace and C-space, TS-RRT may require additional movements within the C-space for the same trajectory in the workspace, and even potentially leading to the omission of feasible solutions. To overcome the issue of TS-RRT failing to find a solution due to multiple regions in C-space mapping to the same region in the workspace, [16] proposed a hierarchical planner that decomposed the workspace and generated a connected graph to guide the planning. EET [17] integrated workspace preprocessing methods into workspace planning by constructing a sphere-tree in the workspace to obtain workspace information. With the sphere-tree, EET adaptively adjusted the sampling strategy in narrow areas, and then rapidly guided the exploration. Notwithstanding, due to the reliance on greedy search, the sphere-tree may provide inappropriate guidance in complex environments. Consequently, EET may fail in environments where the sphere tree connectivity differs from the C-space connectivity, thus lacking the capability to achieve probabilistic completeness. In general, workspace-based SBPs directly utilize the informative low-dimensional workspace, which partially solves the problem of inefficiency in restricted areas. However, few workspace-based SBPs directly focus on restricted areas and handle these areas accordingly. Moreover, SBPs planning in the workspace generally rely on inverse kinematics solutions, which may result in a significant computational burden for robotic systems and tasks with relatively complex inverse kinematics. Furthermore, since planning is carried out in the workspace with little consideration to the C-space, ensuring asymptotic optimality remains a significant challenge.

Given the complementary advantages of C-space planning and workspace planning, this paper proposes a dual-space planner called workspace-guided informed tree (WGIT*). The main contributions of this paper can be summarized as:

- The proposed workspace preprocessing method transforms implicit key information within workspace into concise and precise guidance directly utilizable by WGIT*. As a result, unnecessary workspace information can be eliminated to reduce computational burdens in complex restricted environments.
- The proposed workspace-guided planning strategy utilizes workspace guidance information to enhance sampling and exploration processes, thus preventing excessive sampling and redundant exploration. Consequently, WGIT* can rapidly obtain high-quality solutions in restricted environments.
- The proposed dual-space information updating method dynamically balance two spatial information spaces, mitigates the issues of inconsistency information between the two space and improper guidance from workspace. Therefore, WGIT* can comprehensively utilize the information from both spaces and achieve balanced overall performance.

II. Problem Formulation

This section provides a necessary introduction to the critical notions involved in this study and defines the path planning problem to facilitate the subsequent introduction and theoretical analysis of WGIT*.

A. Preliminaries

The workspace of a robot is denoted as $\mathcal{W} \subset \mathbb{R}^n$ with $n \in \{2, 3\}$. The region of space occupied by the obstacles in the workspace is denoted as $\mathcal{W}_{\text{obs}}$. Hence, the free workspace is defined as $\mathcal{W}_{\text{free}} = \mathcal{W} \setminus \mathcal{W}_{\text{obs}}$. Let $\mathcal{C} \in \mathbb{R}^m$ denote the C-space, where $m \in \mathbb{N}^*$. The configuration of the robot is uniquely determined by a joint vector $q \in \mathbb{R}^m$. Let $\mathcal{A}(q) \in \mathbb{R}^n$ denote the space occupied by the robot body in $\mathcal{W}$ when the robot is in configuration q. The space of obstacles in the configuration space can be represented as $\mathcal{C}_{\text{obs}}$, where $\forall q \in \mathcal{C}_{\text{obs}} : \mathcal{A}(q) \cup \mathcal{W}_{\text{obs}} \neq \emptyset$. Thus, the free configuration space is defined as $\mathcal{C}_{\text{free}} = \mathcal{C} \setminus \mathcal{C}_{\text{obs}}$.

WGIT* uses a search tree constructed in the C-space to achieve motion planning. The search tree in the C-space is represented as $\mathcal{T} = (V, E)$, where $V \subset \mathcal{C}_{\text{free}}$ is the set of nodes in the tree, and E is the set of edges constructed by the nodes in V. The set Q comprises all samples in the C-space. The set of samples that are not connected to $\mathcal{T}$ is denoted as $Q_{\text{uncon}} \subset Q$. The set $V_{\text{unexp}} \subset V$ comprises the nodes in the tree that are not expanded. Two order queues $\mathcal{E}_V$ and $\mathcal{E}_E$ are used to store the nodes to be expanded in $\mathcal{T}$ and the edges to be added to $\mathcal{T}$, respectively. The values of nodes and edges are arranged according to their heuristic function values, which will be described in detail in Section III-C. The cost $\hat{c}(q_i, q_j)$ and $c(q_i, q_j)$ represent the estimated distance and true distance between configurations $q_i, q_j \in Q$, respectively. When the edge (q_i, q_j) is collision-free with $\mathcal{C}_{\text{obs}}$, the value of $c(q_i, q_j)$ is infinity. The path found by WGIT* is denoted as $\sigma_{\text{WGIT*}}$, which undergoes continuous improvement through the planning. The cost of $\sigma_{\text{WGIT*}}$ in the i^{th} iteration is denoted as c_i. The informed set is denoted as $\mathcal{C}_{\text{inf}}$.

Let $f_{\text{FK}} : \mathbb{R}^m \to \mathbb{R}^n$ be the forward kinematics function that computes the position $p \in \mathbb{R}^n$ of the reference point in the workspace. The reference point is a point fixed in the frame of a specific part of the robot. For serial manipulators, the reference point is typically the working point of the end effector. The reference points corresponding to q_{start} and q_{goal} are p_{start} and p_{goal}, respectively. Based on the forward kinematic mapping, $\mathcal{T}$ can be mapped to the workspace to obtain the workspace search tree $\mathcal{T}_{\mathcal{W}}$.

B. Problem Definitions

Definition 1. (Feasible motion planning) *Let q_{start} and q_{goal} denote the starting and goal configurations of the robot, respectively. Let $\sigma : [0, 1] \to \mathcal{C}_{\text{free}}$ denote a feasible solution. Feasible motion planning aims to find a path σ that satisfies $\sigma(0) = q_{\text{start}}, \sigma(1) = q_{\text{goal}}$. If such a path does not exist, the planner returns a failure.*

Definition 2. (Optimal motion planning) *Let Σ denote the set of all feasible solutions. Let $\mathcal{L}_{\text{c}} : \sigma \to \mathbb{R}_{\geq 0}$ denote a cost function. The cost of an infeasible solution is infinite, i.e., for all $\sigma \notin \Sigma$, $\mathcal{L}_{\text{c}}(\sigma) = \infty$. Optimal motion planning aims to find the optimal solution σ^* that satisfies*

$$\sigma^* = \arg\min_{\sigma \in \Sigma} \{\mathcal{L}_{\text{c}}(\sigma) \mid \sigma(0) = q_{start}, \sigma(1) = q_{goal}\}. \quad (1)$$

Definition 3. *(Cost function) To facilitate application in practical robotic systems, a solution is represented by a set of discrete path points denoted as $\sigma = \{\sigma(1), \sigma(2), ..., \sigma(n)\}$, while the specific representation of the cost function is defined as*

$$\mathcal{L}_c(\sigma) = \sum_{i=1}^{n-1} \|(\sigma(i+1) - \sigma(i))\| \tag{2}$$

III. Methodology

In this section, WGIT* is presented in detail, starting with a high-level overview of WGIT* in Section III-A. Subsequently, Sections III-B through III-D delve into the detailed explanation of the primary components of WGIT*. Finally, Section III-E provides a theoretical analysis of WGIT*.

*A. The Framework of WGIT**

The framework of the WGIT* is shown in Alg. 1, and Fig. 1 provides a flowchart of WGIT*. WGIT* consists of three main modules: workspace preprocessing, workspace-guided planning, and dual-space information updating.

1) Workspace preprocessing: To effectively leverage critical information in the workspace, WGIT* employs a series of techniques during the initial planning phase (Alg. 1, lines 1∼4). Firstly, the workspace is decomposed into an octree or a quadtree to facilitate the extraction of valuable information in the workspace. Next, critical regions within the workspace are identified based on the workspace decomposition. Subsequently, the A* algorithm is used to obtain a reference solution in the workspace, which captures essential connectivity information and further refines critical regions. By utilizing the information provided by the reference solution and active critical regions, WGIT* can guide the planning process effectively and exclude areas of low importance, thereby reducing the computational burden.

2) Workspace-guided planning: With the guidance from the workspace, WGIT* initiates the main planning process (Alg. 1, line 5 $\sim$ 15), which includes workspace-guided sampling and workspace-guided exploration. During the workspace-guided sampling phase , WGIT* algorithm leverages the information from active critical regions in the workspace to increase the density of samples in restricted areas of the C-space. The workspace-guided sampling leads to a balanced distribution of samples, which in turn facilitates the detection of connectivity in the C-space. During the workspace-guided exploration phase, the workspace-guided heuristic function guides the search tree toward promising directions based on the reference solution and active critical regions. As a benefit, the number of redundant explorations is reduced, and then the planning efficiency is improved.

