Çankaya University Journal of Science and Engineering
Volume 16, No. 2 (2019) 046–069
Date Received: October 21, 2019
Date Accepted: October 28, 2019

CUJ

Safe and Efficient Path Planning for Omni-directional Robots using an Inflated Voronoi Boundary

Mohammed Rabeea Hashim Al-Dahhan[1], Klaus Werner Schmidt[2]

[1]*Department of Electronic and Communication Engineering, Çankaya University, Turkey,*
[2]*Department of Electrical and Electronics Engineering, Middle East Technical University, Turkey,*
e-mail: alanymohammed5@gmail.com, schmidt@metu.edu.tr

Abstract: Path planning algorithms for mobile robots are concerned with finding a feasible path between a start and goal location in a given environment without hitting obstacles. In the existing literature, important performance metrics for path planning algorithms are the path length, computation time and path safety, which is quantified by the minimum distance of a path from obstacles.

The subject of this paper is the development of path planning algorithms for omni-directional robots, which have the ability of following paths that consist of concatenated line segments. As the main contribution of the paper, we develop three new sampling-based path planning algorithms that address all of the stated performance metrics. The original idea of the paper is the computation of a modified environment map that confines solution paths to the vicinity of the Voronoi boundary of the given environment. Using this modified environment map, we adapt the sampling strategy of the popular path planning algorithms PRM (probabilistic roadmap), PRM* and FMT (fast marching tree). As a result, we are able to generate solution paths with a reduced computation time and increased path safety. Computational experiments with different environments show that the proposed algorithms outperform state-of-the-art algorithms.

Keywords: Path planning, omni-directional robots, sampling-based algorithms, Voronoi diagram, safety.

1. Introduction

Path planning for mobile robots has attracted much attention in the recent years [1, 2, 3, 4, 5]. Path planning is concerned with finding a feasible robot path between a start and goal location, while avoiding obstacles in the robot environment [6, 7]. Hereby, the most common performance metrics to validate the quality of solution paths are computation time, path length and the minimum distance to obstacles, which quantifies path safety [8, 9].

There are different possible scenarios for robotic path planning depending on the availability of information about the environment [10, 11, 12, 13, 14], the type of obstacles (static or dynamic) [15, 16, 17] and the robot type [7, 3, 18]. In this paper, we focus on the path planning for omni-directional robots in known environments with static obstacles.

In the recent literature, sampling-based algorithms are most popular for path planning in known environments with static obstacles [5, 10, 12, 13, 14, 9, 19, 20, 21, 22]. A large majority of such algorithms is based on probabilistic roadmaps (PRM) or rapidly exploring random trees (RRT). On the one hand, PRMs generate random sample nodes and introduce connections between close nodes in the obstacle-free region to determine a solution path [10]. On the other hand, RRTs are based on the idea of growing a tree in the obstacle-free region from the start location to the goal location [13]. Moreover, there are various extensions of these algorithms. The PRM* and RRT* algorithms ensure convergence to an optimal path [12] and the FMT algorithm in [14] combines features of PRMs and RRTs to determine shorter solution paths. The Quick RRT* algorithm in [21] promises faster convergence to the optimal path and the synchronized biased-greedy RRT in [22] grows trees towards the goal location. A common feature of these algorithms is that they focus on the fast computation of minimum length solution paths. Only very recently, confidence random trees (CRT) [9] were introduced as a sampling-based method for path safety.

Methods that take into account path safety are frequently based on the generalized Voronoi diagram (GVD) [23, 24, 25, 26], which partitions an environment in Voronoi regions of points that are closest to an obstacle. Then, the Voronoi boundary represents the border of the Voronoi regions such that each point on the Voronoi boundary has the same distance to its closest obstacles. Using the GVD, methods such as [24, 26] guide the search for a solution path along the GVD while applying sampling-based methods in narrow passages [24] or satisfying differential constraints [26]. The method in [25] combines visibility graphs, GVDs and potential fields to obtain short paths. Although these methods make use of the GVD, they do not specifically address path safety. [23] first determines a graph from the GVD and then finds the shortest path in that graph using Dijkstra's algorithm [27]. In this case, the solution path has the largest possible distance to obstacles but can be unnecessarily long. As a remedy, [8] suggests to first refine the shortest path in the GVD by removing unnecessary turns and then introduces additional points in order to shorten the solution path. Differently, [28] applies the fast marching method on an inflated Voronoi boundary.

The main objective of this paper is the development of sampling-based methods for robotic path planning with a small computation time, short path length and guaranteed safety distance to obstacles. To this end, we propose to first compute an inflated Voronoi boundary, which then serves as the available free space for solution paths. Next, we develop a new sampling strategy that efficiently generates samples exlusively on the inflated Voronoi boundary. Finally, we adapt the sampling-based algorithms PRM, PRM* and FMT in order to obtain connections between node samples that stay within the inflated Voronoi boundary. The resulting methods are denoted as Inflated PRM (IPRM), Inflated PRM* (IPRM*) and Inflated FMT (IFMT). The quality of the computed solution paths regarding path length and safety is demonstrated by comprehensive computational experiments with different environments and a comparison to state-of-the-art methods.

We note that the only existing sampling-based method in combination with the inflated Voronoi boundary is given by our previous work in [29]. This work only considers the algorithm IPRM and does not provide a comprehensive evaluation.

The remainder of the paper is organized as follows. Background information on sampling-based path planning is given in Section 2. The proposed method is introduced in Section 3 and evaluated in Section 4. Conclusions and an outline of future work are given in Section 5.

2. Background

This section summarizes the required background information for the paper. The notation for robotic path planning is introduced in Section 2.1 and Section 2.2 explains the generalized Voronoi diagram. Several relevant state-of-the-art path planning methods are explained in Section 2.3.

2.1. Notation and Path Planning Problem

In this paper, we focus on path planning in two-dimensional (2D) environments. Such environments are suitable for robots that can navigate along paths that consist of straight-line segments. A possible example for such robots are omni-directional robots that can turn on the spot [18].

We represent the 2D environment by the *configuration space* $\mathscr{C} \in \mathbb{R}^2$. Obstacles in $\mathscr{C}$ that should not be hit by the mobile robot are characterized by the *obstacle region* $\mathscr{C}_{\text{obs}} \subseteq \mathscr{C}$ as depicted in Fig. 1. Then, the *obstacle-free region* that can be used for the robot motion is $\mathscr{C}_{\text{free}} = \mathscr{C} \setminus \mathscr{C}_{\text{obs}}$.

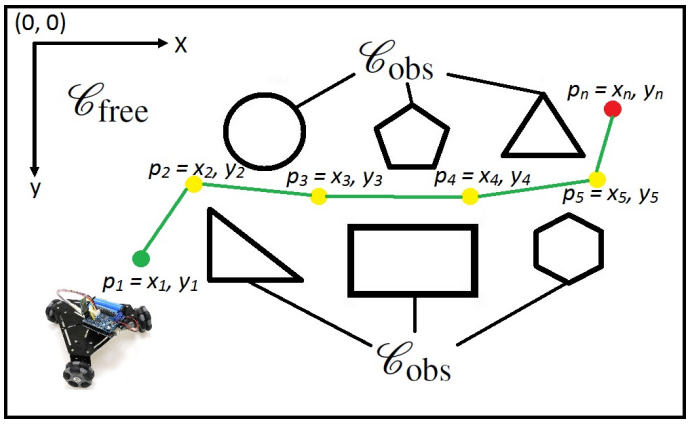

FIGURE 1. Example robot environment.

For any point $p \in \mathscr{C}$, we write $p = (x, y)$, whereby x and y represent the coordinates of p as shown in Fig.1. Then, a *robot path* P can be described by a sequence of vertexes $P = (p_1, p_2, \ldots, p_n)$, whereby $p_i \in \mathscr{C}$ for $i = 1, \ldots, n$. Here, p_1 is the start vertex, p_n is the goal vertex and n is the number of vertexes of the path. Connecting adjacent vertexes p_i and p_{i+1} of a path by straight

lines l_i for each $i = 1,\ldots,n-1$, the overall path consists of the points $\mathscr{P}_P \subseteq \mathscr{C}$ that are covered by the line segments $l_1,\ldots,l_{n-1}$. Accordingly, a path P is *collision-free* if all points covered by the path are in the obstacle-free region, that is, $\mathscr{P}_P \subseteq \mathscr{C}_{\text{free}}$. The set of all obstacle-free paths is

$$\mathscr{A} = \{P | \mathscr{P}_P \subseteq \mathscr{C}_{\text{free}}\}. \tag{1}$$

The distance between two points $p_i = (x_i, y_i), p_j = (x_j, y_j) \in \mathscr{C}$ is given by

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{2}$$

Using (2) and considering a point $p \in \mathscr{C}$ and subsets $\mathscr{C}', \mathscr{C}'' \subseteq \mathscr{C}$, we further write

$$d(p, \mathscr{C}') = \min_{p' \in \mathscr{C}'} d(p, p') \tag{3}$$

for the minimum distance of p from points in the set $\mathscr{C}'$ and

$$d(\mathscr{C}', \mathscr{C}'') = \min_{p' \in \mathscr{C}', p'' \in \mathscr{C}''} d(p', p'') \tag{4}$$

for the minimum distance of points in the sets $\mathscr{C}'$ and $\mathscr{C}''$. Referring to (4), $d(\mathscr{P}_P, \mathscr{C}_{\text{obs}})$ represents the minimum distance of a path P from the obstacle region. In addition, the *path length* of P is

$$L_P = \sum_{i=1}^{n-1} d(p_i, p_{i+1}). \tag{5}$$

Employing the above notation, the path planning problem for mobile robots is concerned with finding a suitable obstacle-free path between a given start position $p_S \in \mathscr{C}_{\text{free}}$ and goal position $p_G \in \mathscr{C}_{\text{free}}$. Here, suitability is specified by performance metrics based on the path length (finding the shortest path), path safety (finding a path distant from obstacles) or the computation time for finding a path. In this paper, we will consider a combination of the stated performance metrics.

