IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.
Cite as T-ASE paper.

RoA-Planner: Rotatable Area-Based Path Planner in Dense Spaces

Yeongwoo Son, Hyunyong Lee, Hansol Kang, Jiman Park, Seongwon Nam, Jaeyoung Oh, Bumsu Yi, Junha Song, Sooyeon Choi, Bogeun Kim, Daegeun Song, and Hyouk Ryeol Choi, *Fellow, IEEE*

***Abstract*—Path planning in obstacle-dense environments is a challenging problem, particularly for robots with asymmetric rectangular footprints. To address this problem, we propose a novel collision-checking approach, called a Rotatable Area, which represents a range of heading angles where the robot can rotate without colliding with obstacles. Based on the relationship between two rotatable areas, we define safe local motion and extend this concept to the RoA-Planner, a path planning framework in SE(2) dense space. We validate our planner through extensive simulations and real-world experiments in complex and narrow environments. The results demonstrate that our method achieves fast planning speed while ensuring safety and robustness, making it suitable for practical applications.**

***Note to Practitioners*—Navigating complex environments is critical for applications, such as warehouse automation, facility management, or urban patrol. Existing path planning methods either suffer from low collision-checking accuracy in complex environments or require a long computation time to ensure fidelity. To tackle this problem, this paper introduces the Rotatable Area, a collision-checking method tailored for asymmetric rectangular robots. By checking local motion between rotatable areas, a collision-free path can be planned in a fast and precise manner. This approach is applicable to any shape of rectangular robots with holonomic movement and particularly effective in obstacle dense and narrow environments. The method was validated in scenarios derived from industrial fields and service domains, demonstrating efficiency and safety in real-time.**

***Index Terms*—Mobile robot, rectangular robot, holonomic robot, path planning, collision avoidance, dense environment.**

I. Introduction

ADVANCEMENTS in robotics navigation technologies are driving the growing integration of mobile robots into various environments around us. Initially confined to industrial sites, these robots are now expanding into human living spaces [1], [2]. Accordingly, Mobile robots are transitioning from being mere tools to performing numerous tasks that could replace human effort. However, despite these remarkable advancements, path planning in dense environments remains a significant challenge, especially for asymmetric rectangular robots. The difficulty arises not only from the challenge of finding a path itself, but also from the inability to guarantee safety even when a path is found.

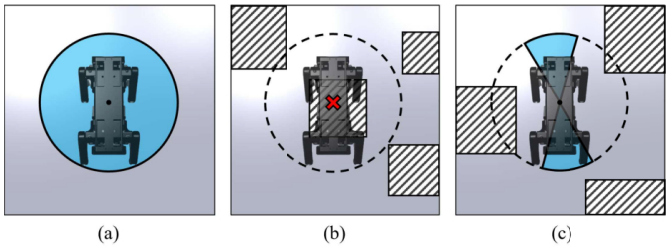


Fig. 1. Three types of rotatable area. (a) Fully-RoA, R_F, (b) Non-RoA, R_N, (c) Partially-RoA, $R_{P,i}$.

In fact, path planning fundamentally involves connecting the start region to the goal region through a sequence of local motions, while ensuring collision avoidance at every stage. However, in dense environments, collision-checking becomes far more demanding. Unlike open field where checking the start and goal poses may suffice, dense environments require continuous verification along the entire trajectory due to the geometric constraints imposed by the robot's shape. This makes path planning time-consuming and highly sensitive to small errors, especially when dealing with rectangular robots whose orientation plays a crucial role in collision avoidance.

A. Related Work

1) Geometric-Based Method: Inflating obstacles in the map to the size of the robot is one of the simplest methodologies for collision-checking, because the robot can be represented as a point [3], [4], [5]. In particular, the method of inflating obstacles with a circumcircle is widely used because it allows the robot's heading angle to be taken as the tangential direction of the generated path. However, these methods tend to overly approximate the robot's collision model, which results in conservative path generation in dense and cluttered environments.

Some works focus on creating a safe corridor that guarantees no collisions with obstacles and optimize waypoints to be located within this corridor [6], [7], [8]. Another common

Received 3 December 2024; revised 8 April 2025 and 20 July 2025; accepted 25 August 2025. Date of publication 22 September 2025; date of current version 14 November 2025.This article was recommended for publication by Associate Editor Z. Liu and Editor J. Yi upon evaluation of the reviewers' comments. This work was supported by the Materials and Parts Technology Development Program (Development and Demonstration of Unmanned Autonomous Operation Technology Based on Field-Use Visualization Sensors and 6-Axis Rotational Angle Sensors) funded by the Ministry of Trade Industry and Energy (MOTIE), South Korea, under Grant RS-2024-00508191. *(Corresponding author: Hyouk Ryeol Choi.)*

Yeongwoo Son, Hansol Kang, Jiman Park, Seongwon Nam, Jaeyoung Oh, Bumsu Yi, Junha Song, Sooyeon Choi, Bogeun Kim, Daegeun Song, and Hyouk Ryeol Choi are with the School of Mechanical Engineering, Sungkyunkwan University, Suwon 16419, South Korea (e-mail: ywson96@gmail.com; choihyoukryeol@gmail.com).

Hyunyong Lee is with AIDIN ROBOTICS Inc., Anyang 14055, South Korea (e-mail: leevsv@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TASE.2025.3612949, provided by the authors.

Digital Object Identifier 10.1109/TASE.2025.3612949

approach involves optimizing the control points of the B-spline curve [9], [10], [11], whose convex hull property ensures the trajectory remains within a safe region. In addition, [12] and [13] plan safe trajectories by combining safe corridors with Bézier curves, which have convex hull property similar to B-spline curves. Neverthelesss, since these methods generally model the robot as a point during trajectory generation, map inflation using the circumcircle is necessary to accommodate rectangular-shaped robots.

To avoid the limitation of the circumcircle in dense spaces, some studies directly employ the expanded rectangular collision model [14], [15], while others utilize collision spheres and signed distance fields (SDF) to approximate rectangular robots [16], [17], showing good performance in collision detection. All of these approaches are built upon sampling-based planners. Notably, the method in [15] enhances the sampling probability in narrow passages through adaptive sampling distribution, providing advantages in path planning within cluttered environments. However, they do not detect continuous collisions but instead rely on discrete collision checks along the local motions. Moreover, these methods suffer from the curse of a large search space, which not only increases the computational burden of path planning but also significantly reduces the number of valid samples in dense environments as the map size grows, often resulting in suboptimal detouring paths.

Due to this inefficiency, the incircle-based method has also been widely adopted in dense environments [18]. TRG-Planner [19] efficiently plans paths by leveraging both the height information within the incircle of each node and an ellipsoid that approximates local motion. However, the incircle method has the disadvantage of not guaranteeing the safety of the generated path, as it uses a collision model that is smaller than the actual robot size. Therefore, additional optimization processes or local path planning are required. In the works of [20], [21], and [22], the initial path is optimized, utilizing the distance between the rectangular footprint and obstacles. Similarly, method attaching collision barrier functions also uses the distance information for optimization [23], [24], [25]. But, these approaches are prone to getting stuck in local minima, highlighting the importance of the initial path due to the convergence.

2) Learning-Based Method: In most deep learning-based studies, path planning relies on a learned model that predicts the feasibility of local motion [26], [27]. A mobile robot collects data—such as success or failure, energy consumption, time, and surrounding environment information—while navigating to a local target pose in 2D [28] or SE(2) [29]. The learned model predicts the success probability of a local motion, and the planner utilizes it as a cost for optimization or the form of traversability map [30], [31]. In addition, the studies in [32], [33], and [34] focus on environments with dense obstacles. In particular, Art-Planner [34] improves collision detection performance in cluttered environments by evaluating valid vertices using a rectangular collision model and assessing edge safety through a learning-based local motion estimator.

These methods can efficiently and continuously predict collisions but have significant limitations. A major issue is their reliance on oversimplified evaluation metrics. They encode the entire local motion process—starting pose, target pose, and continuous poses bridging them—into a single measure, such as success/failure or collision probability. This approach does not directly learn the continuous geometric relationship between the environment and local motion but instead learns indirectly from the statistical correlation between the initial pose, the local target pose, and the final outcome (success or failure). As a result, it lacks information on when and why collisions occur, preventing the model from understanding the logic behind collisions. Consequently, collision detection performance deteriorates, and generalization to new environments becomes challenging.

B. Contributions

As mentioned earlier, although traditional geometric-based methods provide high accuracy, they have inherent drawbacks–conservativeness (circumcircle), aggressiveness (incircle), and a large search space (rectangle). Learning-based methods, on the other hand, can efficiently predict continuous collisions but suffer from limited accuracy and generalization. To overcome these limitations, we make the following key contributions:

1) We present a novel collision-checking concept called a Rotatable Area (RoA). The RoA represents a range of heading angles within which the robot can rotate without collision with obstacles. This enables accurate collision checking not only for a single heading angle but also for rotational motion, forming the theoretical foundation of the proposed path planning method.

2) Based on this concept, we develop a path planning framework called RoA-Planner for obstacle-dense spaces. This framework utilizes a quadtree structure to efficiently represent planning space as a set of RoAs. By treating each RoA as a node and local motions between adjacent RoAs as edges, we design a modified A* algorithm tailored to RoA-based path planning. As a result, the framework efficiently generates collision-free paths for holonomic, asymmetric rectangular robots even in highly constrained environments.

3) To evaluate the feasibility of local motions between RoAs, we define two geometric conditions: *area condition* based on the overlap of rotational ranges, and *edge condition* based on the Euclidean distance between them. Owing to these conditions, the SE(2) path planning problem is transformed into a 2D problem, effectively mitigating the curse of dimensionality. In comparison to existing methods, our approach demonstrates superior performance in terms of conservativeness, aggressiveness, and accuracy. Furthermore, it can be readily applied to various path planning algorithms that represent motion using nodes and edges (e.g., PRM*, RRT).

4) Finally, we validate the proposed method through simulations and real-world experiments using a quadruped robot (AiDIN-VIII [35]). RoA-Planner consistently generates stable and reliable paths even in cluttered and narrow environments, and maintains real-time performance in dynamic and large-scale environments, demonstrating strong practical applicability for real robotic systems.

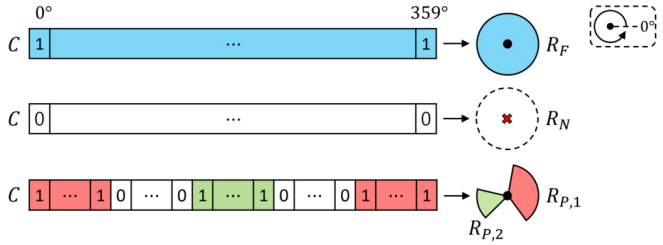


Fig. 2. Generation of three types of RoA from collision-list C.

