Biomimetic Intelligence and Robotics 3 (2023) 100119

Contents lists available at ScienceDirect

Biomimetic Intelligence and Robotics

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

Research Article

Graph-based robot optimal path planning with bio-inspired algorithms

Tingjun Lei [a], Timothy Sellers [a], Chaomin Luo [a,*], Daniel W. Carruth [b], Zhuming Bi [c]

[a] Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State 39762, USA
[b] Center for Advanced Vehicular Systems, Mississippi State University, Mississippi State 39762, USA
[c] Department of Civil and Mechanical Engineering, Purdue University Fort Wayne, Fort Wayne 46805, USA

ARTICLE INFO

Article history:
Received 4 July 2023
Revised 3 August 2023
Accepted 6 August 2023
Available online 16 August 2023

Keywords:
Autonomous robot
Path planning
Bio-inspired algorithm
Graph-based model
Improved seagull optimization algorithm (iSOA)

ABSTRACT

Recently, bio-inspired algorithms have been increasingly explored for autonomous robot path planning on grid-based maps. However, these approaches endure performance degradation as problem complexity increases, often resulting in lengthy search times to find an optimal solution. This limitation is particularly critical for real-world applications like autonomous off-road vehicles, where high-quality path computation is essential for energy efficiency. To address these challenges, this paper proposes a new graph-based optimal path planning approach that leverages a sort of bio-inspired algorithm, improved seagull optimization algorithm (iSOA) for rapid path planning of autonomous robots. A modified Douglas–Peucker (mDP) algorithm is developed to approximate irregular obstacles as polygonal obstacles based on the environment image in rough terrains. The resulting mDP-derived graph is then modeled using a Maklink graph theory. By applying the iSOA approach, the trajectory of an autonomous robot in the workspace is optimized. Additionally, a Bezier-curve-based smoothing approach is developed to generate safer and smoother trajectories while adhering to curvature constraints. The proposed model is validated through simulated experiments undertaken in various real-world settings, and its performance is compared with state-of-the-art algorithms. The experimental results demonstrate that the proposed model outperforms existing approaches in terms of time cost and path length.

1. Introduction

Autonomous robots have found wide-ranging applications in various domains of our daily lives, including autonomous vehicles [1–5], medical robots [6], agriculture robots [7,8], and emergency response robots [9–11]. Robot path planning, a fundamental research area in robotics, has garnered considerable attention and research efforts over the past few decades [12–16]. The primary objective of path planning is to determine a collision-free trajectory that enables a robot to navigate from its initial position to its target position while effectively avoiding obstacles [17,18]. In this context, bio-inspired and evolutionary algorithms have emerged as powerful optimization methods, showcasing their effectiveness and success across diverse robotics areas.

Environmental modeling with robots is a crucial component when utilizing bio-inspired algorithms for autonomous robot path planning (Fig. 1). Most existing algorithms employ grid maps to facilitate path planning [19–23]. The resolution of a grid map determines the size of individual grids in the real-world environment. Higher resolutions result in more precise maps with increased grid density, yielding more accurate planning outcomes [24]. However, approaches relying on point-to-point traversal experience more expensive computational efforts as the number of grids escalates, and conversely, reduced grid numbers lead to faster planning times. While this pursuit of precision is essential, it often translates to significant time requirements when employing bio-inspired algorithms for robot path planning, which is particularly problematic in time-sensitive real-world applications like search and rescue robots in rough terrains [25–27]. Therefore, achieving rapid and high-quality path planning assumes utmost importance, aiming to save valuable time and optimize the limited onboard power resources [28].

1.1. Related work

Over the past few decades, numerous methods for autonomous robot path planning have been proposed, such as artificial potential fields [30,31], sampling-based algorithms [32,33], graph-based models [34–36], and bio-inspired algorithms [37–45].

Artificial potential fields (APF) [30,31] refer to a set of techniques that employ point-to-point traversal path planning. This

* Corresponding author.
E-mail address: Chaomin.Luo@ece.msstate.edu (C. Luo).

https://doi.org/10.1016/j.birob.2023.100119

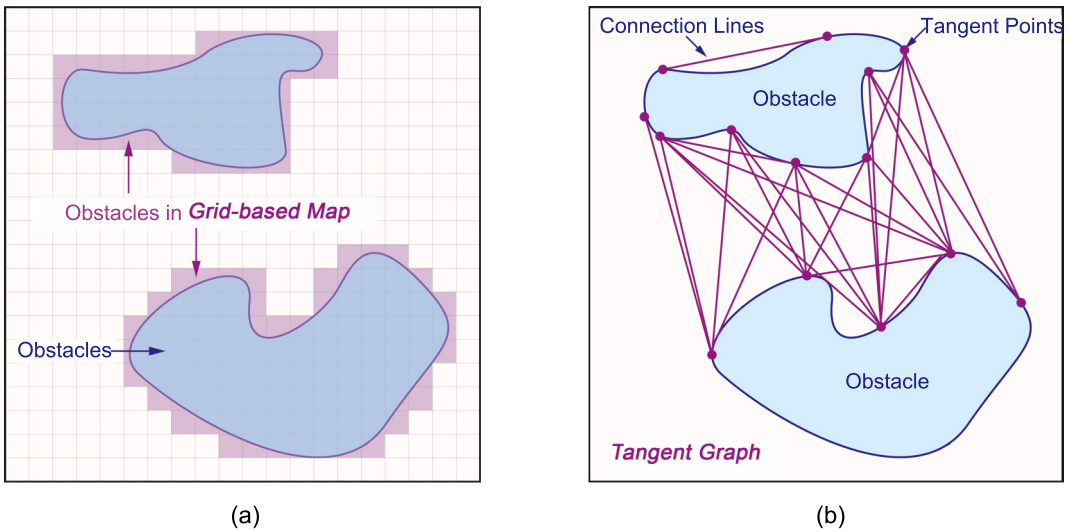

Fig. 1. Environmental modeling as a 2D map. (a) Grid-based Map. Each grid on a grid-based map corresponds to a free or occupied state (obstacle) shown by 0 or 1, respectively. (b) Tangent-based graph map. The graph's nodes stand for common tangent points found on obstacle boundaries, while its edges represent collision-free common tangents connecting the boundaries, and convex boundary segments lying between those tangent points [29].

class of methods can be categorized into two distinct types. Firstly, the object moves through a field of forces, with the target serving as the attractive pole and obstacles generating a repulsive force that reduces as the distance between them increases. Secondly, the field of forces is established by a specific gradient from the beginning point to the endpoint. Nonetheless, these algorithms may encounter a local minimum, rendering them incapable of identifying the optimal path. In addition, various solutions have been developed to tackle this issue. Szczepanski et al. [30] developed an artificial potential field driven planner by augmented reality in which the algorithm anticipates an imminent local minimum and then leverages the enhanced perception of a mobile robot to circumvent it. Due to the absence of stagnation in local minima, the proposed method allows the generation of shorter paths compared to jumping techniques. Orozco-Rosas et al. [31] proposed an improved APF method, which integrates membrane computing with genetic algorithm and APF method to find parameters to generate feasible and safe paths.

Sampling-based algorithms, such as the Rapidly Exploring Random Tree (RRT) and (PRM) algorithms, construct trees incrementally using randomly selected samples from the search space [32]. These trees typically expand towards unexplored regions. Arslan and Tsiotras [46] proposed their version of RRT that is able to ensure that the generated tree roots have the lowest cost path information for improved path planning. To improve the motion planning strategies of autonomous robot, Castro et al. [47] proposed a sampling based algorithm to calculated the control law parameters of autonomous robot to increase the level safety while still reaching the desired target. Starek et al. [48] set out to fill the gap between bi-directional motion planning and asymptotic algorithms by developing an asymptotically-optimal algorithm to help improve the knowledge space between the two methodologies. Li et al. [49] introduces Prioritized Experience Relay (PER) to improve the effectiveness of the original algorithm. However, these sampling-based algorithms still suffer from the sensitivity to sampling density and lack of guarantees. The performance of sampling-based algorithms is dependent highly on the sampling density of the configuration space. Optimal sampling density is crucial in algorithms to avoid missing important features or becoming computationally intractable. However, sampling-based algorithms lack theoretical guarantees of completeness or optimality, potentially resulting in failure to find a solution or sub-optimal solutions.

Graph-based techniques have been developed to solve the path planning issue in robotics such as Delaunay Triangulation, Voronoi Diagram and Maklink diagram. Lei et al. [50] developed a Maklink lines graph in combination with an ant colony optimization algorithm to improve the path planning capabilities. To improve the safety of autonomous robot path planning, Sellers et al. [51] proposed a generalized Voronoi diagram with an adjacent node selection algorithm to generate a trajectory focused on robot safety. Yu and LaValle [52] developed a complete algorithm and effective algorithm to solve this problem, which utilizes integer linear programming. Stahl et al. [53] proposed a multilayer graph-based trajectory planning method for race vehicles in dynamic scenarios, to improve the overall performance in autonomous vehicles. Kularatne et al. [54] developed a graph-based approach to optimal path planning in general flows, which help minimize the cost function of the algorithm for fast and more efficient path planning. Hegedűs et al. [55] proposed a low-complexity graph-based motion planning algorithm for autonomous vehicles. The algorithm is designed to be able to plan collision-free paths in environments with complex obstacles. Dang et al. [56] proposed a graph-based path planning method for subterranean exploration using aerial and legged robots. The method is designed to be able to plan collision-free paths in environments with complex obstacles, and to take into account the different capabilities of the aerial and legged robots. However, aforementioned graph-based representations become memory-intensive and computationally expensive, especially for large and complex environments. Furthermore, graph-based models may struggle to efficiently determine solutions between global and local optima.

Based on the superb optimization capabilities of bio-inspired algorithms, researchers have recently explored numerous bio-inspired approaches for robot path planning and navigation issues. Yang and Luo [37] developed a biologically inspired neural networks method of autonomous robot coverage navigation model in a non-stationary environment, which is extended to unknown workspace by concurrent mapping and navigation [24, 39]. Lei et al. [57] proposed the Bat-Pigeon Algorithm (BPA) as a solution for autonomous vehicles to adaptively regularize their speeds while navigating varying road conditions. In order to effectively find solution to optimizations problems, Dehghani and Trojovskỳ [58] proposed a natural inspired metaheurstic algorithm based on the behavioral patterns of servals. Yu et al. [59]

proposed a nature-inspired rat path planning method based on RatSLAM to provide robots with a more flexible and intelligent navigation methodology. To conquer the problem of simultaneous UAV arrival, Zhang et al. [60] proposed a time window multi-UAV path planning scheme for simultaneous arrival. A path planning method based on ant colony optimization (ACO) for dynamic speed navigation and mapping of autonomous robots is developed in Lei et al. [50]. However, many of the aforementioned bio-inspired models suffer from slow speed to integrate with real workspaces.

Another issue within previous mentioned model is their inefficiency in classifying obstacles and collision check. Path planning results are affected differently by the approximations of irregular obstacles' boundaries. For example, the grids are approximated obstacle in the grid-based environment. If the grid is too large, it is simple to generate a map that prevents the robot from passing between two closely spaced obstacles, which may allow the robot to pass. The size of the grid is related to the accuracy of obstacles and also related to calculation time. The complexity of the typical algorithm applied for grid-based environment is $O(\mathcal{N})$. $\mathcal{N}$ is number of grids in the workspace.

1.2. Proposed algorithm and original contributions

A graph-based optimal path planning approach that incorporates an improved seagull optimization algorithm is proposed in this paper. The proposed model, along with the mDP algorithm and Bezier-curve-based smoothing, offers rapid and high-quality path planning for autonomous robots in real-world scenarios. Initially, the Maklink graph-based theory is integrated with the modified Douglas–Peucker algorithm to reconstruct the free-space of the environment. The proposed method allows the environmental modeling to enclose irregular-shaped obstacles into convex polygons and reduce the number of vertices within the constructed graph. Then, we developed a bio-inspired optimization-based path planning algorithm to traverse the graph's edges and plan safe trajectories. Lastly, a path smoothing technique is applied to smooth the overall planned trajectory. The technical contributions of this paper are summarized as follows:

- The paper develops a framework for autonomous robot path planning focusing on optimizing the trajectory of autonomous robots. The framework employs a graph-based representation of the environment, which facilitates efficient path computation. By leveraging bio-inspired algorithms, the framework provides a structured and efficient approach to autonomous robot path planning.
- A new bio-inspired algorithm, namely, improved seagull optimization algorithm is developed in this paper. The iSOA algorithm enhances the efficiency of path planning by optimizing the trajectory of the autonomous robot within the workspace. By incorporating seagull behavior into the optimization process, iSOA effectively guides the robot towards finding optimal paths, leading to improved performance and navigation efficiency.
- A modified version of the Douglas–Peucker algorithm is proposed in this paper to tackle irregular obstacles present in the environment. The mDP algorithm approximates irregular obstacles as polygonal obstacles based on the environment image, enabling more accurate modeling and path planning. The resulting graph derived from the mDP algorithm is then modeled using Maklink graph theory. This provides a mathematical framework to analyze and optimize the path planning process within the graph-based representation of the environment.
- To generate safer and smoother trajectories for the autonomous robot, a Bezier-curve-based smoothing approach is developed. By considering curvature constraints, the approach ensures that the robot follows a path with controlled and optimized curvature characteristics. This smoothing technique enhances the overall navigation performance of the autonomous robot by reducing sudden changes in direction, resulting in smoother and more efficient movements.