2.2. Generalized Voronoi Diagram

The generalized Voronoi diagram (GVD) is frequently used in robotic path planning [8, 16, 24, 25, 26]. Consider a set of geometric objects $\mathscr{O}_1, \mathscr{O}_2, \ldots, \mathscr{O}_m$ such that $\mathscr{O}_i \subseteq \mathscr{C}$ for $i = 1,\ldots,m$. The Voronoi region $\mathscr{V}_i$ of $\mathscr{O}_i$ is the set of all points $p \in \mathscr{C}$ that are closer to $\mathscr{O}_i$ than to any other object $\mathscr{O}_j$ with $i \neq j$ [24, 25, 26]. Formally,

$$\mathscr{V}_i = \{p \in \mathscr{C} | d(p, \mathscr{O}_i) \leq d(p, \mathscr{O}_j), \forall j \neq i\}. \tag{6}$$

Then, the *generalized Voronoi diagram* (GVD) is the collection of all regions $\mathscr{V}_1, \ldots, \mathscr{V}_m$. Moreover, we introduce the notation $\mathscr{V}$ for the set of all points on the boundary of a Voronoi region and we call $\mathscr{V}$ the *Voronoi boundary*. Hereby, it is the case for any $p \in \mathscr{V}$ that there are at least two objects $\mathscr{O}_i, \mathscr{O}_j$ with $i \neq j$ and such that $d(p, \mathscr{O}_i) = d(p, \mathscr{O}_j)$. In addition, we define the set of *branching points* (BPs) $\mathscr{B}$ as the set of all points on the boundary of at least 3 Voronoi regions.

The existing literature provides a multitude of methods for the computation of GVDs [8, 24, 25, 31]. In this paper, we assume that the robot environment is given in the form of a digital map, where $\mathscr{C}_{\text{free}}$ and $\mathscr{C}_{\text{obs}}$ are represented by white and black pixels, respectively. In this case, the GVD can be computed by using the morphological operation of thinning as illustrated in Fig.2.

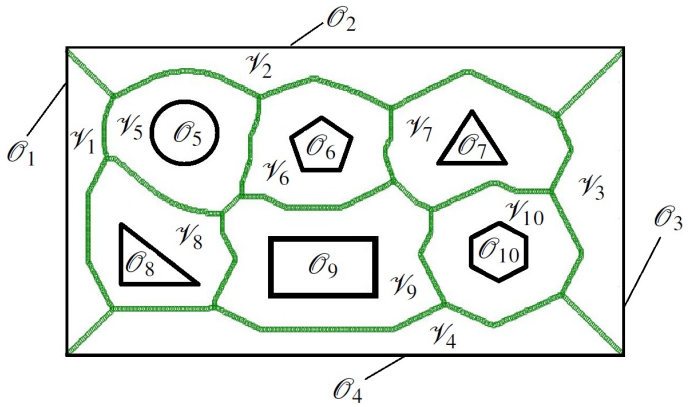

FIGURE 2. Example environment with generalized Voronoi diagram.

2.3. Basic Methods

We next describe several popular algorithms for robotic path planning.

2.3.1. Voronoi Diagram and Dijkstra's Algorithm. GVDs can be used for path planning in order to obtain a path with a maximum distance from the obstacle region [8, 24, 25, 31]. Hereby, the existing methods generate the GVD of the environment and extend it by the shortest obstacle-free connection of the start position p_{S} and goal position p_{G} to the GVD [16, 26, 32]. Then, a graph $G = (V, E)$ is extracted from this extended GVD such that p_{S}, p_{G} and the BPs correspond to vertexes in V. The edges E are introduced between vertexes that are connected by the Voronoi boundary and are labeled with the connection distance. Finally, Dijkstra's algorithm [27] is applied to this graph to determine the shortest path in the extended GVD. The described algorithms are safe but generally produce long paths. An example for the environment in Fig. 1 is given in Fig. 3.

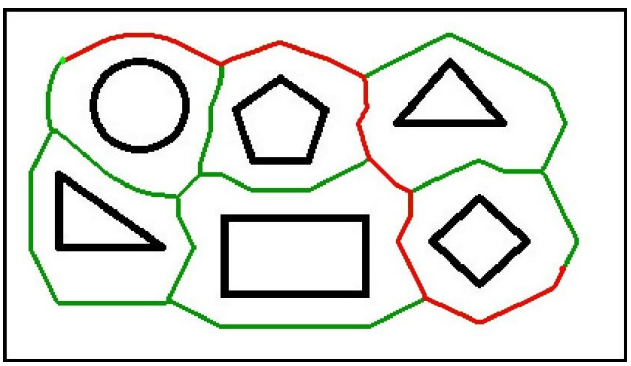

FIGURE 3. Shortest path on the Voronoi boundary.

2.3.2. Relevant Functions for Sampling-based Methods. Sampling-based algorithms for robotic path planning are of high interest due to their fast computation of reasonable solution paths. We next outline common functions of these sampling-based algorithms.

All sampling-based methods are based on the generation of random samples in $\mathscr{C}_{\text{free}}$. We denote the function that generates N samples in $\mathscr{C}_{\text{free}}$ that are drawn from a uniform distribution as

$$X_{\text{rand}} = \texttt{SampleFree}(\mathscr{C}_{\text{free}}, N) \subseteq \mathscr{C}_{\text{free}}.$$

Hereby, each $x \in X_{\text{rand}}$ represents a random sample in $\mathscr{C}_{\text{free}}$. In addition, we introduce the function

$$X_{\text{new}} = \texttt{SampleRad}(x, r, N, \mathscr{C}_{\text{free}}) \subseteq \mathscr{C}_{\text{free}}$$

that generates N random samples X_{new} in $\mathscr{C}_{\text{free}}$ on a circle with radius r around a point x. Further,

$$X_{\text{near}} = \texttt{Near}(V, x, r) = \{\hat{x} \in V | d(x, \hat{x}) \leq r\}$$

determines all points in the set V that lie on a disk with radius r around a point x. Finally,

$$\texttt{CollisionFree}(x, \hat{x}, \mathscr{C}_{\text{obs}}) = \begin{cases} \textbf{true} & \text{if } l_{x,\hat{x}} \cap \mathscr{C}_{\text{obs}} \neq \emptyset \\ \textbf{false} & \text{otherwise} \end{cases}$$

returns **true** if the straight line $l_{x,\hat{x}}$ between the points x and $\hat{x}$ intersects $\mathscr{C}_{\text{obs}}$ and **false** otherwise.

2.3.3. Probabilistiy Roadmap (PRM). The PRM algorithm in Algorithm 1 is one of the most popular algorithms for robotic path planning [10, 11] that has initially been introduced for multi-query applications. However, the PRM algorithm is as well suitable for single-query applications [30]. In this case, the PRM algorithm first generates a set of N_{PRM} random nodes in $\mathscr{C}_{\text{free}}$ and creates a graph $G = (V, E)$. Initially, the vertexes consist of X_{rand}, p_{S} and p_{G} and there are no edges. Then, the PRM algorithm iteratively picks one of the sample nodes (line 4) and determines all neighbor nodes X_{near} of x_{rand} within a radius r_{PRM} (line 5). Edges from nodes $x_{\text{near}} \in X_{\text{near}}$ to x_{rand} are introduced if x_{rand} and x_{near} do not belong to the same connected component in G and if they can be connected by a collision free line (line 6 to 9). Each edge $(x_{\text{rand}}, x), (x, x_{\text{rand}})$ is labeled by its cost $c((x_{\text{rand}}, x)) = c((x, x_{\text{rand}}))$, which is represented by the distance between the related nodes (line 10). Finally, the algorithm returns the shortest (minimum cost) path P from p_{s} to p_{g} in the resulting graph G.

1 **Function** $P = \texttt{PRM}(p_S, p_G, N_{PRM}, \mathscr{C}, \mathscr{C}_{obs}, \mathscr{C}_{free})$
2 Initialize: $X_{\text{rand}} = \texttt{SampleFree}(\mathscr{C}_{\text{free}}, N_{\text{PRM}})$; $V = \{p_{\text{S}}, p_{\text{G}}\} \cup X_{\text{rand}}$; $E = \emptyset$
3 **for** $i = 1, \ldots, N_{PRM}$ **do**
4 Pick $x_{\text{rand}} \in X_{\text{rand}}$; $X_{\text{rand}} = X_{\text{rand}} \setminus \{x_{\text{rand}}\}$
5 $X_{\text{near}} = \texttt{Near}(V, x_{\text{rand}}, r_{\text{PRM}})$
6 **for** $x \in X_{near}$ *in order of increasing* $d(x, x_{rand})$ **do**
7 **if** x_{rand} *and* x *are not in the same connected component of* G **then**
8 **if** $\texttt{CollisionFree}(x_{rand}, x, \mathscr{C}_{obs})$ **then**
9 $E = E \cup \{(x_{\text{rand}}, x), (x, x_{\text{rand}})\}$
10 $c((x_{\text{rand}}, x)) = c((x, x_{\text{rand}})) = d(x_{\text{rand}}, x), (x, x_{\text{rand}})$
11 **return** shortest path P from p_{S} to p_{G} in G.

Algorithm 1: PRM algorithm (for fixed value of r_{PRM}) and PRM* algorithm (for $r_{\text{PRM}} = \gamma_{\text{PRM}} \cdot (\log(n)/n)^2)$).

2.3.4. PRM*. The PRM* algorithm was proposed as a modified version of PRM in [12]. Its only difference to the classical PRM algorithm in Algorithm 1 is that the connection radius r_{PRM} between nodes in line 5 decreases with an increasing number of nodes in the form $r_{\text{PRM}} = \gamma_{\text{PRM}} \cdot \sqrt{\log(n)/n}$. Hereby, γ_{PRM} is a constant that has to be chosen larger than $2 \cdot \sqrt{1.5} \cdot \mu(\mathscr{C}_{\text{free}})/\pi$ for 2D-environments ($\mu(\mathscr{C}_{\text{free}})$ denotes the area of $\mathscr{C}_{\text{free}}$). As a result, the PRM* algorithm is expected to produce shorter solution paths than the PRM algorithm with a possibly increased computational effort.