II. Rotatable Area

A. Definition of Rotatable Area

To define the RoA, we use AiDIN-VIII, a quadruped robot with an asymmetrical rectangular shape, as an example. We define the RoA, denoted as R, as the continuous range of yaw heading angles where the robot can rotate at position (x, y) without collision with obstacles.

$$R = (x, y, \psi_s, \psi_e)^T \tag{1}$$

The parameters $\psi_s, \psi_e \in [0, 2\pi]$ denote the start and end of the RoA, and all parameters are referenced to the world frame. If the robot's heading angle is between ψ_s and ψ_e at (x, y), the robot remains collision-free. Within the range corresponding to RoA, the robot can rotate freely.

The RoA is categorized into three types: Fully-RoA, Non-RoA and Partially-RoA (Fig. 1). Fully-RoA, R_F, represents a situation where the distance from surrounding obstacles is far enough that the robot can exist safely at all heading angles like an open field. Non-RoA, R_N, is a case where collisions with obstacles occur in all heading angles. The robot can't move to a position evaluated as Non-RoA. Partially-RoA, R_P, is a case where the robot can rotate safely in certain heading angles while collisions occur in others. Depending on the configuration of obstacles, the range and number of Partially-RoAs are determined. Multiple Partially-RoAs can exist at the same position, but they are treated as distinct RoAs, and rotating between different Partially-RoAs is impossible.

$$R = \begin{cases} R_F = (x, y, 0, 2\pi)^T \\ R_N = (x, y, \varnothing, \varnothing)^T \\ R_{P,i} = (x, y, \psi_{s,i}, \psi_{e,i})^T \end{cases} \tag{2}$$

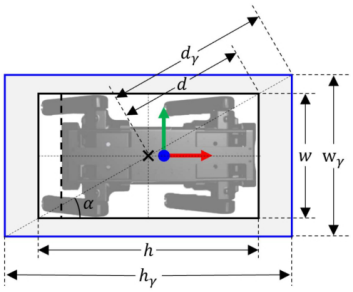
B. Rotatable Area Generation

The RoA is generated through two steps: generating the collision-list and constructing the RoA based on this list (Fig. 2). The collision-list C is generated through collision-checking between obstacles in the map image and the robot footprint. At a given inspection location (x, y), if any pixel within the robot footprint is occupied by obstacles, it is considered as a collision. This collision-checking is performed sequentially for all heading angles, starting from 0°. If a collision occurs at a specific heading angle, the corresponding entry in collision-list is set to *False*; otherwise, it is set to *True*. In this study, the yaw configuration space is discretized with a resolution of 1-degree, resulting in collision list having a size of 360. After evaluating all heading angles, a group of consecutive collision-free angles in collision-list, denoted as C_{free}, forms a single RoA. The type of RoA, as well as the number and range of Partially-RoA, is determined through the size of C_{free}.

$$R = \begin{cases} R_F, & |C_{free}| = 360 \\ R_N, & |C_{free}| = 0 \\ R_{P,i}, & 0 < |C_{free}| < 360 \end{cases} \tag{3}$$

In the case of AiDIN-VIII, collision-checking is performed using an asymmetric rectangular footprint of 1160mm×640mm, which includes the hind limbs (see Table I). Due to the area covering the hind limbs, the distance from the center of rotation to the front and the back differs by about 180 mm. By using an asymmetric collision model, our approach enables precise collision checking in dense spaces. Note that, to apply the theory we established, we check

TABLE I
Notation w.r.t Collision Model



Notation	Meaning
basic footprint (▭), S	Space that covers all parts of the robot
height, h / *width*, w	Height and width of the basic footprint
expansion ratio, γ	Rate of expanding footprint $(1 < \gamma)$
expanded footprint (▭), S_γ	Collision model expanded by expansion ratio
h_γ / w_γ	Height and width of expanded footprint: $h\gamma$ / $w\gamma$
α	Diagonal angle of the footprint
d / d_γ	Half diagonal length of basic footprint and expanded footprint
×	Center of the footprint
	Coordinate of the robot body. The red axis indicates the heading angle.

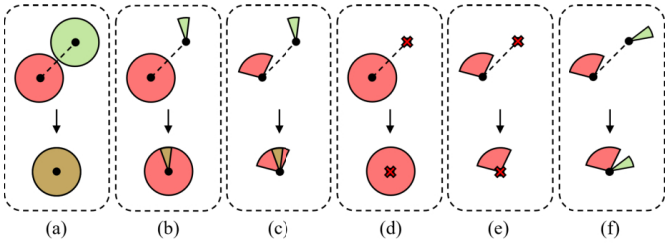


Fig. 3. The *area condition* is determined by whether there are intersections (light brown). (a) R_F-R_F and (b) R_F-R_P pass the *area condition* as they have an intersection, whereas (d) R_N-R_F and (e) R_N-R_P do not. Whether *area condition* of R_P-R_P combination passes (c) or fails (f) is determined by the range of RoAs.

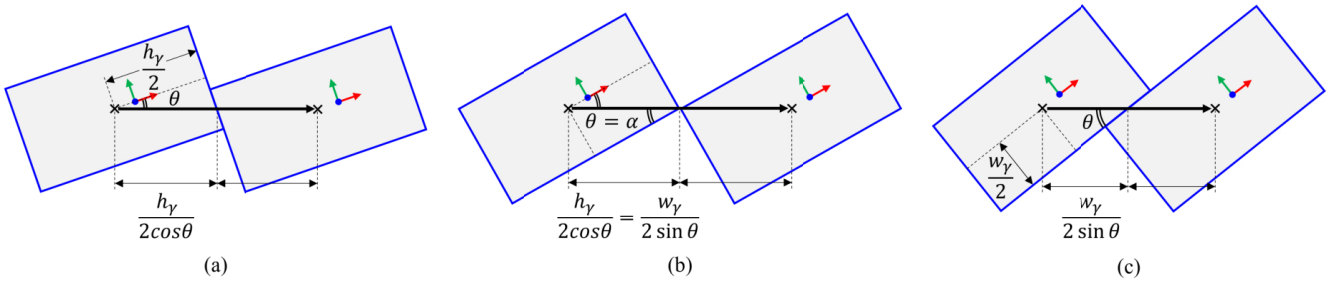


Fig. 4. First condition of the *edge condition*. *Safe space* is connected by (a) width side, (b) vertex, (c) height side.

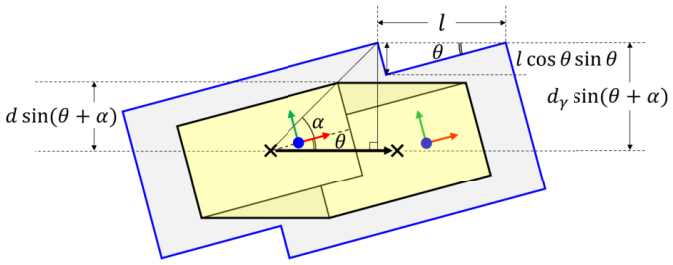


Fig. 5. Second condition of the *edge condition*.

collision using an *expanded footprint* with an *expansion ratio* γ; details are provided in Section III-B.

III. Local Motion Between Rotatable Areas

Our approach regards local motion as a relationship between two RoAs, $R_1 \rightarrow R_2$. Local motion consists of 1) performing rotation at R_1 from the initial heading angle θ_1 to the target heading angle θ_2, and 2) then translation to the target position of R_2.[1] θ_1 and θ_2 are the target heading angles at R_1 and R_2, respectively, and fall within the range of their RoAs. This is the process of determining θ_2 for feasible local motion. If local motion is feasible, a robot can safely traverse between two RoAs without collision. The feasiblility of local motion is determined by *area condition* and *edge condition*.

A. Area Condition

First, to safely rotate from θ_1 to θ_2 at R_1, θ_2 must be included in R_1. To simplify the evaluation of this situation, the *area condition* is defined as follows:

$$\exists R_{inter},\ R_{inter} = R_1 \cap R_2 \tag{4}$$

where R_{inter} denotes the intersection of R_1 and R_2. Since R_{inter} is the subset of R_1, determining θ_2 within R_{inter} ensures a safe rotation from θ_1 to θ_2. Therefore, for feasible local motion, intersections must exist between the two RoAs, and the angles within R_{inter} are considered as candidates for the target heading angle θ_2. The *area condition* based on combinations of RoAs are depicted in Fig. 3.

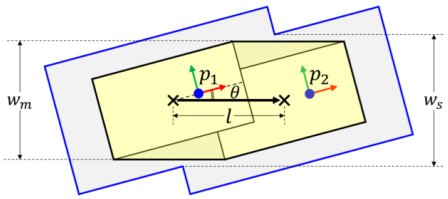
B. Edge Condition

Notations regarding *edge condition* are described in Table II. Let's assume that the *area condition* is satisfied and local motion is executed with a heading angle that belongs to the R_{inter}. In that case, the maximum length of local motion, which guarantees safety in translation motion, could be determined.

$$l \le l_{max}(\gamma, \theta) \tag{5}$$

Equation (5) represents the *edge condition*, where $l_{max}(\gamma, \theta)$ denotes *maximum length* of the safe local motion depending on heading angle and *expansion ratio*. Note that during the process of determining the *edge condition*, the heading angle θ is calculated in a counterclockwise (CCW) direction relative to the local motion direction, but in other situations, it is calculated relative to the world frame. The *maximum length* is

TABLE II
Notation w.r.t Edge Condition



Notation	Meaning
p_1	Start pose of the local motion after rotating by target heading angle θ: $p_1 = (x_1, y_1, \theta)$
p_2	Local target pose of the local motion: $p_2 = (x_2, y_2, \theta)$
local motion [2] ($\longrightarrow$), $\mathcal{L}$	The omni-directional movement of the robot from p_1 to p_2.
heading angle, θ	The heading angle of the robot based on local motion
motion space (), S_m	The space swept by the basic footprint during local motion: $S_m = S(p_1) \oplus \mathcal{L}$ [3]
safe space (), S_s	The space corresponding to expanded footprint at p_1 and p_2. There are no obstacles in safe space, because the heading angle exists within the RoA: $S_s = S_\gamma(p_1) \cup S_\gamma(p_2)$
edge length, l	length of the local motion
motion width, w_m	Maximum length of motion space in the vertical direction of local motion
safe width, w_s	Minimum length of safe space in the vertical direction of local motion

[1]This sequence of in-place rotation followed by linear translation is referred to as the *motion constraint* throughout the paper.