The overall workflow of the proposed bio-inspired graph-based path planning approach is illustrated in Fig. 2. The contributions of this paper include the introduction of a new graph-based optimal path planning approach, utilization of the improved seagull optimization algorithm, development of the modified Douglas–Peucker algorithm, application of Maklink graph theory, introduction of a Bezier-curve-based smoothing approach, and validation through experiments in real-world settings. Together, these contributions offer a rapid and high-quality path planning solution for autonomous robots in real-world scenarios.

The rest of this paper is organized as follows. Section 2 presents the graph-based methodology utilized in the proposed model. Section 3 describes an improved obstacle approximation methodology to reduce the number of vertices within the graph. The proposed algorithm for robot planning is depicted in Section 4. Section 5 presents a path smoothing scheme to reduce the overall path length of the generate trajectory. Section 6 reports simulation results and comparative analyses. Several important properties of the proposed framework are summarized in Section 7.

2. Environmental modeling — Maklink graph

Spatial environmental modeling approaches significantly enhance obstacle collision checking in robot path planning. The non-obstacle spaces within the workspace are known as free spaces where the robot can move freely within the workspace, and the display of obstacles allows the robot to effectively avoid obstacles. Since modeling the robot environment affects the simplicity and computational efficiency of the path planning method, this is a fundamental issue in robot path planning. Thus, there are many modeling techniques, such as grid-based maps, vertex graphs, and generalized Voronoi graphs. However, the construction and update of the model are very complicated and required map configuration with high accuracy, which brings difficulties to practical applications.

In this section, the Maklink graph method is utilized for efficient environmental modeling or map construction in complex environments. Maklink's capacity to depict the environment as a graph, where vertices correspond to prospective robot positions and edges represent feasible trajectories between these positions, is its defining characteristic. Adaptability is a notable characteristic of the Maklink graph, as it may accommodate various types of terrain and obstacles effectively, making it versatile for real-world applications. The method's sophisticated edge-weighting scheme permits flexible modeling of diverse factors such as path length, traversal cost, and environmental constraints, enabling the construction of optimal or near-optimal paths tailored to particular robot capabilities and goals. In addition, the Maklink graph exhibits scalability, which is essential for managing complex and expansive environments while maintaining computational efficiency. Its interpretability enables the robot's path planner to acquire a thorough comprehension of the environment's underlying structure, thereby facilitating the planning of informed decisions. The visual representation of the Maklink graph facilitates the identification of critical navigation points and potential obstacles, enabling the robot to navigate efficiently,

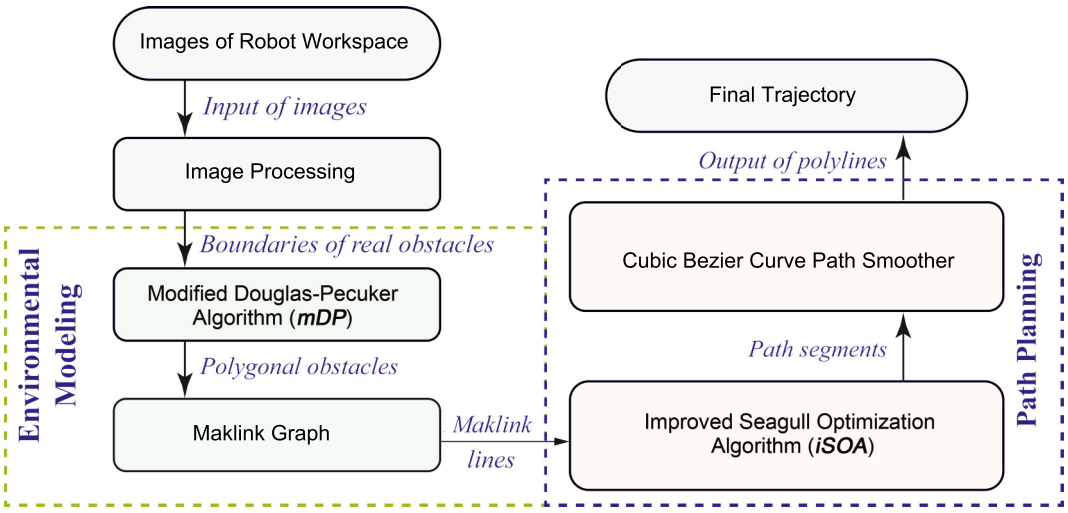

Fig. 2. Illustration of the overall workflow of the proposed bio-inspired graph-based path planning approach.

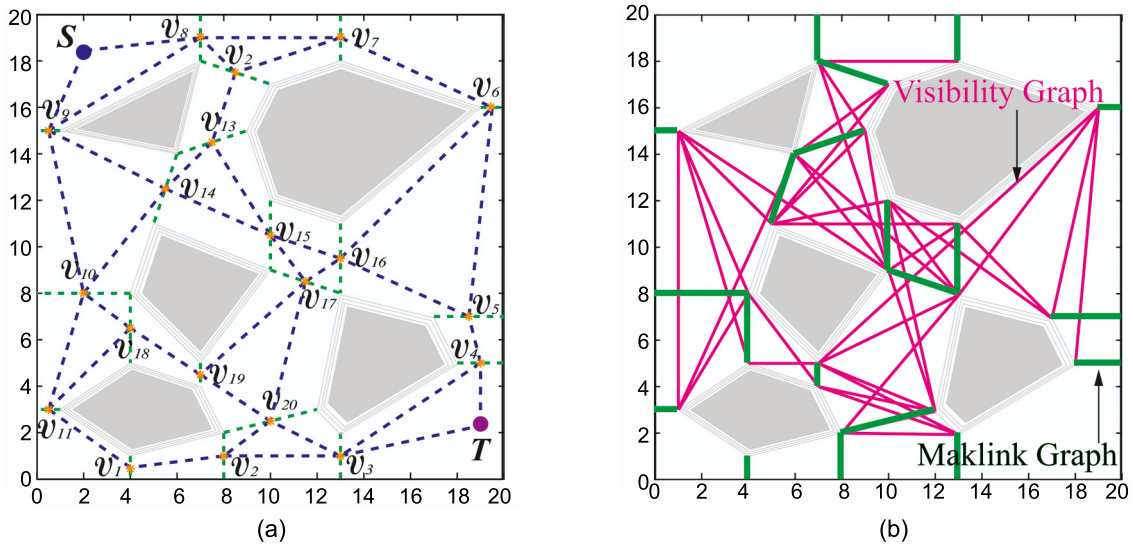

Fig. 3. Illustration of workspace based on the Maklink graph theory. (a) Dashed green lines are free Maklink lines, which connected two vertices from two different obstacles. Orange stars depict the middle points (vertices) $\mathcal{V}_n$ of the free Maklink line. Dashed blue lines are free links, which connected adjacent vertices. (b) The Maklink graph model has fewer edges than the visibility graph model which is tailored for path planning in cluttered environments with obstacles. Pink and green lines denote the visibility graph model and Maklink graph model, respectively.

avoid collisions, and reach its destination without incident. The Maklink graph applied to robot path planning provides a robust, adaptable, and scalable solution, enabling autonomous robots to effectively navigate complex environments, resulting in improved performance, safety, and dependability in real-world scenarios. Therefore, we utilize an efficient *Maklink graph* to model the robot environment, which can reduce the complexity of the model and obtain an optimized path. In order to implement the proposed path planning algorithm, three assumptions are made:

(1) The mobile robot's spatial environment is two-dimensional (2D).

(2) The workspace and obstacles are assumed to be static polygon shapes.

(3) The mobile robot is defined as a single particle, with no specific size.

Polygonal terrains, such as those found in a robot workspace, can be enclosed by a sequence of free Maklink lines. To model this robot workspace, a graph-based motion planning methodology can be employed. This graph is constructed by finding the free Maklink lines and the lines of each node on every expanded virtual obstacle, based on Maklink graph theory:

(1) Free Maklink lines are straight lines, connecting two vertices on two different obstacles, or extending from the apex of an obstacle and perpendicular to the boundary of the environment.

(2) These lines must not go through any obstacles.

Based on the definitions and assumptions outlined above, Maklink graph theory constructs the environment model in the following way:

(1) Identify all free Maklink lines in the environment, as shown by the dashed green lines in Fig. 3.

(2) Obtain the start point S, the target point T, and the middle points $\mathcal{V}_1$, $\mathcal{V}_2$, …, $\mathcal{V}_n$ of all n free Maklink lines.

(3) Connect the middle points of adjacent free Maklink lines, the starting point S to the middle points of adjacent free Maklink lines, and the target T to the middle points of adjacent free Maklink lines.

Each free link provides a collision-free trajectory for mobile robots, depicted as dashed blue lines in Fig. 3.

3. Polygonal obstacle approximation algorithm

Graph-based models offer a structured depiction of the environment, facilitating efficient exploration and navigation. By representing the environment as a graph, nodes are correspond to feasible positions or states, while edges represent valid transitions between them. This feature empowers the proposed methodology and other graph-based models to discover near-optimal solutions for various path planning scenarios. However, real-world environments often feature obstacles with complex shapes and dynamics that cannot be accurately represented by simple geometric primitives. This is where the obstacle approximation technique comes into play in this paper.

For random obstacles with irregular shapes in the real-world environment, the size of grid and the definition of obstacle filling (such as the obstacle that occupies half of the grid is regarded as the obstacle filling the entire grid) can be defined in the grid map to approximate the shape of the entire obstacle [61]. However, for some robot path planning based on graph theory, such as Maklink graph and vertical cell decomposition (VCD), these algorithms are only applied to polygonal obstacles, because of the relationship between different obstacle vertices within the workspace. In an environment with obstacles with curved boundaries, this type of path planning cannot be achieved. Even if the obstacle is approximated as a polygon, its irregular and complex shape still increase the amount of vertices, which lessens the path planning efficiency. The Douglas–Peucker (DP) algorithm is a popular polygon approximation technique used to find a similar piecewise linear curve with fewer points, given a closed curve. In order to ensure that the path generated by the DP algorithm does not collide with the actual obstacle area, a modified DP algorithm is proposed. This algorithm takes into account the fact that, while the obstacle approach result of the DP algorithm can accurately represent the boundaries of an obstacle, it may not include all of the points inside the obstacle area. Therefore, a modified algorithm is necessary in order to take into account all of the points within the obstacle area, so that the generated path may not collide with the actual obstacle area.

The proposed mDP algorithm is based on the concept of dissimilarity, which is determined by the maximum distance ε between the original curve point and its simplified piecewise linear curve. To begin, the set of digitized points of the curvilinear obstacle $\mathfrak{S}= \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N$ is considered. The line connecting two points $\mathcal{P}_a(x_a, y_a)$ and $\mathcal{P}_b(x_b, y_b)$ can be expressed as

$$x\,(y_a - y_b) + y\,(x_b - x_a) + y_b x_a - y_a x_b = 0 \tag{1}$$

Then, find two points $\mathcal{P}_i(x_i, y_i)$, $\mathcal{P}_j(x_j, y_j)$ that has the maximum distance from the list $\mathfrak{S}$. The line passing through the two points $\{\mathcal{P}_i, \mathcal{P}_j\}$ separate the curvilinear obstacle into two curves. Then the deviation d_m of a point $P\mathcal{P}_m(x_m, y_m) \in \{\mathcal{P}_i, \mathcal{P}_{i+1}, \ldots, \mathcal{P}_j\}$ from $\{\mathcal{P}_i, \mathcal{P}_j\}$ the line passing through can be defined by

$$d_m = \frac{\left|x_m\left(y_i - y_j\right) + y_m\left(x_j - x_i\right) + y_j x_i - y_i x_j\right|}{\sqrt{\left(x_j - x_i\right)^2 + \left(y_i - y_j\right)^2}} \tag{2}$$

Therefore, the point with the largest deviation $d_{\max}$ can be found, which is denoted as $\mathcal{P}_{\max}$. Then by utilizing the pairs $\{\mathcal{P}_i, \mathcal{P}_{\max}\}$ and $\{\mathcal{P}_{\max}, \mathcal{P}_j\}$, two new points can be found from the $\mathfrak{S}$ from the concept found in Eqs. (1) and (2). Obviously, as we select the newer point with the largest deviation from the passing through the pair of points, the maximum deviation continues to decrease. This process can be repeated until the maximum deviation of the point contained in its corresponding edge segment is less than the predefined maximum dissimilarity tolerance threshold value ε. It should be noted that there is another deviation d_n of a point $\mathcal{P}_n(x_n, y_n) \in \{\mathcal{P}_j, \mathcal{P}_{j+1}, \ldots, \mathcal{P}_N, \mathcal{P}_1, \mathcal{P}_i\}$ from the line passing through the pair $\{\mathcal{P}_i, \mathcal{P}_j\}$, which follows the same procedure. Finally, all the selected points are stored in the list $\mathfrak{V}$.

The proposed mDP Algorithm sequence is as shown in Fig. 4(a). In this case, since the deviation d_m and d_n is greater than or equal to ε value, new line segments are generated by connecting $(\mathcal{P}_i, \mathcal{P}_m)$, $(\mathcal{P}_m, \mathcal{P}_j)$, $(\mathcal{P}_i, \mathcal{P}_n)$, and $(\mathcal{P}_n, \mathcal{P}_j)$. $\mathcal{P}_m$ and $\mathcal{P}_n$ are the points with the largest deviation d_m and d_n. Then based on the new line segments $(\mathcal{P}_i, \mathcal{P}_m)$, $(\mathcal{P}_m, \mathcal{P}_j)$, $(\mathcal{P}_i, \mathcal{P}_n)$, and $(\mathcal{P}_n, \mathcal{P}_j)$, it recursively finds new largest deviation, as d_1, d_2, d_3, and d_4 in Fig. 4(b). In this case, since the d_1, d_2, d_3, and d_4 are all smaller than ε, the list $\mathfrak{V}$ refuses to include the newly acquired points resulting from the largest deviation.

