A Generalized Voronoi Diagram-Based Efficient Heuristic Path Planning Method for RRTs in Mobile Robots

Wenzheng Chi, Zhiyu Ding, Jiankun Wang, Guodong Chen, and Lining Sun

***Abstract*—The rapidly exploring random tree and its variants (RRTs) have been widely adopted as the motion planning algorithms for mobile robots. However, the trap space problem, such as mazes and S-shaped corridors, hinders their planning efficiency. In this article, we present a generalized Voronoi diagram (GVD)-based heuristic path planning algorithm to generate a heuristic path, guide the sampling process of RRTs, and further improve the motion planning efficiency of RRTs. Different from other heuristic algorithms that only work in certain environments or depend on specified parameter setting, the proposed algorithm can automatically identify the environment feature and provide a reasonable heuristic path. First, the given environment is initialized with a lightweight feature extraction from the GVD, which guarantees that any state in the free space can be connected to the feature graph without any collision. Second, to remove the redundancy of feature nodes, a feature matrix is proposed to represent connections among feature nodes and a corresponding feature node fusion technique is utilized to delete the redundant nodes. Third, based on the GVD feature matrix, a heuristic path planning algorithm is presented. This heuristic path is then used to guide the sampling process of RRTs and achieve real-time motion planning. The proposed GVD feature matrix can be also utilized to improve the efficiency of the replanning. Through a series of simulation studies and real-world implementations, it is confirmed that the proposed algorithm achieves better performance in heuristic path planning, feature extraction of free space, and real-time motion planning.**

***Index Terms*—Feature-matrix-based path planning, GVD, mobile robots, sampling-based motion planning.**

Manuscript received July 25, 2020; revised November 14, 2020, January 15, 2021, and March 10, 2021; accepted April 23, 2021. Date of publication May 13, 2021; date of current version January 7, 2022. The work of Wenzheng Chi was supported by the National Key R&D Program of China under Grant 2019YFB1310003. The work of Wenzheng Chi was supported by the NSFC under Grant 61903267. *(Corresponding authors: Jiankun Wang; Guodong Chen; Lining Sun.)*

Wenzheng Chi, Zhiyu Ding, Guodong Chen, and Lining Sun are with the Robotics and Microsystems Center, School of Mechanical and Electric Engineering, Soochow University, Suzhou 215021, China (e-mail: wzchi@suda.edu.cn; zyding1017@stu.suda.edu.cn; chenguodong@suda.edu.cn; lnsun@suda.edu.cn).

Jiankun Wang is with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: wangjk@sustech.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIE.2021.3078390.

Digital Object Identifier 10.1109/TIE.2021.3078390

I. Introduction

PATH planning is a fundamental research direction in robotics [1], which aims to find a collision-free or a time-efficient path for the robot to move from a start location to a goal location [2], [3]. It has been widely applied in different fields including but not limited to robotic surgery [4], service robots [5], autonomous vehicle [6], and modern industry [7]. Many path planning algorithms have been proposed in terms of different theories, which can be classified into three categories. The grid-based algorithms such as A* [8] can always find a resolution optimal path if it exists. These algorithms depend on the space discretization and do not perform well as the problem scale increases. The artificial potential field (APF) method [9] finds a feasible path by following the steepest decent of a potential field. However, the APF often suffers from the local minimum. The sampling-based algorithms such as probabilistic roadMap [10] and rapidly exploring random tree (RRT) [11] have gained popularity for their capability of efficiently searching the state-space. By implementing a collision-check module, the sampling-based algorithms can avoid the complex construction of the configuration space so that they are suitable to solve the high-dimensional or multiconstrained (kinematic or dynamic constraints) planning problems. However, they are usually trapped in some complex environments, such as mazes, S-shaped passages, and narrow corridors, which means they have difficulty in finding a feasible path or they can only generate a much longer path. Or even worse, they cannot plan a path within reasonable time cost, resulting in a failure.

To address these problems, we utilize the generalized Voronoi diagram (GVD) [12] to aid the sampling-based algorithm to achieve better performance. In practical applications, a global feasible path is generated first for the robot and then a partial motion planning strategy is implemented to maintain real-time movement. Therefore, the key problem is how to quickly generate a high-quality heuristic path connecting the robot and the goal even in some trapped environments. Using the GVD, we can transfer the connectedness of any two nodes to the diagram. Then, a set of feasible edges connecting the two specified nodes can be easily calculated. A heuristic path is naturally generated by traversing these edges from the robot node to the goal node. But this generated path cannot be directly applied into the robot path planning due to the robot kinematic and dynamic constraints or the influence of moving obstacles. To solve this problem, we propose to incorporate the RRT scheme to achieve real-time robot path planning. The nodes on the heuristic path are selected as the sub-goals for the RRT algorithm sequentially so as to guide the motion planning. As the robot moves in the current environment, this tree is also updated so that a heuristic path is always available whenever the goal position changes. It is noted that in the GVD construction process, we propose a feature matrix to represent connections among feature nodes and a feature node fusion technique to delete the unnecessary nodes so that the computation complexity is further reduced, which is beneficial for the real-time execution of the proposed path planning algorithm.

The rest of this article is organized as follows. In Section II, we first review the related work of the sampling-based algorithms and GVD application. The details of the proposed feature extraction and node fusion methods and feature tree construction algorithm are provided in Section IV. We conduct a series of experiments and discuss the results in Section V. Section VI concludes this article.

II. Related Work

Recently, many researchers propose different methods to solve the trap space problem of the sampling-based algorithms. Generally, many of them are based on the nonuniform sampling technique. Gammell *et al.* [13] propose the batch informed trees (BIT*) where they iteratively add a batch of nodes to the state space and use an admissible ellipsoidal heuristic to bias the sampling. Wang *et al.* [14] implement a Gaussian mixture model to adaptively change the sampling region so that a high-quality initial path can be found. Yershova *et al.* [15] introduce the dynamic-domain RRT in which the nodes far from the current tree are ignored in the sampling process. There are also other nonuniform sampling methods which can be found in sampling-based A* [16], Theta*-RRT* [17], and potentially guided bidirectional RRT* [18], among others. While the aforementioned algorithms can improve the performance in certain environments, they are not generally applicable or they require a lot of time to replan the feasible path when the environment changes.

More recently, the learning-based methods are also widely used in the sampling-based algorithms because of their good adaptability to different environments. Wang *et al.* [19] implement a convolutional neural network to predict a promising sampling region for the RRT* algorithm. Zhang *et al.* propose to learn an implicit sampling distribution for the motion planning algorithm. On the contrary, Ichter *et al.* [20] propose to learn an explicit sampling distribution using the conditional variational autoencoder. Other learning methods, including generative adversarial network [21] and recurrent neural network [22], are also incorporated with the sampling-based algorithms and good results are obtained. These learning-based methods can significantly improve the performance of the sampling-based algorithms, but the prediction result is not always correct. If a wrong prediction occurs, the performance of the whole path planning algorithm even gets worse.

However, the GVD method used in this article can always find an appropriate path linking the robot and the goal for two dimensional problems if it exists [23]. Moreover, a lot of efficient methods [23], [24] have been proposed to construct the GVD, making its wide application reliable. Kim *et al.* [25] propose the Cloud RRT* where the GVD is used to geometrically analyze the workspace, but they only pay attention to the scattered nodes and the heuristic connectivity among the nodes is neglected. Wang and Meng [26] also implement the GVD technique in the sampling-based algorithm but the application is not direct and a lot of intermediate calculation is required.

In this article, we propose a GVD feature matrix-based efficient heuristic path planning framework in the complex environment. In order to improve the efficiency of finding a heuristic path, we propose three strategies in this article. First, we use the feature nodes instead of GVD nodes themselves to represent the workspace, which can reduce the redundancy of the representation and decline the nodes on the feature tree exponentially. Second, the proposed feature matrix can quickly retrieve adjacent nodes, avoiding repeated graph search and time-consuming obstacle detection. Third, for the computationally heavy procedure, especially the GVD feature extraction process, we carry out it off-line. For one map, the feature extraction procedure needs to be executed only once, and after that, the feature nodes can be utilized for fast motion planning and replanning.