3) Dual-space information updating: In order to enhance the accuracy of workspace guidance and fully utilize the information from both spaces, continuous exchange and updating of information between the two spaces are necessary. The updating of dual-space information is mainly achieved through workspace guidance switching and dual-space information balancing. Workspace guidance switching phase addresses improper guidance through feedback from the C-space. The workspace information balancing phase dynamically adjusts the weight of information from the two spaces during the planning process, avoiding excessive reliance on a single space and ensuring that WGIT* converges to the optimal solution.

Algorithm 1: WGIT*

```
Input: $q_{\text{start}}$, $q_{\text{goal}}$
Output: $\mathcal{T}$
// Workspce preprocessing
1  $\mathcal{S} \leftarrow$ Decompose($\mathcal{W}, \mathcal{W}_{\text{obs}}$);
2  $\mathcal{N} \leftarrow$ FindCriticalRegions($\mathcal{S}$);
3  $\mathcal{P} \leftarrow$ FindReferenceSol($f_{\text{locate}}(q_{\text{start}}), f_{\text{locate}}(q_{\text{goal}}), \mathcal{S}$);
4  $\mathcal{N}_{\mathcal{P}} \leftarrow \{\eta \in \mathcal{N} \mid \eta \cap \mathcal{P} \neq \emptyset\}$;
// Planner initialization
5  $V \leftarrow q_{\text{start}}; E \leftarrow \emptyset; \mathcal{T} \leftarrow (V, E)$;
6  $Q_{\text{uncon}} \leftarrow q_{\text{goal}}$;
7  $\mathcal{E}_{\text{V}} \leftarrow V; \mathcal{E}_{\text{E}} \leftarrow \emptyset$;;
8  $V_{\text{unexp}} \leftarrow V$;
// Workspace-guided planning
9  while not PTC do
10     if $\mathcal{E}_E = \emptyset$ and $\mathcal{E}_V = \emptyset$ then
11         $Q_{\text{reuse}} \leftarrow$ Prune($\mathcal{T}, Q_{\text{uncon}}, c_i$);
12         $Q_{\text{sample}} \leftarrow$ GuidedSample($\mathcal{N}_{\mathcal{P}}, n$);
13         $Q_{\text{uncon}} \leftarrow Q_{\text{sample}} \cup Q_{\text{reuse}}$;
14     $\mathcal{T}$ = Explore($Q, \mathcal{P}$);
15 $\sigma_{\text{WGIT*}} \leftarrow$ FindPath($\mathcal{T}$);
16 return $\sigma_{\text{WGIT*}}$;
```

Through the coordination between these three modules, WGIT* utilizes workspace information to efficiently generating high-quality paths for robots to complete tasks in restricted environments.

B. Workspace Preprocessing

The main objective of the workspace preprocessing module is to simplify the workspace and extract information, such as narrowness, connectivity and cost, providing guidance for path planning in the C-space. The simplification and extraction of workspace information are achieved through three phases: quadtree/octree decomposition, critical region search, and reference solution search.

1) Octree/Quadtree decomposition: To tap the information from the workspace, WGIT* decomposes the workspace $\mathcal{W}$ into an octree (or quadtree) [18], denoted as $\mathcal{S}$, as shown in Fig. 2. The obtained octree (or quadtree) is composed of cells in different sizes, presenting a hierarchical structure. A cell at level k is represented as s^k, and the set of all cells at level k is denoted by $\mathcal{S}^k$. The octree/quadtree can approximate the workspace at different resolutions based on the narrowness, which can be used to locate narrow areas in

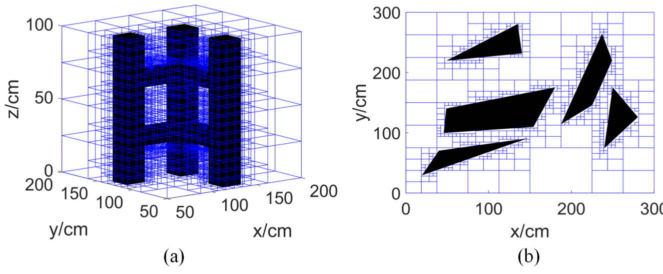

Fig. 2: Workspace decomposition. (a) Decompose $\mathcal{W} \subset \mathbb{R}^3$ into a octotree. (b) Decompose $\mathcal{W} \subset \mathbb{R}^2$ into a quadtree.

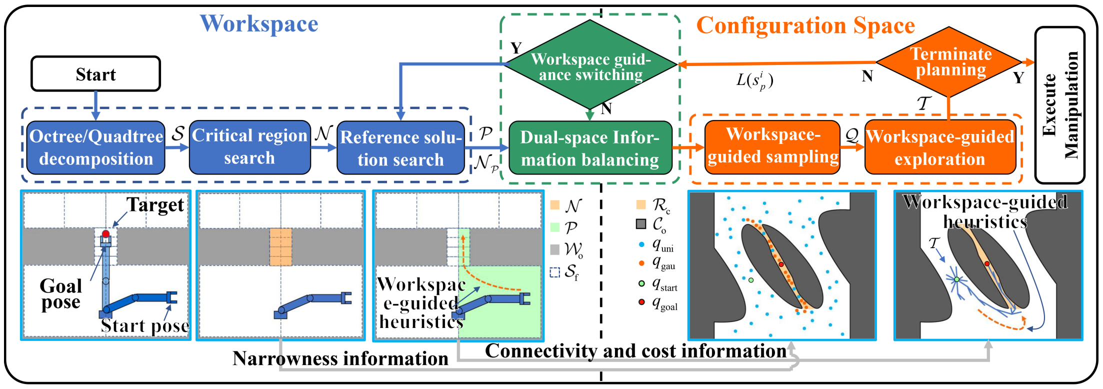
Workspace
Configuration Space
Start
Octree/Quadtree decomposition
Critical region search
Reference solution search
Workspace guidance switching
Dual-space Information balancing
Workspace-guided sampling
Workspace-guided exploration
Terminate planning
Execute Manipulation
Target
Goal pose
Start pose
Workspace-guided heuristics
Narrowness information
Connectivity and cost information

Fig. 1: Framework flowchart of WGIT*. The blue dashed box represents the workspace preprocessing module, the orange dashed box represents the workspace-guided planning module, and the green dashed box represents the dual-space information updating module.

the workspace and simplify information in open areas. Based on the interference between the cells and the obstacles $\mathcal{W}_{\text{obs}}$, the status of a cell can be one of three types: *empty* (indicating that the cell is obstacle-free), *full* (indicating that the cell is completely occupied by obstacles), or *mixed* (indicating that the cell is partially occupied by obstacles). For a given cell, the variables $\varphi_{\text{state}}(s) \in \{\text{empty}, \text{full}, \text{mixed}\}$, $\varphi_{\text{size}}(s) \in \mathbb{R}^+$, and $\varphi_{\text{center}}(s) \in \mathbb{R}^n$ denote its state, maximum edge length, and center coordinates, respectively. The collection of cells with a state of empty is designated as the set of free cells, denoted by $\mathcal{S}_{\text{free}} := \{s \in \mathcal{S} \mid \varphi_{\text{state}}(s) = \text{empty}\}$. The function $f_{\text{locate}} : q \rightarrow \mathcal{S}_{\text{free}}$ locates the free cell containing the given position $f_{\text{FK}}(q)$. The construction of the octree/quadtree begins with the whole workspace $\mathcal{W}$ as the initial cell, and then the mixed cells are recursively subdivided until each cell either reaches a state of empty or full. Alternatively, the subdivision process is terminated when the size of each mixed cell reaches a predetermined resolution.

The constructed octree/quadtree contains information regarding the narrowness and connectivity of the workspace, providing a foundation for extracting critical information from the workspace.

2) Critical region search: The octree/quadtree contains valuable implicit information of narrowness in the form of critical regions, denoted as $\mathcal{N} \subset \mathcal{S}_{\text{free}}$. These critical regions refer to areas in the workspace that are significantly constrained by obstacles $\mathcal{W}_{\text{obs}}$ and are essential for establishing connectivity. Due to the small clearance with obstacles $\mathcal{W}_{\text{obs}}$, the critical regions $\mathcal{N}$ often align with the restricted regions in the C-space. Therefore, utilizing the critical regions $\mathcal{N}$ to guide planning in the C-space is a practical strategy.