[2]Being translation motion after rotation, local motion can be represented at any position and we use center of the footprint to simplify calculations.

[3]The operation $\oplus$ represents the Minkowski sum.

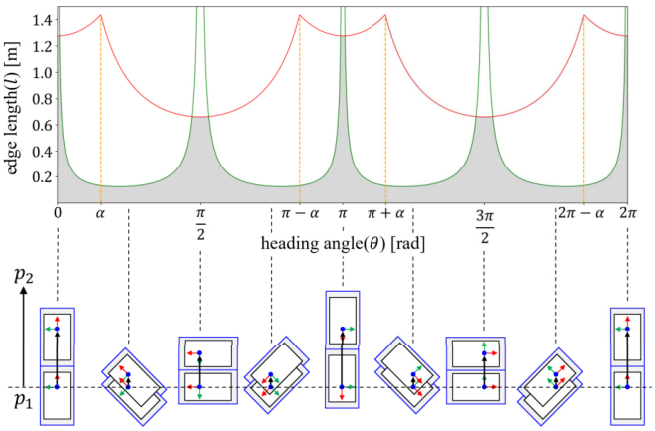


Fig. 6. Graph of *edge condition* when *expansion ratio* is set to 1.1. Red line by Equation (7) and green line is drawn by Equation (11). The figure below expresses the local motion with *maximum length* at representative heading angles.

Algorithm 1 Local Motion Evaluation

Feasibility evaluation of local motion from R_1 to R_2 through *area condition* and *edge condition.*

Input: R_1, R_2
Output: θ_2
procedure *evaluateLocalMotion*(R_1, R_2)
  $R_{inter} \leftarrow$ *checkAreaCondition*(R_1, R_2)
  if R_{inter} is empty **then** ▷area condition not met
    return $\theta_2 \leftarrow$ *Null*
  $l \leftarrow$ *Distance*(R_1, R_2)
  $R'_{inter} \leftarrow$ *checkEdgeCondition*(R_{inter}, l) ▷Alg. 2
  if R'_{inter} is empty **then** ▷edge condition not met
    return $\theta_2 \leftarrow$ *Null*
  else ▷edge condition met
    return $\theta_2 \leftarrow$ *TargetHeadingAngle*(R'_{inter})
  end if
end procedure

determined under the assumption that the *motion space* must be completely contained in the *safe space*:

$$S_m \subseteq S_s \tag{6}$$

If we use the *basic footprint* to generate RoA, the *motion space* becomes larger than the *safe space* and Equation (6) cannot be satisfied. That's why we use the *expanded footprint* in RoA generation. The *maximum length* that satisfies Equation (6) is calculated by two simple geometric analyses.

1) Condition 1: The *safe space* must be connected. Otherwise, the safety of the space in between cannot be guaranteed. Fig. 4 depicts the minimum conditions under which *safe space* can be connected, and the *maximum length* at this time is determined by the connecting side. Through the analysis in Fig. 4(a) and (c), the *maximum length* is calculated when *width* or *height* is connected. As shown in Fig. 4(b), the connecting side is switched based on when the *safe space* is connected by a vertex ($\theta = \alpha$). Therefore the first condition of the *maximum length*, defined from 0 to 2π is as follows.

$${}^1l_{max} = \begin{cases} \left|\dfrac{h\gamma}{\cos\theta}\right| & (\theta \in (2\pi-\alpha, \alpha) \cup (\pi-\alpha, \pi+\alpha)) \\ \left|\dfrac{w\gamma}{\sin\theta}\right| & (\theta \in (\alpha, \pi-\alpha) \cup (\pi+\alpha, 2\pi-\alpha)) \end{cases} \tag{7}$$

2) Condition 2: The *safe width* must be larger than the *motion width*. Since the *safe width* and the *motion width* are symmetrical based on the direction of the local motion, we set up the formula below:

$$\frac{w_m}{2} \leq \frac{w_s}{2} \tag{8}$$

Through the analysis of Fig. 5, Equation (8) is converted to Equation (9). And by organizing this, the *maximum length* when the heading angle is from 0 to $\pi/2$ is obtained as in Equation (10).

$$d\sin(\theta+\alpha) \leq d_\gamma \sin(\theta+\alpha) - l\cos\theta\sin\theta \tag{9}$$

$$l \leq (d_\gamma - d)\frac{\sin(\theta+\alpha)}{\cos\theta\sin\theta} \tag{10}$$

In a similar way, we define the second condition of the *maximum length* in each quadrant as follows.

$${}^2l_{max} = \begin{cases} \left|(d_\gamma - d)\dfrac{\sin(\theta+\alpha)}{\cos\theta\sin\theta}\right| & \left(\theta \in \left(0, \frac{\pi}{2}\right) \cup \left(\pi, \frac{3\pi}{2}\right)\right) \\ \left|(d_\gamma - d)\dfrac{\sin(\theta-\alpha)}{\cos\theta\sin\theta}\right| & \left(\theta \in \left(\frac{\pi}{2}, \pi\right) \cup \left(\frac{3\pi}{2}, 2\pi\right)\right) \end{cases} \tag{11}$$

We define the *maximum length* that satisfies both Equation (7) and (11) as follows:

$$l_{max} = \min({}^1l_{max}, {}^2l_{max}) \tag{12}$$

Fig. 6 depicts the *maximum length* for each heading angle, drawn by Equation (7) and (11). The gray area, satisfying Equation (12), represents the lengths of local motion that guarantee safety according to the heading angle. Due to the configuration of the *safe space*, it is observed that the robot's longitudinal ($\theta = 0, \pi$) and lateral motions ($\theta = \frac{1}{2}\pi, \frac{3}{2}\pi$) have a greater *maximum length* compared to its diagonal motions ($\theta = \frac{1}{4}\pi, \frac{3}{4}\pi, \frac{5}{4}\pi, \frac{7}{4}\pi$). Note that we precompute the *maximum length* offline and use it to determine the feasibility of local motion online.

C. Feasibility of Local Motion

To determine a feasible local motion in a planning problem, we use the *area condition* and *edge condition* defined above. In Algorithm 1, we first identify the candidates for valid heading angles of local motion, R_{inter}, by checking the *area condition* between the two areas. If R_{inter} is empty, it indicates that the *area condition* is not satisfied, and the local motion is deemed infeasible. Otherwise, we proceed to verify whether any heading angle within R_{inter} has a longer *maximum length* than the distance between the two areas (line 8 → Algorithm 2). In other words, R_{inter} is filtered to R'_{inter}, which includes only heading angles that satisfy the *edge condition*. Therefore, if R'_{inter} is not empty, the local motion is feasible and target heading angle θ_2 is determined within R'_{inter}.

Algorithm 2 Edge Condition Checking

Identification of the area that satisfies the *edge condition*.

Input: R_{inter}, l
Output: R'_{inter}
procedure *checkEdgeCondition*(R_{inter}, l)
  for each θ in R_{inter} **do**
    if $l \leq l_{max}[\theta]$ **then**
      Add θ to R'_{inter}
  end for
  return R'_{inter}
end procedure

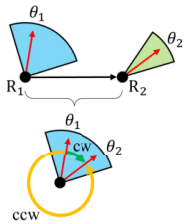


Fig. 7. Rotating from θ_1 to θ_2, clockwise is valid, but counterclockwise is not because it exceeds range of R_1.

For complete safety, our approach uses the middle value of R'_{inter}.

To track the local motion, robot should rotate at the position of R_1 from θ_1 to θ_2. As described in Fig. 7, there are two directions to rotate from θ_1 to θ_2: clockwise and counterclockwise. Rotations in both directions create opposing ranges, and at least one range is included in R_1. The included one is selected as the rotation direction, d_r, for safe rotation.

By using this method, it can be ensured that the local motion is collision-free in continuous space. Additionally, unlike existing methods that sample directly in the SE(2) space of (x, y, θ), our method achieves higher efficiency by determining valid heading angles from samples generated solely in the 2D space of (x, y). This method can be applied to various path planning structures that check local motion between adjacent nodes. The *area condition* and *edge condition* are useful criteria for selecting candidates of near nodes, which guarantees safety according to the heading angle and *expansion ratio*. An expanded collision model also helps to prevent collisions caused by control, sensor or mapping errors since the RoA is conservatively generated.

IV. RoA-Planner

In this section, we extend our local motion evaluation method to path planning framework, RoA-Planner. While single-query methods like RRT or RRT* are applicable, we use a multi-query approach to generate a RoA-Map. This approach leverages GPU acceleration to enhance the generation speed of multiple RoAs. Details of the parallelization are discussed in Section V-A.

The generated RoA-Map is used in the graph search algorithm. We demonstrate the utilization of the structure of the A* algorithm, but the structure of other graph-based methods can also be used. The overall framework of the planner consists of two stages: mapping and planning (Fig. 8).

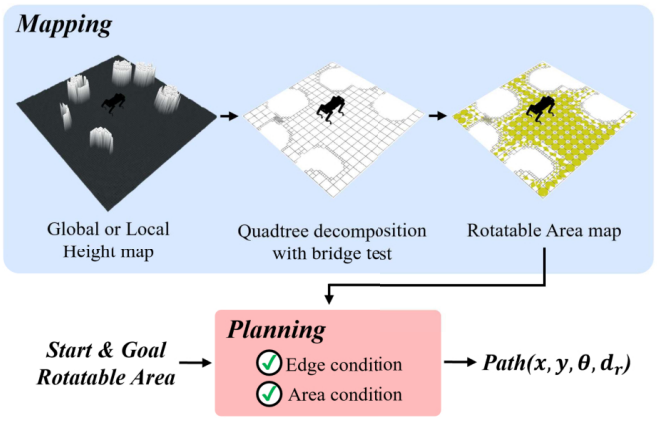


Fig. 8. Framework of RoA-Planner.