Since the line segment of the polygon initially obtained is less than maximum dissimilarity tolerance threshold value ε from the original boundary of the obstacle. By moving the line segment of the polygon toward the boundary of the obstacle a new polygon is formed, which encloses the original polygon. As shown in Fig. 4(b), since d_1, d_2, d_3, and d_4 are smaller than ε, we move line segments $(\mathcal{P}_i, \mathcal{P}_m)$, $(\mathcal{P}_m, \mathcal{P}_j)$, $(\mathcal{P}_i, \mathcal{P}_n)$, and $(\mathcal{P}_n, \mathcal{P}_j)$ toward the obstacle curve as d_1, d_2, d_3, and d_4, respectively. The intersection points can be found for extended new line segment. Therefore, as the whole obstacle can be wrapped in the new polygon. The pseudocode of the modified Douglas–Peucker algorithm for curvilinear obstacle as shown in Algorithm 1.

Algorithm 1: The proposed modified Douglas–Peucker (mDP) algorithm

Input
A list $\mathfrak{S}$ containing the digitized point of the obstacle.
Threshold value ε for maximum dissimilarity tolerance.
Output
A list of points $\mathfrak{V}$ represents the final result of polygonal approximation of the obstacle.
Step 1: Line
$\mathfrak{V}$={}; `// Create the polygonal approximation point list`
Find the two points $\{\mathcal{P}_i, \mathcal{P}_j\}$ in $\mathfrak{S}$ with the maximum distance from each other;
Separate the curvilinear obstacle into two curves, C_1 and C_2
Step 2: Maximum deviation
Find deviation of points in C_1 and C_2 from the line $(\mathcal{P}_i, \mathcal{P}_j)$;
Find maximum deviation d_{max} as d_m and d_n and points $\mathcal{P}_m$ and $\mathcal{P}_n$ corresponding to d_m and d_n;
Step 3: Termination/recursion condition
if $d_{max} \leq \varepsilon$ **then**
  $\mathfrak{V} = \{\mathfrak{V}, \mathcal{P}_i, \mathcal{P}_j\}$; `// Add new point in the list`
else
  $\mathfrak{V} = \{\mathfrak{V}, \text{DP\_max}(\mathcal{P}_i, \mathcal{P}_m)\}$; `// Add new point in the list`
  $\mathfrak{V} = \{\mathfrak{V}, \text{DP\_max}(\mathcal{P}_m, \mathcal{P}_j)\}$;
  $\mathfrak{V} = \{\mathfrak{V}, \text{DP\_max}(\mathcal{P}_i, \mathcal{P}_n)\}$;
  $\mathfrak{V} = \{\mathfrak{V}, \text{DP\_max}(\mathcal{P}_n, \mathcal{P}_j)\}$;
  end

4. Proposed algorithm for robot path planning

This section describes the use of Dijkstra's algorithm combined with an iSOA method to enable robot path planning based on Maklink graph-based environmental modeling. An adjacency matrix with weights is defined in order to calculate the shortest path and a smooth scheme is used for the computation in this paper.

4.1. Initial path planning

Dijkstra's algorithm is a popular approach for finding the shortest path between a starting point S and a target point T on a

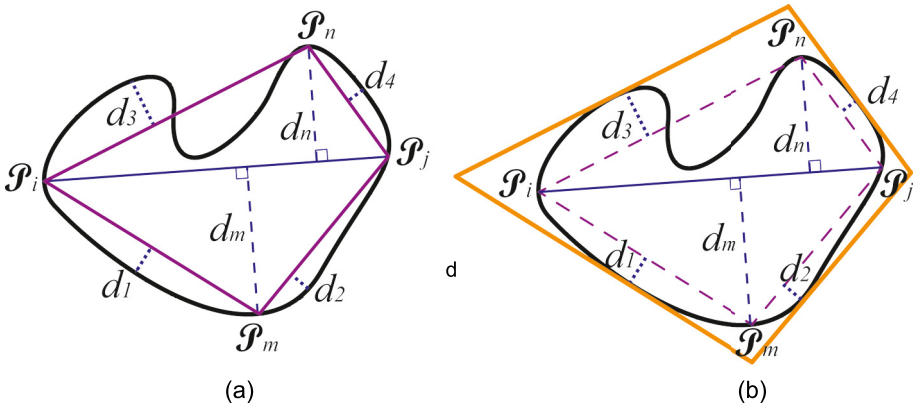

Fig. 4. Processing of the modified Douglas–Peucker Algorithm. (a) Initial polygonal approximation result. (b) Final polygonal approximation result obtained by proposed mDP algorithm. $(\mathcal{P}_i, \mathcal{P}_j)$ denotes the line segment with the maximum distance in the obstacle. According to the line $(\mathcal{P}_i, \mathcal{P}_j)$, the curvilinear obstacle can be separated into two curves, to find the largest deviation d_m and d_n of the two curves respectively.

network graph. The cost function is usually calculated as the sum of the weights of all the line segments of the path. In this case, the weight of each line segment is represented by its length. Before using the Dijkstra algorithm, an adjacency matrix must be created that includes weights for each edge between two adjacent path points v_i and v_j. This matrix can then be used to calculate the shortest path. The value of each element in the matrix represents the length of the straight-line segment between the two adjacent points. Therefore, the element at position (i, j) in the adjacency matrix can be defined as

$$\text{Adj}[i,j] = \begin{cases} length\left(v_i, v_j\right) & \text{if } v_i, v_j \text{ are adjacent} \\ \infty & \text{other} \end{cases} \tag{3}$$

Dijkstra's algorithm can be used to plan out a sub-optimal path in the Maklink graph, represented by waypoints W_0, W_1, W_2, …, W_d, and W_{d+1}, where W_0 and W_{d+1} are the starting point S and the target point T, respectively.

4.2. Improved seagull optimization algorithm

The original Seagull Optimization Algorithm (SOA) has several drawbacks, including poor solution quality, premature convergence, and a propensity to easily degenerate into a local optimum. In this section, an improved SOA is developed to overcome these drawbacks. The SOA is a bio-inspired heuristic optimization technique in light of the natural behavior of seagulls. For instance, the migration of seagulls is a seasonal movement from one place to another in search of the most abundant food sources, providing enough energy to survive in winter conditions. Studies into seagull have revealed their aggressive tendencies when migrating from one place to another, often attacking other birds that are migrating at sea in a spiral motion. The SOA focuses on the migration and attack behavior of seagulls. The mathematical descriptions of the behavior of search agents in SOA are shown as follows:

(a) Avoid collisions: To prevent collisions between adjacent agents, an additional control factor $\mathcal{A}$ is used to adjust the location of the seagull (search agent) in question. This helps to avoid any potential for collisions. The potential location for search agents that does not conflict with other search agents in the environment is denoted by $\vec{\mathcal{S}}_P = \mathcal{A} \times \vec{\mathcal{S}}_c(i), i = 0, 1, 2, \ldots, \max(i)$. the current position of the search agent is denoted by S_c, the current iteration is indicated by i, and the motion behavior of the search agent in a given search space is represented by $\mathcal{A}$.

$$\mathcal{A} = f_c - \left(i \times \left(\frac{f_c}{\max(i)}\right)\right) \tag{4}$$

where f_c denotes the frequency control interval of variable $\mathcal{A}$, which decreased linearly from f_c to 0.

(b) After avoiding collisions with other neighbors, the candidates try to move in the direction of the best neighbor (the best solution) by taking cues from their experience. The candidate search agent positions $\vec{\mathcal{S}}_c(i)$ towards the best fitting candidate search agent $\vec{\mathcal{S}}_b(i)$ are described by $\vec{d}_e = \mathcal{B} \times \left(\vec{\mathcal{S}}_b(i) - \vec{\mathcal{S}}_c(i)\right)$. The coefficient $\mathcal{B} = 2 \times \mathcal{A}^2 \times rand$ is a random value that balances exploitation and exploration. $\mathcal{B}$, where *rand* denotes the random value between 0 and 1.

(c) Finally, search agents update their position based on the best solution by utilizing the following formula in order to move toward the best solution (search agent). $\vec{\mathcal{D}}_e = \left|\vec{\mathcal{S}}_N + \vec{d}_e\right|$ describes the difference between the agents and the best cost. During migration, seagulls can adjust the angle and velocity of their attack continuously. By utilizing their weight and wings, they can remain in the air and spiral in the x, y, and z planes as $\hat{x} = r \times \cos(t); \hat{y} = r \times \sin(t); \hat{z} = r \times \theta$. The random value θ is described in the interval 0 and 2π, and the radius of the spiral turns r, is formulated as $r = \alpha \times e^{\beta\theta}$. Here, e denotes the natural logarithm base, and α and β shape the spiral. The updated position of agents is calculated in Eq. (5).

$$\vec{\mathcal{S}}_c(i+1) = \left(\vec{\mathcal{D}}_e \times \hat{x} \times \hat{y} \times \hat{z}\right) + \vec{\mathcal{S}}_b(i) \tag{5}$$

where $\vec{S}_c(i)$ keeps the best results as well as the updated positions of the search agents.

However, the original SOA has a number of limitations, such as poor solution quality, premature convergence, and a propensity to readily degenerate into a local optimum. The developed improved seagull optimization algorithm incorporated inertia weight and mutation operators in this paper enhances the balancing between the exploration and exploitation stages and increases the opportunity to reach the optimal value. Since the optimization process presents a *non-linear* declining curve, simulating the migration process of the seagull population by decreasing the additional control factor $\mathcal{A}$ linearly in Eq. (4) may reduce the actual algorithm's optimization search capability.

$$\mathcal{A} = f_c - f_c \times \left(2 \times \left(\frac{i}{\max(i)}\right) - \left(\frac{i}{\max(i)}\right)^2\right) \tag{6}$$

The values of additional variable control factor $\mathcal{A}$ decrease in a nonlinear fashion during the iSOA optimization process in Eq. (6), which can enhance the algorithm's local search capability. In contrast to the original linear control factor $\mathcal{A}$, this nonlinear

control factor facilitates the local exploitation in a greater number of iterations. It accelerates the convergence capability of the algorithm and improves the algorithm's search precision.

When using a nonlinear control factor, it is known that the precision of individuals provided after the total iteration may be reduced due to a reduction in exploration around the area of the present optimal individual. In addition, in each iteration, the previous prospective individual set will be supplanted by the new updated individual set, and only the current optimal individual will be retained for the subsequent iteration. A solitary candidate will not contribute to the organization's capacity to provide search directions. This is the reason why fundamental SOA exploration is inefficient. The new candidate individuals are generated using only the difference between the current individual and the global optimal individual, according to Eq. (5). The current population member will be attracted to and congregated around the global optimal individual. It may result in premature population convergence and limited individual precision.

The position of the personal best history is a crucial piece of information obtained by population-based meta-heuristic approaches during the iteration process, and it plays a crucial role in directing the algorithm to search the unexplored regions. Nevertheless, the fundamental SOA does not utilize the personal best history data during the iterative process. In other terms, SOA is a memory-less population-based meta-heuristic algorithm. Therefore, the formulation of SOA is modified to account for the best personal historical position and to direct the population search procedure. The proposed position update rule is expressed in Eq. (7).

$$\vec{S}_c(i+1) = \left(\left|\mathcal{A}\cdot\vec{S}_c(i)+\mathcal{B}\cdot\left(\vec{S}_b(i)-\vec{S}_c(i+1)\right)\right|\times\hat{x}\times\hat{y}\times\hat{z}\right)+\vec{S}_P(i) \quad (7)$$

The position of the new candidate search agent is determined by analyzing data pertaining to the current seagull, the global best seagull, and the candidate's own best position in the past. The proposed iSOA increases the accuracy of candidates by identifying plausible regions of the search space.

4.3. Implementation of iSOA for path planning

The initial sub-optimal path, determined by Dijkstra's algorithm in the Maklink graph, is represented by waypoints W_0, W_1, W_2, …, W_d, and W_{d+1}, with W_0 and W_{d+1} representing the starting point S and the target point T, respectively. Since these path points are the middle points of the corresponding free Maklink lines, they are not ideal for traversing open areas. Thus, the positions of the path points need to be adjusted and optimized on their free Maklink lines in order to achieve the shortest path. Let the free Maklink lines crossed by the initial path be denoted by L_i (with $i = 1, 2, \ldots, n$). If W_i^1 and W_i^2 are the two endpoints of line L_i, then the location of W_i on its free Maklink line $W_i^1W_i^2$ can be expressed as follows:

$$W_i(\gamma_i) = W_i^1 + \left(W_i^2 - W_i^1\right)\times\gamma_i, \quad i = 1, 2, \ldots, n \quad (8)$$

where γ_i is the scale factor, $\gamma_i \in [0, 1]$; n is the number of free Maklink lines.

A new robot path may be generated by a set of optimal scale factors $(\gamma_1, \gamma_2, \ldots, \gamma_n)$, thereby obtaining the optimal and shortest path. The objective function of the optimization problem can be formulated as follows:

$$L = \sum_{i=0}^{d} length\left\{W_i(\gamma_i), W_{i+1}(\gamma_{i+1})\right\} \quad (9)$$

where length $\{W_i(\gamma_i), W_{i+1}(\gamma_{i+1})\}$ denotes a straight line distance between two adjacent points W_i and W_{i+1}.