The contributions of our work are summarized as follows:

- an efficient heuristic path planning framework for mobile robots;
- a GVD feature node fusion algorithm which guarantees that any node from the free space can be connected to at least one feature node in the feature sets without any collision;
- a feature matrix-based GVD feature tree construction algorithm for efficient heuristic path planning.

III. Problem Statements

In this section, we first present the definitions used in this article and formulate the heuristic path planning problem. The general steps of an RRT motion planner include node sampling, parent node selection, collision checking, and node connection. A node is sampled randomly from the free space. Then, the nearest node on the tree is selected as the parent node, and if there are no obstacles between the sampled node and its parent node, the sampled node evolves as a new node with a steering function and then connects to the parent node. This process repeats until a feasible trajectory is found.

Let $\mathcal{X} \in \mathbb{R}^n$ be the state-space. Let $\mathcal{X}_{\text{obs}}$ be the obstacle space and $\mathcal{X}_{\text{free}} = \mathcal{X} \backslash \mathcal{X}_{\text{obs}}$ be the obstacle-free space. The trap space problem is caused by the probability of the samples generated from a feasible trap area approaching zero

$$p(x) \sim \mathcal{X}_{\text{trap}} \to 0, \text{ when } \frac{\mathcal{X}_{\text{trap}}}{\mathcal{X}_{\text{free}}} \to 0. \tag{1}$$

Herein, $\mathcal{X}_{\text{trap}} \subset \mathcal{X}_{\text{free}}$. The trap space changes with different maps. Even within the same map, the trap space changes with different initial states. Therefore, how to increase the probability of samples in the trap space for different scenarios is the key to solve this problem.

In this work, we propose to generate a heuristic path to guide the sampling of RRTs on the basis of the GVD feature matrix. The GVD of a given map is defined as the set of nodes which have the same Euclidean distance to the nearest obstacles. In other words, a GVD node f is required to satisfy the following conditions:

$$\begin{cases} x_{\text{nerast}} = \arg \min\limits_{x_i \in \mathcal{X}_{obs}} \|f - x_i\| \\ \text{num}(\text{x}_{\text{nerast}}) \geq 2 \\ f \in \mathcal{X}_{\text{free}} \end{cases} . \tag{2}$$

The distance between x_{nerast} and f is defined as the radius of f. We extract the feature node from the original GVD node set and use the feature node to represent the free space of the environment, which will be explained in Section IV. With the proposed algorithm, each grid in the free space has one corresponding feature node.

Let the initial pose and the goal pose of the robot be x_{start} and x_{goal}. The corresponding feature nodes of x_{start} and x_{goal} are denoted with f_{start} and f_{goal}, respectively. The heuristic planning problem is defined to find a path τ in the GVD feature set

$$\begin{aligned} \tau &: [0, 1] \mapsto F \\ \text{s.t. } \tau(0) &= f_{\text{start}} \\ \tau(1) &= f_{\text{goal}}. \end{aligned} \tag{3}$$

Herein, F represents the GVD feature set. In the motion planning process, $\tau(k)$ are adopted as subgoals one by one to guide the sampling of the RRT motion planner so as to improve the probability of samples in the trap space. Thus, the key problem is how to generate a heuristic path in real time.

IV. Algorithms

In this article, we propose a GVD feature matrix-based efficient path planning strategy for mobile robots in 2D grid map. A lightweight metric filter is adopted to extract the preliminary feature nodes from GVD of the map. A feature matrix structure is proposed to represent and retrieve connections among feature nodes. A node fusion algorithm is presented to reduce the redundant feature nodes and provide a concise representation for the map. Based on GVD feature matrix, an efficient heuristic path planning algorithm is presented for further motion planning guidance. The proposed feature matrix structure can facilitate both the feature fusion and the heuristic path planning process.

A. Preliminary GVD Feature Extraction

In this part, we aim to extract feature nodes to represent the feasible area in the map so as to accelerate the heuristic path planning. As shown in Fig. 1(a), A and B are two points on the obstacles that have the same distance r to the GVD node f_n. We define the feature area of f_n as a circle centered at f_n with the radius r. It is obvious that there is no obstacle inside the feature area and each grid inside the feature area can be connected to

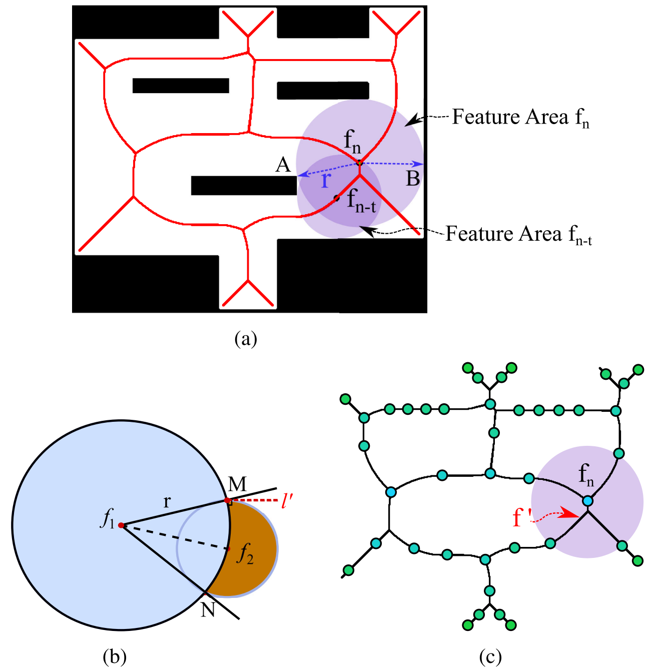

Fig. 1. A preliminary feature extraction for GVD. (a) GVD nodes and feature areas. (b) Metric filter. (c) Preliminary feature extraction.

the GVD node without any collision. Therefore, the GVD node can be utilized to represent the feature area. However, we can also find that there is a lot of overlap between the feature area of f_n and that of its surrounding GVD node, as shown with dark purple in Fig. 1(a). This representation of the feasible area still has many redundancy.

Herein, we first utilize a metric filter [27] to cluster GVD nodes inside one feature area into one feature node. It is noteworthy that the clustering starts from the GVD node with the largest GVD radius. The feature areas of the removed nodes are then connected to the feature node after clustering. As shown in Fig. 1(b), after the metric filtering, the orange area that originally belongs to f_2 will be connected to f_1. As long as f_2 is within the feature area of f_1, it is obvious that all the grids inside the feature area of f_2 can be connected to f_1 without any collision. Therefore, we can use f_1 to represent all the GVD nodes inside its feature area. The feature nodes extracted by the metric filter are illustrated in Fig. 1(c), where f_n represents the feature node and f' denotes the removed node.

