Robotica (2025), 1–24
doi:10.1017/S0263574725000281

REVIEW ARTICLE

Comparative analysis of popular mobile robot roadmap path-planning methods

Ben Beklisi Kwame Ayawli[1], John Kwao Dawson[1], Esther Badu[2], Irene Esinam Beklisi Ayawli[3] and Dawda Lamusah[1]

[1]Department of Computer Science, Sunyani Technical University, Sunyani, Ghana
[2]The ICT Directorate, Sunyani Technical University, Sunyani, Ghana
[3]Department of Pharmacy, Sunyani Technical University, Sunyani, Ghana
Corresponding author: Ben Beklisi Kwame Ayawli; Email: bbkayawli@yahoo.com

Received: 21 April 2024; **Revised:** 29 November 2024; **Accepted:** 4 February 2025

Keywords: voronoi diagram; rapidly exploring random trees; probabilistic roadmap; roadmap; mobile robot; path planning

Abstract
Global path planning using roadmap (RM) path-planning methods including Voronoi diagram (VD), rapidly exploring random trees (RRT), and probabilistic roadmap (PRM) has gained popularity over the years in robotics. These global path-planning methods are usually combined with other path-planning techniques to achieve collision-free robot control to a specified destination. However, it is unclear which of these methods is the best choice to compute the efficient path in terms of path length, computation time, path safety, and consistency of path computation. This article reviewed and adopted a comparative research methodology to perform a comparative analysis to determine the efficiency of these methods in terms of path optimality, safety, consistency, and computation time. A hundred maps of different complexities with obstacle occupancy rates ranging from 50.95% to 78.42% were used to evaluate the performance of the RM path-planning methods. Each method demonstrated unique strengths and limitations. The study provides critical insights into their relative performance, highlighting application-specific recommendations for selecting the most suitable RM method. These findings contribute to advancing robot path-planning techniques by offering a detailed evaluation of widely adopted methods.

1. Introduction

Robotics as a branch of Artificial Intelligence aims to build embodied intelligence systems to perform tasks in structured and unstructured environments [1]. Robot path planning, a crucial component of robotics, includes choosing the best route for a robot to take from its current location to a specified objective location while avoiding obstacles and abiding by a variety of limitations.

Research is advanced in achieving intelligent navigation and task performance of robots. Recent advancements in robotic path planning have introduced various innovative approaches that address the challenges of navigating complex environments. For example, the study of Boscariol et al. [2] focused on developing and refining path-planning algorithms for special robotic operations, where robots are deployed in complex environments requiring high precision, safety, and efficiency. Orozco-Rosas et al. [3] also proposed Q-learning artificial potential field technique to address path-planning problem.

Various path-planning methods categorized into nature-inspired computation, conventional, and hybrid methods were used by researchers to address path-planning problems [4]. The popular category of global path-planning methods is roadmap (RM) path-planning methods including probabilistic roadmap (PRM), rapidly exploring random trees (RRT), and Voronoi diagram (VD) [5, 6]. PRM is particularly notable for its efficiency in exploring large configuration spaces by randomly sampling the environment, as discussed by Santiago et al. [7]. RRT have gained popularity for their ability to quickly explore vast and unstructured spaces, making them ideal for real-time applications [8–13]. VD is a spatial partitioning method used in robotics for efficient path planning, dividing space based on proximity to predefined points, and enhancing navigation by complementing other techniques [14]. Another global path-planning method apart from PRM, VD, and RRT is the visibility graph (VG) [5]. The VG technique involves constructing a graph where nodes represent key positions (such as the start, goal, and obstacle vertices), and edges connect nodes that have a direct line-of-sight, allowing for the computation of the shortest path in environments with well-defined obstacles.

To provide efficient path-planning algorithms, most researchers explore hybrid methods. Some of the popular proposed hybrid methods considered RM path-planning techniques including PRM, VD, and RRT. For example, in ref. [5], VD was combined with morphological dilation to propose an algorithm for mobile robot path planning in complex and dynamic environments. Also, an extreme learning machine (ELM) was combined with improved RRT in ref. [15] to address path-planning problems. Moreover, Qiao et al. [16] combine artificial potential field and PRM to address path-planning problems in complex environments. The integration of recent methodologies with these established RM techniques underscores the ongoing evolution of path-planning strategies. Despite the popularity and successes chucked by the integration of RM path-planning methods with others over the years compared to other global path-planning methods, the challenge of achieving intelligent path planning in complex environments persists [17, 18]. It remains a significant challenge to determine the appropriate choice of RM path-planning methods to integrate with other methods to compute a safe and optimal path with minimal computational time [4, 19–21].