2.3.5. Fast Marching Tree (FMT). The FMT algorithm was proposed in [14]. Its set of nodes V is generated in the same way as for the PRM algorithm. Moreover, it keeps sets of closed (V_{closed}), open (V_{open}) and unvisited (V_{un}) nodes that are initialized in line 2. Hereby, $c(x)$ represents the cost (distance) of traveling from p_{S} to x along the graph $G = (V, E)$. In each iteration, the FMT algorithm selects the node with the lowest cost in the open set (line 18) and finds all of its neighbors X_{near} in the unvisited set (line 5). In analogy to PRM*, neighbors are defined based on a connection distance $r_{\text{FMT}} = \gamma_{\text{FMT}} \cdot \sqrt{\log(n)/n}$ that decreases with the number of iterations. Each neighbor $x \in X_{\text{near}}$ is then connected to the closest neighbor y_{min} in the open set Y_{near} if the connection is collision free and the relevant sets are updated (line 7 to 15). The algorithm terminates without success if no more nodes are left to be processed ($V_{\text{open}} = \emptyset$ in line 16). Otherwise, a solution path is returned if p_{G} is found to be the unprocessed node with the lowest cost.

1 **Function** $P = \texttt{FMT}(p_S, p_G, N_{FMT}, \mathscr{C}, \mathscr{C}_{obs}, \mathscr{C}_{free})$
2 Initialize: $V = \{p_S, p_G\} \cup \texttt{SampleFree}(\mathscr{C}_{\text{free}}, N_{\text{FMT}})$; $E = \emptyset$; $V_{\text{un}} = V \setminus p_S$; $V_{\text{open}} = \{p_S\}$; $V_{\text{closed}} = \emptyset$; $c(p_S) = 0$; $z = p_S$
3 **while** $z \neq p_G$ **do**
4 $\hat{V}_{\text{open}} = \emptyset$
5 $X_{\text{near}} = V_{\text{un}} \cap \texttt{Near}(V \setminus \{z\}, z, r_{\text{FMT}})$
6 **for** $x \in X_{near}$ **do**
7 $Y_{\text{near}} = V_{\text{open}} \cap \texttt{Near}(V \setminus \{x\}, x, r_{\text{FMT}})$
8 $y_{\min} = \arg\min_{y \in Y_{\text{near}}} \{c(y) + d(y,x)\}$
9 **if** $\texttt{CollisionFree}(y_{min}, x, \mathscr{C}_{obs})$ **then**
10 $E = E \cup \{(y_{\min}, x)\}$
11 $\hat{V}_{\text{open}} = \hat{V}_{\text{open}} \cup \{x\}$
12 $V_{\text{un}} = V_{\text{un}} \setminus \{x\}$
13 $c(x) = c(y_{\min}) + d(y_{\min}, x)$
14 $V_{\text{open}} = (V_{\text{open}} \cup \hat{V}_{\text{open}}) \setminus \{z\}$
15 $V_{\text{closed}} = V_{\text{closed}} \cup \{z\}$
16 **if** $V_{open} = \emptyset$ **then**
17 **return** no path found
18 $z = \arg\min_{x \in V_{\text{open}}} \{c(x)\}$
19 **return** shortest path P from p_S to p_G in G.

Algorithm 2: FMT algorithm.

2.3.6. Confidence Random Tree (CRT). The previously discussed algorithms are designed for finding short paths with a small computation time. Including path safety as a performance metric, the confidence random tree (CRT) algorithm tries to generate solution paths that stay away from obstacles [9]. To this end, the CRT algorithm introduces the notion of *confidence* of a node $x \in \mathscr{C}_{\text{free}}$ as

$$\texttt{Conf}(x, \mathscr{C}_{\text{obs}}) = \min\{d(x, \mathscr{C}_{\text{obs}})/c_{\max}, 1\}.$$

Hereby, $c_{\max}$ is a maximum distance parameter and the confidence gives an indication of the distance of x to the obstacle region. Then, the CRT algorithm expands a tree $G = (V, E)$ starting from p_S (line 2). In each iteration, a set X_{new} of new nodes is determined at a distance $d = \texttt{Conf}(x, \mathscr{C}_{\text{obs}}) \cdot c_{\max}$ from each node x in the current open set X_{open} (line 5 to 9). Hereby, each element of X_{new} stores the generated node x_{new} and its parent node x (line 9). In order to limit the number of nodes, the CRT algorithm includes a node rejection method to avoid generating nodes in previously explored areas (line 10 to 23). Here, nodes are rejected if they are too close to previously explored nodes in X_{closed} (line 14 to 17) or if they are too close to an accepted node x_{new} with a higher confidence (line 21 to line 23). Accepted nodes x_{new} are added to X_{open} for processing and an edge to the parent node is introduced in G (line 19 and 20). The CRT algorithm terminates without a solution path if X_{open} is empty or with a solution path P if a connection to p_G is found (line 26).

Due to the consideration of node confidence, the CRT algorithm generates safe solution paths at the expense of an increased path length.

Function $P = \texttt{CRT}(p_S, p_G, c_{max}, c_{min}, \mathscr{C}, \mathscr{C}_{obs}, \mathscr{C}_{free})$
  Initialize: $X_{\text{open}} = \{p_{\text{S}}\}$; $X_{\text{closed}} = \emptyset$; $V = \{p_{\text{S}}\}$, $E = \emptyset$
  while $X_{open} \neq \emptyset$ **do**
    $X_{\text{closed}} = X_{\text{closed}} \cup X_{\text{open}}$; $X_{\text{new}} = \emptyset$
    for $x \in X_{open}$ **do**
      $X_{\text{rand}} = \texttt{SampleRad}(x, \texttt{Conf}(x, \mathscr{C}_{\text{obs}}) \cdot c_{\text{max}}, n_{\text{CRT}}, \mathscr{C}_{\text{free}})$
      for $x_{new} \in X_{rand}$ **do**
        if $\texttt{Conf}(x_{new}, \mathscr{C}_{obs}) \geq c_{min}$ **then**
          $X_{\text{new}} = X_{\text{new}} \cup \{(x_{\text{new}}, x)\}$
    Sort X_{new} with decreasing confidence
    $X_{\text{open}} = \emptyset$
    while $X_{new} \neq \emptyset$ **do**
      Take first element (x_{new}, x) from X_{new}; $X_{\text{new}} = X_{\text{new}} \setminus \{(x_{\text{new}}, x)\}$; $f_{\text{accept}} = 1$
      for $\hat{x} \in X_{closed}$ **do**
        if $\texttt{Conf}(\hat{x}) \cdot c_{max} > d(x_{new}, \hat{x})$ **then**
          $f_{\text{accept}} = 0$
          break
      if $f_{accept} \neq 0$ **then**
        $X_{\text{open}} = X_{\text{open}} \cup \{x_{\text{new}}\}$
        $V = V \cup \{x_{\text{new}}\}$; $E = E \cup \{(x, x_{\text{new}})\}$
        for $(\hat{x}_{new}, \hat{x}) \in X_{new}$ **do**
          if $\texttt{Conf}(x_{new}) \cdot c_{max} > d(x_{new}, \hat{x}_{new})$ **then**
            $X_{\text{new}} = X_{\text{new}} \setminus \{(\hat{x}_{\text{new}}, \hat{x})\}$
    $x_{\text{near}} = \arg\min_{x \in X_{\text{open}}} d(x, p_{\text{G}})$
    if $Conf(x_{near}) \cdot c_{max} > d(x_{near}, p_G)$ **then**
      return path P from p_{S} to p_{G} in $G = (V, E)$
  return $P = \emptyset$

Algorithm 3: CRT algorithm.

For illustration, Fig. 4 shows solution paths for the described algorithms. It is readily observed that the paths obtained for PRM, PRM* and FMT come very close to the obstacles. This is expected since these algorithms try to minimize the path length and do not account for path safety. On the other hand, the example path for CRT in Fig. 4 (d) stays away from the obstacles but leads to an increased path length. The main focus of this paper is the adaptation of algorithms such as PRM, PRM* and FMT in order to address path safety without a significant increase in path length.

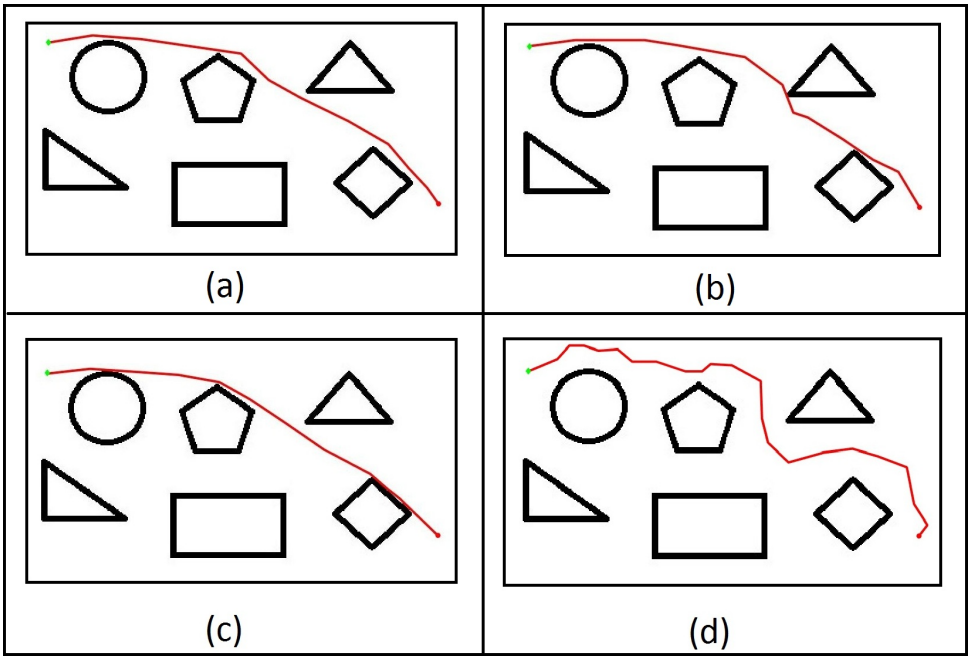

FIGURE 4. Example solution paths: (a) PRM; (b) PRM*; (c) FMT; (d) CRT.