B. GVD Feature Matrix

After the preliminary feature extraction, we obtain a set of feature nodes and each node has a feature index. In this part, we further remove the redundancy of the feature nodes based on a feature matrix. Herein, we propose an N-order square matrix to represent the geometric connections among feature nodes, where N is the number of feature nodes. When obtaining the set of feature nodes, for each feature node, we first extract its adjacent nodes. As shown in Fig. 2, the adjacent nodes of f_n include f_2, f_4, and f_{n-1}. Based on adjacent nodes, a feature vector is constructed for f_n. Each element in the feature vector corresponds

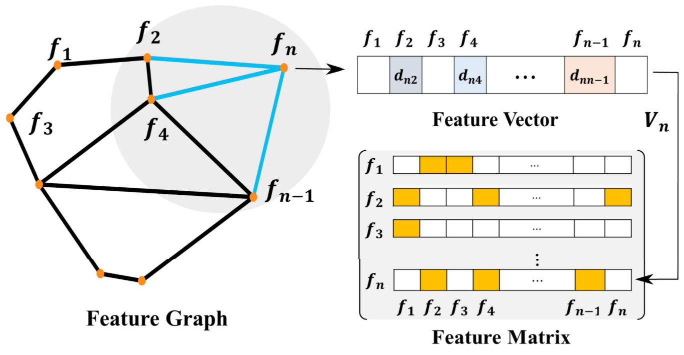

Fig. 2. Feature matrix construction with the extracted feature nodes.

to the geometric connection between f_n and the other nodes. If a node belongs to adjacent nodes of f_n, then the element with its index is set as the distance between f_n and the current node; otherwise, the element is set as zero. As shown in Fig. 2, the feature vector of f_n is $V_n = [0, d_{n2}, 0, d_{n4}, 0, \ldots, d_{nn-1}, 0]$. The feature vectors of all feature nodes constitute a feature matrix. With the feature matrix, the feature nodes are connected to form a feature graph. In order to represent the correspondence between the grids in the map and the feature nodes, we create a feature map with the same size as the original map. In the free area of the feature map, the value of the element is set as the feature index of its corresponding feature node. For the obstacle area, the value is set as zero. Therefore, each node in the free area can be connected to the feature graph by using a simply mapping.

By using the feature matrix, we can directly retrieve the neighbor nodes of arbitrary feature node without performing any collision checking operation, which is usually one of the most computation-intensive parts in the searching process. This good property of the feature matrix will help to improve the efficiency of subsequent feature fusion and heuristic path planning.

C. GVD Feature Node Fusion

In order to further remove redundant feature nodes, we propose a feature node fusion strategy. Herein, the redundant node refers to a node that deleting it does not destroy the connections among the remaining nodes and all grids it represents can be connected to its neighbor nodes. Therefore, the feature node fusion can still guarantee the representative completeness, which means that each node in the free area of the map can be connected to at least one feature node. The pseudo-code of the feature node fusion algorithm is shown in Algorithm 1.

First, we sort the feature nodes according to the length of the GVD radius, as shown from line 1 to line 3 in Algorithm 1. After the preliminary filter, the feature node with a larger GVD radius usually contains richer information while the smaller the radius, the lower the information richness it contains. Our idea is to select feature nodes around the node with larger GVD radius as potential redundant nodes. The feature matrix and the feature map are then initialized with the updated index.

Algorithm 1: GVD Feature Node Fusion.

```
Input: The preliminary feature set $F_P$, feature matrix $F_{MP}$, and feature map $M_{IP}$.
Output: Feature set $F$, feature matrix $F_M$, and feature map $M_I$.
1  $F_P.sort(radius)$
2  $F_{MP}.init(F_P)$
3  $M_{IP}.init(F_P)$
4  for all $f_i \in F_P$ do
5      $V_{open}.init(1, N)$
6      $V_i = featureVec(F_{MP}, i)$
7      $f_c = findCandidate(V_i, V_{open})$
8      while $f_c$ is not empty do
9          $V_c = featureVec(F_{MP}, c)$
10         $F_N = filterNeighbor(V_i, V_c)$
11         if collision checkings among $F_N$ succeed then
12             $A_c = featureArea(M_{IP}, c)$
13             $C_c = findCountour(A_c)$
14             if collision checkings for $C_c$ succeed then
15                 $allocateFeature(A_c)$
16                 $M_{IP}.update(A_c)$
17                 $F_{MP}.clear(c)$
18                 $F_{MP}.add(F_N)$
19         $V_{open}.clear(c)$
20         $V_i = featureVec(F_{MP}, i)$
21         $f_c = findCandidate(V_i, V_{open})$
22 return $F_P$, $F_{MP}$, and $M_{IP}$
```

After the feature node sorting, we retrieve the fusion candidate by finding the neighbor nodes of the feature node sequentially. Since the geometric connections among feature nodes have been recorded in the feature matrix, we then use the feature matrix to retrieve the neighbor nodes of a given feature node and the details are as follows. Take the ith feature node f_i as an example. As shown in line 6 of Algorithm 1, the feature vector of f_i is obtained from the ith row of the feature matrix. Herein, we define an open vector to record the checked node and the open vector is initialized as an N-dimensional vector, with all elements equal to 1, as shown in line 5 of Algorithm 1. Then, the candidate nodes can be retrieved directly by traversing the elements that larger than zero in the Hadamard product of feature vector and the open vector

$$f_c = \arg\min_{f_i \in F_P}(V_i \circ V_{\text{open}} > 0) \tag{4}$$

where the Hadamard product refers to component-wise multiplication of two vectors with same dimension. This candidate node-retrieving method can avoid traversing repeated neighbor nodes. In this part, the neighbor retrieving with the proposed feature matrix reduces the calculation complexity in the neighbor nodes searching by avoiding the time-consuming collision checking and thus improves the efficiency of the feature node fusion.

When the candidate node is found, we would like to check the connections among its neighbor nodes after its deletion and whether all grids it represents can be connected to its neighbor

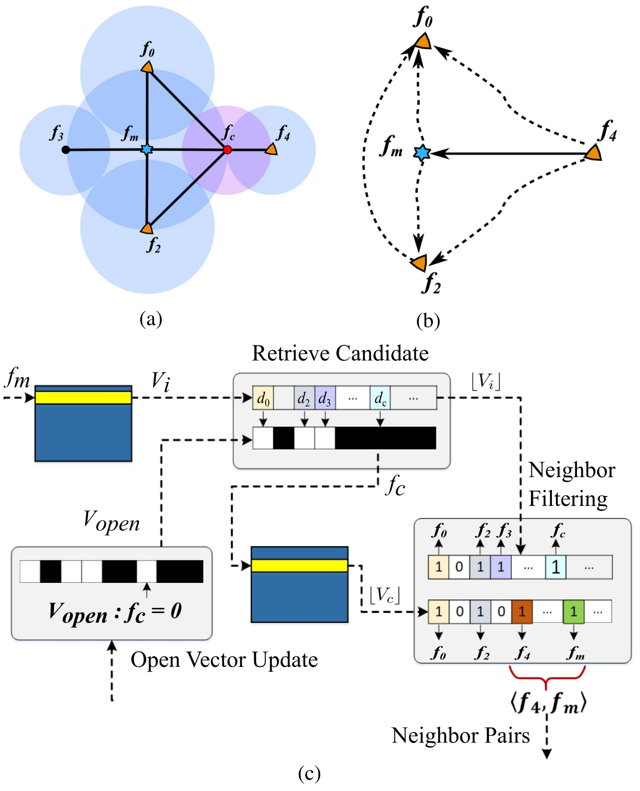

Fig. 3. GVD feature node fusion. (a) Retrieve candidate node. (b) Collision checking. (c) The system architecture of feature fusion.