Noted that the iSOA's dimensions are the free Maklink lines traversed by the path. As shown in Fig. 5(a), $\{W_{i-1}^1, W_{i-1}^2\}$, $\{W_i^1, W_i^2\}$, and $\{W_{i+1}^1, W_{i+1}^2\}$ are three different dimensions in the Maklink graph model. W_i is both the endpoint of the Maklink line and the obstacle's vertex P in Fig. 5(b). Therefore, in order to limit the search dimension of the iSOA algorithm thereby obtaining the result promptly, it is essential to decrement the number of the polygonal obstacles' vertices in Section 3. The Maklink graph model is adopted in this paper partially due to the same ability to reduce the search dimension. For instance, if the obstacles' vertices are all connected, there are more connecting lines, such as $\{P_1, P_4\}$ in Fig. 5(b). The proposed iSOA algorithm is then utilized to optimize on the Maklink lines to determine the optimal passing position as shown by the green lines in Fig. 5(b).

Algorithm 2: Implementation of iSOA for path planning

Parameter Initialization
Initialize the parameters $\mathcal{A}$, $\mathcal{B}$, f_c, α, β and max iteration times max(i).
Population Initialization
Random initialize all search agents on the d Maklink lines.
for $i = 1 :$ max(i) **do**
  for $j = 1 : d$ **do**
    Calculate the fitness $Fit(j)$ using Equation (9);
    if $Fit(j) \le Fit_{best}$ **then**
      $Fit_{best} = Fit(j)$;
    end
  end
  $\vec{S}_b \leftarrow Fit_{best}$; // Store best candidate
  $\mathcal{A} = f_c - f_c \times \left(2\times\left(\frac{i}{\max(i)}\right) - \left(\frac{i}{\max(i)}\right)^2\right)$; // Update additional control factor
  $Rand$ = Rand(0, 1);
  $\mathcal{B} = 2\times\mathcal{A}^2\times$ rand; // Update coefficient
  $\vec{D}_e = |\vec{S}_N + \vec{d}_e|$; // Update difference
  θ = Rand(0, 2π);
  $r = \alpha \times e^{\beta\theta}$;
  $\mathcal{S} = \hat{x}\times\hat{y}\times\hat{z}$;
  $\vec{S}_c(i+1) = \left(\left|\mathcal{A}\cdot\vec{S}_c(i)+\mathcal{B}\cdot\left(\vec{S}_b(i)-\vec{S}_c(i+1)\right)\right|\times\mathcal{S}\right)+\vec{S}_P(i)$;
  // Update position of search agent
  $i = i + 1$;
end

The complexity of the algorithm is a key factor in assessing its performance. The computational complexity of the proposed iSOA algorithm is $\mathcal{O}\left(\text{Max}_{iteration}\times n_o\times n_p\times o_f\right)$. Here, n_o represents the number of objectives, n_p stands for the population size, and o_f refers to the objective function for the problem. The space complexity for iSOA is given as $\mathcal{O}\left(n_0\times n_p\right)$. The implementation of iSOA for path planning is summarize in Algorithm 2.

5. Cubic Bezier curve path smoother

The proposed cubic Bezier curve path smoother offers several advantages over other smoothing techniques, allowing outcomes to be readily controlled. Most significantly, the curve always stays within the convex hull of the control points. By examining the convexity, it is possible to determine whether the resulting line may intersect with any obstacles. Another benefit is the simple control of the approach angle towards the goal position. In a cubic Bezier curve, which is defined by four control points ($\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$), the approach to $\mathcal{T}_3$ is achieved from point $\mathcal{T}_2$. Thus, by adjusting the position of $\mathcal{T}_2$, it is possible to control the approach angle to $\mathcal{T}_3$. Similarly, the curve is generated by moving from $\mathcal{T}_0$ towards $\mathcal{T}_1$.

The basic Bezier curve is described in detail in Tayebi Arasteh and Kalisz [62]. Therefore, we concentrate solely on the control

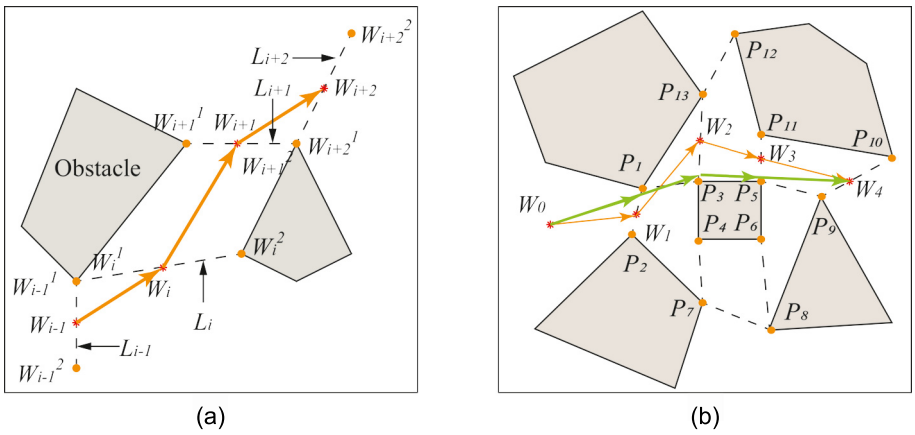

Fig. 5. Illustration of proposed iSOA path planning method in Maklink graph. (a) The initial path is determined by Dijkstra's algorithm. It determines n free Maklink lines to go through, which is the number of dimension in iSOA. (b) The final optimized path is obtained by optimizing the position passing through the Maklink line through iSOA. Dashed black lines represent free Maklink lines. The orange lines represent the initial paths obtained by the Dijkstra's algorithm. The green lines represent the final paths obtained by iSOA.

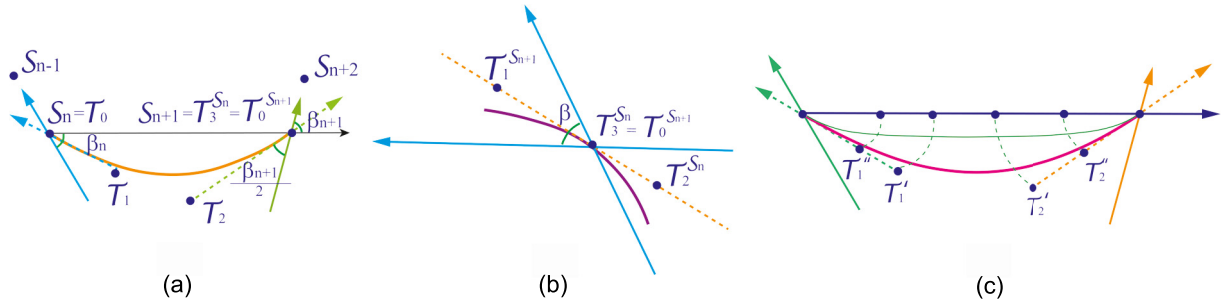

Fig. 6. Illustration of properties of the Bezier curve. (a) A Bezier curve is generated between nodes S_n and S_{n+1}. (b) The collinearity between points S_2^{Nn}, S_3^{Nn}, and S_1^{Nn+1} ensures a smooth connection between curves. (c) The influence of control points on the Bezier curve can be visualized by changing their positions along the vectors.

point design. The Bezier curve is utilized to create a path between two nodes, N_n and N_{n+1}. In this context, the initial node becomes $\mathcal{T}_0$, while the subsequent node in the sequence is designated as $\mathcal{T}_3$. We engineer the path curvature to ensure that the approach towards $\mathcal{S}_{n+1}$ is directed towards the next node $\mathcal{S}_{n+2}$, as depicted in Fig. 6(a). We determine the angle β between two vectors represented by $[\mathcal{S}_n, \mathcal{S}_{n+1}]$ and $[\mathcal{S}_{n+1}, \mathcal{S}_{n+2}]$. To achieve a smooth curve around the node $\mathcal{S}_{n+1}$, we calculate half of the angle β and construct a vector with this angle relative to the line between $\mathcal{S}_n$ and $\mathcal{S}_{n+1}$. The position of point $\mathcal{T}_2$ is then established along this vector to ensure that the curve makes half of the turn towards the subsequent node $\mathcal{S}_{n+2}$. The remaining half of the turn is controlled by properly placing the node $\mathcal{T}_1$ in the next section. The point $\mathcal{T}_1$ of the current section is positioned on a vector with half of the angle β from the previous section, which is calculated between the lines connecting points $[\mathcal{S}_{n-1}, \mathcal{S}_n]$ and $[\mathcal{S}_n, \mathcal{S}_{n+1}]$ as illustrated in Fig. 6(a). Initially, $\mathcal{T}_1$ and $\mathcal{T}_2$ are positioned at the midpoint of the line connecting $\mathcal{S}_n$ to $\mathcal{S}_{n+1}$. Utilizing the angle information of the two vectors, we construct two rotation matrices, $\mathcal{D}_{\beta_n}$ and $\mathcal{D}_{\beta_{n-1}}$, and rotate the points $\mathcal{T}_2$ and $\mathcal{T}_1$, respectively. The rotation matrix $\mathcal{D}_\beta$ is constructed as follows:

$$\mathcal{D} = \begin{bmatrix} \cos(\frac{\beta}{2}) & -\sin(\frac{\beta}{2}) \\ \sin(\frac{\beta}{2}) & 3\cos(\frac{\beta}{2}) \end{bmatrix} \tag{10}$$

where β is the angle of β_{n-1} or the angle of β_n. However, the points are calculated as

$$\mathcal{T}_n = \mathcal{D}_\beta \times \begin{bmatrix} \mathcal{T}_{nx} \\ \mathcal{T}_{ny} \end{bmatrix} \tag{11}$$

The value of n is either 1 or 2 to determine the control point $\mathcal{T}_n$ and $\mathcal{D}_\beta$ are the rotation matrix. Placing the control points $\mathcal{T}_2^n$ and $\mathcal{T}_1^{n+1}$ on the same vector ensures collinearity and a smooth connection between the generated curves of the node sections between $\mathcal{S}_n$ to $\mathcal{S}_{n+1}$ and $\mathcal{S}_{n+1}$ to $\mathcal{S}_{n+2}$, as illustrated in Fig. 6(b). Moreover, the robot's final position can be controlled by manually inputting the angle β for the final node. This approach directs the path to approach the final position from the desired angle.

In Euclidean geometry, the shortest path between two points is a straight line connecting them. To generate a curve with the shortest arc length, it is necessary to have a curve that is as close to a straight line as possible. To achieve this, we need to position points $\mathcal{T}_1$ and $\mathcal{T}_2$ as close as possible to the starting node $\mathcal{S}_n$ and the goal node $\mathcal{S}_{n+1}$, respectively. However, since we are developing a smooth path planner with a curvature constraint, we must also consider the steepness of the curve. Thus, the goal of positioning the control points is to find the shortest distance to the nodes while ensuring that the curve's steepness does not exceed the maximum allowable limit. The curvature can be defined as the reciprocal of the radius of a circle, denoted by $\mathcal{P} = \frac{1}{\mathcal{D}}$.

To approximate the curvature, we can calculate the radius of a circle described by three consecutive points on the Bezier curve. A circle can be fitted to three equidistant points $\mathcal{T}_{a1}, \mathcal{T}_{a2}, \mathcal{T}_{a3}$ on the curve. We select the second point, $\mathcal{T}_{a2}$, as the origin of the circle, and recalculate points $\mathcal{T}_{a1}$ and $\mathcal{T}_{a3}$ as $\mathcal{G}$ and $\mathcal{Z}$, represented by: $\mathcal{G} = \mathcal{T}_{a1} - \mathcal{T}_{a2}$ and $\mathcal{Z} = \mathcal{T}_{a3} - \mathcal{T}_{a2}$. The coloration between the equation of a circle and the radius is known as: $(x - x_a)^2 + (y - y_a)^2 = x_a^2 + y_a^2 = \mathcal{D}^2$. It can be rewritten as, $2x_a x + 2y_a y = x^2 + y^2 = \mathcal{U}$.

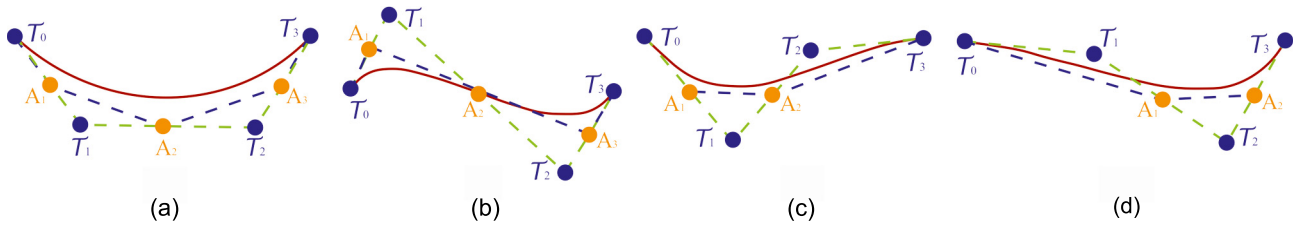

Fig. 7. Verification by polylines $\{\mathcal{T}_0, A_1, A_2, A_3, \mathcal{T}_3\}$ in (a) and (b) or $\{\mathcal{T}_0, A_1, A_2, \mathcal{T}_3\}$ in (c) and (d). The polylines allows the newly constructed path to be shorten and more smooth than the original path by moving between the ordinal line's most extreme points. This allows the curve to be formed in real-time with low computational effort.