To identify the critical region $\mathcal{N}$ from $\mathcal{S}$, a critical area recognition method, denoted as `FindCriticalRegions`, is developed based on watershed algorithm [19]. The algorithm `FindCriticalRegions` performs a layer-by-layer search of the free cells $\mathcal{S}_{\text{free}}$, assigning different labels to non-adjacent regions. As the search depth increases, the regions with different labels will gradually expand and eventually intersect. The intersecting regions will be marked as watersheds. The cells with suitable size near each watershed can be merged

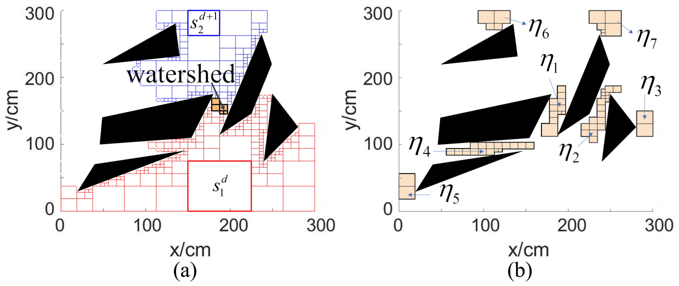

Fig. 3: Identify critical regions in a $\mathbb{R}^2$ workspace. (a) The process of identifying critical regions. (b) The critical regions $\mathcal{N}$ in the workspace

to obtain the critical regions $\mathcal{N}$. For instance, in the cluttered environment as depicted in Fig. 3(a), the blue and red regions continuously expand, meet in the narrow central passage, and form one of the watersheds. Subsequently, this watershed merges with other adjacent cells to form the critical region η_1 in Fig. 3(a), and the remaining six critical regions are obtained similarly.

Through the critical area search phase, workspace information has been simplified into a set of critical regions, denoted as $\mathcal{N} := \{\eta_1, \eta_2, ..., \eta_n\}$. However, for specific planning problems, not all of these critical areas are involved. Thus, a search for a workspace reference solution is necessary to eliminate the regions that are less significant for robot motion and to furnish planning with additional crucial information.

3) Reference solution search: The reference solution, denoted as $\mathcal{P}$, is a path of the reference point of robot in the workspace, encompassing information such as connectivity and cost within the workspace. According to the forward kinematic mapping and Definition 1, collision-free continuous paths in C-space correspond to collision-free continuous paths in the workspace. Thus, the reference solution can be used to assess the corresponding feasible path in C-space, which guides the exploration phase of WGIT*.

The reference solution $\mathcal{P}$ is derived from $\mathcal{S}$. As the workspace is decomposed into cells, the connectivity of the workspace can be conveniently represented as a graph, denoted as $\mathcal{G}$. The nodes of $\mathcal{G}$ are the free cells $\mathcal{S}_{\text{free}}$. Two nodes are adjacent if their corresponding cells share a common surface (for octree) or edge (for quadtree). The edges of $\mathcal{G}$ represent the adjacency between neighboring cells, and the cost of each edge is defined as the distance between the centers of the two corresponding cells. Based on the constructed graph, the reference solution $\mathcal{P}$ can be obtained by applying the A* algorithm with the start and end nodes set as $f_{\text{locate}}(q_{\text{start}})$ and $f_{\text{locate}}(q_{\text{goal}})$, respectively. The reference solution $\mathcal{P}$ is composed of a sequence of adjacent cells $s_{\mathcal{P},i} \in \mathcal{S}_{\text{free}}$, as shown in Fig. 4.

According to the relationship between the reference solution and the feasible path in the C-space, the critical regions traversed by the reference solution are likely to correspond to the regions that are necessary to be traversed in the C-space. These regions are referred to as key regions in the C-space and denoted as $\mathcal{R}_{\mathcal{C}}$. Consequently, the critical regions traversed by the reference solution are designated as active critical regions, denoted as $\mathcal{N}_{\mathcal{P}} \subset \mathcal{N}$, while the remaining critical regions are classified as inactive critical regions. Inactive critical regions are not passed on to the next module as workspace guidance, thereby effectively reducing computational burden, especially in environments with many restricted areas.

Through workspace preprocessing, the workspace information is simplified into a reference solution and active critical regions. The reference solution contains connectivity and distance information of the workspace, while the active critical regions contain information about narrow regions of the workspace. Due to the close relationship between the workspace and the C-space, the workspace information can effectively guide planning in the C-space.

C. Workspace-Guided planning

The objective of workspace-guided planning is to utilize critical information from the workspace to provide appropriate guidance for sampling and exploration. The first step in workspace-guided planning is to initialize the planner and establish the required data structures for planning (Alg. 1, line 5∼8). Once the planner is initialized, the planning process alternates between the workspace-guided sampling phase and the workspace-guided expansion phase until the planning termination condition (PTC) is satisfied, such as available planning time or expected path cost.

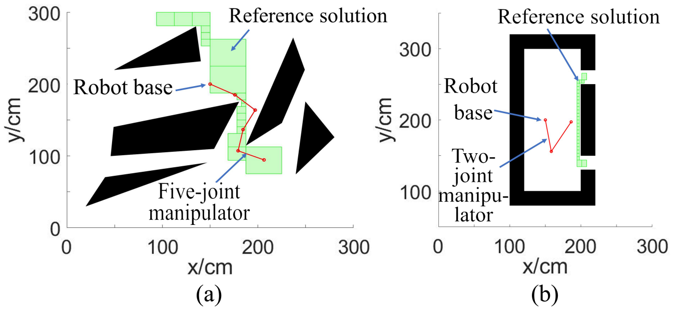

Fig. 4: Workspace reference solution. The green cells represent the workspace reference solution $\mathcal{P}$. (a) The reference solution for a five-joints manipulator. (b) The reference solution for a two-joints manipulator.

1) Workspace-guided sampling: The information from the active critical regions includes the narrowness of the workspace, which can assist in quickly identifying narrow regions in the C-space. To leverage this information and improve the efficiency of sampling, WIGT* employs a workspace-guided sampling strategy, as depicted in Alg. 2.

In the sampling phase, WGIT* first uniformly sample in the informed set $\mathcal{C}_{\text{inf}}$ (Alg. 2 line 3). Any uniform sample q_{uni} satisfying $f_{\text{locate}}(q_{\text{uni}}) \in \mathcal{N}_{\mathcal{P}}$ is likely to be located in the key region in C-space. Such samples are referred to as critical samples. For each critical sample, n_{gau} samples following a Gaussian distribution $\mathrm{N}(q_{\text{uni}}, \sigma)$, denoted as gaussian samples Q_{gau}, are taken (Alg. 2, line 7). The deviation δ is proportional to $\varphi_{\text{size}}(q_{\text{uni}})$ with a proportionality constant of F_{dev} (Alg. 2, line 6).

Workspace-guided sampling increases the density of samples in restricted regions, thereby achieving a balanced distribution of samples. Furthermore, this approach prevents excessive sampling in environments with abundant restricted regions, thereby reducing the computational burden.

Algorithm 2: GuidedSample

```
Input: $\mathcal{N}_{\mathcal{P}}, n$
Output: $Q_{\text{sample}}$
1  $Q_{\text{sample}} \leftarrow \emptyset$;
2  while $|Q_{\text{sample}}| < n$ do
3      $q_{\text{uni}} \leftarrow$ UniformSample($\mathcal{C}_{\text{inf}}$);
4      $Q_{\text{sample}} \overset{+}{\leftarrow} q_{\text{uni}}$;
5      if $f_{\text{locate}}(q_{\text{uni}}) \in \mathcal{N}_{\mathcal{P}}$ then
6          $\delta = F_{\text{dev}} \cdot \varphi_{\text{size}}(f_{\text{locate}}(q_{\text{uni}}))$ ;
7          $Q_{\text{gau}} \leftarrow$ GaussianSample($q_{\text{uni}}, \delta, n_{\text{gau}}$);
8          for $q_{\text{gau}} \in Q_{\text{gau}}$ do
9              if $q_{\text{gau}} \in \mathcal{C}_{\text{inf}}$ then
10                 $Q_{\text{sample}} \overset{+}{\leftarrow} q_{\text{gau}}$;
11 return $Q_{\text{sample}}$
```

2) Workspace-guided exploration: By utilizing information from the reference solution in the workspace and active critical regions, the search tree can be guided efficiently toward directions anticipated to yield high-quality paths, while avoiding redundant expansions.