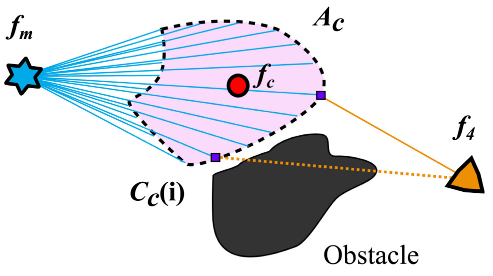

Fig. 4. Collision checking between the contour of the feature area A_c of the candidate node f_c and its neighbor nodes (f_m and f_4).

nodes. Since the connections between some neighbor nodes are already represented in the feature matrix, we can filter out these nodes and only do collision checking between the remaining nodes so as to reduce the computing cost. As shown in Fig. 3(a), the candidate node f_c has four neighbor nodes, namely f_m, f_0, f_2, and f_4. The node f_m also has four neighbor nodes, namely f_c, f_0, f_2, and f_3. To remove f_c, the feasible connections of its neighbor pairs should be guaranteed and that is (f_0, f_m), (f_2, f_m), (f_2, f_0), (f_0, f_4), (f_2, f_4), (f_m, f_4), as represented with dashed and solid lines in Fig. 3(b). However, since f_0 and f_2 are also neighbors of f_m, the connections between them do not need to be checked redundantly. In this case, we only carry out the collision checking between f_m and f_4. If the collision checking between f_m and f_4 succeeds, the connection between f_0 and f_4 can be guaranteed by an intermediate node f_m, namely $f_0 \to f_m \to f_4$. The same is true for (f_2, f_4). Then, we can guarantee that the deletion of f_c does not affect the connection among its neighbor nodes. Herein, we propose a neighbor filter based on the feature vector. The remaining neighbor nodes that require the collision checking are filtered by the following function:

$$F_N = \arg_{f_n \in F_P} (\lfloor V_c \rfloor - \lfloor V_i \rfloor) > 0 \tag{5}$$

$$\text{where } \lfloor x \rfloor = \begin{cases} 1, \text{if } x > 0 \\ 0, \text{otherwise.} \end{cases} \tag{6}$$

V_i represents the feature vector of the current node [f_m in Fig. 3(a)], and V_c represents the feature vector of the candidate node [f_c in Fig. 3(a)]. As mentioned in Section IV-B, the feature vector records the distance between two nodes. Herein, we adopt the function $\lfloor\rfloor$ to transform the feature vector to a $0-1$ vector. Then, (5) returns a vector with -1, 0, and 1. The corresponding nodes with the value 1 are the neighbor nodes of f_c that are not the neighbor nodes of f_m. As shown in Fig. 3(c), the neighbor pair (f_4, f_m) is obtained in the neighbor filtering.

If each pair of nodes in F_N can pass the collision checking, it can be considered that the deletion of the candidate node does not affect the connection of the remaining nodes. Furthermore, we should also guarantee that all grids that the candidate node represents can be connected to its neighbor nodes. As shown in Fig. 4, we denote the grids represented by the candidate node f_c as the feature area A_c, which can be retrieved directly by the feature map. To simplify the collision checking between the feature area and the neighbor nodes, we extract the contour of the feature area, namely C_c, which can also represent the feature area well. In Fig. 4, the collision checking between C_c and f_m succeeds, whereas that between C_c and f_4 fails. If the collision checking between C_c and any neighbor node (e.g. f_m in Fig. 4) succeeds, then we can reallocate the feature area to this node and update the feature map accordingly. In other words, all the grids that originally represented by f_c will be represented by f_m after the feature fusion. In the feature matrix, the corresponding row and column of f_c are then set as zero vector. The connections among F_N are then added in the feature matrix.

After each iteration, the corresponding element of the candidate node in the open vector is set as zero. The new fusion candidate is then found according to the updated feature matrix and the open vector. It is noteworthy that the feature matrix will change after each iteration and the open vector scheme can guarantee the continuous feature node fusion with the updated feature matrix.

D. Feature Matrix-Based Heuristic Path Planning

In the path planning process, feature nodes of the robot initial pose and the goal pose are first retrieved from the feature map. The feature map represents feasible areas corresponding to different feature nodes, which are illustrated with different colors, as shown in Fig. 5(a). The feature extraction should guarantee that each grid in the free space can be connected

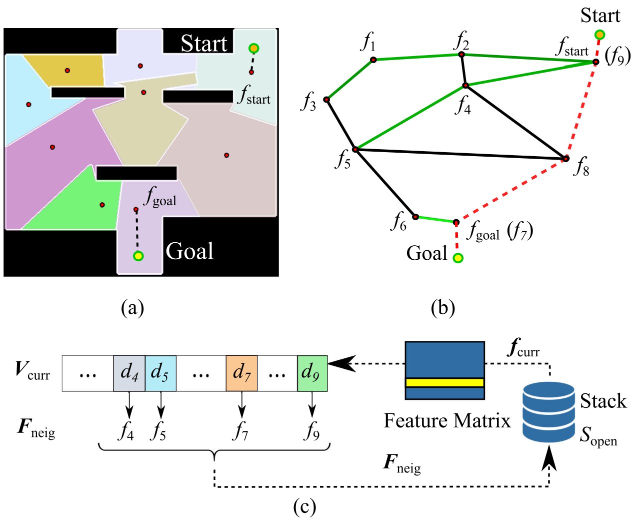

Fig. 5. Feature node searching through feature matrix. (a) retrieve feature node. (b) feature tree and heuristic path. (c) feature node searching through feature matrix.

Algorithm 2: Feature Matrix based Path Planning.

```
Input: The feature set $F$, feature matrix $F_M$, feature
       map $M_I$, start pose $x_{start}$, and goal pose $x_{end}$.
Output: A heuristic path $\tau$.
1  $f_{start} = M_I(x_{start})$
2  $f_{goal} = M_I(x_{end})$
3  $\mathcal{T}.init(F, f_{start})$
4  $\mathcal{S}_{open}.push\_back(f_{start})$
5  while $\mathcal{S}_{open}.size() > 0$ do
6      $f_{curr} = \mathcal{S}_{open}.front()$
7      $\mathcal{S}_{open}.pop()$
8      $V_{curr} = featureVec(F_M, f_{curr}.i)$
9      $F_{neig} = find(V_{curr} > 0)$
10     for all $f_{new} \in F_{neig}$ do
11         $cost_{new} = f_{curr}.cost + V_{curr}(f_{new}.i)$
12         if $cost_{new} < f_{new}.cost$ then
13             $f_{new}.cost = cost_{new}$
14             $f_{new}.parent = f_{curr}$
15             update $\mathcal{T}$
16             $\mathcal{S}_{open}.push\_back(f_{new})$
17 $\tau = \mathcal{T}(f_{goal} \rightarrow f_{start})$
18 return $\tau$
```

to at least one feature node. Once an initial pose is given, its corresponding feature node can be found immediately through a simple mapping from the feature map. The utilization of feature map allows us to directly skip the feature node searching for the initial pose and goal pose and the time-consuming obstacle detection process.

With feature nodes of the initial pose and the goal pose, namely f_{start} and f_{goal}, we construct a feature tree for the heuristic path planning. Each node f_i on the feature tree has the following information:

- the pose $pose = [x, y]$;
- the node index i;
- the cumulative cost cost;
- the parent node f_{parent}.

As illustrated in Algorithm 2, in the initialization, the feature tree is composed of all feature nodes and their original index numbers in the feature set. The cost of f_{start} is set as zero and the cost of the other feature nodes is set as positive infinity. All the parent nodes are set as null. Indeed, the feature tree construction process is to find a parent node for each feature node expect for f_{start}. Then we put f_{start} at the end of the open node stack $\mathcal{S}_{\text{open}}$, which stores the candidate nodes that need to be searched.

In the tree construction process, the front node in $\mathcal{S}_{\text{open}}$ is taken out as the current node f_{curr}. Through the index of f_{curr}, we can find its corresponding feature vector V_{curr} from the $(f_{\text{curr}}.i)$th row of the feature matrix. Then we can find the neighbor nodes of f_{curr} by extracting elements greater than zero in the feature vector and record the feature nodes corresponding to these elements as F_{neig}, as shown in Fig. 5(c).

For each feature node f_{new} in F_{neig}, the new cumulative cost from the root through f_{curr} to f_{new} is calculated as cost_{new}. If cost_{new} is less than the original cost of f_{new}, the cost of f_{new} is replaced by cost_{new} and the parent node of f_{new} is set as f_{curr}. The feature tree is updated with new connection from f_{curr} to f_{new}. f_{new} is pushed back to $\mathcal{S}_{\text{open}}$ and then one round of iteration is finished.