Therefore, a system can be formed as

$$\begin{cases} x_a x_{\mathcal{G}} + y_a y_{\mathcal{G}} \\ x_a x_{\mathcal{Z}} + y_a y_{\mathcal{Z}} \end{cases} \tag{12}$$

Thus, by fitting the radius calculation, an approximate curvature of $\mathcal{P}$ at any given point of the inverse radius of a circle as

$$\mathcal{P} = \frac{1}{\sqrt{x_a^2 + y_a^2}} = 2 \times \frac{x_{\mathcal{G}} y_{\mathcal{Z}} - x_{\mathcal{Z}} y_{\mathcal{G}}}{\sqrt{(\mathcal{U}_{\mathcal{G}} x_{\mathcal{Z}} - \mathcal{U}_{\mathcal{Z}} x_{\mathcal{G}})^2 + (\mathcal{U}_{\mathcal{G}} y_{\mathcal{Z}} - \mathcal{U}_{\mathcal{Z}} y_{\mathcal{G}})^2}} \tag{13}$$

The golden section search method is used to determine the optimal positions for $\mathcal{T}_1$ and $\mathcal{T}_2$. Initially, the maximum and minimum candidate positions for each node are set at nodes $\mathcal{S}_n$ and $\mathcal{S}_{n+1}$ respectively, with the midpoint between the two serving as the starting candidate position. If the maximum curvature of a curve generated using the proposed control point is less than the threshold, the current position becomes the minimum of the section; otherwise, it becomes the maximum. This process continues until the section becomes sufficiently small, allowing for the near-optimal distance to curvature ratio to be found for each point. Fig. 6(c) provides a visualization of the process of changing the location of the control points for the creation of the Bezier curve. As the control points are moved closer to their respective nodes, the arc length of the curve decreases while the curvature increases.

In similar approaches, the convexity of control points is verified to ensure the produced trajectory. However, in our approach, as the straight line between the nodes is obstacle-free, it is not necessary to check the entire convex. Instead, we validate the polyline that bounds the curve. Since we use a cubic Bezier curve, we can create three lines between all control points. The curve is bounded by the midpoints of these lines, as depicted in Fig. 7. The bounding polyline can be represented by line segments $\{\mathcal{T}_0, A_1\}$, $\{A_1, A_2\}$, $\{A_2, A_3\}$, and $\{A_3, \mathcal{T}_3\}$.

When control points $\mathcal{T}_1$ and $\mathcal{T}_2$ are positioned beneath the curve, as shown in Figs. 7 (c) and (d), the curve is not bounded on the outer side. To address this, if $\mathcal{T}_1$ is positioned under the curve, the middle point between $\mathcal{T}_1$ and $\mathcal{T}_2$, denoted as A_1, is selected. In this case, A_3 is not selected. Similarly, if $\mathcal{T}_2$ is positioned under the curve, A_3 is not calculated and A_2 is directly connected to $\mathcal{T}_3$. The selection process for bounding polyline points is outlined in Algorithm 3.

To ensure that the generated path is obstacle-free, we verify the polylines $\{\mathcal{T}_0, A_1, A_2, \mathcal{T}_3\}$ or $\{\mathcal{T}_0, A_1, A_2, A_3, \mathcal{T}_3\}$. This method provides a more precise verification process by minimizing the chance of false obstacle detection. Unlike other approaches that check the whole convex, our proposed method only checks the polyline that bounds the curve. However, if the bounding polylines intersect with one or more obstacles, we cannot be confident that a safe path can be obtained under the given conditions. In such cases, the trajectory generation may fail. The following is a summary of the benefits of the cubic Bezier curve path smoother.

Algorithm 3: Implementation of Cubic Bezier Curve Path Smoother

Input: iSOA generated trajectory
for i = *1 : length*($\mathcal{P}_t$) **do**
  points = Path[$i : i + 2$] // Extract the path
  if *straintLine(points) == True* **then**
    Break
  else
    $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2$ = point; // Separate the point into 3 variables
    $n = 1$;
    if $\mathcal{T}_1$ *above line* $\{\mathcal{T}_0, \mathcal{T}_1\}$ **then**
      $\mathcal{A}_n = \{\frac{\mathcal{T}_{0x}+\mathcal{T}_{1x}}{2}; \frac{\mathcal{T}_{0y}+\mathcal{T}_{1y}}{2}\}$; // Find the mid-point between $\{T_0, T_1\}$
      $n = n + 1$;
      $\mathcal{A}_n = \{\frac{\mathcal{T}_{1x}+\mathcal{T}_{2x}}{2}; \frac{\mathcal{T}_{1y}+\mathcal{T}_{2y}}{2}\}$; // Find the mid-point between $\{T_1, T_2\}$
      $n = n + 1$;
    else
      $\mathcal{A}_n = \{\frac{\mathcal{T}_{1x}+\mathcal{T}_{2x}}{2}; \frac{\mathcal{T}_{1y}+\mathcal{T}_{2y}}{2}\}$; // Find the mid-point between $\{T_1, T_2\}$
      $n = n + 1$;
    End
    Curved points = BezierCurve($(\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2), \mathcal{A}_n$);
    NewPath = [New Path, Curved Points];
  End
End

(1) The smoothed path is tangent and curvature continuity, *i.e.*, the robot has the smooth steering command, which prevents acceleration discontinuities and makes the trajectory safer for robots to follow. (2) The seamless trajectory is only impacted by the two lines at the corner. Adjustments to alternative paths have no effect on the uniform trajectory. (3) We could readily modify the flattened path based on constraints imposed by the environment or the robot.

6. Simulations and comparison studies

In this section, simulations and comparison studies are performed to validate our proposed graph-based optimal path planning model. In the first subsection, numerical experiments and comparison studies are conducted by utilizing different parameters of the proposed *mDP* algorithm, and the results are discussed and compared with original DP algorithm. In the second subsection, the proposed graph-based *iSOA* path planning is applied to the map of real-world scenarios in comparison with other state-of-the-art path planning algorithms.

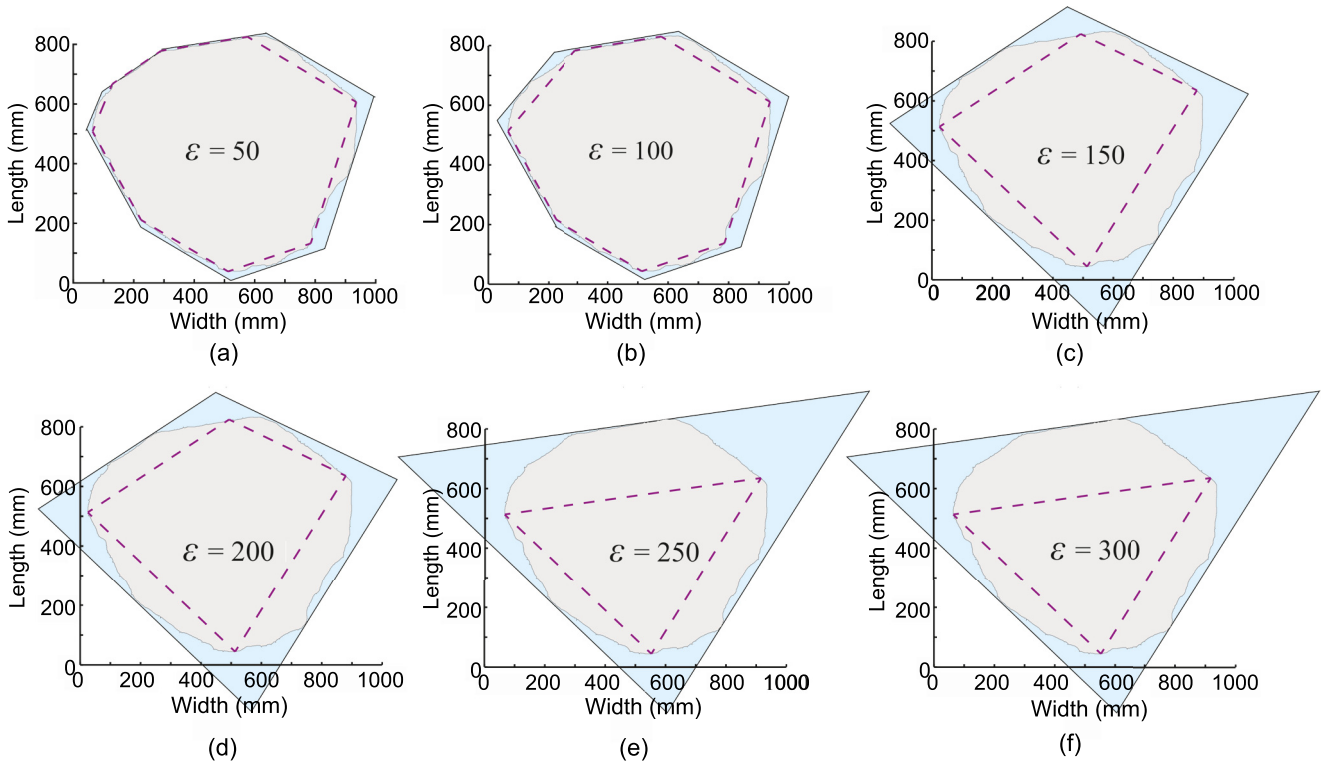

Fig. 8. Comparison of polygonal approximation result of DP and mDP Algorithms according to ε value. The gray area represents the original obstacle, the purple dash lines enclose DP results, and the light blue area represents the approximated polygonal obstacle generated by the mDP algorithm. ε denotes the maximum dissimilarity tolerance threshold value.

Table 1
Numbers of vertices and redundant area ratio of different ε in Fig. 8.

ε	Number of Vertices	Redundant Area Ratio
50	8	9.14%
100	7	10.39%
150	**4**	**29.36%**
200	**4**	**29.36%**
250	3	57.76%
300	3	57.76%

6.1. Numerical experiments and comparison studies

The fusion of graph-based models and obstacle approximation holds immense importance across diverse domains, particularly in the realm of path planning and motion control. In order to explore the significance of obstacle approximation in graph-based methodologies and examine the correlation between robot size and algorithm parameters, numerical experiments and comparison studies are first carried out to validate the proposed convex mDP algorithm for obstacle approximation and its performance in graph-based environmental modeling.

The entire obstacle is computed and then enclosed by the mDP algorithm, while it inevitably increases the area of the original obstacle. Based on a real irregularly shaped obstacle, the comparison results of DP and mDP algorithm with different maximum dissimilarity tolerance threshold value ε (with interval 50) are shown in Fig. 8. The mDP algorithm results are summarized in Table 1. As ε increases, the generated polygon vertices decease, while the redundant space occupied by the obstacle polygon increases. The reduction of the vertex angle may reduce the calculation time in the global path planning. However, increasing the extra space for obstacles may eliminate the possibility of a feasible path. Therefore, the maximum dissimilarity tolerance threshold value ε should be adjusted to balance the number of vertices and redundant area ratio, which affects computational complexity in graph theory and the optimal of the result in robot trajectory, respectively. As in Section 2, the autonomous robot in the Maklink graph is considered as a particle. To ensure that established path is not to close to the obstacles, the boundary of the polygon obstacle is expanded according to the addition of the maximum value to the distance occupied by the robot diameter with the minimum distance required for correct sensing. Therefore, in our proposed mDP algorithm, the threshold ε affects the expansion of obstacles should be proportional to the robot's size.

In the second simulation, the mDP algorithm simplifies the intricate geometry of obstacles in the real-world map into a representation that can be efficiently processed by the graph-based model. The comparison of mDP polygonal approximation results of different ε in the real-world environment is depicted in Fig. 9. It showcases the influence of the parameter ε on the execution of obstacle approximation. The interval range of ε represents the approximate polygonal obstacle stability. Therefore, this is also one of the criteria considered when we select the parameter ε of the real-world map.