Algorithm 3 Quadtree w/ Bridge test

Recursive construction of a quadtree with bridge test and RoA generation for each cell.
$\mathcal{M}, c$: Map image and decomposed cell
s, s_{max}, s_{min}: Cell size, maximum and minimum cell size
$\mathcal{Q}$: RoA-Map in quadtree format

Input: $\mathcal{M}, c$
Output: $\mathcal{Q}$
procedure *buildQuadtree*(c)
  if $s < s_{max}$ **then** ▷ split cell until maximum size
    createChildren(c) & *buildQuadtree*(c)
  else
    if *IsOccupied*(c) **then** ▷ occupied
      createChildren(c) & *buildQuadtree*(c)
    else ▷ free
      if $s = s_{min}$ **then**
        if *testBridge*(c) **then**
          generateRotatableArea(c)
      else
        if *testBridge*(c) **then** ▷ narrow space
          createChildren(c) & *buildQuadtree*(c)
        else ▷ open space
          generateRotatableArea(c)
        end if
      end if
    end if
  end if
end procedure

A. Mapping

In the mapping stage, we generate a RoA-Map, indicating RoA in each region. In detail, nodes are distributed throughout the given map, and assigned the corresponding RoA following the method described in Section II-B. To get nodes, we can use various sampling methods or a grid map; however, for robustness in dense environments, we utilize quadtree decomposition [36] and bridge test [37]. Quadtree decomposition is an algorithm that recursively divides the space into four

Algorithm 4 Modified A*

A* algorithm to utilize RoA.
R_{start}, R_{goal}: Start and goal RoA
$\mathcal{Q}$: RoA-Map in quadtree format
O: Open list in priority queue format
$\mathcal{P}$: Path composed of RoAs

1: **Input:** $R_{start}, R_{goal}, \mathcal{Q}$
2: **Ouput:** $\mathcal{P} = \{R_{start}, \ldots, R_{goal}\}$
3: **procedure** *ModifiedAstar*(R_{start}, R_{goal}, $\mathcal{Q}$)
4: Add R_{start} to O
5: **while** O is not empty **do**
6: Pop R_{cur} from O with lowest f
7: **if** $R_{cur} = R_{goal}$ **then**
8: **return** $\mathcal{P}$
9: **for each** R_{near} in *searchNeighbor*($\mathcal{Q}$, R_{cur}) **do**
10: $\theta_{near} \leftarrow$ *evaluateLocalMotion*(R_{cur}, R_{near})
11: ▷ Alg. 1
12: **if** θ_{near} is *Null* **then**
13: **continue**
14: Calculate cost of R_{near}
15: Add R_{near} to O
16: **end for**
17: **end while**
18: **end procedure**

cells if the space is occupied by obstacles. And the bridge test is an algorithm that detects milestones, which are samples in narrow passages. We perform a bridge test on the cells decomposed by the quadtree (Algorithm 3). If a cell passes the bridge test, it is further subdivided even if it is free of obstacles (line 14-15). Through this, the cells are densely distributed in confined spaces and sparsely distributed in open field. We use the center point of each leaf cell as a node. At this node, a collision list is generated and subsequently transformed into RoAs (line 12, 17).

B. Planning

As shown in Algorithm 4, we develop an A* algorithm suitable for RoA. The overall structure follows the standard A* algorithm, but uses a RoA instead of a node. Rather than pre-constructing a graph, the feasibility of local motion is evaluated simultaneously with the A* search through Algorithm 1. Given start and goal pose, the RoAs are generated, and the RoA containing each heading angle are designated as R_{start} and R_{goal}. The algorithm searches a sequence of RoAs that connect R_{start} and R_{goal}. The generated path consists of the x, y coordinates, target heading angles θ, and rotation directions d_r of each waypoint RoA.

Just like the standard A* algorithm, the search is performed around R_{cur}, which has the lowest cost in open list O. If R_{cur} and R_{goal} are equal, the path is returned (line 7-8). In the RoA-Map $\mathcal{Q}$, R_{near} within the search radius from R_{cur} are retrieved using quadtree range search (line 9). The search radius is defined as the largest value of the *maximum length* since the heading angle is yet to be determined. For every R_{near}, the feasibility of the local motion is evaluated and the target heading angle is determined (line 10 → Algorithm 1). After calculating the cost, R_{near} is added to O (lines 14-15).

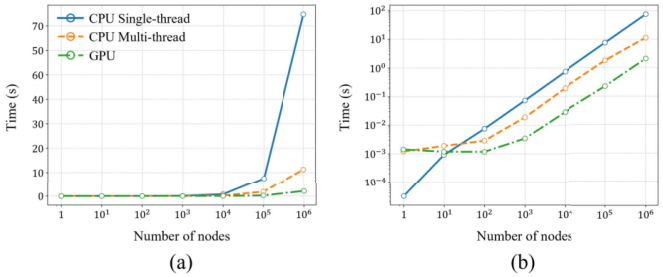


Fig. 9. Graph (a) illustrates the collision-list generation time across three computing platforms—CPU single-thread, CPU multi-thread, and GPU—as a function of the number of nodes. Graph (b) presents the same data with a logarithmic scale applied to the y-axis for better visualization of time comparison in the small number of nodes.

The cost function is defined as follows.

$$f(R_n) = g(R_n) + \lambda_h h(R_n)$$
$$g(R_n) = \sum_{k=1}^{n} l_k + \lambda_r \sum_{k=1}^{n} |\theta_k - \theta_{k-1}|$$
$$h(R_n) = \left\| \begin{bmatrix} x_{goal} \\ y_{goal} \end{bmatrix} - \begin{bmatrix} x_n \\ y_n \end{bmatrix} \right\|_2 + \lambda_r |\theta_{goal} - \theta_n| \quad (13)$$

where λ_h and λ_r are weights for heuristic cost and heading angle change, respectively. θ_n is the target heading angle assigned to R_n, as observed in the world frame. The cost g represents the actual motion cost, calculated as the sum of the local motion lengths and heading angle changes to reach R_n. The cost h is the heuristic cost, defined as the Euclidean distance and minimum heading angle change from R_n to R_{goal}. By adjusting λ_h, the planner can prioritize distant RoA, naturally favoring forward motion, while seeking paths with minimal distance and rotation overall.

V. Analysis

This section presents a detailed analysis of the proposed method, and the results demonstrate its scalability. All performance measurements were conducted on the NVIDIA Jetson AGX Orin 32GB.

A. Parallelization

In the mapping stage, the collision-list generation step constitutes the dominant portion of the overall computation time. As shown in Fig. 9, the generation of collision-list for 10^6 nodes requires approximately 74.7 seconds when using a single-threaded CPU. This high computational cost stems from the need to check the occupancy status of $w_\gamma \cdot h_\gamma$ pixels across 360 heading angles for each node, resulting in a substantial per-node workload. However, since the collision-list generation is completely independent for each node, the process is highly amenable to parallelization. In practice, when utilizing CPU multi-threading under the same conditions, the execution time is reduced to approximately 11.23 seconds. Furthermore, leveraging a GPU results in a significant acceleration, achieving a runtime of only 2.09 seconds. These results demonstrate the effectiveness of parallel processing for this task.

Therefore, in methods based on the PRM framework—where a large number of nodes are processed

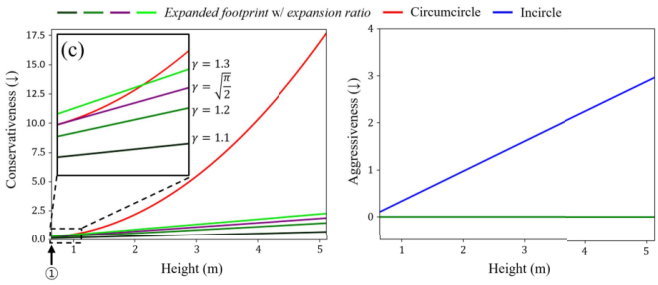


Fig. 10. (a), (c) conservativeness and (b) aggressiveness graph. As the height increases, the asymmetry of the rectangular footprint becomes more pronounced. ① represents a square footprint (0.64m×0.64m).

in batch—substantial performance gains can be achieved by employing CPU multi-threading or GPU acceleration. For single-query approaches like RRT or scenarios with a small number of nodes, single-threaded execution is more efficient since it incurs no parallelization overhead. These results demonstrate that our method can be effectively integrated with various types of planning algorithms and is capable of supporting real-time operation. Notably, 10^6 nodes correspond to the number required to cover a large-scale open field of 320m×320m under our experimental conditions. The successful real-time performance at this scale, enabled by parallelization, confirms the practicality of the proposed approach for large environments. Since our planning method is based on a quadtree structure and processes many nodes concurrently, all subsequent experiments adopt the GPU-based collision-list generation approach.

B. Conservativeness and Aggressiveness

We also examine the conservativeness and aggressiveness of our collision model (*expanded footprint* with *expansion ratio*) across various robot shapes, from square to rectangular robots with extreme aspect ratios. The conservativeness, defined as the area difference between the robot's actual footprint and collision model, is compared with circumcircle. The conservativeness of the circumcircle and ours is calculated as Equation (14) and (15), respectively. High conservativeness indicates lower robustness.

$$\pi\left(\left(\frac{w}{2}\right)^2+\left(\frac{h}{2}\right)^2\right)-wh \tag{14}$$

$$wh\gamma^2-wh \tag{15}$$

We fix the width at 0.64m and increase the height from 0.64m. As shown in Fig. 10(a), for most heights, the circumcircle is notably more conservative, with the gap widening as height increases. However, Fig. 10(c) reveals that when *expansion ratio* is 1.1 and 1.2, ours is less conservative, while when 1.3, it becomes more conservative at certain heights. Through this analysis, we can determine the value of *expansion ratio* that ensures the conservativeness of our method is smaller than that of the circumcircle across all heights, as shown in Equation (16). By defining this value as the guideline for *expansion ratio*, the conservativeness can be maintained for rectangular robots of all aspect ratios.

$$\gamma_{guide}=\sqrt{\frac{\pi}{2}}\approx 1.253 \tag{16}$$

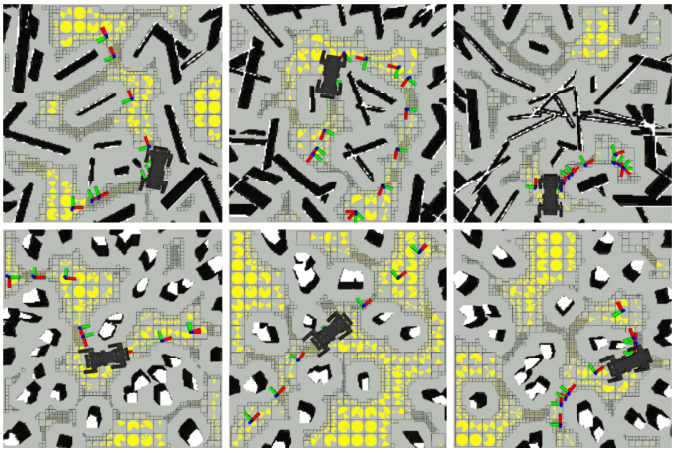


Fig. 11. Simulation test in dense spaces with AiDIN-VIII. The grid represents the quadtree cells, and the yellow sectors indicate the RoAs.