During the expansion phase, the utilization of workspace information is mainly achieved through heuristic functions. Compared to the commonly used heuristic functions, $f(q) = g(q) + h(q)$ [20], WGIT* incorporates a workspace heuristic term $h_{\mathcal{W}}$ to formulate a workspace-guided heuristic function. The term $h_{\mathcal{W}}$ is expressed as follows:

$$h_{\mathcal{W}}(q) = F_{\text{d}} w_{\text{d}}(q) + F_{\text{s}} w_{\text{s}}(q) + F_{\text{n}} w_{\text{n}}(q) \tag{3}$$

where F_{d}, F_{s}, and F_{n} are constant parameters, while ω_{d}, ω_{s}, and ω_n are cost factors defined as follows:

$$\begin{cases} w_{\text{d}}(q) = \|f_{\text{FK}}(q) - s_{\text{n}}(q)\| / d_{\mathcal{W}} \cdot d_{\mathcal{C}} \\ w_{\text{s}}(q) = \varphi_{\text{size}}(s_{\text{n}}(q))[\varphi_{\text{size}}(s_{\text{n}}(q)) > l_{\text{size}}] / D_{\mathcal{W}} \cdot d_{\mathcal{C}} \\ w_{\text{n}}(q) = [s_{\text{n}}(q) \in \mathcal{N}_{\mathcal{P}}] \cdot d_{\mathcal{C}} \\ s_{\text{n}}(q) = \arg\min_{s \in \mathcal{P}} \{\|f_{\text{FK}}(q) - \varphi_{\text{center}}(s)\|\} \end{cases} \tag{4}$$

where $d_{\mathcal{W}} = \|p_{\text{start}} - p_{\text{goal}}\|$, $d_{\mathcal{C}} = \|q_{\text{start}} - q_{\text{goal}}\|$ and $D_{\mathcal{W}}$ represents the maximum scale of $\mathcal{W}$. The incorporation of $d_{\mathcal{W}}$, $d_{\mathcal{C}}$, and $D_{\mathcal{W}}$ is intended to transform the metrics from the workspace into C-space. The symbol $[\cdot]$ is the Iverson bracket, which takes the value 1 for which the statement is true and takes the value 0 otherwise.

The cost factor ω_{d} depends on the deviation of the reference solution $\mathcal{P}$. This deviation is evaluated through the distance between the reference point $p = f_{\text{FK}}(q)$ and the nearest reference solution cell, represented as $s_{\text{n}}(q)$. Particularly, as this distance increases, the estimated additional cost also increases. With the incorporation of ω_{d}, the search tree in C-space expands preferentially in the regions mapping to the vicinity of the reference solution. Then, in the workspace, the reference point moves along the path close to the reference solution $\mathcal{P}$.

The value of $\omega_{\text{s}}(q)$ depends on the openness of the corresponding region, which is measured by the size of $s_{\text{n}}(q)$. When this size exceeds a specified threshold, denoted as l_{size}, an additional estimated cost is added in proportion to the size. Through the introduction of $\omega_{\text{s}}(q)$, the search tree reduces excessive expansion in open regions.

The value of $\omega_{\text{n}}(q)$ is determined by the proximity to the active critical regions $\mathcal{N}_{\mathcal{P}}$. This proximity is measured by whether $s_{\text{n}}(q)$ falls within the active critical region $\mathcal{N}_{\mathcal{P}}$. An additional estimation cost is imposed if $s_{\text{n}}(q)$ is not in $\mathcal{N}_{\mathcal{P}}$. This measure biases the search tree exploration towards the key regions $\mathcal{R}_{\mathcal{C}}$ to quickly establish connectivity.

Algorithm 3: Explore

```
Input: $Q$, $\mathcal{P}$
Output: $\mathcal{T}$
1  while BestQueueValue($\mathcal{E}_V$) $\leq$ BestQueueValue($\mathcal{E}_E$) do
2      ExpandNextVertex($\mathcal{E}_V, \mathcal{E}_E, c_i$);
3  $(q_{\text{head}}, q_{\text{end}}) \leftarrow$ PopBestInQueue($\mathcal{E}_E$);
4  if $g(q_{\text{head}}) + \hat{c}(q_{\text{head}}, q_{\text{end}}) + h(q_{\text{end}}) + h_{\mathcal{W}}(q_{\text{end}}) < c_i$ then
5      if $g(q_{\text{head}}) + \hat{c}(q_{\text{head}}, q_{\text{end}}) < g(q_{\text{end}})$ then
6          $c_{\text{edge}} \leftarrow c(q_{\text{head}}, q_{\text{end}})$;
7          if $g(q_{\text{head}}) + c_{\text{edge}} + h(q_{\text{end}}) + h_{\mathcal{W}}(q_{\text{end}}) < c_i$ then
8              if $g(q_{\text{head}}) + c_{\text{edge}} < g(q_{\text{end}})$ then
9                  if $q_{\text{end}} \in V$ then
10                     $v_{\text{parent}} \leftarrow$ Parent($q_{\text{end}}$);
11                     $E \xleftarrow{-} (v_{\text{parent}}, q_{\text{end}})$;
12                 else
13                     $Q_{\text{uncon}} \xleftarrow{-} q_{\text{end}}$;
14                     $V \xleftarrow{+} q_{\text{end}}$;
15                     $\mathcal{E}_V \xleftarrow{+} q_{\text{end}}$;
16                     $V_{\text{unexp}} \xleftarrow{+} q_{\text{end}}$;
17                 $E \xleftarrow{+} (q_{\text{head}}, q_{\text{end}})$;
18                 $n_{\text{exp}} \leftarrow n_{\text{exp}} + 1$;
19             if $\|f_{\text{FK}}(q_{\text{end}}) - s_{\text{n}}(q_{\text{end}})\| < r_{\mathcal{P}}(s_{\text{n}}(q_{\text{end}}))$ and
               Indix($s_{\text{n}}(q_{\text{end}})$) $> I_{\text{con}}$ then
20                 $I_{\text{con}} \leftarrow$ Indix($s_{\text{n}}(q_{\text{end}})$);
21                 $n_{\text{exp}} \leftarrow 0$;
22             else if $n_{\text{exp}} > N_{\text{exp}}$ then
23                 $\mathcal{P} \leftarrow$ UpdateReferenceSol($\mathcal{P}, S_{\text{block}}$);
24                 $\mathcal{N}_{\mathcal{P}} \leftarrow \{\eta \in \mathcal{N} \mid \eta \cap \mathcal{P} \neq \emptyset\}$;
25 return $\mathcal{T}$
```

To illustrate the impact of the workspace-guided heuristic function on the exploration phase, an example of planning a two-link manipulator arm shown in Fig. 4(b) is considered, with its C-space depicted in Fig. 5. The values of $h(q)$ for all samples, depicted in Fig. 5(a), are solely dependent on the distance to q_{goal} and then cannot be adjusted based on $\mathcal{C}_{\text{obs}}$. Upon incorporating the workspace guidance term $h_{\mathcal{W}}(q)$, the heuristic guidance for each sample is closely aligned with the shape of $\mathcal{C}_{\text{obs}}$, as demonstrated in Fig. 5(b). The exploration of the search tree under the two heuristics is depicted in Fig. 5(c) and 5(d), respectively. The figures indicate that the search tree with workspace guidance finds a feasible path with fewer explorations.

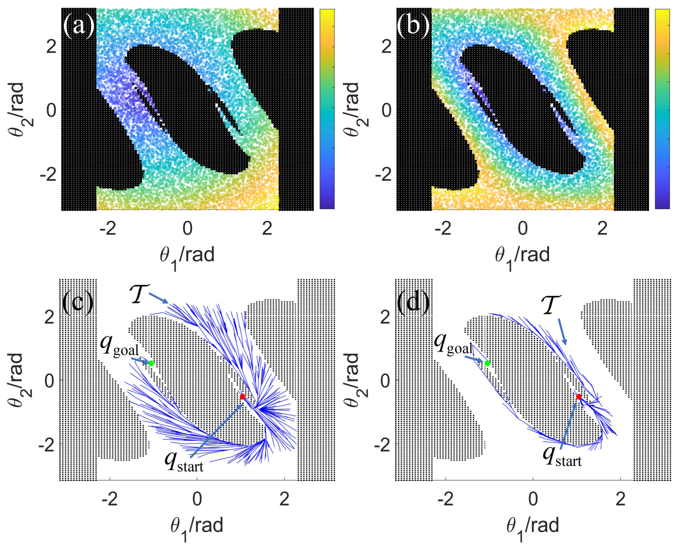

Fig. 5: The heuristics and searching tree in the C-space corresponding to the two-joint manipulator in Fig. 4(b). The first and second columns correspond to the search results of BIT* and WGIT*, respectively. The black area indicates $\mathcal{C}_{\text{obs}}$ and the blue lines indicate the search tree $\mathcal{T}$. (a) The value distribution map of samples using heuristic $h(q)$. (b) The value distribution map of samples using heuristic $h(q) + h_{\mathcal{W}}(q)$. (c) The search tree of BIT*. (d) The search tree of WGIT*.

Thus, the workspace-guided heuristic function involving $h_{\mathcal{W}}$ effectively leverages the available workspace information and optimizes the exploration sequence of the search tree. This approach enables the search tree to pass through the restricted region efficiently, resulting in comprehensive exploration.

In summary, the workspace-guided planning module flexibly transfers the key information from the workspace to the C-space, providing important guidance for planning in complex and restricted C-spaces. As a benefit, WGIT* is able to avoid redundant searches and achieve rapid convergence to high-quality paths.