When the ε is in the interval from 0.57 m to 0.61 m, an approximated polygonal obstacle is obtained with 10 vertices. The generated approximate polygonal obstacle has too many vertices, which may affect the performance of the graph-based model. When the ε is in the interval from 0.82 m to 1.92 m, an approximated polygonal obstacle is generated with 6 vertices. For graph-based models, this is a realistic polygonal obstacle that effectively obtains a balance between the number of vertices with the redundant area. The ε in this interval is also the diameter

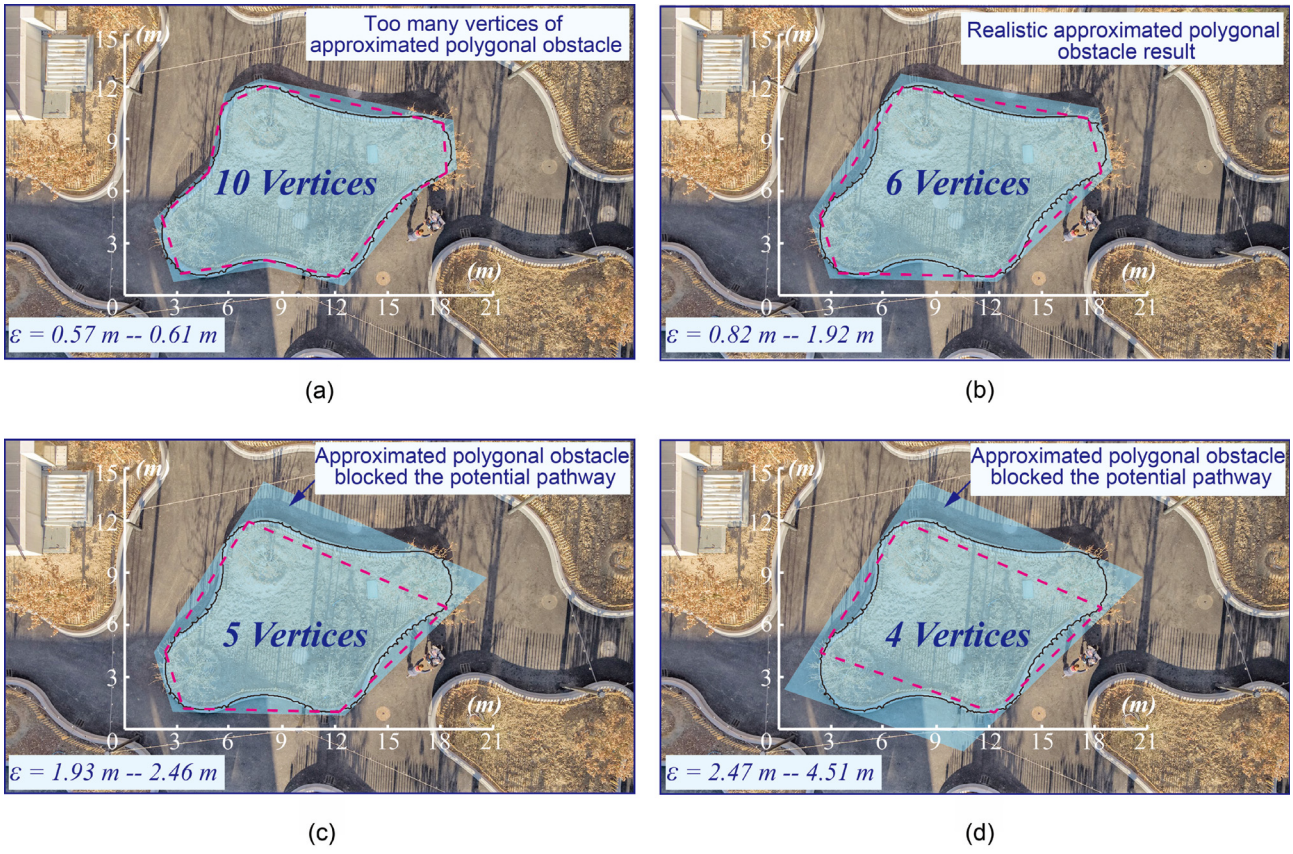

Fig. 9. Comparison of DP and mDP polygonal approximation results of different ε in real-world environment. The pink dash lines encloses DP model results and the cyan polygons are the mDP results. (a) The ε is in the interval from 0.57 m to 0.61 m, generating an approximated polygonal obstacle with 10 vertices. (b) The ε is in the interval from 0.82 m to 1.92 m, generating an approximated polygonal obstacle with 6 vertices. (c) The ε is in the interval from 1.93 m to 2.46 m, generating an approximated polygonal obstacle with 5 vertices. (d) The ε is in the interval from 2.47 m to 4.51 m, generating an approximated polygonal obstacle with 4 vertices.

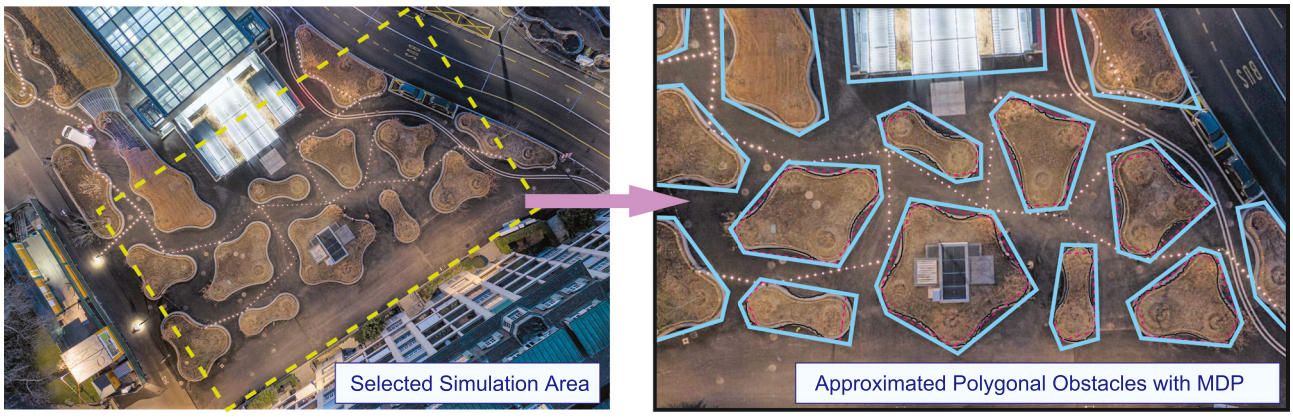

Fig. 10. The simulation in the real-world environment. (a) The selected rectangle area for simulation. The size is 70 m in width and 110 m in length. (b) The approximated polygonal obstacles with the mDP method.

of the autonomous robot that is adequate for the current environment. When the ε is in the interval from 1.93 m to 2.46 m and 2.47 m to 4.51 m, the approximated polygonal obstacle is generated with 5 vertices and 4 vertices, respectively. Their resulting approximate polygonal obstacles have too much redundant area, which may block the original potential pathway. Upon analyzing the appropriate value of ε, we observed that in most cases, an improperly tuned parameter would result in excessive redundancy in the new obstacle border, as depicted in Figs. 10(c) and (d). This results in a significant increase in the obstacle border compared to the original obstacle, thereby reducing the available space for the robot's traversal. By appropriately coupling the ε parameter with the robot's overall size, the mDP algorithm generates new obstacle boundaries without excessive redundancy within the model, as demonstrated in Table 2.

Table 2
Numbers of vertices and redundant area ratio of different ε in Fig. 9.

Number of Vertices	ε (m)	ε Interval (m)	Redundant Area Ratio
10	0.57–0.61	0.04	14.85%
9	0.62–0.72	0.10	17.02%
8	0.73–0.78	0.05	19.54%
7	0.79–0.81	0.02	21.17%
6	**0.82–1.92**	**1.10**	**24.64%**
5	1.93–2.46	0.53	30.83%
4	2.47–4.51	2.04	42.67%

6.2. Path planning under real-world application environments

In this section, three simulated experiments and comparison studies are performed in the selected real-world settings to demonstrate the feasibility and robustness of the proposed algorithms. Six state-of-art path planning algorithms, namely A*, Bi-directional A*, D*-Lite, Breadth First Search (BFS), Rapidly-exploring Random Tree (RRT), and Batch Informed Trees (BIT*) are selected for comparison studies. A* algorithm is an informed search method that efficiently finds the shortest path between two nodes in a graph by combining the cost of the path traveled so far with an estimated cost to the destination node. Bidirectional A* is a variant of the A* algorithm that simultaneously explores the graph from the start and goal nodes, meeting in the middle to find the shortest path more efficiently than A* alone. The D*-Lite path planning algorithm is an incremental search algorithm that efficiently finds the shortest path in a graph with dynamically changing costs, using a combination of dynamic programming and A* search techniques. It updates only the affected portions of the graph when costs change, minimizing the computational overhead while maintaining optimality. BFS is an uninformed search algorithm that explores all the neighbors of a node before moving on to the next level, making it suitable for finding the shortest path in an unweighted graph. RRT is a probabilistic complete algorithm that incrementally builds a tree of randomly sampled configurations, enabling efficient path planning in complex and high-dimensional spaces. BIT* is a sampling-based motion planning algorithm that incrementally builds a graph representation of the configuration space, providing efficient path planning in high-dimensional and continuous spaces. A* algorithm, Bidirectional A* algorithm, D*-Lite, BFS algorithm are based on grid maps with a grid size of 1 m $\times$ 1 m, which is a total of 110 $\times$ 70 grids in the selected area. RRT and BIT* are sampling-based model with 10,000 maximum iteration and 0.5 m step length.

The original environment with complex-shaped obstacles is illustrated in Fig. 11(a), The selected simulation area is a rectangular area with a length of 110m and a width of 70m as shown in Fig. 11(b). It also demonstrates the utilization of the proposed mDP algorithm to approximate the obstacles as polygons. Three scenarios are utilized to validate the proposed graph-based bio-inspired path planning algorithm. The starting and target points are randomly set in Scenario 1 and Scenario 2. Scenario 3 considers the impact of randomly added obstacles in the original Scenario 2 on the proposed graph-based bio-inspired path planning algorithm. It necessitates map reconstruction, unlike grid-based maps. The proposed graph-based model requires only the cancellation of the connecting lines near to the new obstacles as shown in Fig. 13(b). Thus, the bio-inspired path planning algorithm may re-executed easily, which saves a substantial amount of effort during map creation.

The simulation results of the proposed method for Scenarios 1, 2, and 3 are shown in Figs. 11(a), 12(a), and 13(a), respectively. The blue lines represent the initial path planning by Dijkstra's algorithm. The green edges are the alternative paths for iSOA selection. The purple line shows the final trajectory obtained by iSOA. Since the iSOA-obtained trajectory is a line segment comprised of points and edges, we utilize the proposed cubic Bezier path smoother to smooth the robot's movements, allowing for a smoother steering command. Figs. 11(b), 12(b), and 13(c) illustrate the various smoothed trajectories in light of the obtained IOSA results. The convergence curves in Figs. 12(c) and 13(d) show the iSOA algorithms performance.

The results of comparison studies in terms of minimum path length, average path length, and average execution time are outlined in Table 3. From Table 3, it reveals that the execution times of the A* algorithm, Bi-directional A* algorithm, D*-Lite, and BFS algorithms are extremely short, which are also in line with their characteristics. However, their resulting path length is also dependent on the grid size. An optimal solution cannot be obtained based on the existing grid size. Our proposed algorithm obtains the optimal solution in Scenarios 1 and 3, and the solution in Scenario 2 is also close to the optimal result of BIT*. However, our algorithm outperforms the BIT* algorithm in terms of average path length and average execution time, which demonstrates the stability and efficiency of our proposed bio-inspired algorithm. The trajectory results and search ranges of A* algorithm, Bidirectional A* algorithm, D*-Lite, BFS algorithm, RRT, and BIT* in Scenarios 1, 2, and 3 are shown in Figs. 11(c)–(h), Figs. 12(d)–(h), and Figs. 13(e)–(h), respectively. It exhibits the search areas in space for different algorithms. In general, a smaller search space corresponds to an algorithm with more optimal space complexity. Our algorithm has a relatively limited search space, which practically corresponds to an improvement in algorithm time (Figs. 11(a), 12(a), and 13(a)).

6.3. Analyses and discussions

By incorporating mDP polygonal obstacle approximation into graph-based models, we can mitigate computational complexity, resulting in faster planning and control algorithms. This can be observed in the number of vertices and edges required to construct the graph for each obstacle. Obstacles with more intricate curves necessitate additional points along their edges to create the graph, leading to an increase in the number of vertices and edges. This, in turn, diminishes the effectiveness of the proposed model, which is why we have integrated obstacle approximation into our approach. The integration of graph-based models and obstacle approximation techniques allows for efficient and precise path planning in environments with complex obstacles. It enables the navigation of robots, autonomous vehicles, or virtual characters through cluttered spaces while considering the dynamics and constraints imposed by the obstacles. By leveraging the strengths of both approaches, this combination enhances the efficiency, scalability, and real-time performance of the proposed model. The advantages of our proposed mDP algorithm with Maklink graph-based model can be summarized as follows:

(a) Accurate Representation of Irregular Barriers: One of the primary benefits of employing the mDP algorithm lies in its ability to accurately represent irregular barriers in the environment. As the mDP algorithm simplifies the raw obstacle data while preserving essential geometry, the resulting polygonal impediments closely approximate the shape of the original barriers. This accurate representation ensures that the Maklink Graph method can effectively navigate around intricate and complex obstacles, leading to more reliable and collision-free path planning for autonomous robots.