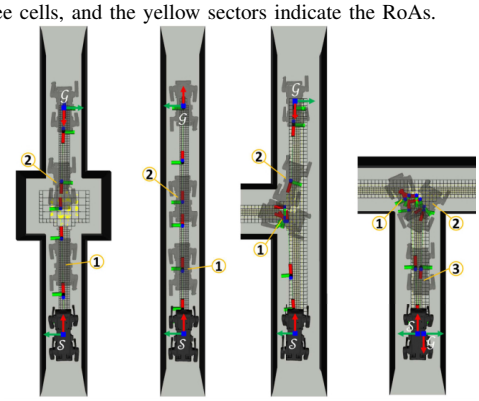


Fig. 12. Simulation test in narrow environments such as a hallway. The numbers represent the order in which the robot moves.

The aggressiveness is how boldly the robot collision model is expressed. It is compared with the incircle representation, which is calculated as follows:

$$wh-\pi\left(\frac{w}{2}\right)^2 \tag{17}$$

Unlike the incircle, our approach does not underestimate the collision model, resulting in zero aggressiveness. Fig. 10(b) shows that the incircle's aggressiveness remains significantly higher.

In conclusion, by setting the *expansion ratio* of the RoA-Planner to be smaller than γ_{guide}, it provides advantages in terms of both robustness (conservativeness) and safety (aggressiveness) compared to existing methods, regardless of the aspect ratio of the rectangular footprint.

VI. Experiment

We demonstrate our work with asymmetric rectangular robot AiDIN-VIII in simulation and real-world scenarios.[4] And we compare our method using the configuration of symmetric rectangular robot ANYmal-C [38]. The *expansion ratio* was set in accordance with the guideline in Section V-B. The whole framework of the planner runs on NVIDIA Jetson AGX Orin 32GB.

A. Simulation

We evaluated the path planning performance of RoA-Planner using AiDIN-VIII within gazebo simulation. The

[4]https://youtu.be/d1c2L-7MnsU

TABLE III

PATH PLANNING COMPARISON IN DENSE ENVIRONMENTS

Best / Second-best

Evaluation Metrics	*Wall 1*				*Wall 2*				*Polygon 1*				*Polygon 2*			
	C (↓)	T (↓)	D (↓)	R (↓)	C (↓)	T (↓)	D (↓)	R (↓)	C (↓)	T (↓)	D (↓)	R (↓)	C (↓)	T (↓)	D (↓)	R (↓)
RoA-Planner	**0**	**0.114s**	5.78m	**140.0°**	**0**	**0.178s**	7.37m	**374.0°**	**0**	**0.079s**	5.42m	**232.0°**	**0**	**0.04s**	**5.69m**	**44.0°**
Baseline 1	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Baseline 2	**0**	0.264s	**5.75m**	270.8°	1	0.41s	7.77m	521.1°	**0**	0.138s	**5.35m**	339.1°	**0**	0.126s	5.71m	105.5°
Wellhausen *et al.* [15]	**0**	0.3s	6.97m	408.2°	**0**	0.3s	9.29m	441.2°	**0**	0.3s	6.81m	350.9°	**0**	0.3s	6.73m	586.0°
Art-Planner [34]	**0**	0.3s	6.9m	276.5°	**0**	0.3s	9.41m	417.6°	**0**	0.3s	7.16m	308.6°	1	0.3s	8.67	379.6°
TRG-Planner [19]	16	0.244s	5.82m	-	29	0.212s	**7.27m**	-	8	0.383s	5.69m	-	9	0.434s	6.08m	-

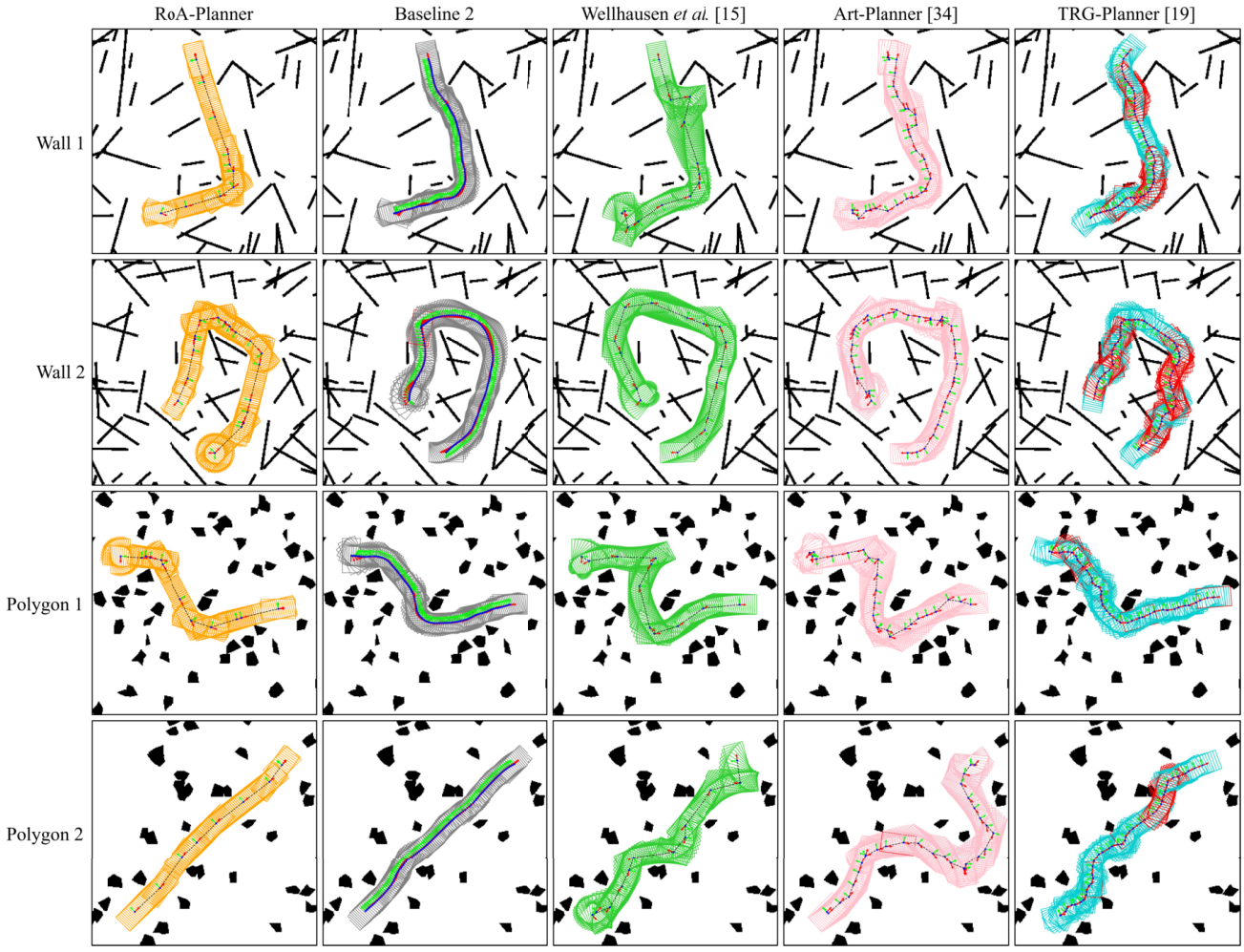


Fig. 13. Visualization of path footprints generated by the RoA-Planner and baselines under the ANYmal-C configuration. States that collided with obstacles are highlighted in red on the path.

environments is 5m×5m square, and the start and goal poses are configured to require traversal of a narrow and cluttered area. As shown in Fig. 11, by quadtree with bridge test, the complex environments were partitioned efficiently, creating fewer RoAs in open fields and more RoAs in narrow passages. In the RoA-Map, modifed A* algorithm searched sequences of RoAs that satisfy both *area condition* and *edge condition*. The resultant path was followed by the robot, allowing it to safely reach its goal without any collisions. Through this experiment, we validated that the RoA-Planner successfully generates the shortest collision-free path in highly-dense spaces.

Moreover, as illustrated in the Fig. 12, we specifically evaluated performance in narrow passages by setting the yaw of the goal pose opposite to that of the start pose. Due to spatial constraints, the critical challenge was finding suitable rotation region within tight spaces, where the robot often cannot rotate freely. The planner found sequence of pose that satisfy challenging start and goal pose constraints, demonstrating flexibility and robustness. This capability is particularly valuable for real-world service application, such as mobile manipulation, parking, or charging station docking.

Next, we compare RoA-Planner and six planners consisting of geometric and learning-based methods using ANYmal-C

TABLE IV
SPECIFICATION OF BASELINES

	Algorithm basis	Type of local motion evaluation
Baseline 1	Grid	Geometric (circumcircle)
Baseline 2	Grid + Optimization	Geometric (incircle + rectangle)
Wellhausen *et al.* [15]	Sampling	Geometric (rectangle)
Yang *et al.* [29]	Grid	Learning
Art-Planner [34]	Sampling	Learning + Geometric (rectangle)
TRG-Planner [19]	Sampling	Geometric (incircle)

TABLE V
PATH PLANNING COMPARISON IN NARROW CORRIDORS

	Narrow corridor 1	*Narrow corridor 2*	*Narrow corridor 3*
RoA-Planner	✓	✓	✓
Baseline 2	×	×	×
Wellhausen *et al.* [15]	5/10	5/10	4/10
Art-Planner [34]	6/10	3/10	6/10
TRG-Planner [19]	×	×	×

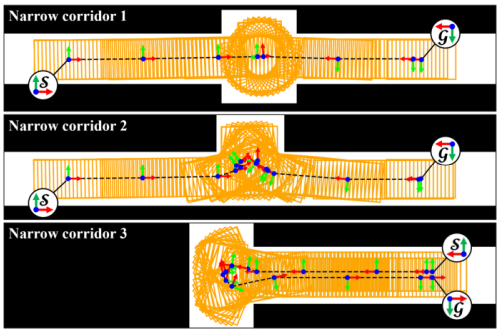
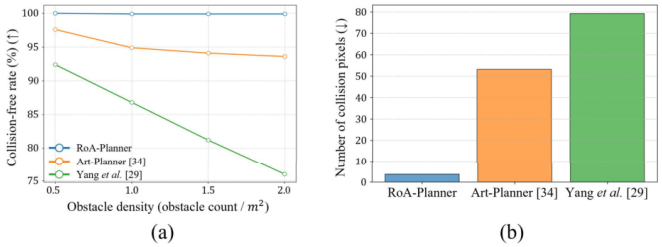


Fig. 14. Path planning scenario in narrow corridor environments with the start and goal heading angle set in completely opposite directions. The path shown in the figure was generated by RoA-Planner.