3. Proposed Method

This section develops the method for safe and fast path planning based on the knowledge of the extended GVD and the corresponding Voronoi boundary $\mathcal{V}$ in Section 2.3.1. Section 3.1 describes the general methodology and Section 3.2 and 3.3 combine the proposed methodology with the classical path planning algorithms described in Section 2.3.

3.1. Inflated Path

We assume that the Voronoi boundary is available in the form of a set of points $\mathcal{V} \subseteq \mathcal{C}_{\text{free}}$ and the start and goal point p_{S} and p_{G} are elements of $\mathcal{V}$ in analogy to Section 2.3.1. Specifically, we consider the case where a solution path exists in the extended GVD.[1] As the first step of our algorithm, we suggest to prune the extended GVD in order to remove parts of $\mathcal{V}$ that cannot lie on a solution path from p_{S} to p_{G} similar to [25].

Next, we define a distance D_{I} and we inflate $\mathcal{V}$ by the width D_{I}. Considering that the environment is given in the form of a digital image (such as JPEG), whereby $\mathcal{V}$ is represented by pixels in this image, the inflated Voronoi boundary $\mathcal{V}_{\text{I}}$ can be computed by a morphological dilation operation:

$$\mathcal{V}_{\text{I}} = \mathcal{V} \oplus B_{D_{\text{I}}} = \bigcup_{b \in B_{D_{\text{I}}}} \mathcal{V}_b, \tag{7}$$

[1]We note that this is not a restriction of the general case. If there is no solution in the extended GVD, there is generally no solution of the path planning problem.

whereby B_{D_I} represents a disk with radius D_I, $\oplus$ represents the dilation operation and $\mathscr{V}_b$ is the translation of $\mathscr{V}$ by $b \in B_{D_\mathrm{I}}$. The resulting map for the environment in Fig. 2 with the inflated pruned Voronoi boundary $\mathscr{V}_\mathrm{I}$ for $D_\mathrm{I} = 6$ and $D_\mathrm{I} = 12$ is shown in Fig. 5.

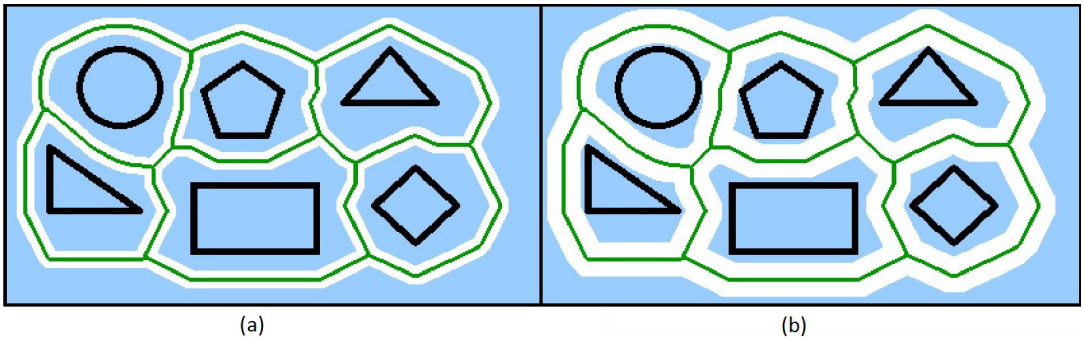

FIGURE 5. Example environment with different path width: (a) D = (6 pixels) (b) D = (12 pixels)

Using $\mathscr{V}_\mathrm{I}$, the main idea of this paper is a modification of the sampling method `SampleFree` used in the sampling-based path planning algorithms in Section 2.3. Instead of generating random samples in the free space $\mathscr{C}_\mathrm{free}$, we suggest to generate samples only in $\mathscr{V}_\mathrm{I}$. To this end, we next both develop an efficient method for generating such samples and provide a formula for deciding on the number of required node samples.

In order to efficiently generate samples in $\mathscr{V}_\mathrm{I}$, we first observe that all such samples should have a maximum distance of D_I from the original Voronoi boundary $\mathscr{V}$. That is, we first select a number of N_V random points $P_\mathscr{V}$ from $\mathscr{V}$. For each point $p \in P_\mathscr{V}$, we generate random values $d \in [0, D_\mathrm{I}]$ and $\theta \in [0, 2, \pi]$ and determine the sample

$$v = p + d \cdot \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}. \tag{8}$$

That is, each node sample is represented by a point, whose distance to $\mathscr{V}$ is bounded by D_I. The proposed procedure is summarized in Algorithm 4.

Writing $|\mathscr{V}|$ for the overall sum of all path lengths on $\mathscr{V}$, we observe that the number of required node samples increases with $|\mathscr{V}|$ and decreases with D_I since a larger value of D_I leaves more free space for obstacle-free connections (compare Fig. 5 (a) and (b)). Hence, we suggest to compute the number of node samples as

$$N_\mathscr{V} = \gamma_\mathscr{V} \cdot \frac{|\mathscr{V}|}{D_\mathrm{I}}, \tag{9}$$

1 **Function** $S = \texttt{SampleInflated}(N_V, \mathscr{V}, D_I)$
2 Initialize: $S = \emptyset$
3 Select N_{V} random points P_{V} from $\mathscr{V}$
4 **for** $p \in P_V$ **do**
5 Generate random $d \in [0, D_{\mathrm{I}}]$
6 Generate random $\theta \in [0, 2, \pi]$
7 Generate new sample v according to (8)
8 $S = S \cup \{v\}$
9 **return** S

Algorithm 4: Computation of samples close to $\mathscr{V}$.

whereby $\gamma_{\mathscr{V}}$ is a safety coefficient that can be adjusted depending on the specific environment. We next point out the proposed modifications of the algorithms (PRM, PRM', FMT) in Section 2.3.

3.2. Inflated-path PRM (IPRM) and Inflated-path PRM* (IPRM*)

The original PRM algorithm (Section 2.3.3, Algorithm 1) computes node samples in the overall free space using $\texttt{SampleFree}(\mathscr{C}, \mathscr{C}_{\mathrm{obs}}, N_{\mathrm{PRM}})$ and checks if connections between nodes are collision-free using $\texttt{CollisionFree}(x_{\mathrm{rand}}, x, \mathscr{C}_{\mathrm{obs}})$ with the obstacle region $\mathscr{C}_{\mathrm{obs}}$. The proposed algorithms inflated-path PRM (IPRM) and inflated-path PRM* (IPRM*) generate node samples as described in Section 3.1 and check collision-freeness using the modified obstacle region $\hat{\mathscr{C}}_{\mathrm{obs}} = \mathscr{C} \setminus \mathscr{V}_{\mathrm{I}}$. That is, line 2 in Algorithm 1 is replaced by

$$X_{\mathrm{rand}} = \texttt{SampleInflated}(N_{\mathscr{V}}, \mathscr{V}, D_{\mathrm{I}}); V = \{p_{\mathrm{S}}, p_{\mathrm{G}}\}; E = \emptyset$$

and line 8 in Algorithm 1 is replaced by

$$\textbf{if } \texttt{CollisionFree}(x_{\mathrm{rand}}, x, \hat{\mathscr{C}}_{\mathrm{obs}}) \textbf{ then}.$$

3.3. Improved FMT

Similar to the modification of PRM and PRM*, we suggest to change the sampling method and the obstacle region of the FMT algorithm in Section 2.3.5. To this end, we replace line 2 in Algorithm 2 by

$$V = \{p_{\mathrm{S}}, p_{\mathrm{G}}\} \cup \texttt{SampleInflated}(N_{\mathscr{V}}, \mathscr{V}, D_{\mathrm{I}}); E = \emptyset; V_{\mathrm{un}} = V \setminus p_{\mathrm{S}}; V_{\mathrm{open}} = \{p_{\mathrm{S}}\};$$

In addition, we replace line 9 in Algorithm 2 by

$$\textbf{if } \texttt{CollisionFree}(y_{\mathrm{min}}, x, \hat{\mathscr{C}}_{\mathrm{obs}}) \textbf{ then}.$$

4. Evaluation

We next perform a comparison of the proposed methods and the existing methods in Section 2.3 regarding the resulting path length, safety distance and computation time. Section 4.1 explains the setup of the computational experiments and Section 4.2 to 4.5 evaluate the considered algorithms for different environments. A discussion of the obtained results is given in Section 4.6.

4.1. Experimental Setup and Maps

We apply the described algorithms to the environments in Figure 6 which are given as binary images, where pixels in $\mathscr{C}_{\text{free}}$ are white and pixels in $\mathscr{C}_{\text{obs}}$ are black. The start position and the goal position are shown by a green diamond and red circle, respectively.

The maps are selected according to their different properties as follows. The polygon map in Fig. 6 (a) has different obstacles that are represented by polygon shapes and that leave sufficient free space for multiple routes between p_{S} and p_{G}. In the maze map in Fig. 6 (b), obstacles are represented by straight lines and there are multiple routes between p_{S} and p_{G}. The U-map in Fig. 6 (c) offers U-shaped obstacles, where candidate paths can be trapped. The maze map in Fig. 6 (d) provides a single long and narrow circular solution route.