When there is no node in $\mathcal{S}_{\text{open}}$, the construction of the feature tree is finished. On the feature tree, each node only has one parent node. Therefore, starting from the goal node f_{goal}, by connecting the current node and its parent node till the root on the tree, we can find a feasible path, as shown in Fig. 5(b).

From Algorithm 2, we can find that the most frequent statement in this algorithm is in line 9; therefore, the complexity of the program is $\mathcal{O}(n^2)$ by using the feature matrix strategy. Since the connection relationship has been recorded in the feature matrix, no additional collision checking among the feature nodes is needed, which reduces the algorithm complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$.

E. Fast Motion Planning With RRTs

When the heuristic path planning is completed, the nodes on the generated heuristic path are taken out one by one and served as sub-goals for the motion planning with RRTs. In this work, we take the risk-RRT planner [28] for motion planning, which is a classic sampling-based motion planner for dynamic environments. The risk-RRT tree grows by using the node $\tau(k)$ on the heuristic path τ as the local target and generates the partial trajectory σ_k

$$\begin{aligned} &\sigma_k : [0, T_k] \mapsto \mathcal{X}_{\text{free}}(t) \\ \text{s.t. } &\sigma_{\text{k}}(0) = x_{\text{start}} \\ &\sigma_k(T_k) \in \mathcal{X}_{\text{free}}(\tau(k), T_k) \\ &\sigma_k(t) \in \mathcal{X}_{\text{free}}(t),\ \forall t \in [0, T_k]. \end{aligned} \tag{7}$$

Herein, we use the time t to represent changes in dynamic environment and $\mathcal{X}_{\text{free}}(\tau(k), T_k)$ denotes the free space around $\tau(k)$ at time T_k. When one subgoal is reached, the next is then

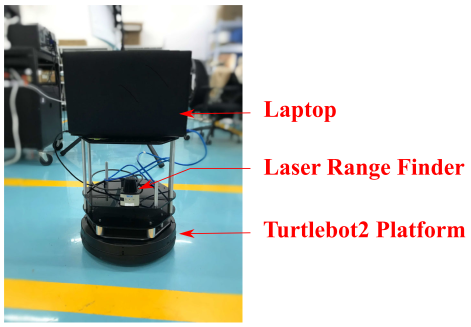

Fig. 6. Experiment platform.

chosen as a new local target. This process repeats until the risk-RRT tree reaches the final goal.

According to the sampling scheme of RRTs, the local target $\tau(k)$ will be sampled by a certain probability. Since τ is a feasible heuristic path, a bias sampling around the node on it will effectively increase the probability of samples in the trap space related to the given motion planning problem. Different from other heuristic algorithms on the basis of nonuniform sampling in a certain area, the node on τ also has a better representation of the geometric information of the environment and thus it can obtain good guidance performance in arbitrary environment. Moreover, as a meta-algorithm, the heuristic path can be also applied into other RRT variants to further improve the performance.

V. Experimental Studies and Results

In this section, we carry out experimental studies on the basis of the robot operating system (ROS). For the experimental studies, we adopt the Turtlebot2 with a Sick Laser Range Finder as the experimental platform, as shown in Fig. 6. The laptop is with Ubuntu 14.04 on an Intel i7-4600 U CPU with 8 GB of RAM. A computer with Ubuntu 18.04 on an Intel i7-9700 K CPU with 16 GB of RAM is adopted for the simulation studies. First, the efficiency of the heuristic path planning algorithm is tested. Second, we give a evaluation of the compactness and the representativeness of the feature extraction algorithm. Third, the analysis of the proposed algorithm in real-time motion planning is given. Finally, the limitation of the proposed algorithm is discussed and presented. The navigation runtime and the length of the whole trajectory are recorded in the real-time motion planning process. Herein, the navigation runtime refers to the time cost of the robot navigation from the start to the goal, including both the planning time and the execution time.

A. Efficiency of the Heuristic Path Planning

We first test the algorithm efficiency on finding a heuristic path. The A* algorithm [8], the bidirectional RRT (Bi-RRT) algorithm [29], and the GVD feature map (GVD-FM) algorithm [30] are adopted as comparison. The planning time, the total number of nodes traversed, and the length of the heuristic path are recorded for the evaluation. Four scenarios

TABLE I
Statistics of the Heuristic Path Planning Experiments in Four Typical Scenarios

Experiment	Time	Nodes	Path Length	Homotopy
back_forth				
A*	12.1±0.1 s	141668	440	100%
Bi-RRT	168.8±161.4 ms	165±88	694±126	30%
GVD-FM	73.7±8.4 ms	54	483	100%
Ours	73.4±6.2 μs	24	606	100%
s_corridor				
A*	45.9±0.4 s	560734	936	100%
Bi-RRT	7.2±1.4 s	1967±223	1562±118	100%
GVD-FM	126.9±2.8 ms	90	1258	100%
Ours	87.8±5.9 μs	26	1268	100%
shb_4				
A*	58.1±0.3 s	533626	666	100%
Bi-RRT	3.6±2.9 s	1650±1376	1069±192	70%
GVD-FM	521.7±10.2 ms	668	772	100%
Ours	322.4±14.3 μs	104	774	100%
maze				
A*	74.1±1.4 s	508072	1741	100%
Bi-RRT	26.6±17.2 s	7245±2501	2268±69	100%
GVD-FM	1494.0±41.1 ms	2773	2064	100%
Ours	567.3±8.8 μs	167	2064	100%

have been adopted in this experiment, as shown in Fig. 7. The feature graph and the feature matrix are demonstrated in Fig. 7(a)–(d) and in Fig. 7(e)–(h), respectively. For the feature graph, the red node represents the feature node and the green line represents the connections among feature nodes. For the feature matrix, different colors show the different distance between two neighbor nodes. We can find that the dimension of the feature matrix increases as the complexity of the map becomes larger.

Table I lists the data that we obtain from 50 times of repeated trials. We can find that with the proposed algorithm, the heuristic path planning time is obviously decreased. Especially in the complicated maze scenario, the planning time by the proposed algorithm is less than 0.6 ms, which verifies that the proposed algorithm can still guarantee real-time performance in the complicated environments. We can also find that the number of nodes to be searched is greatly reduced compared to the other three methods, which significantly reduces processing time. It is noteworthy that the planning time in Table I records the time that has been spent since the system receives a start pose and a goal pose each time. In addition, since the proposed algorithm is used to generate heuristic paths for further motion planning, compared with the path length, we pay more attention to whether the generated heuristic path and the optimal path are in the same homotopy class [31]. Two trajectories with the same start and the same end are said to be in the same **Homotopy Class** iff one can be smoothly deformed into the other without intersecting obstacles. Otherwise they belong to different homotopy classes. In Table I, we can find that all the heuristic paths generated by the proposed algorithm are in the same homotopy class of the A* algorithm. Therefore, although the path length of the proposed algorithm is longer than the optimal path, it can still provide an effective guide for motion planning.