(930mm×530mm). The specification of all comparison algorithms are summarized in Table IV. Baseline 1 generates paths using A* algorithm on maps where obstacles are inflated to the size of a circumcircle. Baseline 2 utilizes the incircle size for raw path generation and assigns the heading angle as the tangential direction of the path. A gradient-based optimizer then refines the path for collision avoidance and smoothness, considering a rectangular footprint in x, y, and yaw. Wellhausen et al. and Art-Planner, both based on PRM*, were configured to generate paths within a specified time limit for each comparison. And their rectangular collision models are expanded by the same *expansion ratio* as RoA-Planner. Since TRG-Planner plans positions relied on incircle-based method, we set the heading angles to follow the tangential direction of the generated path. In comparison, collisions (C) were counted by dividing the path into discrete states and measuring pixel-level overlaps between each state's footprint and the map image. And we denote the planning time as T, the total path length as D, and the accumulated heading angle change as R.

First, in Table III and Fig. 13, we compare RoA-Planner with five planners—Baseline 1, Baseline 2, Wellhausen et al., Art-Planner, and TRG-Planner—to analyze path planning performance in highly obstacle-dense environments. Planning was performed on four 5m×5m maps, each with the same



(a) (b)

Fig. 15. Comparison of collision-free rate and severity of random local motion under varying obstacle densities.

start and goal poses. Baseline 1 failed to plan paths due to over-approximation of the collision model. And Baseline 2 almost successfully planned smooth and collision-free paths. However, since it requires collision checking for the rectangular model at every iteration for all waypoints, it takes longer than RoA-Planner. Wellhausen et al. and Art-Planner were allocated 0.3 seconds for planning, which exceeds the planning time of RoA-Planner. Despite achieving near-accurate collision detection, these methods failed to produce optimized paths due to the large search space (clearly visible in *Wall 2* and *Polygon 2*). And because of the inherent limitation of its incircle-based approach, which neglects heading angle considerations, TRG-Planner resulted in many collisions. In contrast, RoA-Planner generated collision-free paths with the fastest planning times. In most environments, our method outperformed other algorithms in terms of energy efficiency, as measured by total distance and total heading angle change.

Secondly, we compare RoA-Planner against Baseline 2, Wellhausen et al., Art-Planner, and TRG-Planner in terms of robustness when navigating narrow corridors (Table V and Fig. 14). In this comparison, the goal pose was set directly opposite to the start, and most space do not allow full turns. In experimental results, Baseline 2 diverged during optimization due to its inability to establish proper initial heading angles in *narrow corridor 1* and *narrow corridor 2*. Furthermore, in *narrow corridor 3*, there was a significant limitation due to an insufficient number of waypoints for optimization. Although Wellhausen et al. and Art-Planner were allocated 4 seconds for planning, they achieved low success rates (3 to 6 successful attempts out of 10), primarily due to the extensive search space required by their rectangular collision model. And also because TRG-Planner generates paths based solely on position, it cannot account for constraints where the start and goal have opposite heading angles in narrow environments. On the other hand, RoA-Planner efficiently generated collision-free paths in all experimental settings and demonstrated superior robustness in narrow passages compared to other algorithms.

To further validate the safety performance of local motion evaluation, we compares RoA-Planner with Yang et al. and Art-Planner under varying obstacle densities. In the Fig. 15(a), obstacle density is defined as the number of obstacles with random sizes present within a 1m×1m area. The collision-free rate represents the percentage of local motions classified as safe by each algorithm that did not result in actual

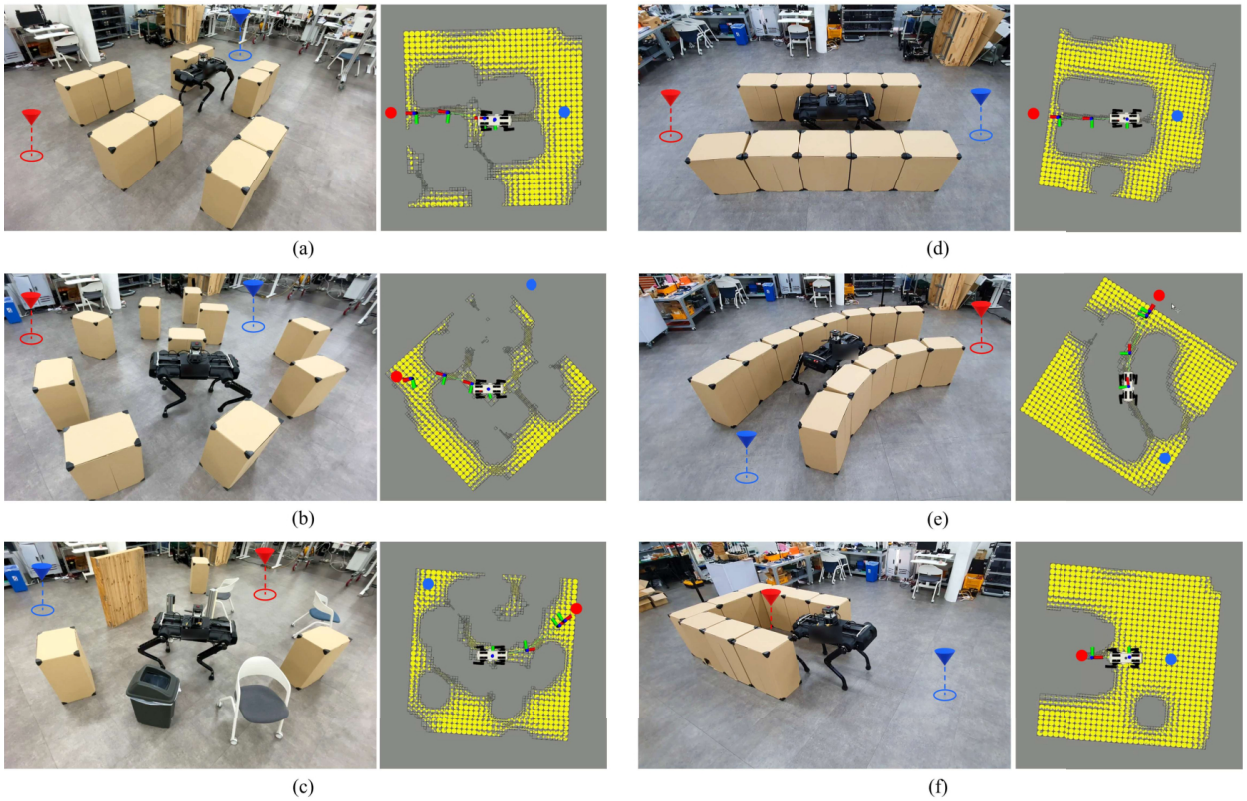


Fig. 16. Path planning test in various scenario including obstacle-dense environments and structured settings. The blue and red point represent the start and goal region, respectively, while the right side illustrates the generated RoA-Map and the path.

collisions. Each algorithm generated 1,000 feasible random local motions in environments with obstacle densities of 0.5, 1.0, 1.5, and 2.0. The results show that RoA-Planner maintained an almost 100% collision-free rate regardless of obstacle density, whereas Yang et al. and Art-Planner exhibited lower collision-free rates, which declined sharply as obstacle density increased. This result originated from the insufficient accuracy and generalization ability of the learning-based local motion estimator. While RoA-Planner is theoretically collision-free, minor collisions were detected due to pixel-level collision detection. Additionally, Fig. 15(b) illustrates the severity of collisions, showing the number of overlapping pixels between the footprint and obstacles while local motion. Yang et al. and Art-Planner recorded 53.2 and 79.2 overlapping pixels on average, indicating that they frequently misclassified high-impact local motions as safe. In contrast, RoA-Planner averaged just 3.6 overlapping pixels, demonstrating that only minor, negligible collisions occurred.

B. Real-World

In real robot experiments, AiDIN-VIII with an Ouster lidar OS0-64 navigates obstacle spaces. A local height map is generated via elevation mapping [39], converted into a gray-scale image, and dichotomized based on the robot's step-over height. If the goal pose is within the local map, the nearest RoA containing the goal heading angle is selected as a target; otherwise, the closest RoA to the goal that contains the robot's current heading angle is chosen. The robot executes the first local motion from the generated path, and the process repeats until the goal pose is reached.

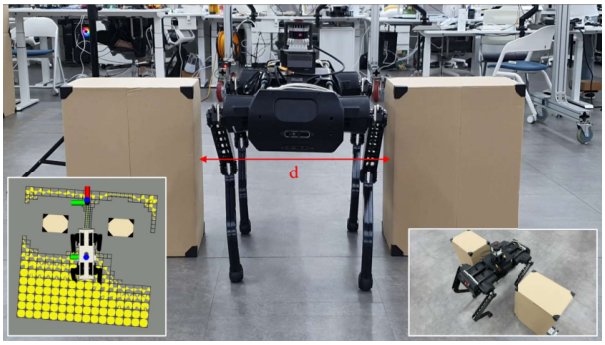
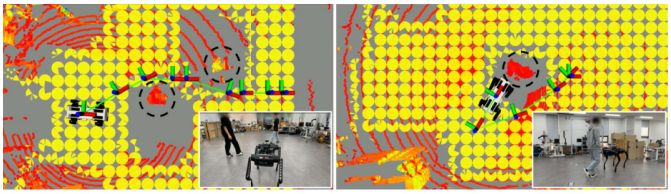


Fig. 17. Path planning test in narrow gap. The figure above demonstrates successful navigation with d = 0.8 m, achieving success at 1.0, 0.95, 0.9, and 0.85 m.

In the first experiment, two obstacles were positioned at varying distances apart to test the robot's ability to navigate through narrow gaps (Fig. 17). The spacing was reduced incrementally from 1.0m to 0.8m in 0.05m increments. The robot successfully traversed gaps across all tested distances without collisions. While the theoretical limit for gap navigation is 0.704m (w_γ), practical challenges like localization errors and sensor noise made path generation difficult near this thresh-



(a) (b)

Fig. 18. Responsiveness test in the presence of dynamic obstacles. The dashed circles indicate dynamic obstacles.

old. Despite practical issues, the robot consistently navigated 0.8m gaps collision-free, demonstrating robust performance in narrow spaces.