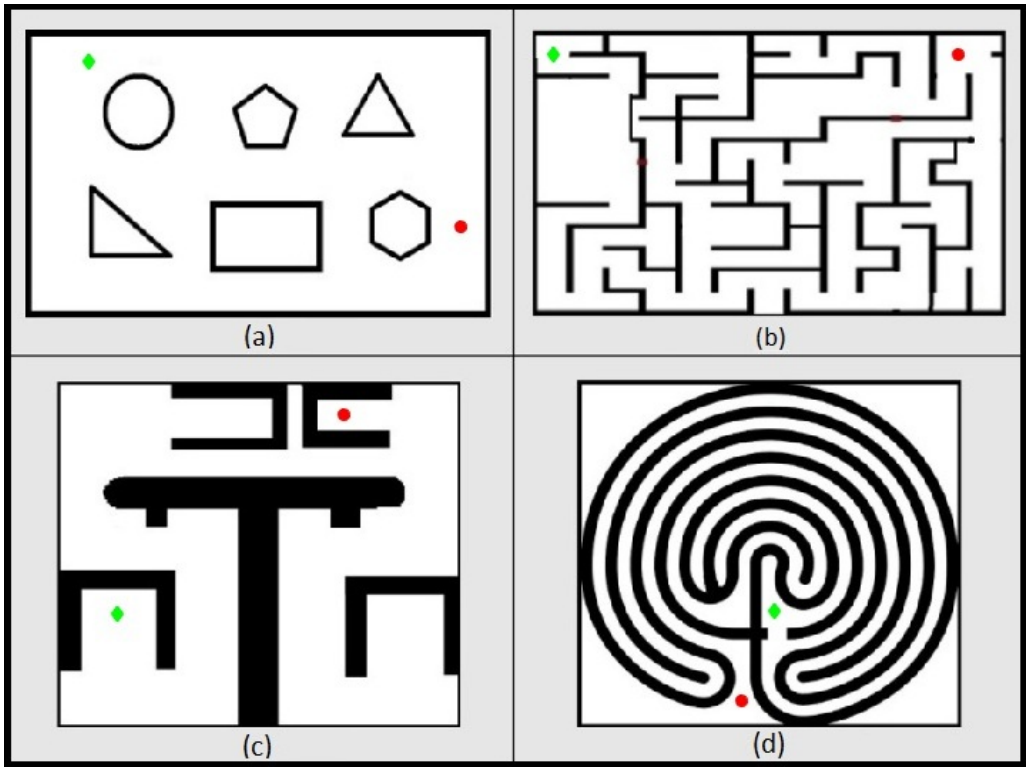

FIGURE 6. Environments used for the evaluation.

In order to perform a fair evaluation, all the algorithms were implemented in Matlab using the same functions for common tasks of the different algorithms as indicated in Section 2.3.2. The experiments were run on a personal computer with Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz and 8.00 GB RAM. For each environment, 100 test runs of each algorithm were performed.

4.2. Polygon Map

We first consider the polygon map in Fig. 6 (a). Following the procedure in Section 2.2 and 3.1, we determine the extended GVD and the inflated Voronoi boundary as shown in Fig. 7 for different values of $D_{\mathrm{I}} = 14$, $D_{\mathrm{I}} = 10$ and $D_{\mathrm{I}} = 6$. Here, $\mathscr{V}_{\mathrm{I}}$ is shown in white, $\mathscr{V}$ is shown in green and $\mathscr{C}_{\mathrm{obs}}$ is shown in black. The light blue region represents the part of $\mathscr{C}_{\mathrm{free}}$ that is not close enough to $\mathscr{V}$ and hence is not considered for solution paths.

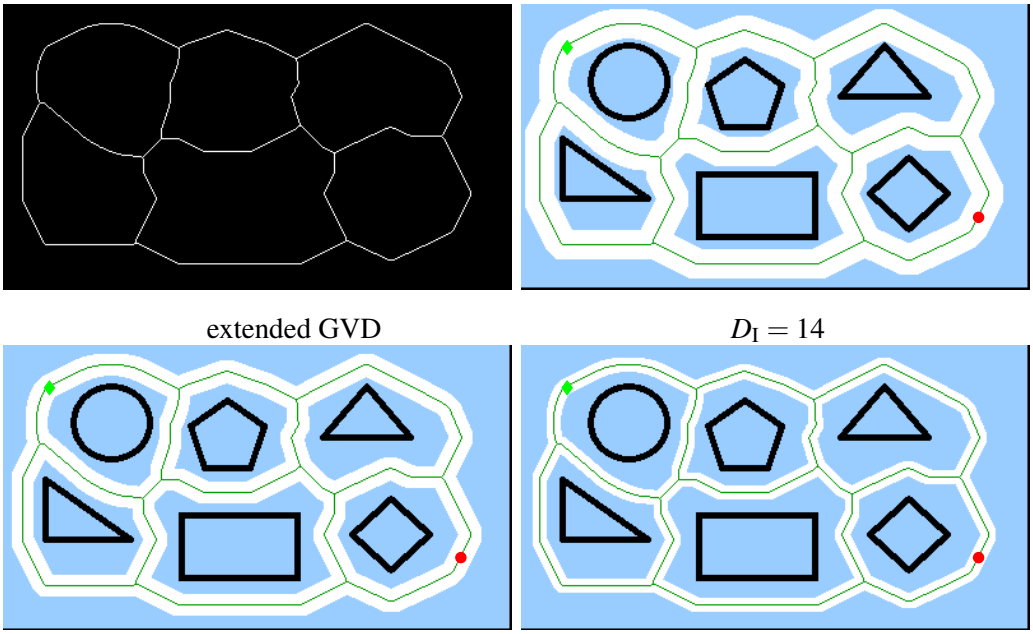

FIGURE 7. Polygon map: extended GVD and inflated Voronoi boundary.

The computational results for the polygon map are shown in Fig. 8. For each method the average values of computation time ($\bar{T}_{\mathrm{comp}}$), path length ($\bar{L}_{\mathrm{path}}$), minimum obstacle distance ($\bar{D}_{\mathrm{min}}$) and number of nodes ($\bar{N}_{\mathrm{nodes}}$) for 100 runs (represented by a bar) are displayed. In addition, the error bars show the maximum and minimum value among the 100 test runs. We point out the following main observations from the figure.

- Regarding the computation time, it is observed that the proposed algorithms are always faster than the related classical algorithm, whereby the computation time increases for a narrower inflated path. This change in the computation time is directly related to the number of nodes $\bar{N}_{\mathrm{nodes}}$ as computed with (9).

- Although it is the case that PRM, PRM* and FMT produce short solution paths, these algorithms do not account for path safety, that is, $\bar{D}_{\text{min}}$ is small. All the proposed algorithms achieve increased safety depending on the value of D_{I}. Most interestingly, path safety is comparable to the results of the CRT algorithm if D_{I} is chosen small enough, whereas the computation time, the path length and the variation among solutions are smaller for the proposed algorithms.
- Only the VD algorithm can achieve safer paths than the proposed algorithms but with a significantly increased path length.

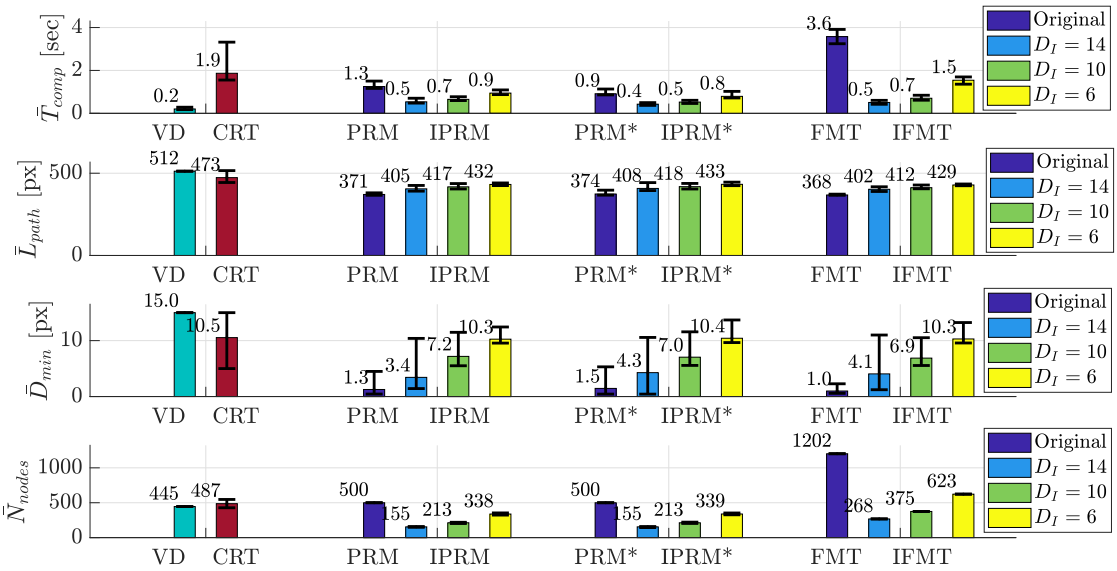

FIGURE 8. Comparison of the performance metrics for the polygon map.

In summary, the proposed algorithms clearly outperform the existing algorithms for the polygon map when taking into account computation time, path length and safety. For illustration, Fig. 9 compares solution path examples for the different methods ($D_{\text{i}} = 6$). It can be seen that the path for CRT has unnecessary turns, which extend the path compared to the paths generated by IPRM, IPRM* and IFMT, which attempt to find the shortest path within the inflated Voronoi boundary.

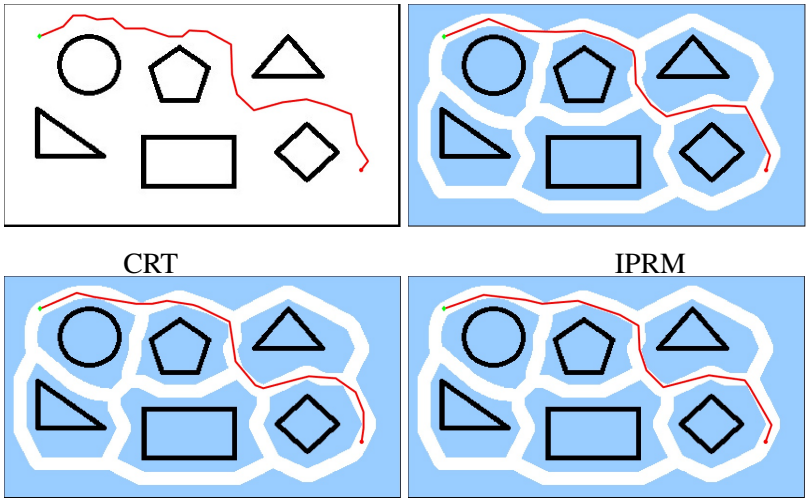

FIGURE 9. Solution paths for the polygon map.