For the A* algorithm and the Bi-RRT algorithm, each time they receive the path planning target, a new searching process

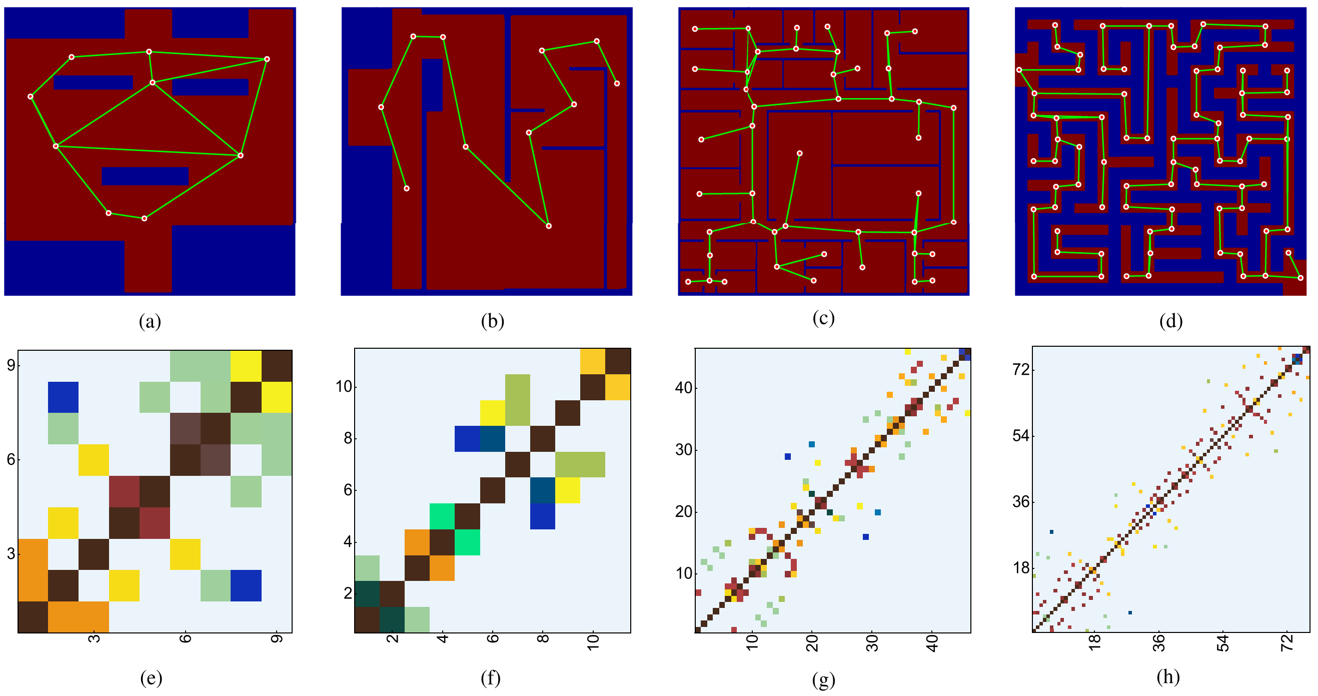

Fig. 7. Feature graph and feature matrix of maps with different complexity. The upper row shows the feature graph. The lower row shows the feature matrix. (a) back_forth. (b) s_corridor. (c) shb_4. (d) maze. (e) back_forth. (f) s_corridor. (g) shb_4. (h) maze.

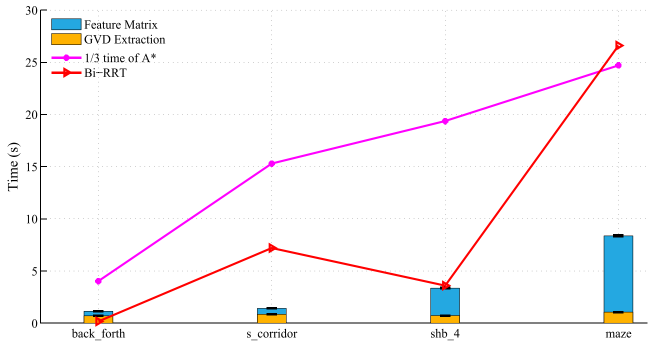

Fig. 8. Time required in the GVD extraction and feature matrix construction. One-third of the average planning time of the A* algorithm and the average planning time of the Bi-RRT algorithm are utilized as references.

needs to be carried out. For the proposed method, the feature matrix only needs to be constructed once for one environment, and then can be reused in the following path planning targets with different start and goal poses. Therefore, we additionally show the required time in the GVD extraction and the feature matrix construction in Fig. 8. We can find that even in the complicated maze environment, the entire preprocessing time is less than 10 s, which illustrates the feasibility of the proposed method in the actual system. One-third of the average planning time of the A* algorithm and the average planning time of the Bi-RRT algorithm are utilized as references. Except for the relatively simple environment, namely the $back\_forth$ scenario, our algorithm still outperforms the A* algorithm and the Bi-RRT algorithm even with the preprocessing time. Through comparison, it can be indicated that the advantages of the proposed algorithm are more obvious in complex environments, while in the simple environment, the sampling-based Bi-RRT algorithm has more advantages.

B. Compactness and Representativeness of Feature Nodes

In this section, we propose two evaluation indicators for the feature extraction of the map. The first one is the compactness of feature nodes, which is defined as the ratio of the number of feature nodes to the number of grids in the free area. Herein, we use a C_{score} to represent the compactness of feature nodes

$$C_{\mathrm{score}} = \frac{\mathrm{count}(\text{feature nodes})}{\mathrm{count}(\text{free grids})}. \tag{8}$$

For a map, the smaller the C_{score} is, the fewer feature nodes are required to represent the free area and the fewer nodes need to be traversed when performing the heuristic path planning.

The second one is the representativeness of feature nodes, which is defined as the ratio of the number of grids in the free area that cannot be connected to any feature node to the total number of grids in the free area. We use a R_{score} to represent the representativeness of feature nodes

$$R_{\mathrm{score}} = \frac{\mathrm{count}(\text{disconnect grids})}{\mathrm{count}(\text{free grids})}. \tag{9}$$

The best value of R_{score} is zero, which means that all points in the free area can be connected to the feature tree, thus guaranteeing the completeness of the heuristic path planning. Therefore, the feature point extraction is the process to minimize these two indicators.

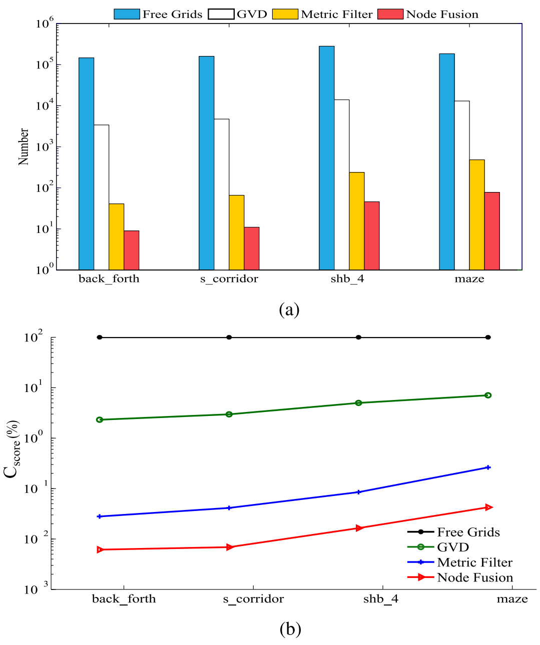

Fig. 9. Compactness of the feature node in four scenarios. (a) Number of feature nodes. (b) Compactness of feature nodes.

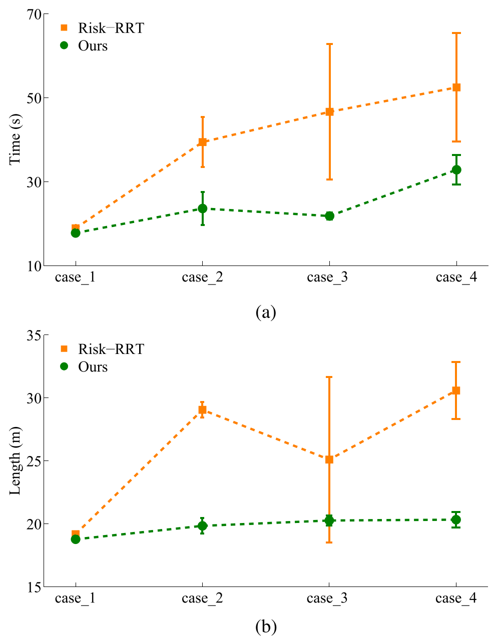

Fig. 10. Performance of the proposed algorithm in four cases. (a) Navigation runtime. (b) Trajectory length.