D. Dual-space Information Updating

The workspace information, while convenient to obtain and process, cannot fully reflect the spatial properties of the C-space due to the lower dimensionality of the workspace compared to the C-space and the lack of consideration for kinematic constraints. To enhance the guidance, continuous updates of both the workspace information and C-space information are required during the planning process. The update of the dual space consists of two main phases: workspace guidance switching and dual-space information balancing.

1) Workspace guidance switching: In cases where the connectivity of the workspace is not aligned with that of the C-space, following the reference solution may result in decreased planning efficiency. To address the decrease from the mismatch of dual-space, a workspace guidance switching strategy is developed, which utilizes feedback from the C-space information during the exploration process to correct the workspace reference solution $\mathcal{P}$, providing proper guidance, as shown in Alg.3, lines 19∼24.

The misaligned connectivity leads to a significant deviation between the workspace search tree $\mathcal{T}_{\mathcal{W}}$ and reference solution $\mathcal{P}$. To measure this deviation and detect inappropriate guidance, each cell in $\mathcal{P}$ is assigned a neighboring region, which is a spherical region with a center of $\varphi_{\text{center}}(s_{\mathcal{P},i})$ and a radius of $r_{\mathcal{P}}(s_{\mathcal{P},i})$. To improve measurement accuracy, $r_{\mathcal{P}}(s_{\mathcal{P},i})$ is appropriately increased when $\varphi_{\text{size}}(s_{\mathcal{P},i}) > l_{\text{size}}$ and is appropriately reduced when $s_{\mathcal{P},i} \in \mathcal{N}_{\mathcal{P}}$. The connectivity index I_{con} records the expansion of $\mathcal{T}_{\mathcal{W}}$. When $\mathcal{T}_{\mathcal{W}}$ expands to an unvisited neighboring region, I_{con} is updated to the index of the corresponding cell, i.e., its order in $\mathcal{P}$. If I_{con} remains unchanged after N_{exp} expansions, The corresponding region in the C-space for $\mathcal{P}$ is likely discontinuous and cannot provide adequate guidance for sampling and exploration. In such cases, a reference solution switching strategy is employed to update the reference solution, as shown in Alg. 4.

The procedure for updating the reference solution is illustrated in Fig. 6. The cells along the initial reference solution are inflated until their boundaries come into contact with $\mathcal{W}_{\text{obs}}$ (Alg. 4, line 3), as shown in Fig. 6(2). The resulting inflated region is referred to as $\mathcal{S}_{\text{block}}$, whose bounds are achieved through binary search. To obtain a new reference solution, the A* algorithm is re-invoked, with the cells passing through $\mathcal{S}_{\text{block}}$ incurring additional costs. Ultimately, a new reference solution for the workspace is identified, and then the active critical regions $\mathcal{N}_{\mathcal{P}}$ are updated accordingly.

2) Dual-Space Information balancing: Since planning is directly conducted in C-space, C-space information is more accurate and complete than workspace information. Moreover, the workspace-guided heuristic function is inadmissible, which may affect the convergence toward the optimal path. Therefore, a proper balance between workspace and C-space information is necessary to ensure the efficiency of the planning process and the quality of the solution.

WGIT* balances the information between the workspace and C-space by gradually reducing the weight of the workspace-guided information. The gradual detachment of

Algorithm 4: UpdateReferenceSol

```
  Input: $\mathcal{P}, \mathcal{S}_{\text{block}}$
  Output: $\mathcal{P}$
1 for $s \in \mathcal{P}$ do
2     if $s \notin \mathcal{S}_{\text{block}}$ then
3         $s_{\text{exp}}$ = ExpandCell($s$);
4         $\mathcal{S}_{\text{block}} \xleftarrow{+} s_{\text{exp}}$;
5 $\mathcal{P} \leftarrow$ AstarPlan($p_{\text{start}}, p_{\text{goal}}, \mathcal{S}, \mathcal{S}_{\text{block}}$);
6 return $\mathcal{P}$
```

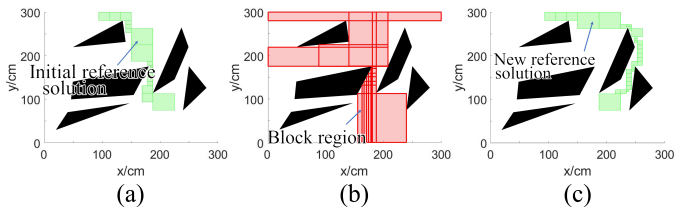

Fig. 6: Workspace guidance switching. (a) Initial workspace reference solution. (b) Expanded cells along the initial reference solution. (c) New workspace reference solution.

workspace information is achieved through the incorporation of a balancing factor into the workspace-guided heuristic function. The balancing factor is defined as $e^{-\alpha i}$, where $\alpha \in R^+$ is a constant and i represents the iteration number of WGIT*. Ultimately, the heuristic function of the algorithm is formulated as:

$$f_{\mathcal{W}}(q) = g(q) + h(q) + e^{-\alpha i} h_{\mathcal{W}}(q). \tag{5}$$

As the algorithm iterates, the balance factor decreases, thereby diminishing the workspace guidance. When the balance factor eventually reaches zero, the workspace-guided heuristic function $f_{\mathcal{W}}(q)$ reverts back to its original form $f(q)$. This approach gradually transforms the heuristics into admissible to enhance the efficiency of searching for high-quality solutions.

Dual-space information updating module enables bidirectional information exchange instead of unidirectional information flow from the workspace to the C-space. As a result, the planner can fully utilize the information from both spaces in different planning stages, reducing misleading guidance and improving planning efficiency. More importantly, it can guarantee convergence to the optimal path.

E. Theoretical Analysis of WGIT*

In this section, the properties of the WGIT* are analyzed theoretically as follows.

Theorem 1. (Probabilistic completeness) *For a given planning problem (q_{start}, q_{goal}), if a feasible solution exists, the probability of WGIT* finding a feasible solution is unity as the number of samples approaches to infinity,*

$$P\left(\lim_{N\to\infty} \sigma_{\text{WGIT}^*} \in \Sigma, \sigma(0) = q_{\text{start}}, \sigma(1) = q_{\text{goal}}\right) = 1 \tag{6}$$

where N is the number of samples, Σ is the set of all feasible solutions.

Proof: The proof of Theorem 1 can be derived from the proof of Theorem 2, which concerns almost-sure asymptotic optimality of WGIT*. ■

Theorem 2. (Almost-surely asymptotic optimality) *For a given planning problem (q_{start}, q_{goal}), if an optimal solution σ^* exists, the probability of WGIT* asymptotically converging to the optimal solution is unity as the number of samples approaches to infinity,*

$$P\left(\lim_{N\to\infty} \mathcal{L}_{\text{c}}(\sigma_{\text{WGIT}^*}) = \mathcal{L}_{\text{c}}\left(\sigma^*\right)\right) = 1 \tag{7}$$

Proof: In [21], a sufficient but not necessary condition is presented for SBPs to guarantee almost-surely asymptotically optimal. The condition consists of two parts: (1) The approximation almost-surely contains an asymptotically optimal solution. (2) The graph-search algorithm is resolution-optimal.

Firstly, for each batch of samples, WGIT* consider edges $\mathcal{E}_E$ consist of samples with their neighbors within a radius r^* that scales as in PRM* [23] to construct an implicit random geometric graph (RGG). The samples are connected in ascending order according to eq. (5). This process only alters the order of connection while preserving the structure of the implicit RRG. Therefore, WGIT* implicitly encompasses all edges of PRM*. As the approximation of PRM* almost-surely contains an asymptotically optimal solution [23], the approximation of WGIT* guarantees to contain the same solution. Hence, WGIT* satisfies the part (1) of the condition.

Secondly, when the number of samples approaches infinity, the iteration of WGIT* tends towards infinity since each batch of samples is finite. Consequently, as the balance factor $e^{-\alpha i}$ approaches 0, the heuristic function tends to $f_{\mathcal{W}}(q) = g(q) + h(q)$ according to eq. (5). Hence, the graph search process of WGIT* ultimately converges to that of A*, which is resolution optimal [20]. Consequently, WGIT* satisfies part (2) of the condition as the number of samples approaches infinity.

In conclusion, WGIT* satisfies both part of the conditions simultaneously, thereby establishing its almost-surely asymptotic optimality. Furthermore, since probability completeness is a necessary but not sufficient condition for asymptotic optimality, Theorem 1 is proven. ■

IV. Simulations and Experiments

In this section, simulations and experiments are designed and performed to verify the effectiveness of WGIT*.