4.3. Maze Map with Straight Lines

We next consider the maze map in Fig. 6 (b). The extended GVD and the inflated Voronoi boundary for this map are shown in Fig. 10 for $D_{\mathrm{I}} = 9$, $D_{\mathrm{I}} = 7$ and $D_{\mathrm{I}} = 5$, whereas Fig. 11 depicts the computational results for this map.

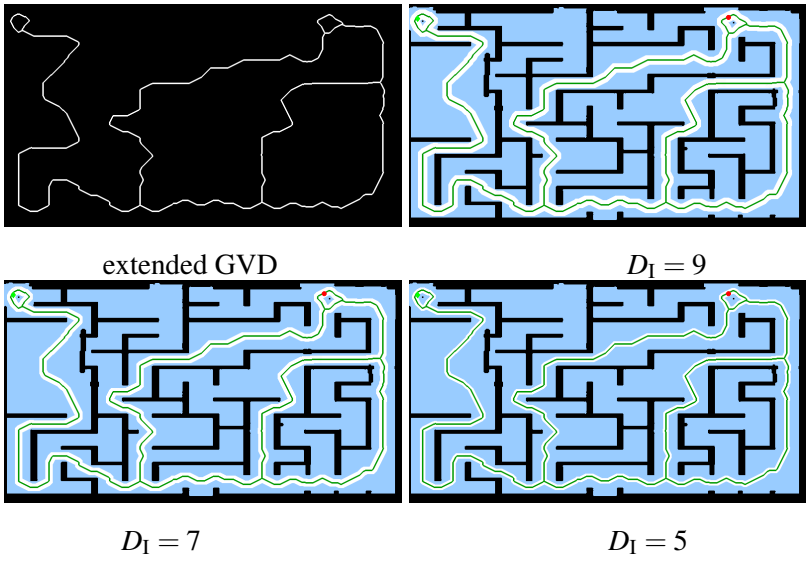

FIGURE 10. Maze map: extended GVD and inflated Voronoi boundary.

We point out the following main observations from this experiment.

- Regarding the computation time, it can again be seen that the proposed algorithms are generally faster than the related classical algorithm, whereby the IPRM* algorithm is fastest.

- Path safety can be significantly increased compared to the classical algorithms when using the proposed algorithms. Moreover, an increase in path safety compared to the CRT algorithm is possible at a significantly reduced computation time, path length and variation of the obtained solutions. Here, the main reason for the increased computation time of the CRT algorithm is the generation of node samples in parts of the map that are not relevant for finding a solution path. The proposed algorithms only generate node samples along the possible routes along the inflated Voronoi boundary.

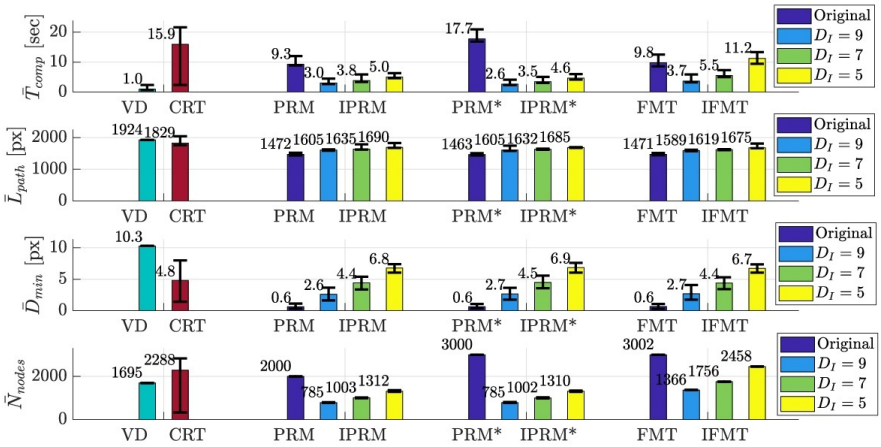

FIGURE 11. Comparison of the performance metrics for the maze map with straight lines.

Fig. 12 compares solutions paths of the different methods. Similar to previous section, CRT generates longer paths due to unnecessary turns when following straight passages. On the contrary, the solution paths of IPRM, IPRM* and IFMT use straight line connections in such passages.

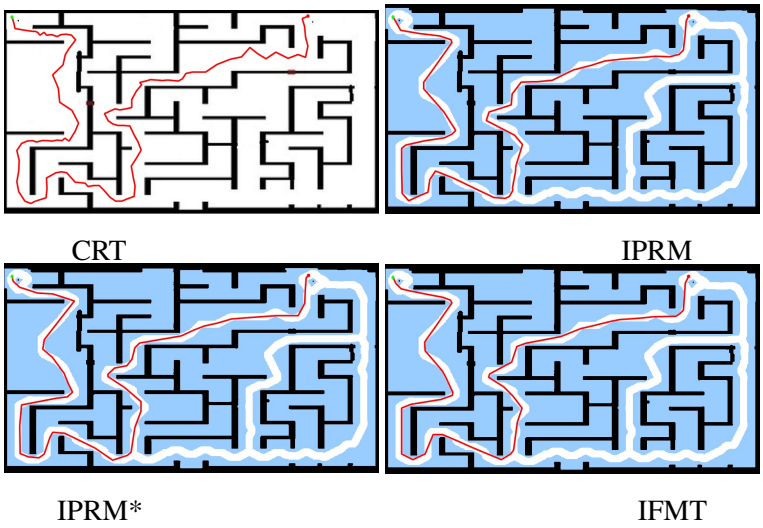

FIGURE 12. Solution paths for the maze map with straight lines.

4.4. U-Map

We further study the U-map in 6 (c). The extended GVD and the inflated Voronoi boundary for this map are shown in Fig. 13 for $D_{\mathrm{I}} = 20$, $D_{\mathrm{I}} = 15$ and $D_{\mathrm{I}} = 10$, whereas Fig. 11 depicts the computational results for this map.

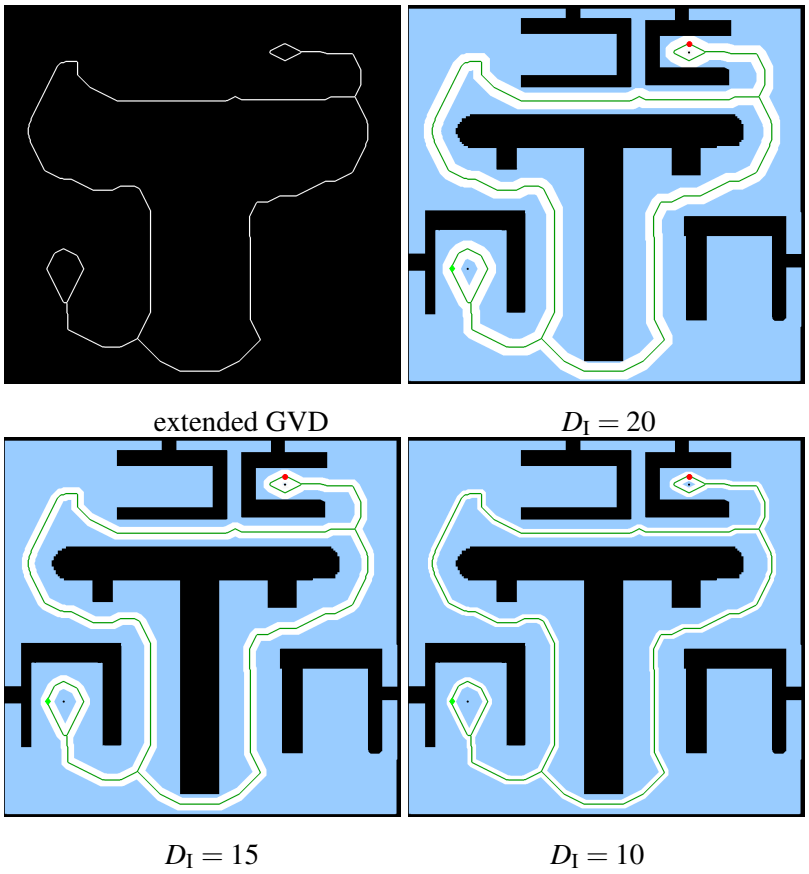

FIGURE 13. U-map: extended Voronoi diagram and inflated Voronoi boundary.

The main observations from this experiment are summarized as follows.

- The proposed algorithms lead to a reduced computation time compared to the existing algorithms except for the VD algorithm, which generates a very long path.
- The proposed algorithms allow adjusting path safety by selecting an appropriate value of D_{I}. For this environment, it has to be mentioned that the CRT algorithm generates safe paths with a similar path length as the proposed algorithms. Nevertheless, the CRT algorithm still leads to a larger computation time and significant variations in the minimum distance from the obstacle region. The main reason is that the CRT algorithm generates random samples that can be more or less close to the obstacle region depending on the found confidence values. On the other hand, the node samples of the proposed algorithms are restricted to the inflated Voronoi boundary such that the lower bound for D_{min} is well-defined.

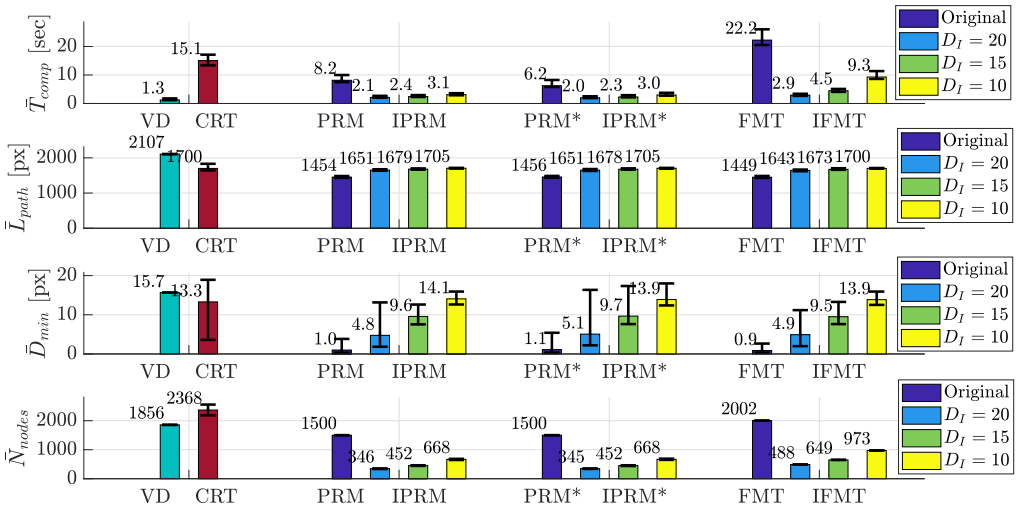

FIGURE 14. Comparison of the performance metrics for the U-map.