The number of feature nodes in the feature extraction process in different scenarios is shown in Fig. 9(a). We record the number of feature nodes after the GVD extraction, after the preliminary feature extraction, and after the feature node fusion, respectively. The total number of grids in the free area is also listed as a reference. The corresponding C_{score} is illustrated in Fig. 9(b). Even in the maze scenario, C_{score} is still able to be below 0.05%. Moreover, we also calculate R_{score} for the four scenarios and all of them equal to zero, which means that the proposed algorithm can represent the map well.

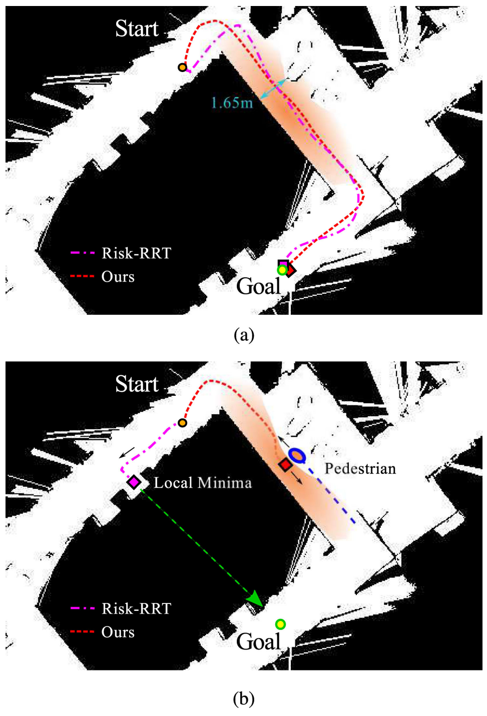

Fig. 11. Demonstration of the proposed algorithm and the risk-RRT algorithm. (a) case_1: t = 24.10s. (b) case_4: t = 9.60s.

C. Performance in Real-Time Motion Planning

The proposed algorithm aims to guide the sampling process and solve the trap space problem of sampling-based motion planning algorithms in complex environments. Therefore, whether it can generate heuristic paths in real time and provide effective guidance for motion planning will be a significant evaluation factor. In the motion planning process, to guarantee real-time performance, sampling-based motion planning algorithms usually adopt a partial motion planning scheme. The representative one is risk-RRT algorithm [28]. In this article, we adopt the Risk-RRT algorithm for the comparative experiment. In our method, we utilize the heuristic path to guide the partial motion planning. In the comparative experiments, the partial motion planning is carried out independently.

The experimental scenario is shown in Fig. 11. The narrowest width of the feasible area (corridor) is around 1.65 m. Except for static obstacles, we set three parameters to illustrate the performance of the proposed algorithm in the complex cases.

- **Size of the Robot**: The size of Turtlebot 2 is 0.36 m × 0.36 m. We add different inflation for the robot in the four cases. The larger the size of the robot, the longer it will take to sample and find a feasible path in a narrow corridor.
- **Sampling Window**: The sampling window is generally set equal to the map size to guarantee the completeness of the path planning. However, the larger the sampling window size, the lower the sampling probability per unit

TABLE II
PARAMETER SETTINGS OF REAL-TIME MOTION PLANNING EXPERIMENTS

Parameter	Feasible Area			Sampling Window
	Corridor	Pedestrian	Robot	Size
case_1	1.65	–	0.40	10*10
case_2	1.65	–	0.80	10*10
case_3	1.65	–	0.60	30*30
case_4	1.65	0.65	0.80	10*10

area in limited time. The trade-off between the sampling efficiency and the completeness of the path planning is usually selected by adjusting the size of the sampling window.

- **Pedestrian**: The pedestrian occupies the feasible area, which will further increase the uncertainty and the difficulty of sampling in the narrow corridor.

The parameter settings of four cases in the real-time motion planning experiments are listed in Table II.

We record the navigation runtime and the length of the trajectory in the four cases for evaluations. For each case, ten times of repeated trials are conducted and the experimental results are shown in Fig. 10. It is obvious that compared with the risk-RRT algorithm, the proposed algorithm can obtain more robust motion planning results in four cases, with less navigation runtime and shorter trajectory lengths. As illustrated in Fig. 11(a), the size of the robot and the sampling window is relatively small and there is no pedestrian in the feasible area in case_1. The magenta dotted line denotes the trajectory generated by the risk-RRT algorithm and the red dashed line denotes the trajectory generated by the proposed method. The square denotes the current position of the robot. In this case, the original motion planning algorithm can still obtain effective sampling nodes in the corridor. Therefore, the risk-RRT algorithm and the proposed algorithm perform similarly. However, as illustrated in Fig. 11(b), the size of the robot increases and a pedestrian is walking in the corridor in case_4. The blue oval denotes the current position of the pedestrian. The trajectory of the pedestrian is represented with blue dashed line. In this case, it is difficult for the risk-RRT algorithm to sample points in the more narrow feasible area, as represented with the orange area in Fig. 11(b). Therefore, it plans a path to a local minima, which is in the opposite direction of the path to the goal. The green dashed line indicates the attraction of the goal to the local minima. We can find that by using the proposed algorithm, invalid sampling caused by narrow passages, pedestrians, large sampling windows, etc., will be greatly reduced, thereby improving navigation efficiency.

D. Discussions

The proposed GVD feature matrix can help mobile robots to quickly find a heuristic path at the advantage of its sparse representation of the environment. The aforementioned experiments have revealed its effectiveness in guiding the sampling-based motion planning of mobile robots. In this part, we discuss the performance of the proposed algorithm in the less complex environment with broad feasible area.

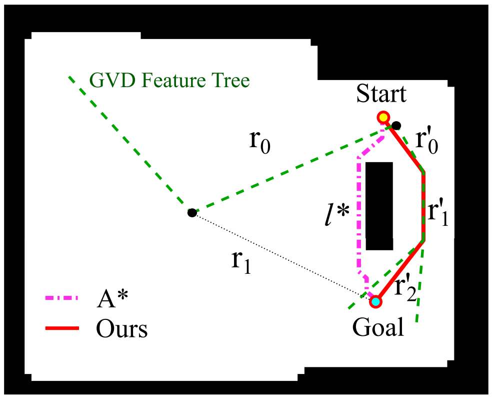

Fig. 12. Heuristic paths generated by the A* algorithm and the proposed algorithm in a relatively simple environment.

A typical example is illustrated in Fig. 12. We can find that the heuristic path generated by the proposed algorithm belongs to the nonhomotopy class of the optimal path that is generated by the A* algorithm. This is caused by the cumulative distance error in the node connection of the feature tree. The cumulative distance error of a path is defined as follows:

$$e = \sum_{i=0}^{N} r_i - l^* \tag{10}$$

where r_i denotes the distance between two GVD feature nodes on the path and l^* denotes the length of the optimal path. In Fig. 12, the cumulative distance errors from the two candidate paths, namely $r_0 \rightarrow r_1$ and $r'_0 \rightarrow r'_1 \rightarrow r'_1$, are calculated as follows:

$$e = \sum_{i=0}^{1} r_i - l^* > e' = \sum_{k=0}^{2} r'_k - l^*. \tag{11}$$

Although $r_0 \rightarrow r_1$ is in the same homotopy class of the optimal path, its larger cumulative distance error hinders it to be selected as the heuristic path since the connection to a node with a larger r will result a greater cumulative distance to the root node.

It is obvious that reducing the cumulative distance error is the key to solving this problem. We can also find that a larger r_i indicates a larger cumulative distance error due to the sparse representation. Therefore, the nonhomotopy problem can be solved by setting an upper threshold for r_i in the node fusion process so as to avoid the extreme large distance between two feature nodes. As the maximum distance between two GVD feature nodes decreases, the cumulative distance error decreases, and thus paths from the nonhomotopy class of the optimal path reduces.