A. Simulation Setup

To verify the effectiveness of WGIT*, a series of simulations were performed. The algorithms are coded and simulated based on MATLAB R2021b and CoppeliaSim Edu V4.4.0 on a computer running Microsoft Windows 10 with Intel Core i5-7300HQ and 16 GB of RAM. For the simulations in $\mathbb{R}^2$ workspace, a three-joint and a five-joint manipulator are utilized, while for the simulations in $\mathbb{R}^3$ workspace, a UR5 robot with six degrees of freedom is employed. The simulations compare four algorithms: WGIT*, BIT* [13], GuILD [24], TS-RRT [15], and EET [17]. BIT* solely utilizes C-space information for collision checking. GuILD reduces the planning space to improves efficiency in restricted environments by introducing local informed subsets. TS-RRT plans directly in the workspace via inverse kinematics. Similar to TS-RRT, EET conducts searches directly in the workspace. However, EET features a workspace preprocessing. The comparison of the five algorithms aims to investigate the effectiveness of combining C-space and workspace information in WGIT*.

The critical parameters of WGIT* are set as follows: $F_{\rm d} = 5.0$, $F_{\rm s} = 7.0$, $F_{\rm n} = 0.1$, $\alpha = 1$. The values of these parameters are selected through trial and error in various scenarios. $F_{\rm d}$ is the weight of the reference solution deviation factor $\omega_{\rm d}$. An increase in $F_{\rm d}$ will align the planned solution more with the reference solution. However, it may excessively rely on the reference path and increase the number of workspace guidance switching. F_s is the weight of the open area factor $\omega_{\rm s}$. While a larger F_s guides the search tree to swiftly pass through open areas and focus the search more on narrow regions, an excessively large F_s might lead the search to avoid necessary open areas. $F_{\rm n}$ is the weight of the critical areas proximity factor ω_n. A larger $F_{\rm n}$ can concentrate the search on active critical areas, while a smaller value can provide higher efficiency when updates are required in active critical areas. α is the decay factor of the workspace heuristic function, used to balance dual-space information during the planning process. A large α causes the workspace heuristic to vanish rapidly, reducing the guidance from workspace. Conversely, a too small α slows down the convergence towards the optimal solution.

The key comparison metrics include the time to find the initial solution $t_{\rm init}$, the cost of the initial solution $c_{\rm init}$, the number of collision checks performed when finding the initial solution $n_{\rm check}$, the cost of the solution when the planning process is complete $c_{\rm final}$, the success rate of the planning $\phi_{\rm s}$ and the CPU load during algorithm execution $\phi_{\rm CPU}$. Each simulation runs 100 times, and the average value of each metric is taken, excluding failed planning results. The available planning time is 20 seconds. For WGIT* and BIT*, the number of samples per batch is 500.

B. Sumulation Verification & Analysis

Two simulation scenarios $\mathcal{W} \in \mathbb{R}^2$ are constructed to investigate the planning performance of WGIT* in planer robots. In scenario 1, two planning tasks are performed by a three-joint manipulator and a five-joint manipulator, as depicted in Fig. 7(a) and 7(c), respectively. The manipulators are required to navigate through a restricted area and place the end effector within another restricted area. Scenario 2 involves two planning tasks carried out by a three-joint manipulator and a five-joint manipulator, as depicted in Fig. 7(e) and 7(g), respectively. The manipulators need to navigate through cluttered obstacles and pass through a challenging narrow passage. Upon reaching the target pose, a significant portion of the robotic arm is located within the passage, with very little clearance from the obstacles. It is noteworthy that the initial reference solution in scenario 2 is not entirely within the reachable areas of the manipulator, as the red path illustrated in Fig 7(f). Therefore, a workspace guidance switching needs to be executed to change the path from the red one to the green one in Fig 7(f). The planning results of the three-joint manipulator and five-joint manipulator in simulation scenario 1 and scenario 2 are shown in Table I, and the search trees in the workspace are illustrated in Fig 7.

In Scenario 1, WGIT* achieves initial solution search speeds 23.70% and 52.50% faster than BIT*, with initial solution costs of 94.38% and 89.63% of the counterpart from BIT*, in $\mathcal{C} \in \mathbb{R}^3$ and $\mathcal{C} \in \mathbb{R}^5$, respectively. Besides, In the two tasks, WGIT* finds initial solution 17.20% and 31.33% faster than

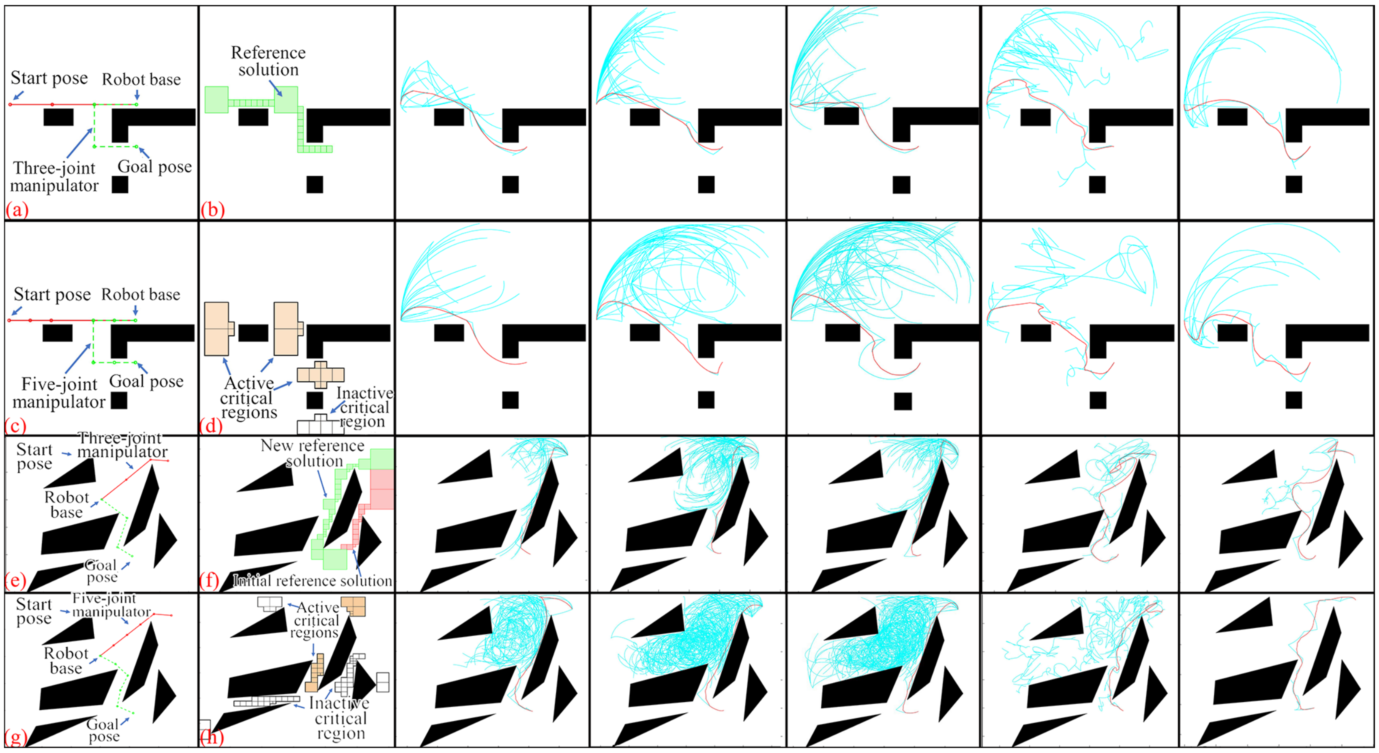

Fig. 7: Comparative simulation results in scenarios 1 and 2. The first column represents four planning tasks, namely three-joint manipulator in scenarios 1, five-joint manipulator in scenarios 1, three-joint manipulator in scenarios 2, and five-joint manipulator in scenarios 2, respectively. The red and green colors in the first column represent the initial and target configurations of the manipulator, respectively. The second column shows the distribution of critical regions and workspace reference solutions by proposed method, respectively. The planning results of WGIT*, BIT*, GuILD, TS-RRT, and EET are presented in columns 3-7, respectively, where the blue lines represent the end-effector trajectories corresponding to the search tree branches, while the red lines represent the end-effector trajectories corresponding to the initial paths.

GuILD, while the initial solution costs of WGIT* are 96.59% and 94.22% of the counterpart from GuILD, respectively. The workspace searching tree illustrated in Fig. 7 and the number of edge collision checks indicate that WGIT* performs fewer explorations than BIT* and GuILD under the guidance of workspace information. As a benefit, WGIT* finds initial solutions faster. Moreover, when the planning time is reached, WGIT*, BIT* and GuILD converge to paths with almost the same cost, indicating that WGIT*, BIT* and GuILD have similar abilities to quickly converge to high-quality paths (with a difference of no more than 3%). In $\mathcal{C} \in \mathbb{R}^3$, both TS-RRT and EET take longer to find initial solutions than WGIT, BIT* and GuILD, but perform fewer edge collision checks. This is mainly because the algorithms that directly plan in the workspace have difficulty finding collision-free inverse solutions in restricted environments. Consequently, even though inverse kinematic sampling can reduce exploration, the planning remains time-consuming. The challenges associated with inverse kinematics sampling and the issue of getting trapped in particular configurations contribute to lower success rates of TS-RRT and EET. In $\mathcal{C} \in \mathbb{R}^5$, EET has the second fastest initial solution search speed after WGIT*, primarily due to workspace pre-processing that reduces redundant explorations.