We further note that the solution paths in Fig. 15 confirm the observations from the previous sections and solution paths of CRT might come close to obstacles.

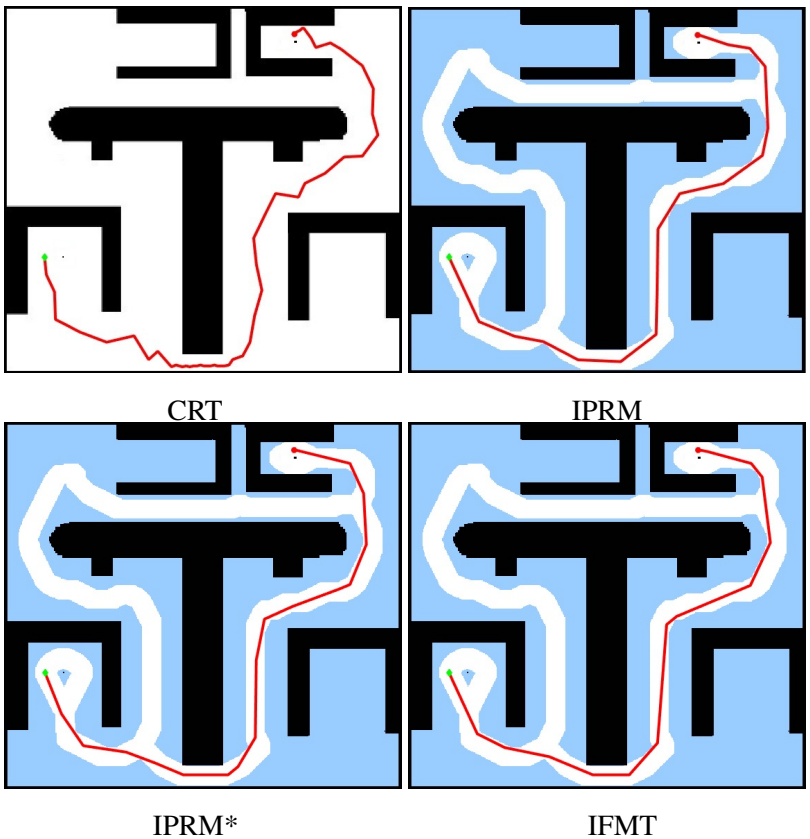

FIGURE 15. Solution paths for the U-map.

4.5. Maze Map with Spiral

We finally investigate the maze map in 6 (d) that has a very long solution path. The extended GVD and the inflated Voronoi boundary for this map are shown in Fig. 16 for $D_{\mathrm{I}} = 8$, $D_{\mathrm{I}} = 6$ and $D_{\mathrm{I}} = 4$, whereas Fig. 17 depicts the computational results for this map.

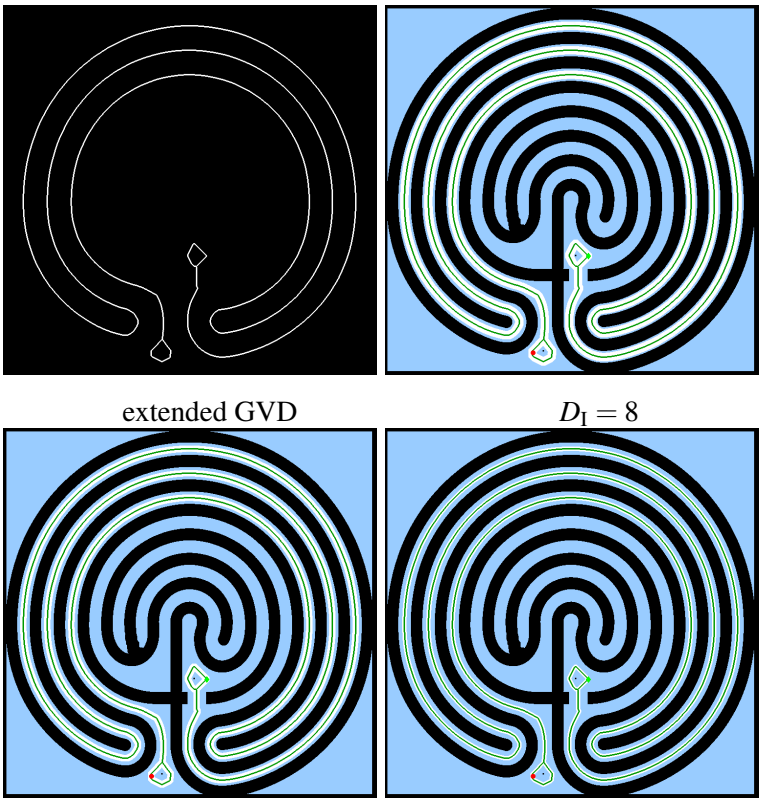

FIGURE 16. Polygon map: extended GVD and inflated Voronoi boundary.

We next describe the main observations from this experiment.

- The proposed algorithms mostly lead to significantly smaller computation times compared to the existing algorithms. It can only be observed that a very small value of D_{I} should be avoided due to the increase in the required number of nodes.
- For this map with a narrow space between obstacles, the proposed algorithms enable a significant increase of path safety without much increase in the path length compared to the classical methods. In addition, the proposed algorithms outperform the CRT algorithm in all performance metrics.

Finally, the solution paths in Fig. 18 again support the superiority of the proposed methods compared to CRT.

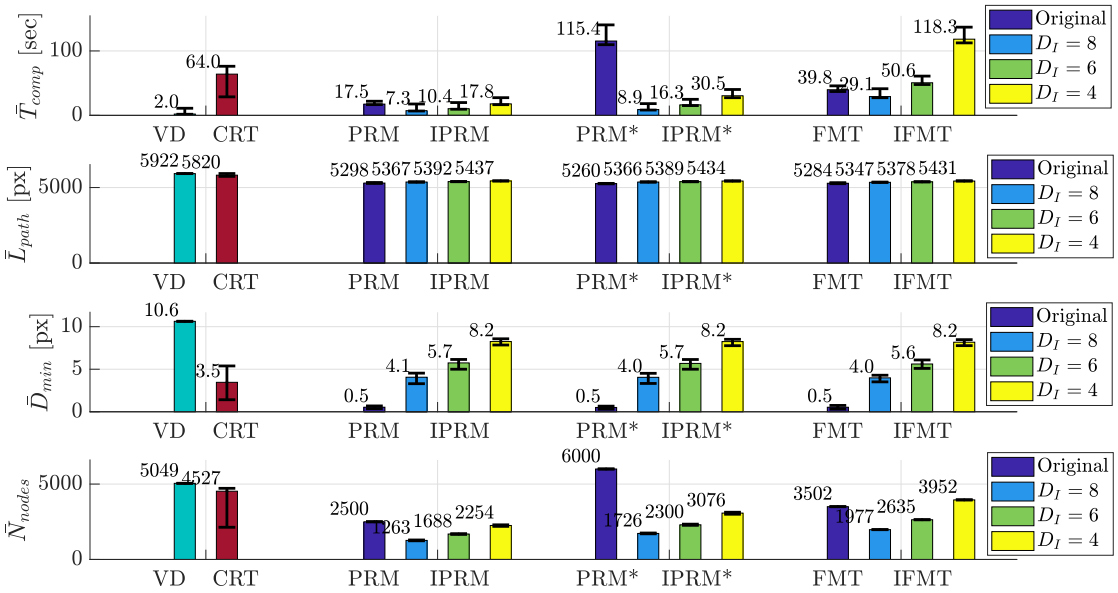

FIGURE 17. Comparison of the performance metrics for the map in Fig. 6 (d).

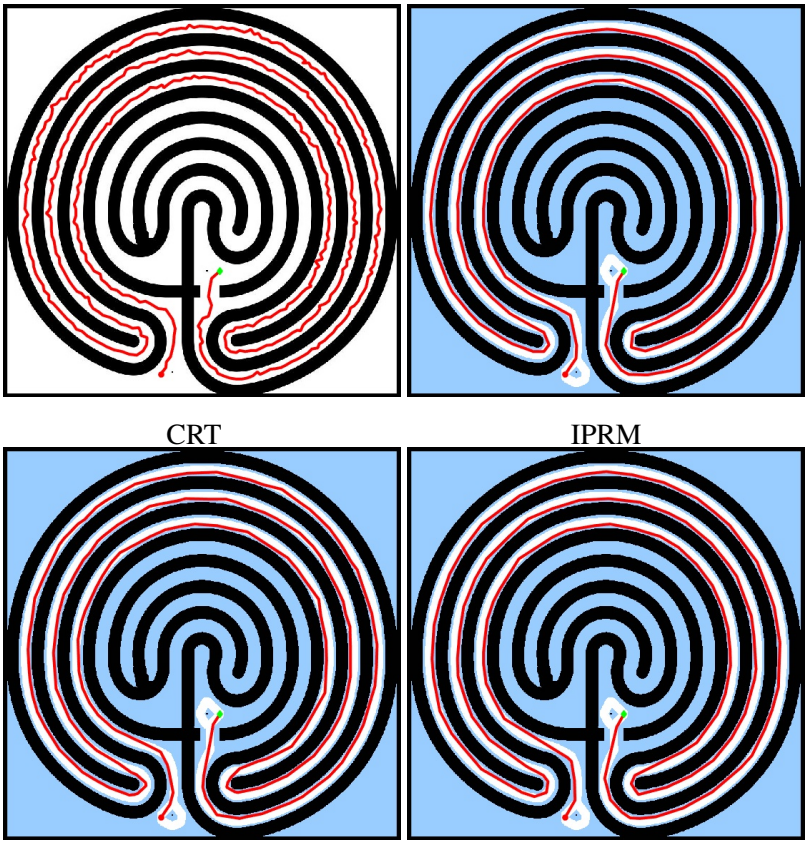

FIGURE 18. Solution paths for the maze map with spiral.