VI. CONCLUSION

In this article, a GVD feature matrix-based efficient heuristic path planning framework was proposed for mobile robot motion planning in dynamic environments. First, based on the GVD of the map, a feature extraction and fusion method was proposed to give a concise representation for the map. A feature matrix structure was proposed to represent the connection among feature nodes. Based on the GVD feature matrix, an efficient heuristic path planning algorithm was presented. In the experimental studies, experiments on the efficiency of heuristic path generation, feature extraction evaluation, and real-time motion planning were been carried out. Two indicators were proposed to evaluate the compactness and the representativeness of the feature nodes. The experimental results verified that our proposed method can achieve satisfactory performance in environments with different complexity. By using the proposed method, the effectiveness and robustness of robot motion planning in a two-dimensional environment were substantially improved.

References

[1] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer, Jul. 2016.

[2] X. Bai, W. Yan, and M. Cao, "Clustering-based algorithms for multivehicle task assignment in a time-invariant drift field," *IEEE Robot. Automat. Lett.*, vol. 2, no. 4, pp. 2166–2173, Jul. 2017.

[3] X. Bai, W. Yan, S. Ge, and M. Cao, "An integrated multi-population genetic algorithm for multi-vehicle task assignment in a drift field," *IEEE Trans. Autom. Sci. Eng.*, vol. 453, pp. 227–238, Jul. 2018, doi: 10.1109/TASE.2019.2914113.

[4] K. Leibrandt, C. Bergeles, and G.-Z. Yang, "Concentric tube robots: Rapid, stable path-planning and guidance for surgical use," *IEEE Robot. Automat. Mag.*, vol. 24, no. 2, pp. 42–53, Jun. 2017.

[5] W. Chi, C. Wang, J. Wang, and M. Q.-H. Meng, "Risk-DTRRT-based optimal motion planning algorithm for mobile robots," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 3, pp. 1271–1288, Jul. 2019.

[6] K. Berntorp, "Path planning and integrated collision avoidance for autonomous vehicles," in *Proc. Amer. IEEE, Control Conf.*, 2017, pp. 4023–4028.

[7] J. Wang and M. Q.-H. Meng, "Socially compliant path planning for robotic autonomous luggage trolley collection at airports," *Sensors*, vol. 19, no. 12, p. 2759, Jan. 2019.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[9] O. KHATIB, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

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

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

[12] J. Y. Lee and H. Choset, "Sensor-based exploration for convex bodies: A new roadmap for a convex-shaped robot," *IEEE Trans. Robot.*, vol. 21, no. 2, pp. 240–247, Apr. 2005.

[13] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 3067–3074.

[14] J. Wang, W. Chi, M. Shao, and M. Q.-H. Meng, "Finding a high-quality initial solution for the RRTS algorithms in 2d environments," *Robotica*, vol. 37, no. 10, pp. 1677–1694, Oct. 2019.

[15] A. Yershova, L. Jaillet, T. Siméon, and S. M. LaValle, "Dynamic-domain RRTS: Efficient exploration by controlling the sampling domain," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2005, pp. 3856–3861.

[16] S. M. Persson and I. Sharf, "Sampling-based a* algorithm for robot path-planning," *Int. J. Robot. Res.*, vol. 33, no. 13, pp. 1683–1708, 2014.

[17] L. Palmieri, S. Koenig, and K. O. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 2775–2781.

[18] Z. Tahir, A. H. Qureshi, Y. Ayaz, and R. Nawaz, "Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments," *Robot. Auton. Syst.*, vol. 108, pp. 13–27, 2018.

[19] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural RRT*: Learning-based optimal path planning," *IEEE Trans. Automat. Sci. Eng.*, vol. 17, no. 4, pp. 1748-1758, Oct. 2020.

[20] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7087–7094.

[21] B. Kim, L. P. Kaelbling, and T. Lozano-Pérez, "Guiding search in continuous state-action spaces by learning an action sampler from off-target search experience," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Apr. 2018, pp. 6509–6516.

[22] Y.-L. Kuo, A. Barbu, and B. Katz, "Deep sequential models for sampling-based planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 6490–6497.

[23] H. Choset and J. Burdick, "Sensor based motion planning: The hierarchical generalized Voronoi graph," *Algorithms Robot Motion Manipulation*, Mar. 1996, pp. 47–61.

[24] K. E. HoffIII, J. Keyser, M. Lin, D. Manocha, and T. Culver, "Fast computation of generalized voronoi diagrams using graphics hardware," in *Proc. 26th Annu. Conf. Comput. Graph. Interactive Techn.*, 1999, pp. 277–286.

[25] D. Kim, J. Lee, and S.-e. Yoon, "Cloud RRT*: Sampling cloud based RRT*," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 2519–2526.

[26] J. Wang and M. Q.-H. Meng, "Optimal path planning using generalized voronoi graph and multiple potential functions," *IEEE Trans. Ind. Electron.*, vol. 67, no. 12, pp. 10621–10630, Dec. 2020.

[27] W. Chi, Z. Ding, J. Wang, G. Chen, and S. Lining, "A reusable gvd feature tree for fast robot motion planning in trapped environments," *IEEE Trans. Cybern.*, in press.

[28] C. Fulgenzi, A. Spalanzani, C. Laugier, and C. Tay, "Risk based motion planning and navigation in uncertain dynamic environment," INRIA, Res. Rep., 2010, p. 14.

[29] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," *in Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symp.*, 2000, pp. 995–1001.

[30] W. Chi, J. Wang, Z. Ding, G. Chen, and L. Sun, "A reusable generalized Voronoi diagram based feature tree for fast robot motion planning in trapped environments," *IEEE Sensors J.*, 2021, pp. 1–1.

[31] G. Diaz-Arango, H. Vázquez-Leal, L. Hernandez-Martinez, M. T. S. Pascual, and M. Sandoval-Hernandez, "Homotopy path planning for terrestrial robots using spherical algorithm," *IEEE Trans. Automat. Sci. Eng.*, vol. 15, no. 2, pp. 567–585, Apr. 2018.

Wenzheng Chi received the B.E. degree in automation from Shandong University, Jinan, China, in 2013, and the Ph.D. degree in biomedical engineering from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, in 2017.

She was a Visiting Scholar at The University of Tokyo, Japan. She was a Postdoctoral Fellow with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, from 2017 to 2018. She is currently an Associate Professor with the Robotics and Microsystems Center, School of Mechanical and Electric Engineering, Soochow University, Suzhou, China. Her research interests include mobile robot path planning, intelligent perception, human–robot interaction, etc.

Zhiyu Ding received the bachelor's degree in mechanical and electronic engineering from the Nanjing Institute of Technology, Nanjing, China, in 2018. He is currently working toward the master's degree with the School of Mechanical and Electrical Engineering, Soochow University, Suzhou, China.

His research interests include mobile robot motion planning and socially adaptive navigation.

Jiankun Wang received the B.E. degree in automation from Shandong University, Jinan, China, in 2015, and the Ph.D. degree in electronic engineering from the Department of Electronic Engineering, Chinese University of Hong Kong, Hong Kong, in 2019.

He is currently a Research Assistant Professor with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, China. His research interests include motion planning and control, human–robot interaction, and machine learning in robotics.

Guodong Chen was born in 1983. He received the Ph.D. degree in mechanical engineering from the Harbin Institute of Technology, Harbin, China, in 2011.

He is currently an Associate Professor with Soochow University, Suzhou, China. His research interests include robot vision and intelligent industrial robots.

Lining Sun was born in Hegang, Heilongjiang Province, China in January 1964. He received the Ph.D. degree in engineering from the Mechanical Engineering Department, Harbin Institute of Technology (HIT), Harbin, China, in 1993.

He is currently with the Soochow University. His research interests include the creation and development of robot-related disciplines.

Dr. Sun has received National Outstanding Youth Fund Winner, and Changjiang Scholar Distinguished Professor by the Ministry of Education.