In Scenario 2, WGIT* outperforms the other four competitors in terms of both initial solution search time and initial solution cost. Furthermore, the final converged solutions are comparable to BIT* and GuILD. However, TS-RRT and EET have significantly lower success rates in this environment. This is mainly due to the large number of obstacles and narrow space, making inverse kinematics sampling more difficult than in Scenario 1. It is worth noting that for EET, the decrease in initial solution search speed and success rate is significant. This is mainly due to the inconsistency between the constructed workspace guidance and the connectivity of the C-space, resulting in inefficient workspace sampling. However, WGIT* is not significantly affected due to the presence of workspace path switching.

To investigate the efficacy of WGIT* in $\mathcal{W} \in \mathbb{R}^3$, simulation scenario 3 is developed, as depicted in Fig. 8. In simulation scenario 3, a six-joint UR5 robot needs to navigate between two adjacent shelves while avoiding collisions. The planning results are shown in Table I.

It can be observed from Table I that WGIT* requires 83.76%, 87.16%, 35.05%, and 45.91% of the initial solution time of BIT*, GuILD, TS-RRT, and EET, respectively. The cost of the initial path generated by WGIT* is approximately 6.11% and 11.29% lower than that of BIT* and GuILD, respectively. As non-optimal algorithms, TS-RRT and EET produce significantly higher initial solution costs than WGIT*.

A series of simulation results for robots with different de-

TABLE I: Statistical results in simulations

Scenario	Algorithm	t_{init} (s)	c_{init} (rad)	n_{check}	c_{final} (rad)	ϕ_{s} (%)	ϕ_{CPU} (%)
Scenario 1 $n_{\text{joint}} = 3$	WGIT*	1.01	4.53	248.50	3.78	100	33.80
	BIT*	1.25	4.80	286.14	3.79	100	36.23
	GuILD	1.22	4.69	245.20	3.82	100	32.34
	TS-RRT	2.03	16.10	231.05	16.10	92	27.38
	EET	1.63	9.89	149.68	9.89	94	28.36
Scenario 1 $n_{\text{joint}} = 5$	WGIT*	1.60	7.01	263.00	4.69	100	33.53
	BIT*	2.44	7.81	405.73	4.63	100	35.41
	GuILD	2.33	7.44	255.11	4.58	100	33.95
	TS-RRT	3.08	29.65	266.44	29.65	74	25.13
	EET	1.98	22.70	165.00	22.70	84	25.44
Scenario 2 $n_{\text{joint}} = 3$	WGIT*	1.20	6.02	256.79	5.58	100	30.56
	BIT*	1.33	6.13	302.00	5.779	100	32.41
	GuILD	1.34	6.18	273.00	5.56	100	31.31
	TS-RRT	2.66	11.78	221.48	11.78	86	28.48
	EET	4.73	14.80	538.95	14.80	67	26.62
Scenario 2 $n_{\text{joint}} = 5$	WGIT*	3.96	9.92	352.81	6.95	100	31.68
	BIT*	4.75	10.33	561.62	7.67	100	32.68
	GuILD	4.54	10.29	498.40	7.17	100	30.82
	TS-RRT	9.64	24.51	406.61	27.89	70	27.95
	EET	6.77	21.29	829.41	21.29	54	26.83
Scenario 3 $n_{\text{joint}} = 6$	WGIT*	1.29	3.38	449.10	2.46	100	42.12
	BIT*	1.54	3.60	541.20	2.56	100	47.49
	GuILD	1.48	3.81	489.10	2.38	100	42.17
	TS-RRT	3.68	6.69	286.54	6.69	92	44.10
	EET	2.81	5.28	100.30	5.28	96	43.22

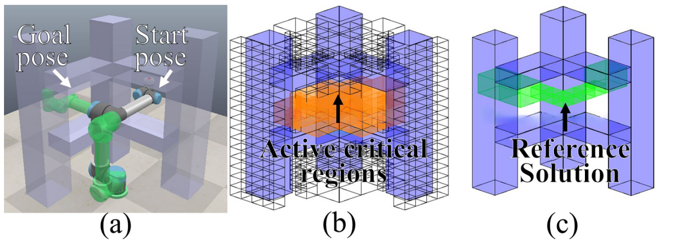

Fig. 8: Simulation scenario 3: a six-joint UR5 manipulator operating within a 3-dimensional workspace. (a) Start configuration and goal configuration. (b) The active critical regions $\mathcal{N}_{\mathcal{P}}$, depicted in orange cells. (c) The reference solution $\mathcal{P}$, depicted in green cells.

grees of freedom in $\mathcal{W} \subset \mathbb{R}^2$ and $\mathcal{W} \subset \mathbb{R}^3$ show that WGIT* is efficient at finding the initial solution under workspace guidance. Moreover, WGIT* quickly converges to high-quality paths without introducing excessive computational burden. These results highlight the effectiveness and universality of WGIT* in planning within restricted environments.

C. Experiment Verification & Analysis

To further validate the effectiveness of WGIT*, experiments are designed using the Fetch robot with a seven-degree-of-freedom arm to perform grasping and placing tasks in two restricted environments, as shown in Fig. 9. The robot is required to pick up the target object from the restricted area and place it at the location in another restricted area while avoiding obstacles and self-collision. In order to obtain high-quality paths, BIT* and WGIT* are utilized to plan motions for the Fetch. As optimizing planners, both BIT* and WGIT* algorithms minimize unnecessary motion, allowing the Fetch robot to efficiently complete tasks. Each algorithm runs 20 times, and the key parameters are consistent with the simulation. Taking into account a comprehensive consideration of safety and efficiency, the operational speed is set at 60% of the Fetch robot's velocity limit. The specific values for joints 1 to 7 (unit: rad/s) are listed as follows: 0.75, 0.87, 0.94, 0.91, 0.94, 1.35, 1.35. The experimental results are shown in Table II.

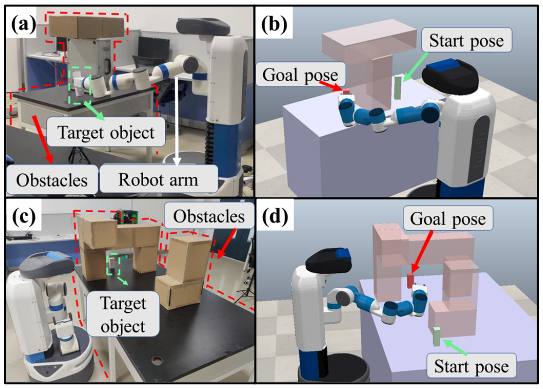

Fig. 9: Experiment scenarios. (a) Experimental scenario 1. (b) Simulation scenario for (a). (c) Real-world experimental scenario 2. (d) Simulation scenario for (c).

TABLE II: Statistical results in experiments

Scenario	Algorithm	t_{init}(s) (s)	c_{init} (rad)	n_{check} n_{check}	c_{final} (rad)	ϕ_{s}(%) (%)	ϕ_{CPU} (%)
Scenario 1 $n_{\text{joint}} = 7$	WGIT*	7.47	5.68	325.40	4.55	100	45.12
	BIT*	9.05	5.89	559.00	4.59	100	43.22
Scenario 2 $n_{\text{joint}} = 7$	WGIT*	3.15	3.79	498.10	2.51	100	49.83
	BIT*	4.72	3.85	639.65	2.71	100	47.20

Table II demonstrates that WGIT* can find the initial solution 21.15% and 33.26% more rapidly than BIT* in experimental scenario 1 and 2, respectively, while achieving superior quality initial solutions. From the perspective of robotic arm motion, this result is supported by the observation that the robot employing WGIT* is capable of executing grasping and placing tasks with simpler movements, as illustrated in Fig. 10. Furthermore, during the process of obtaining the initial solution, WGIT* performs fewer edge collision checks, reducing collision checks by 41.86% and 22.13% respectively, resulting in a reduction of redundant explorations and mitigation of the computational burden. Upon exhausting the planning time, the solutions obtained by WGIT* have costs that are respectively 0.87% and 7.38% lower than those obtained by BIT*. The similarity of their final solutions suggests that both WGIT* and BIT* can rapidly converge to high-quality paths. The conformity between the experimental and simulation results validates that the proposed WGIT* algorithm effectively enhances the performance of robots in realistic restricted environments.

Combining a series of simulations and experiments, it is demonstrated that WGIT* can reduce redundant sampling and exploration under the guidance of workspace information, thereby finding feasible solutions rapidly. Furthermore, WGIT* can rapidly converge to a high-quality solution. In summary, WGIT* can effectively assist robots in completing tasks within restricted environments.