4.6. Discussion

Overall, the computational experiments for different environments indicate that the proposed methods are superior to both existing sampling-based methods such as PRM [11], PRM* [12] and FMT [14] as well as recent methods for path safety such as CRT [9]. In particular, the proposed IPRM algorithm outperforms the original PRM algorithm, the proposed IPRM* algorithm outperforms the original PRM* algorithm and the proposed IFMT algorithm outperforms the original FMT algorithm regarding both computation time and path safety while accepting a slight increase in path length. Moreover, all the proposed algorithms (IPRM, IPRM* and FMT*) lead to a reduced computation time and path length while providing comparable path safety as the CRT algorithm. When comparing the proposed algorithms IPRM, IPRM* and IFMT, it can be observed that all of these algorithms provide similar results regarding computation time, path length and path safety. It is only the case that the IFMT algorithm leads to an increased computation time in case of a small width of the inflated Voronoi boundary and for environments with very long solution paths.

In this context, it has to be noted that the proposed algorithms benefit from confining the generated node samples to the inflated Voronoi boundary. This helps avoiding the exploration of irrelevant regions of $\mathscr{C}_{\text{free}}$. Moreover, this ensures the generation of reliable solution paths with small variations in the path length, minimum distance from obstacles and computation time.

5. Conclusions

The subject of this paper is the path planning problem for mobile robots in a two-dimensional configuration space with obstacles. That is, the presented work specifically addresses the case of omni-directional robots that can perform turning maneuvers on the spot.

As the main contribution, the paper proposes three new path planning algorithms that are extensions of the PRM (probabilistic roadmap) algorithm, PRM* algorithm and FMT (fast marching tree) algorithm. The underlying idea for defining the new algorithms is to first compute a generalized Voronoi diagram (GVD) of the robot environment. The Voronoi boundary of this GVD is then inflated by a certain width. Each of the stated algorithms (PRM, PRM* and FMT) is adapted such that node samples are only generated on the inflated Voronoi boundary and node connections lie fully within the inflated Voronoi boundary. The resulting algorithms are denoted as IPRM (Inflated PRM), IPRM* (Inflated PRM*) and IFMT (Inflated FMT). As a particular feature, the proposed algorithms require fewer nodes when determining a solution path and ensure a minimum distance to obstacles by appropriately choosing the width of the inflated Voronoi boundary.

The proposed algorithms were evaluated by computational experiments with different environments. In these experiments, it was confirmed that the proposed methods IPRM, IPRM* and IFMT outperform the existing methods PRM, PRM* and FMT regarding computation time and

path safety at a slight increase of the path length. Moreover, a comparison with the recent confidence random tree (CRT) algorithm that specifically addresses path safety was performed. This comparison indicates that the proposed algorithms are significantly faster, generate shorter paths and lead to a comparable path safety. Furthermore, large variations of these performance metrics that are observed for the CRT algorithm can be avoided for the proposed algorithms since solution paths are confined to the inflated Voronoi boundary. As an important result of the paper, we conclude that it is preferable to apply proven algorithms such as PRM, PRM* or FMT on pre-processed environment maps instead of designing specific algorithms such as CRT for the original environment map.

Based on these observations, our future work will focus on additional methods for pre-processing environment maps and the extraction of map properties such as the density of obstacles.

References

[1] Z. Kingston, M. Moll, L. E. Kavraki, Sampling-Based Methods for Motion Planning with Constraints, *Annual Review of Control, Robotics, and Autonomous Systems*, **1**(1), (2018), 159–185.

[2] M. M. Costa, M. F. Silva, A Survey on Path Planning Algorithms for Mobile Robots, *IEEE International Conference on Autonomous Robot Systems and Competitions*, (2019), 1–7.

[3] A. Khan, I. Noreen, Z. Habib, On Complete Coverage Path Planning Algorithms for Non-holonomic Mobile Robots: Survey and Challenges, *Journal of Information Science and Engineering*, **33**, (2017), 101–121.

[4] A. S. H. H. V. Injarapu, S. K. Gawre, A survey of autonomous mobile robot path planning approaches, *International Conference on Recent Innovations in Signal Processing and Embedded Systems*, (2017), 624–628.

[5] T. T. Mac, C. Copot, D. T. Tran, R. De Keyser, Heuristic approaches in robot path planning: A survey, *Robotics and Autonomous Systems*, **86** (2016), 13–28.

[6] P. Corke, Robotics, vision and control: fundamental algorithms in MATLAB 2nd ed, *Springer*, **(118)**, (2017).

[7] H. M.Choset, S.Hutchinson, K. M.Lynch, G.Kantor, W. Burgard, Kavraki, L. E., S. Thrun, Principles of robot motion: theory, algorithms, and implementation. *MIT press*, (2005).

[8] P. Bhattacharya, M. L.Gavrilova, Roadmap-Based Path Planning Using the Voronoi Diagram for a Clearance-Based Shortest Path, *IEEE Robotics & Automation Magazine*, **15**(2), (2008), 58–66.

[9] N. Y, D. Kim, W. I. Ko, H. Suh, Confidence random tree-based algorithm for mobile robot path planning considering the path length and safety, *International Journal of Advanced Robotic Systems*, **16**(2), (2019), 1–10.

[10] L. E. Kavraki, P. Svestka, J. C. Latombe, M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Transactions on Robotics and Automation*, **12**(4), (1996), 566-–580.

[11] L. Kavraki, J.Latombe, Randomized preprocessing of configuration for fast path planning, *IEEE International Conference on Robotics and Automation*, **3**, (1994), 2138–2145.

[12] S. Karaman, E.Frazzoli, Sampling-based algorithms for optimal motion planning, *The International Journal of Robotics Research*, **30**(7), (2011), 846-–894.

[13] S. M. LaValle, J. J. Kuffner, Randomized Kinodynamic Planning, *The International Journal of Robotics Research*, **20**(5), (2001), 378–400.

[14] L. Janson, M. Pavone, Fast Marching Trees: A Fast Marching Sampling-Based Method for Optimal Motion Planning in Many Dimensions, *Robotics Research, Springer Tracts in Advanced Robotics*, **114**, (2016).

[15] B. K. Patle, G. Babu L, A. Pandey, D. R. K. Parhi, A. Jagadeesh, A review: On path planning strategies for navigation of mobile robot, *Defence Technology*, **15**(4), (2019), 582–606.

[16] D. A. L. Garcla, F. Gomez-Bravo, Vodec: A fast Voronoi algorithm for car-like robot path planning in dynamic scenarios, *Robotica*, **30**(7), (2012), 1189-1201.

[17] B. B. K. Ayawli, X. Mei, M. Shen, A. Y. Appiah, F. Kyeremeh, Mobile Robot Path Planning in Dynamic Environment using Voronoi Diagram and Computation Geometry Technique, *IEEE Access*,(2019), 86026-86040.

[18] M. R. H. Al-Dahhan, M. M. Ali, Path tracking control of a mobile robot using fuzzy logic, *In 2016 13th International Multi-Conference on Systems, Signals and Devices (SSD)*, (2016), 82–88.

[19] L. Gang, J. Wang, PRM path planning optimization algorithm research, *Wseas Transactions on Systems and control*, (2016), (11), 81-86.

[20] I. B. Jeong, S. J. Lee, J. H. Kim, Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate, *Expert Systems with Applications*, (2019), 82-90.

[21] T. Bai, Z. Fan, M. Liu, S. Zhang, R. Zheng, Multiple Waypoints Path Planning for a Home Mobile Robot, *In 2018 Ninth International Conference on Intelligent Control and Information Processing (ICICIP) IEEE*, (2018), 53–58).

[22] K. Yang, Anytime synchronized-biased-greedy rapidly-exploring random tree path planning in two dimensional complex environments, *International Journal of Control, Automation and Systems*, (2011), **9**(4), 750.

[23] H. Dong, W. Li, J. Zhu, S. Duan, The path planning for mobile robot based on Voronoi diagram, *In 2010 Third International Conference on Intelligent Networks and Intelligent Systems*, (2010), 446–449.

[24] M. Foskey, M. Garber, M. C. Lin, D. Manocha, A Voronoi-based hybrid motion planner, *IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium*, (2001), 55–60.

[25] E. Masehian, M. R. Amin-Naseri, A voronoi diagram-visibility graph-potential field compound algorithm for robot path planning. *Journal of Robotic Systems*, {bf 21}(6). (2004), 275–300.

[26] Q. Wang, M. Wulfmeier, B. Wagner, Voronoi-Based Heuristic for Nonholonomic Search-Based Path Planning, *Intelligent Autonomous Systems 13. Advances in Intelligent Systems and Computing*, **302**. (2016), 445–458.

[27] E. W. Dijkstra, A Note on Two Problems in Connection with Graphs.*Numerische Mathematik*. **1**, (1959), 269–271.

[28] S. Garrido, L. Moreno, M. Abderrahim, F. Martin, Path planning for mobile robot navigation using voronoi diagram and fast marching, *In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems IEEE*, (2006), 2376–2381).

[29] M. R. H. Al-Dahhan, K. W. Schmidt, Path Planning Based on Voronoi Diagram and PRM for Omnidirectinal Mobile Robots, *II. International Conference on Digital Transformation and Smart Systems*, (2019).

[30] M. ALANDES, Comparison among different sampling-based planning techniques, Politedcnco Di Milano university, (2015).

[31] K. E. Hoff, III, J. Keyser, M. Lin, D. Manocha, T. Culver, Fast computation of generalized Voronoi diagrams using graphics hardware. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, (1999), 277–286.

[32] D. Ji, J. Cheng, B. Wang, Path planning for mobile robots in complex environment via laser sensor, *Chinese Control And Decision Conference*, (2018), 2715-2719.