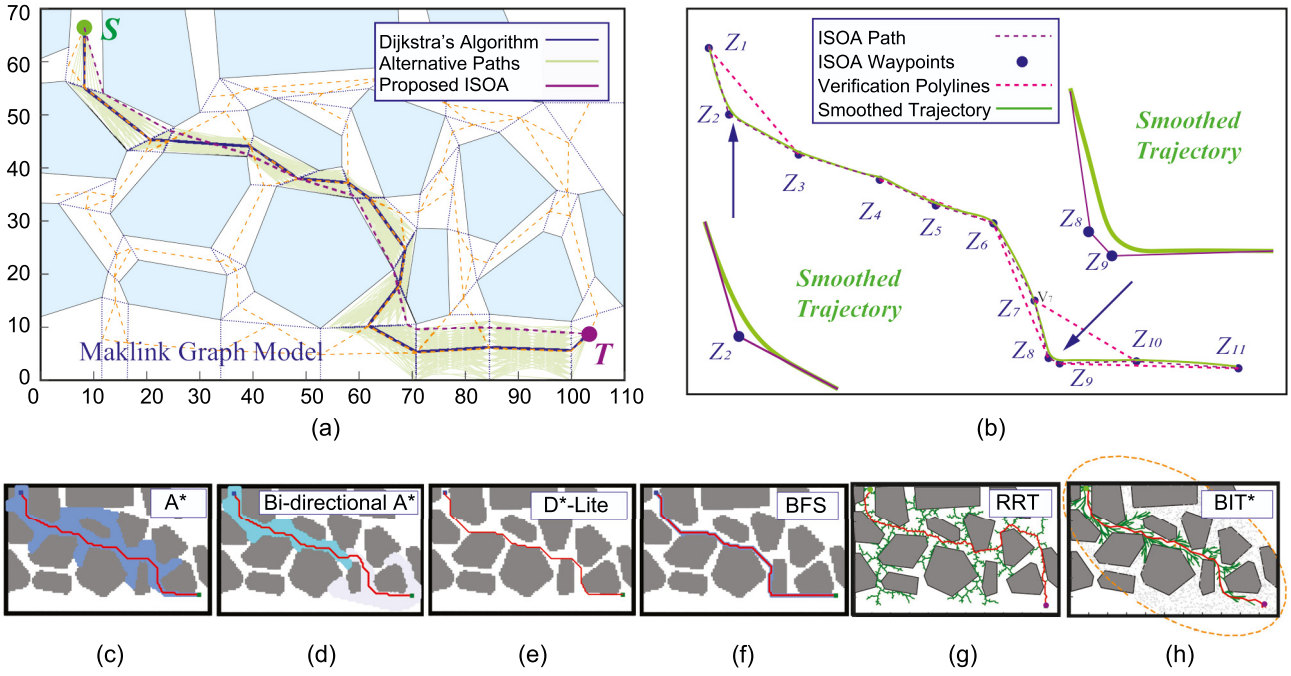

Fig. 11. The simulation and comparison results of Scenario 1 in the real-world environment. (a) The trajectory results of the proposed iSOA in the selected rectangle area. The blue and purple lines are the initial path planned by Dijkstra's algorithm and the final trajectory by the proposed iSOA, respectively. (b) The smoothed trajectory based on the cubic Bezier cure path smoother. (c)-(h) are the trajectory results and the search ranges of A* algorithm, Bidirectional A* algorithm, D*-Lite, BFS algorithm, RRT, and BIT*, respectively.

Table 3
Comparison of minimum path length, average path length, and algorithm average execution time of the compared algorithms and the proposed graph-based iSOA. The values are reported for 50 executions.

Scenario	Algorithm	Minimum Path Length (m)	Average Path Length (m)	Average Execution Time (s)
Scenario 1	A*	128.91	128.91	2.09
	Bidirectional A*	129.74	129.74	0.06
	D*-Lite	128.91	128.91	0.58
	BFS	132.67	132.67	0.02
	RRT	139.82	153.74	9.84
	BIT*	125.69	142.48	87.84
	Proposed Method	**125.16**	**127.01**	**17.19**
Scenario 2	A*	82.36	82.36	2.06
	Bidirectional A*	82.36	82.36	0.03
	D*-Lite	82.35	82.35	0.55
	BFS	82.36	82.36	0.02
	RRT	102.47	114.45	9.09
	BIT*	77.20	110.55	75.32
	Proposed Method	**79.10**	**80.24**	**14.28**
Scenario 3	A*	100.25	100.25	2.07
	Bidirectional A*	101.08	101.08	0.04
	D*-Lite	100.25	100.25	0.65
	BFS	107.57	107.57	0.02
	RRT	116.21	124.87	9.47
	BIT*	96.53	117.54	96.86
	Proposed Method	**95.01**	**96.34**	**10.03**

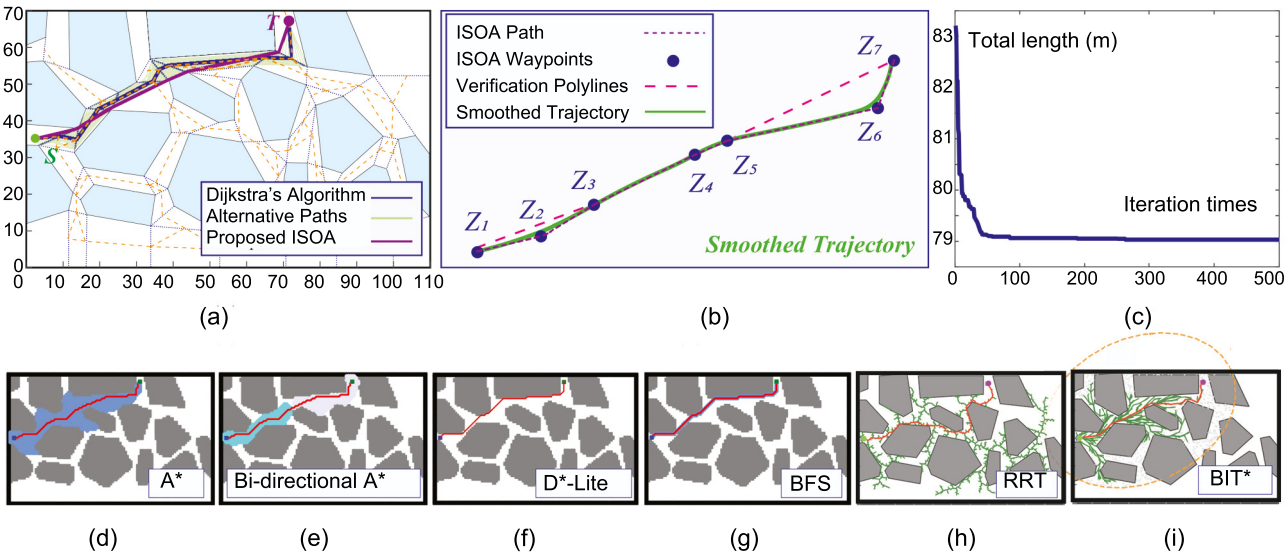

Fig. 12. The simulation and comparison results of Scenario 2 in the real-world environment. (a) The trajectory results of the proposed iSOA in the selected rectangle area. The blue and purple lines are the initial path planned by Dijkstra's algorithm and the final trajectory by the proposed iSOA, respectively. (b) The smoothed trajectory based on the cubic Bezier cure path smoother. (c) The convergence curve shows the iSOA algorithm performance. (d)-(i) are the trajectory results and the search ranges of A* algorithm, Bidirectional A* algorithm, D*-Lite, BFS algorithm, RRT, and BIT*, respectively.

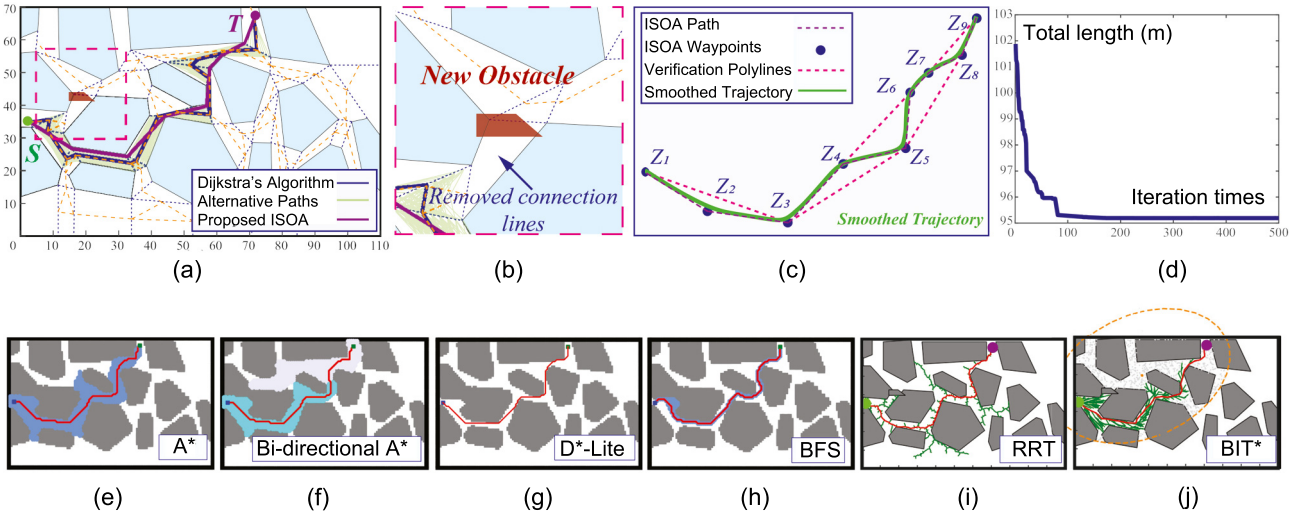

Fig. 13. The simulation and comparison results of Scenario 3 in the real-world environment. The same starting and target points are in Scenario 2. A new obstacle is discovered and updated on the map. (a) The trajectory results of the proposed iSOA in the selected rectangle area with the newly observed obstacle. The blue and purple lines are the initial path planned by Dijkstra's algorithm and the final trajectory by the proposed iSOA, respectively. (b) Updated changes to Maklink graph model for the new obstacle. (c) The smoothed trajectory based on the cubic Bezier cure path smoother. (d) The convergence curve shows the iSOA algorithm performance. (e)-(j) are the trajectory results and the search ranges of A* algorithm, Bidirectional A* algorithm, D*-Lite, BFS algorithm, RRT, and BIT*, respectively.

(b) Adaptability to Robot Dimensions: The mDP algorithm's adaptability to the robot's physical dimensions enhances the Maklink Graph's path planning capabilities. By considering the robot's size and shape during the simplification process, the generated polygonal impediments account for the space required by the robot to maneuver safely. This feature is crucial for ensuring that the planned paths avoid collisions with the robot's own body, leading to more feasible and practical paths for real-world robot navigation.

(c) Efficient Computation and Graph Representation: The mDP algorithm's efficiency in simplifying the representation of irregular barriers contributes to a more streamlined graph-based map for the Maklink Graph method. The reduced number of vertices in the polygonal impediments leads to a less computationally intensive graph representation, enabling faster path planning and navigation for autonomous robots. This efficiency is especially beneficial when dealing with large and complex environments, where computational speed is crucial.

(d) Robustness to Noise and Variability: In real-world environments, obstacles may exhibit noise and irregularities due to sensor inaccuracies or dynamic changes. The mDP algorithm's robustness to noise ensures that the generated polygonal impediments can handle these variations effectively. By capturing the essential structure of the barriers while accommodating irregularities, the Maklink Graph method becomes more reliable in dynamic and uncertain environments, enhancing its practicality for real-world applications.

(e) Improved Exploration and Optimization: The accurate and adaptable polygonal impediments obtained through the mDP algorithm contribute to improved exploration and optimization capabilities of the Maklink Graph method. With a more precise representation of the environment, the Maklink Graph can make

informed decisions during path planning, leading to better optimization of paths, avoidance of local minima, and enhanced overall performance in finding near-optimal routes.

The proposed bio-inspired algorithm, iSOA, is adequate for graph-based model implementations. Due to the algorithm's multiple iteration limitations, its execution time cannot be compared to that of superior path planning algorithms. Nevertheless, its optimal solution outperforms these standard path planning algorithms, such as A*. In comparison to the sampling-based path planning model like BIT*, the proposed graph-based iSOA algorithm is more stable and executes with less time. Consequently, the proposed algorithm is applicable to robot path planning. Despite the promising results in simulations and comparison studies, the proposed Maklink Graph model does have some limitations. Firstly, the accuracy of the path planning heavily depends on the quality of the environmental data and the resolution of the graph representation. In scenarios with highly complex and noisy data, the path planning performance might be affected. Secondly, while the Maklink Graph addresses dynamic obstacles, there is room for improvement in handling rapidly changing environments with frequent obstacle updates. Future research should focus on enhancing the model's adaptability and efficiency in dynamic scenarios.

7. Conclusion

This paper proposes a graph-based optimal path planning approach with bio-inspired algorithms to achieve rapid path planning for autonomous robotics. The introduction of the modified Douglas–Peucker algorithm enables the approximation of irregular obstacles as polygonal obstacles based on the acquired environment image. Subsequently, the mDP-derived graph is modeled using Maklink graph theory, and the improved seagull optimization algorithm is employed for optimal path planning. Furthermore, a Bezier-curve-based approach is developed to smoothen trajectories while adhering to curvature constraints. To evaluate the effectiveness of the proposed model, extensive experiments are conducted in diverse real-world scenarios and compared against state-of-the-art algorithms. The experimental results demonstrate that the proposed model outperforms existing approaches in terms of path length and time cost, validating its superior performance. By introducing the graph-based approach, leveraging bio-inspired algorithms, and incorporating trajectory smoothing techniques, this research contributes to the advancement of rapid path planning for autonomous off-road robotics. The proposed model showcases its effectiveness and potential for enhancing the efficiency and performance of autonomous robots in real-world applications. Future research directions may involve further optimization and refinement of the proposed model, considering additional factors such as dynamic environments, real-time constraints, and its implementation on actual robots.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was partially supported by the Mississippi Space Grant Consortium under NASA EPSCoR RID grant.