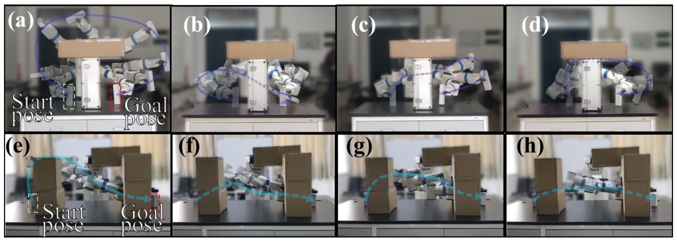

Fig. 10: Grasping and placing task path planning. Columns 1-4 are initial solution from BIT*, initial solution from WGIT*, final solution from BIT*, and Final solution from WGIT*, respectively.

V. Conclusion

This paper has proposed a robot motion planning algorithm for restricted environments, namely the workspace-guided informed tree (WGIT*). WGIT* refines key information in the workspace, such as workspace reference solutions and critical regions, to guide the sampling and exploration processes. Active critical regions help focus the sampling process on the restricted areas in C-space, resulting in a more balanced distribution of samples. Guidance from the workspace reference solution enables the search tree to explore in promising directions, reducing redundant expansions and improving the efficiency of finding feasible solutions. Furthermore, WGIT* incorporates a balanced strategy between workspace and C-space information to guarantee high-quality solutions. Various challenging simulations and experiments have demonstrated the satisfactory performance of WGIT* in restricted environments.

In future research, a wider range of workspace information decomposition methods and heuristic functions will be integrated into WGIT* to address challenges in different environments and tasks. Moreover, WGIT* will be applied to a broader range of robotic applications.

References

[1] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378-400, May 2001.

[2] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566-580, Aug. 1996.

[3] R. Wang, X. Zhang, Y. Fang, and B. Li, "Virtual-goal-guided RRT for visual servoing of mobile robots with FOV constraint," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 4, pp. 2073-2083, Apr. 2022.

[4] Z. Wang, Y. Li, H. Zhang, C. Liu, and Q. Chen, "Sampling-based optimal motion planning with smart exploration and exploitation," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 5, pp. 2376-2386, Oct. 2020.

[5] O. Salzman, "Sampling-based robot motion planning," *Commun. ACM*, vol. 62, no. 10, pp. 54-63, Sep. 2019.

[6] Y. Chen, Z. Zhang, Z. Wu, Z. Miao, H. Zhang, and Y. Wang, "SET: Sampling-enhanced exploration tree for mobile robot in restricted environments," *IEEE Trans. Ind. Inform.*, pp. 1-11, 2023.

[7] Z. Wu, Y. Chen, J. Liang, B. He, and Y. Wang, "ST-FMT*: A Fast optimal global motion planning for mobile Robot," *IEEE Trans. Ind. Electron.*, pp. 3854-3864, May 2021.

[8] Y. Chen et al., "Safety-enhanced motion planning for flexible surgical manipulator using neural dynamics," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 5, pp. 1711-1723, Sep. 2017.

[9] L. Jiang, S. Liu, Y. Cui, and H. Jiang, "Path planning for robotic manipulator in complex multi-obstacle environment based on improved_RRT," *IEEE/ASME Trans. Mechatronics*, pp. 1–12, Dec. 2022.

[10] A. Tahirovic and M. Ferizbegovic, "Rapidly-exploring random vines (R-RV) for motion planning in configuration spaces with narrow passages," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2018, pp. 7055-7062.

[11] X. Shu, F. Ni, Z. Zhou, Y. Liu, H. Liu, and T. Zou, "Locally guided multiple bi-RRT* for fast path planning in narrow passages," in *Proc. IEEE Int. Conf. Robot. Biomim.*, Dec. 2019, pp. 2085-2091.

[12] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 966-984, Aug. 2018.

[13] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (BIT*): Informed asymptotically optimal anytime search," *Int. J. Robot. Res.*, vol. 39, no. 5, pp. 543-567, Apr. 2020.

[14] Z. Meng, H. Qin, H. Sun, X. Shen, and M. H. Ang, "Obstacle-guided informed planning towards robot navigation in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Biomim.*, Dec. 2017, pp. 332-337.

[15] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space Voronoi bias," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 2061-2067.

[16] G. Mesesan, M. A. Roa, E. Icer, and M. Althoff, "Hierarchical path planner using workspace decomposition and parallel task-space RRTs," in *Proc. Int. Conf. Intell. Robot. Syst.*, 2018, pp. 1-9.

[17] M. Rickert, A. Sieverling, and O. Brock, "Balancing exploration and exploitation in sampling-based motion planning," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1305-1317, 2014.

[18] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robot.*, vol. 34, no. 3, pp. 189-206, Apr. 2013.

[19] J. P. van den Berg and M. H. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," *Int. J. Robot. Res.*, vol. 24, no. 12, pp. 1055-1071, Dec. 2005.

[20] P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Syst. Cybern.*, vol. 4, no. 2, pp. 100-107, Jul. 1968.

[21] M. P. Strub and J. D. Gammell, "Adaptively informed trees (AIT*) and effort informed trees (EIT*): Asymmetric bidirectional sampling-based path planning," *Int. J. Robot. Res.*, vol. 41, no. 4, pp. 390-417, Apr. 2022.

[22] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *Int. J. Robot. Res.*, vol. 34, no. 7, pp. 883-921, June. 2015.

[23] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846-894, Jun. 2011.

[24] R. Scalise, A. Mandalika, B. Hou, S. Choudhury, and S. S. Srinivasa, GuILD: Guided incremental local densification for accelerated sampling-based motion planning, in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2023, pp. 10212-10218.

Zhixing Zhang received the B.S. degree in mechanical design manufacture and automation from Guangdong University of Technology, Guangzhou, China, in 2021. He is currently working toward the M.S. degree in mechatronic engineering from Fuzhou University, Fuzhou, China. His research interests include motion planning and mobile robot.

Yanjie Chen received the B.S. degree in electrical engineering and its automation from Southwest Jiaotong University, Chengdu, China in 2011, the M.S. and Ph.D. degrees in control science and engineering from Hunan University, Changsha, China, in 2013 and 2017, respectively. He was awarded the Newton International Fellowships 2022 by the Royal Society, UK. He is currently an Associate Professor with School of Mechanical Engineering and Automation, Fuzhou University, Fuzhou, China. He is also a Royal Society Newton International Fellow with Department of Computer Science, Aberystwyth University, Aberystwyth, UK, and an Scientist with the National Engineering Research Center of Robot Visual Perception and Control Technology, Changsha, China. His research interests include robotics, aerial manipulator, motion planning and artificial intelligence.

Feng Han received the B.S. degree in mechanical design manufacture and automation from Fuzhou University, Fuzhou, China, in 2022, where he is currently working toward the M.S. degree in mechatronic engineering from Fuzhou University, Fuzhou, China. His research interests include motion planning.

Junwei Fan received the B.S.degree in mechanical design manufacture and automation from Fuzhou University, Fuzhou, China, in 2022 where he is currently working toward the M.S. degree in mechanical engineering from Fuzhou University, Fuzhou, China. His research interests include motion planning.

Hongshan Yu received the B.S., M.S., and Ph.D. degrees in control science and technology in electrical and information engineering from Hunan University, Changsha, China, in 2001, 2004, and 2007, respectively. From 2011 to 2012, he was a Post-Doctoral Researcher with the Laboratory for Computational Neuroscience, University of Pittsburgh, USA. He is currently a Professor with Hunan University and the Associate Dean of the National Engineering Laboratory for Robot Visual Perception and Control. His research interests include autonomous mobile robot and machine vision.

Hui Zhang received the B.S., M.S., and Ph.D. degrees in pattern recognition and intelligent system from Hunan University, Changsha, China, in 2004, 2007, and 2012, respectively. He is currently a Professor with the School of Robotics and the National Engineering Research Center of Robot Visual Perception and Control Technology, Hunan University. He was a Visiting Scholar with Common Vulnerability Scoring System Laboratory, Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada, in 2017. His research interests include machine vision, sparse representation.

Yaonan Wang received the B.S. degree in computer engineering from East China University of Science and Technology, Fuzhou, China, in 1981 and the M.S. and Ph.D. degrees in control engineering from Hunan University, Changsha, China, in 1990 and 1994, respectively. He was a Post-Doctoral Research Fellow with the National University of Defense Technology, Changsha, from 1994 to 1995, a Senior Humboldt Fellow in Germany from 1998 to 2000, and a Visiting Professor with the University of Bremen, Bremen, Germany, from 2001 to 2004. He has been a Professor with Hunan University since 1995. He has been an academician of China Engineering Academy since 2019. His research interests include intelligent control, information processing, and image processing.