In the second experiment, path planning was tested in challenging scenarios, including dense obstacles and structured settings (Fig. 16). The goal pose was placed beyond narrow, intricate areas obstructed by obstacles. The robot successfully avoided collisions with diverse obstacles like rectangular obstacles, walls, pillars, and chairs (Fig. 16(a)(b)(c)). It also performed well in structured environments, handling straight and curved corridors (Fig. 16(d)(e)). Practical tasks such as returning to charging or parking stations were reliably completed with safe heading angles (Fig. 16(f)).

Furthermore, we tested the responsiveness of the RoA-Planner to dynamic obstacles. To this end, we set a fixed goal pose in 10m×10m local map and the planner periodically generated a path from the robot's current pose. In the first experiment (Fig. 18(a)), multiple moving obstacles frequently crossed the planned path, creating an dynamically obstacle-dense environment. Despite these unpredictable environmental changes, RoA-Planner demonstrated fast replanning capabilities by generating and exploring appropriate RoAs and maintaining collision-free paths. In the second experiment (Fig. 18(b)), a person continuously blocked the robot's path during navigation, invalidating the initially planned path. In response, the RoA-Planner quickly regenerated a new safe path whenever the path was obstructed. In addition, It exhibited push-back motions when encountering sudden obstacles, enabling the robot to continue without colliding with the person. These experimental results demonstrate the real-time performance of RoA-Planner in dynamic obstacle environments. However, we observed that the robot exhibited somewhat static and unnatural motions while tracking the generated path. This behavior is attributed to the *motion constraint* described in Section III and is discussed in more detail in Section VIII.

Finally, we compared the global path planning performance of RoA-Planner with Art-Planner and TRG-Planner in a large-scale real-world environment (See Table VI and Fig. 19). For the experiment, we used LIO-SAM [40] to generate a 40m×25m global map of a laboratory and projected it onto the x–y plane, converting it into a 2D image map. According to the experimental results, the TRG-Planner successfully generated short paths through narrow passages, but it resulted in a high number of collisions because of the

TABLE VI

GLOBAL PATH PLANNING COMPARISON IN LARGE-SCALE REAL-WORLD

Evaluation Metrics	*Path 1*				*Path 2*			
	C (↓)	T (↓)	D (↓)	R (↓)	C (↓)	T (↓)	D (↓)	R (↓)
RoA-Planner	**0**	**0.925s**	**39.5m**	**611.0°**	**0**	**0.519s**	**13.55m**	**239.0°**
Art-Planner [34]	3	180s	54.15m	1677.7°	1	180s	27.43m	664.5°
TRG-Planner [19]	38	1.476s	41.5m	-	33	1.462s	13.72m	-

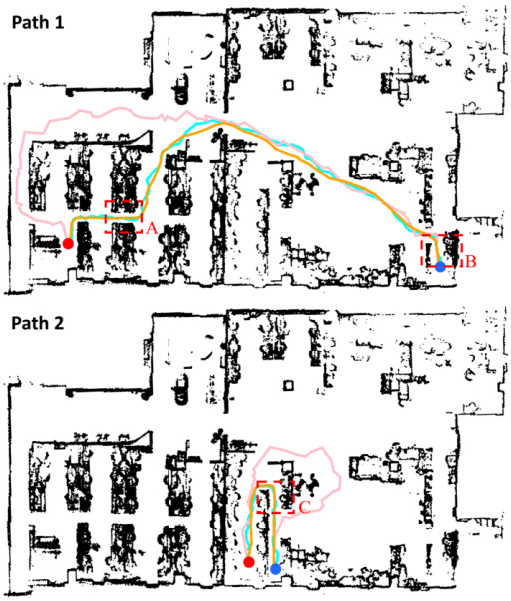


Fig. 19. Global path planning results in a 40m×25m laboratory using the RoA-Planner (orange), Art-Planner (pink), and TRG-Planner (cyan). The environment includes a mix of narrow corridors and open fields. The width of region A is approximately 0.8m, and that of region C is approximately 1m.

nature of incircle representation. And the Art-Planner was unable to obtain a sufficient number of valid samples in narrow regions A and C. Consequently, the planner generated wider, yet longer, alternative paths. Moreover, the planning time required was at least 180 seconds for both paths. In contrast, RoA-Planner successfully identified narrow passages A and C, generating the most efficient paths within 1 second (*path 1*: 0.925s, *path 2*: 0.519s). These experimental results suggest that RoA-Planner performs effectively in large-scale real-world environments, showing promising capability in identifying narrow passages with relatively fast computation speed.

Particularly, a notable number of collisions involving the Art-Planner were observed in region B, due to uncertainty caused by sensor noise and low resolution. These factors led to ambiguous or weak representations of obstacle boundaries, making it difficult for the learning model to accurately identify and characterize obstacles. If the training data does not sufficiently account for these uncertainties, learning-based methods inevitably suffer from the sim-to-real problem. Even with data augmentation techniques or collecting real-world data, it is impossible to account for all possible real-world scenarios. Unlike that, RoA-Planner, which relies solely on geometric information, is fundamentally immune to the sim-to-real problem. Consequently, it demonstrates robustness against

environmental uncertainties in real-world, successfully generating safe paths.

VII. Conclusion

In this work, we propose a novel collision-checking method, called the Rotatable Area, which precisely evaluates collisions based on the robot's heading angle. Leveraging simple geometric conditions of the RoAs, this method accurately enables the identification of safe local motions. The approach exhibits strong scalability, making it applicable to any waypoint-based path planning framework and adaptable to various types of rectangular robots. By further extending this evaluation scheme into a complete path planning algorithm, we significantly reduce the search space of rectangular robots in obstacle-dense environments, enabling fast and safe path generation. The effectiveness and robustness of our method have been validated through extensive simulations and real-world experiments, confirming its practical applicability to real-world navigation task.

VIII. Limitation And Future Work

Due to the nature of the proposed local motion evaluation method, tracking a planned path requires the *motion constraint*: first in-place rotate and then translate (as discussed in Section III). By enforcing this two-phase motion pattern, the collision-free motion is gauranteed. However, this constraint makes path tracking difficult compared to continuous motion that perform rotation and translation simultaneously. The limitation is especially evident in dynamic environments requiring frequent and fast replanning, where the sequential motion lacks responsiveness and adaptability. To overcome this challenge, our future work focus on transitioning from sequential to continuous motion primitives. By leveraging optimization theory, we aim to enable smoother and more flexible local path planning that better handles dynamic environments.

Appendix

A. Time Complexity Analysis

This section analyzes the time complexity of the RoA-Planner by dividing it into three main components: collision-list generation, quadtree decomposition with bridge test, and modified A*. The symbols used in the analysis are defined as follows: N denotes the total number of nodes, M is the map size, which is represented as number of pixels in the map, V represents the number of RoAs, and E is the number of valid edges that satisfy both the *area condition* and *edge condition*.

The collision-list generation performs collision checks for all pixels within the robot footprint across 360 degrees of orientation, leading to a time complexity of $O(360 \cdot w_\gamma h_\gamma \cdot N) \approx O(N)$. Fig. 9(b) shows that the collision-list generation time increases linearly with respect to the number of nodes, which aligns well with the theoretical analysis.

The quadtree with bridge test has the same time complexity as the standard quadtree. The standard quadtree has different time complexities for construction and query. The constructing process has a time complexity of $O(M)$, while range search have $O(\log M)$.

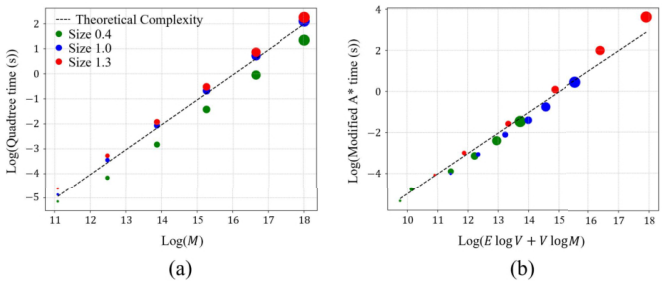


Fig. 20. Graphs (a) and (b) show log-log plots of the experimental time complexity data for quadtree construction with bridge test and modified A*, respectively. The increasing marker sizes represent the corresponding map sizes, approximately 10m, 20m, 40m, 80m, 160m and 320m. The dashed line represents the theoretical time complexity, with a slope of 1 in the log-log plot.

The time complexity of modified A* consists of two major components. First, a range search using the quadtree is performed to find neighboring RoAs for each of the V, which results in a complexity of $O(V \log M)$. Second, a graph search is conducted by inserting valid edges between neighboring RoAs into a priority queue. The complexity of this step is $O(E \log V)$. Therefore, the overall theoretical time complexity of modified A* is $O(E \log V + V \log M)$.

Note that both E and V are affected by the size of the map and the distances between obstacles. To capture this effect, we conducted experiments not only by varying the M, but also by changing the size of obstacles placed on the center of a regular 2.5m×2.5m grid. The height of the square map was exponentially increased from approximately 10m to 320m by doubling at each step, while obstacle sizes of 0.4m, 1.0m, and 1.3m were considered. For each combination of map and obstacle sizes, a quadtree was constructed, and the numbers of M, explored E and explored V were recorded. For each fixed obstacle size, both the quadtree construction with bridge test and the modified A* consistently followed the expected performance trend as the map size grew, supporting the validity of the proposed complexity model (Fig. 20).

References

[1] Y. Li, Y. Ma, X. Huo, and X. Wu, "Remote object navigation for service robots using hierarchical knowledge graph in human-centered environments," *Intell. Service Robot.*, vol. 15, no. 4, pp. 459–473, Sep. 2022.

[2] Y. Chen and Y. Lou, "A unified multiple-motion-mode framework for socially compliant navigation in dense crowds," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3536–3548, Oct. 2022.

[3] A. D. Sabiha, M. A. Kamel, E. Said, and W. M. Hussein, "Real-time path planning for autonomous vehicle based on teaching–learning-based optimization," *Intell. Service Robot.*, vol. 15, no. 3, pp. 381–398, Jul. 2022.

[4] Y. Pi et al., "OMEPP: Online multi-population evolutionary path planning for mobile manipulators in dynamic environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 6234–6245, 2025.

[5] T. Dudzik et al., "Robust autonomous navigation of a small-scale quadruped robot in real-world environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 3664–3671.

[6] M. Kennedy III, D. Thakur, M. Ani Hsieh, S. Bhattacharya, and V. Kumar, "Optimal paths for polygonal robots in SE (2)," *J. Mechanisms. Robot.*, vol. 10, no. 2, 2018, Art. no. 021005.

[7] C. Li et al., "Marden-based homotopic enclosed safe motion corridor generation for UAV navigation in complex environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 17486–17500, 2025.