References

[1] L. Wang, C. Luo, J. Cai, A variable interval rescheduling strategy for dynamic flexible job shop scheduling problem by improved genetic algorithm, J. Adv. Transp. 2017 (2017).
[2] Z. Chu, F. Wang, T. Lei, C. Luo, Path planning based on deep reinforcement learning for autonomous underwater vehicles under ocean current disturbance, IEEE Trans. Intell. Veh. (2022).
[3] C. Zhang, T. Liu, S. Song, J. Wang, M.Q.-H. Meng, Dynamic wheeled motion control of wheel-biped transformable robots, Biomim. Intell. Robotics 2 (2) (2022) 100027.
[4] S. Ortiz, W. Yu, Autonomous navigation in unknown environment using sliding mode SLAM and genetic algorithm, Intell. Robot 1 (2) (2021) 131–150.
[5] J. Wang, C. Luo, Automatic wall defect detection using an autonomous robot: A focus on data collection, in: ASCE International Conference on Computing in Civil Engineering 2019, American Society of Civil Engineers Reston, VA, 2019, pp. 312–319.
[6] W. Zhao, R. Lun, C. Gordon, et al., Liftingdoneright: A privacy-aware human motion tracking system for healthcare professionals, Int. J. Handheld Comput. Res. (IJHCR) 7 (3) (2016) 1–15.
[7] G. Li, G.D. Chesser, J.L. Purswell, C. Magee, R.S. Gates, Y. Xiong, et al., Design and development of a broiler mortality removal robot, Appl. Eng. Agric. 38 (6) (2022) 853–863.
[8] T. Lei, G. Li, C. Luo, L. Zhang, L. Liu, R. Gates, An informative planning-based multi-layer robot navigation system as applied in a poultry barn, Intell. Robotics 2 (4) (2022) 313–332.
[9] L. Wang, C. Luo, M. Li, J. Cai, Trajectory planning of an autonomous mobile robot by evolving ant colony system, Int. J. Robotics Autom. 32 (4) (2017) 1500–1515.
[10] T. Lei, C. Luo, T. Sellers, Y. Wang, L. Liu, Multitask allocation framework with spatial dislocation collision avoidance for multiple aerial robots, IEEE Trans. Aerosp. Electron. Syst. 58 (6) (2022) 5129–5140.
[11] Z. Chu, D. Zhu, C. Luo, Adaptive neural sliding mode trajectory tracking control for autonomous underwater vehicle without thrust model, in: 2017 13th IEEE Conference on Automation Science and Engineering, (CASE), IEEE, 2017, pp. 1639–1644.
[12] E. Jayaraman, T. Lei, S. Rahimi, S. Cheng, C. Luo, Immune system algorithms to environmental exploration of robot navigation and mapping, in: Advances in Swarm Intelligence: 12th International Conference, ICSI 2021, Qingdao, China, July 17–21, 2021, Proceedings, Part II 12, Springer, 2021, pp. 73–84.
[13] C. Luo, J. Gao, Y.L. Murphey, G.E. Jan, A computationally efficient neural dynamics approach to trajectory planning of an intelligent vehicle, in: 2014 International Joint Conference on Neural Networks, (IJCNN), IEEE, 2014, pp. 934–939.
[14] T. Lei, P. Chintam, C. Luo, L. Liu, G.E. Jan, A convex optimization approach to multi-robot task allocation and path planning, Sensors 23 (11) (2023) 5103.
[15] C. Luo, M.F. Anjos, A. Vannelli, Large-scale fixed-outline floorplanning design using convex optimization techniques, in: 2008 Asia and South Pacific Design Automation Conference, IEEE, 2008, pp. 198–203.
[16] Y. Yang, Q. Deng, F. Shen, J. Zhao, C. Luo, A shapelet learning method for time series classification, in: 2016 IEEE 28th International Conference on Tools with Artificial Intelligence, (ICTAI), IEEE, 2016, pp. 423–430.
[17] T. Sellers, T. Lei, G.E. Jan, Y. Wang, C. Luo, Multi-objective optimization robot navigation through a graph-driven PSO mechanism, in: Advances in Swarm Intelligence: 13th International Conference, ICSI 2022, Xi'an, China, July 15–19, 2022, Proceedings, Part II, Springer, 2022, pp. 66–77.
[18] L. Liu, C. Luo, F. Shen, Multi-agent formation control with target tracking and navigation, in: 2017 IEEE International Conference on Information and Automation, (ICIA), IEEE, 2017, pp. 98–103.
[19] T. Sellers, T. Lei, H. Rogers, D.W. Carruth, C. Luo, Autonomous multi-robot allocation and formation control for remote sensing in environmental exploration, in: Autonomous Systems: Sensors, Processing and Security for Ground, Air, Sea, and Space Vehicles and Infrastructure 2023, Vol. 12540, SPIE, 2023, pp. 225–241.
[20] A. Koval, S. Karlsson, G. Nikolakopoulos, Experimental evaluation of autonomous map-based spot navigation in confined environments, Biomim. Intell. Robotics 2 (1) (2022) 100035.
[21] C. Luo, S.X. Yang, X. Yuan, Real-time area-covering operations with obstacle avoidance for cleaning robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 3, IEEE, 2002, pp. 2359–2364.
[22] S. Liang, B. Song, D. Xue, Landing route planning method for micro drones based on hybrid optimization algorithm, Biomim. Intell. Robotics 1 (2021) 100003.
[23] J. Chen, K. Xu, X. Ding, Adaptive gait planning for quadruped robot based on center of inertia over rough terrain, Biomim. Intell. Robotics 2 (1) (2022) 100031.

[24] T. Lei, T. Sellers, C. Luo, L. Zhang, A bio-inspired neural network approach to robot navigation and mapping with nature-inspired algorithms, in: Advances in Swarm Intelligence: 13th International Conference, ICSI 2022, Xi'an, China, July 15–19, 2022, Proceedings, Part II, Springer, 2022, pp. 3–16.
[25] T. Sellers, T. Lei, D. Carruth, C. Luo, Deep learning-based heterogeneous system for autonomous navigation, in: Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping VIII, 12539, SPIE, 2023, pp. 140–153.
[26] G.E. Jan, C. Luo, L.-P. Hung, S.-T. Shih, A computationally efficient complete area coverage algorithm for intelligent mobile robot navigation, in: 2014 International Joint Conference on Neural Networks, (IJCNN), IEEE, 2014, pp. 961–966.
[27] R. Tong, C. Qiu, Z. Wu, J. Wang, M. Tan, J. Yu, Na-CPG: A robust and stable rhythm generator for robot motion control, Biomim. Intell. Robotics 2 (4) (2022) 100075.
[28] R. Cimurs, J. Hwang, I.H. Suh, Bezier curve-based smoothing for path planner with curvature constraint, in: 2017 First IEEE International Conference on Robotic Computing, (IRC), IEEE, 2017, pp. 241–248.
[29] Z. Yao, W. Zhang, Y. Shi, M. Li, Z. Liang, Q. Huang, ReinforcedRimJump: Tangent-based shortest-path planning for two-dimensional maps, IEEE Trans. Ind. Inform. 16 (2) (2019) 949–958.
[30] R. Szczepanski, A. Bereit, T. Tarczewski, Efficient local path planning algorithm using artificial potential field supported by augmented reality, Energies 14 (20) (2021) 6642.
[31] U. Orozco-Rosas, O. Montiel, R. Sepúlveda, Mobile robot path planning using membrane evolutionary artificial potential field, Appl. Soft Comput. 77 (2019) 236–251.
[32] C. Li, F. Meng, H. Ma, J. Wang, M.Q.-H. Meng, Relevant region sampling strategy with adaptive heuristic for asymptotically optimal path planning, Biomim. Intell. Robotics 3 (3) (2023) 100113.
[33] T. Lei, P. Chintam, D.W. Carruth, G.E. Jan, C. Luo, Human-autonomy teaming-based robot informative path planning and mapping algorithms with tree search mechanism, in: 2022 IEEE 3rd International Conference on Human-Machine Systems, (ICHMS), IEEE, 2022, pp. 1–6.
[34] J. Chen, C. Luo, M. Krishnan, M. Paulik, Y. Tang, An enhanced dynamic delaunay triangulation-based path planning algorithm for autonomous mobile robot navigation, in: Intelligent Robots and Computer Vision XXVII: Algorithms and Techniques, Vol. 7539, SPIE, 2010, pp. 253–264.
[35] T. Lei, P. Chintam, C. Luo, S. Rahimi, Multi-robot directed coverage path planning in row-based environments, in: 2022 IEEE Fifth International Conference on Artificial Intelligence and Knowledge Engineering, (AIKE), IEEE, 2022, pp. 114–121.
[36] J.W. Jung, B.C. So, J.G. Kang, D.W. Lim, Y. Son, Expanded Douglas–Peucker polygonal approximation and opposite angle-based exact cell decomposition for path planning with curvilinear obstacles, Appl. Sci. 9 (4) (2019) 638.
[37] S.X. Yang, C. Luo, A neural network approach to complete coverage path planning, IEEE Trans. Syst. Man Cybern. B 34 (1) (2004) 718–724.
[38] T. Lei, C. Luo, G.E. Jan, Z. Bi, Deep learning-based complete coverage path planning with re-joint and obstacle fusion paradigm, Front. Robotics AI 9 (2022).
[39] C. Luo, S.X. Yang, X. Li, M.Q.H. Meng, Neural-dynamics-driven complete area coverage navigation through cooperation of multiple mobile robots, IEEE Trans. Ind. Electron. 64 (1) (2016) 750–760.
[40] D. Short, T. Lei, D.W. Carruth, C. Luo, Z. Bi, A bio-inspired algorithm in image-based path planning and localization using visual features and maps, Intell. Robotics 3 (2) (2023) 222–241.
[41] F. Bai, F. Meng, J. Liu, J. Wang, M.Q.-H. Meng, Hierarchical policy with deep-reinforcement learning for nonprehensile multiobject rearrangement, Biomim. Intell. Robotics 2 (3) (2022) 100047.
[42] J. Wang, W. Chen, X. Xiao, Y. Xu, C. Li, X. Jia, M.Q.H. Meng, A survey of the development of biomim. intell. robotics, Biomim. Intell. Robotics 1 (2021) 100001.
[43] Y. Ma, X. Xiao, H. Ren, M.Q.H. Meng, A review of bio-inspired needle for percutaneous interventions, Biomim. Intell. Robotics (2022) 100064.
[44] D. Zhu, C. Tian, X. Jiang, C. Luo, Multi-AUVs cooperative complete coverage path planning based on GBNN algorithm, in: 2017 29th Chinese Control and Decision Conference, (CCDC), IEEE, 2017, pp. 6761–6766.
[45] P. Wang, X. Zuo, C. Guo, R. Li, X. Zhao, C. Luo, A multiobjective genetic algorithm based hybrid recommendation approach, in: 2017 IEEE Symposium Series on Computational Intelligence, (SSCI), IEEE, 2017, pp. 1–6.
[46] O. Arslan, P. Tsiotras, Use of relaxation methods in sampling-based algorithms for optimal motion planning, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 2421–2428.
[47] L.I.R. Castro, P. Chaudhari, J. Tuumova, S. Karaman, E. Frazzoli, D. Rus, Incremental sampling-based algorithm for minimum-violation motion planning, in: 52nd IEEE Conference on Decision and Control, IEEE, 2013, pp. 3217–3224.
[48] J.A. Starek, J.V. Gomez, E. Schmerling, L. Janson, L. Moreno, M. Pavone, An asymptotically-optimal sampling-based algorithm for bi-directional motion planning, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS), IEEE, 2015, pp. 2072–2078.
[49] Y. Li, C. Li, Y. Zhang, G. Zhang, Mobile robot path planning based on improved SAC algorithm, J. Comput. Appl. 43 (2) (2023) 654.
[50] T. Lei, C. Luo, J.E. Ball, S. Rahimi, A graph-based ant-like approach to optimal path planning, in: 2020 IEEE Congress on Evolutionary Computation, (CEC), IEEE, 2020, pp. 1–6.
[51] T. Sellers, T. Lei, C. Luo, G.E. Jan, J. Ma, A node selection algorithm to graph-based multi-waypoint optimization navigation and mapping, Intell. Robotics 2 (4) (2022) 333–354.
[52] J. Yu, S.M. LaValle, Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics, IEEE Trans. Robot. 32 (5) (2016) 1163–1177.
[53] T. Stahl, A. Wischnewski, J. Betz, M. Lienkamp, Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios, in: 2019 IEEE Intelligent Transportation Systems Conference, (ITSC), IEEE, 2019, pp. 3149–3154.
[54] D. Kularatne, S. Bhattacharya, M.A. Hsieh, Going with the flow: A graph based approach to optimal path planning in general flows, Auton. Robots 42 (2018) 1369–1387.
[55] T. Hegedűs, B. Németh, P. Gáspár, Design of a low-complexity graph-based motion-planning algorithm for autonomous vehicles, Appl. Sci. 10 (21) (2020) 7716.
[56] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, M. Hutter, Graph-based subterranean exploration path planning using aerial and legged robots, J. Field Robotics 37 (8) (2020) 1363–1388.
[57] T. Lei, C. Luo, T. Sellers, S. Rahimi, A bat-pigeon algorithm to crack detection-enabled autonomous vehicle navigation and mapping, Intell. Syst. Appl. 12 (2021) 200053.
[58] M. Dehghani, P. Trojovskỳ, Serval optimization algorithm: A new bio-inspired approach for solving optimization problems, Biomimetics 7 (4) (2022) 204.
[59] S. Yu, H. Xu, C. Wu, X. Jiang, R. Sun, L. Sun, Bionic path planning fusing episodic memory based on RatSLAM, Biomimetics 8 (1) (2023) 59.
[60] C. Zhang, Y. Liu, C. Hu, Path planning with time windows for multiple UAVs based on gray wolf algorithm, Biomimetics 7 (4) (2022) 225.
[61] G.E. Jan, C. Luo, H.-T. Lin, K. Fung, Complete area coverage path-planning with arbitrary shape obstacles, J. Autom. Control Eng. 7 (2) (2019) 80–85.
[62] S. Tayebi Arasteh, A. Kalisz, Conversion between cubic Bezier curves and Catmull–Rom splines, SN Comput. Sci. 2 (2021) 1–9.