[8] W. Liu, Y. Ren, and F. Zhang, "Integrated planning and control for quadrotor navigation in presence of suddenly appearing objects and disturbances," *IEEE Robot. Autom. Lett.*, vol. 9, no. 1, pp. 899–906, Jan. 2024.

[9] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "EGO-planner: An ESDF-free gradient-based local planner for quadrotors," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 478–485, Apr. 2021.

[10] Y. Wang, J. Ji, Q. Wang, C. Xu, and F. Gao, "Autonomous flights in dynamic environments with onboard vision," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 1966–1973.

[11] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, "EGO-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 4101–4107.

[12] N. T. Nguyen, P. T. Gangavarapu, N. F. Kompe, G. Schildbach, and F. Ernst, "Navigation with polytopes: A toolbox for optimal path planning with polytope maps and B-spline curves," *Sensors*, vol. 23, no. 7, p. 3532, Mar. 2023.

[13] N. Mohammad, J. Higgins, and N. Bezzo, "A GP-based robust motion planning framework for agile autonomous robot navigation and recovery in unknown environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2024, pp. 2418–2424.

[14] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 586–601, Jun. 2018.

[15] L. Wellhausen and M. Hutter, "Rough terrain navigation for legged robots using reachability planning and template learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 6914–6921.

[16] E. Jelavic, K. Qu, F. Farshidian, and M. Hutter, "LSTP: Long short-term motion planning for legged and legged-wheeled systems," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4190–4210, Dec. 2023.

[17] C. Chen, J. Frey, P. Arm, and M. Hutter, "SMUG planner: A safe multi-goal planner for mobile robots in challenging environments," *IEEE Robot. Autom. Lett.*, vol. 8, no. 11, pp. 7170–7177, Nov. 2023.

[18] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krusi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 1184–1189.

[19] D. Lee, I. M. A. Nahrendra, M. Oh, B. Yu, and H. Myung, "TRG-planner: Traversal risk graph-based path planning in unstructured environments for safe and efficient navigation," *IEEE Robot. Autom. Lett.*, vol. 10, no. 2, pp. 1736–1743, Feb. 2025.

[20] P. Wang, X. Zhou, Q. Zhao, J. Wu, and Q. Zhu, "Search-based kinodynamic motion planning for omnidirectional quadruped robots," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, Jul. 2021, pp. 823–829.

[21] Z. Zhang, J. Yan, X. Kong, G. Zhai, and Y. Liu, "Efficient motion planning based on kinodynamic model for quadruped robots following persons in confined spaces," *IEEE/ASME Trans. Mechatronics*, vol. 26, no. 4, pp. 1997–2006, Aug. 2021.

[22] Q. Liao, Z. Li, A. Thirugnanam, J. Zeng, and K. Sreenath, "Walking in narrow spaces: Safety-critical locomotion control for quadrupedal robots with duality-based optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2023, pp. 2723–2730.

[23] R. Buchanan, L. Wellhausen, M. Bjelonic, T. Bandyopadhyay, N. Kottege, and M. Hutter, "Perceptive whole-body planning for multilegged robots in confined spaces," *J. Field Robot.*, vol. 38, no. 1, pp. 68–84, Jan. 2021.

[24] R. Buchanan, T. Bandyopadhyay, M. Bjelonic, L. Wellhausen, M. Hutter, and N. Kottege, "Walking posture adaptation for legged robot navigation in confined spaces," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2148–2155, Apr. 2019.

[25] M. Gaertner, M. Bjelonic, F. Farshidian, and M. Hutter, "Collision-free MPC for legged robots in static and dynamic scenes," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 8266–8272.

[26] G. Kahn, P. Abbeel, and S. Levine, "BADGR: An autonomous self-supervised learning-based navigation system," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1312–1319, Apr. 2021.

[27] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 3948–3955.

[28] J. Guzzi, R. O. Chavez-Garcia, M. Nava, L. M. Gambardella, and A. Giusti, "Path planning with local motion estimations," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2586–2593, Apr. 2020.

[29] B. Yang, L. Wellhausen, T. Miki, M. Liu, and M. Hutter, "Real-time optimal navigation planning using learned motion costs," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 9283–9289.

[30] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, and A. Giusti, "Learning ground traversability from simulations," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1695–1702, Jul. 2018.

[31] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, "Fast traversability estimation for wild visual navigation," in *Proc. Robot. Sci. Syst.*, Jul. 2023, pp. 1–16.

[32] J. Frey, D. Hoeller, S. Khattak, and M. Hutter, "Locomotion policy guided traversability learning using volumetric representations of complex environments," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, Oct. 2022, pp. 5722–5729.

[33] Y. Kim, C. Kim, and J. Hwangbo, "Learning forward dynamics model and informed trajectory sampler for safe quadruped navigation," in *Proc. Robot. Sci. Syst.*, Jun. 2022, pp. 1–18.

[34] L. Wellhausen and M. Hutter, "ArtPlanner: Robust legged robot navigation in the field," *Field Robot.*, vol. 3, no. 1, pp. 413–434, Mar. 2023.

[35] AIDIN ROBOTICS. *AIDIN ROBOTICS Official Website*. Accessed: Sep. 27, 2025. [Online]. Available: https://www.aidinrobotics.co.kr/

[36] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Inf.*, vol. 4, no. 1, pp. 1–9, 1974.

[37] J. Zhong and J. Su, "Robot path planning in narrow passages based on probabilistic roadmaps," *Int. J. Robot. Autom.*, vol. 28, no. 3, pp. 1–11, 2013.

[38] ANYbotics. *ANYmal Quadrupedal Robot*. Accessed: Sep. 27, 2025. [Online]. Available: https://www.anybotics.com/robotics/anymal/

[39] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3019–3026, Oct. 2018.

[40] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5135–5142.

Yeongwoo Son received the B.S. degree in robotics engineering from Kwangwoon University, Seoul, South Korea, in 2022. He is currently pursuing the Ph.D. degree in robotics innovatory with the School of Mechanical Engineering, Sungkyunkwan University, Suwon, South Korea. His research interests include legged robots and motion planning.

Hyunyong Lee received the B.S. degree in mechanical engineering from Sungkyunkwan University, Suwon, South Korea, in 2016, where he is currently pursuing the Ph.D. degree in robotics innovatory with the School of Mechanical Engineering.

He is the Co-Founder of Aidin Robotics Inc., Suwon. His research interests include complete robotics systems for mobile and legged robots.

Hansol Kang received the B.S. degree in mechanical engineering from Ajou University, Suwon, South Korea, in 2017. He is currently pursuing the Ph.D. degree in robotics innovatory with the School of Mechanical Engineering, Sungkyunkwan University, Suwon. His research interests include learning-based legged robot control.

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

Jiman Park received the B.S. degree in mechanical engineering from Sungkyunkwan University, Suwon, South Korea, in 2020, where he is currently pursuing the Ph.D. degree in robotics innovatory with the School of Mechanical Engineering. His research interests include legged robot systems, system integration, actuation systems, and optimal design and control.

Seongwon Nam received the B.S. degree in mechanical engineering from Sungkyunkwan University, Suwon, South Korea, in 2021, where he is currently pursuing the Ph.D. degree in robotics innovatory with the School of Mechanical Engineering. His research interests include legged robots and optimal control.

Jaeyoung Oh received the B.S. degree in mechanical engineering from Sungkyunkwan University, Suwon, South Korea, in 2022, where he is currently pursuing the Ph.D. degree in robotics innovatory with the School of Mechanical Engineering. His research interests include legged robots and state estimation.

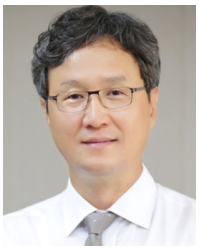
Bumsu Yi received the B.S. degree in mechanical engineering from Sungkyunkwan University, Suwon, South Korea, in 2022, where he is currently pursuing the Ph.D. degree in robotics innovatory with the School of Mechanical Engineering. His research interests include legged robots and perceptive locomotion, and motion planning.

Junha Song received the B.S. degree in mechanical engineering from Sungkyunkwan University, Suwon, South Korea, in 2023, where he is currently pursuing the M.S. degree in robotics innovatory with the School of Mechanical Engineering. His research interests include legged robots and optimal control.

Sooyeon Choi received the B.S. degree in mechanical engineering from Hanyang University, Ansan, South Korea, in 2023. She is currently pursuing the M.S. degree in robotics innovatory with the School of Mechanical Engineering, Sungkyunkwan University, Suwon, South Korea. Her research interests include legged robots, manipulators, and mechanism design.

Bogeun Kim received the B.S. degree in mechanical engineering from Sungkyunkwan University, Suwon, South Korea, in 2023, where he is currently pursuing the M.S. degree in robotics innovatory with the School of Mechanical Engineering. His research interests include quadruped robots and design.

Daegeun Song received the B.S. degree in mechanical engineering from Kyungpook National University, Daegu, South Korea, in 2022. He is currently pursuing the M.S. degree in robotics innovatory with the School of Mechanical Engineering, Sungkyunkwan University, Suwon, South Korea. His research interests include legged robots and optimal control.



Hyouk Ryeol Choi (Fellow, IEEE) received the B.S. degree from Seoul National University, Seoul, South Korea, in 1984, the M.S. degree from Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 1986, and the Ph.D. degree from Pohang University of Science and Technology, Pohang, South Korea, in 1994, all in mechanical engineering.

From 1986 to 1989, he was an Associate Research Engineer with the IT Research Center, LG Electronics. From 1993 to 1995, he was a Post-Doctoral Researcher at Kyoto University, Kyoto, Japan. From 1999 to 2000, he visited the National Institute of Advanced Industrial Science and Technology, Japan, as a JSPS Fellow. From 2008 to 2009, he was a Visiting Professor at the University of Washington, Seattle, WA, USA. Since 1995, he has been a Professor at the School of Mechanical Engineering, Sungkyunkwan University, Suwon, South Korea. His research interests include soft robotics, robotic mechanisms, field applications of robots, dexterous robotic hands, and manipulation. He was the Founding Co-Chair of the IEEE RAS Technical Committee on Robot Hand, Grasping and Manipulation. He was the General Chair of the 2012 IEEE Conference on Automation Science and Engineering (CASE), Seoul. He was an Associate Editor of IEEE TRANSACTIONS ON ROBOTICS, a Technical Editor of IEEE/ASME TRANSACTIONS ON MECHATRONICS, and the Editor-in-Chief of *Intelligent Service Robotics*.