To contribute to research in this area, this article reviewed and performed a comparative analysis of the popular RM path-planning methods including PRM, RRT, and VD to assess the safety of the computed path, path optimality, and the computation time in environments with different complexities and obstacle occupancy rate.

The main contributions of this review paper are summarized as follows:

- To perform a comparative analysis of PRM, RRT, and VD RM path-planning methods in terms of path length, computation time, and path safety.
- Demonstration of the stability of PRM, RRT, and VD path-planning methods across various environmental complexities and obstacle densities.
- Identification of specific scenarios where each method excels, offering insights into the selection of integration of appropriate path-planning methods for different applications.
- Proposal of future improvements for each method based on observed limitations.

The following sections of this article are organized to first provide an overview of the RM path-planning methods (Section 2), followed by a detailed description of the methodology used in this study (Section 3). The comparative analysis results are presented in Section 4, and the article concludes with a discussion of the findings and suggestions for future work in Section 5.

2. Overview of RM path planning

Mobile robots operate in a workspace W made up of obstacles W_{obs} of different shapes. The rest of the W is described as the free workspace W_{free}. Robot navigation is expected in the W_{free}. A RM can be described as a set of edges and points such that for a given initial and goal position in W_{free}, a path can be connected. RM is made up of a set of paths between two given points in the W where the robot can navigate without colliding with obstacles. Figure 1(a) and (b) shows samples of workspace with obstacles, Start, and Goal positions. This collection of paths can be stored as a graph with vertices and edges $\mathcal{G}(V, E)$ [22]. From each point on the RM, a goal point can be accessed and connected.

Algorithm 1. Dijkstra algorithm

```
Input: $\mathcal{G}(V, E)$, initial position $s$, goal position $g$
output: vertices and distance of the shortest path
1:  $Q = \{\}$ //nodes not visited
2:  for each $v \in \mathcal{G}$
3:      $dist[v] \leftarrow \infty$ //current distance unknown
4:      $prev[v] \leftarrow \varnothing$ //previous node visited
5:      $Q = Q \cup v$
6:  end for
7:  $dist[s] \leftarrow 0$; //distance from source to source
8:  while $Q \neq \varnothing$
9:      $u \leftarrow mindist(Q, dist[v])$ //vertex with the minimum distance
10:     $Q = Q - u$ //remove u from the queue
11:     For all $uofv$ //for each neighbor of current node
12:        $p \leftarrow dist[u] + length(u, v)$;
13:        if $p < dist[v]$
14:           $dist[v] \leftarrow p$; //shorter distance found
15:           $prev[v] \leftarrow u$; //vertices of shorter path
16:        end if
17:     end for
18:  end while
19:  return $dist, prev$;
```

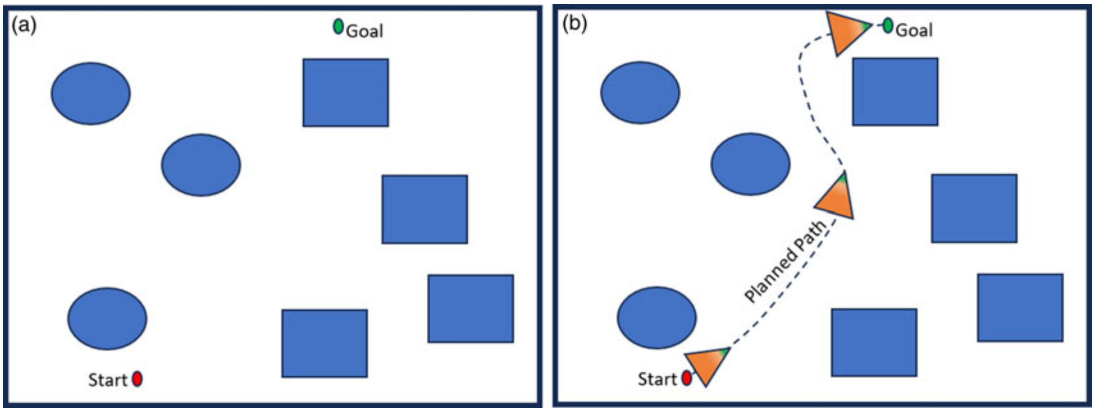

Figure 1. (a) Configuration space with obstacles, start, and goal positions (b) Expected path for a mobile robot in the configuration space.

There is, therefore, a path in $RM \in W_{free}$ from $V_{start} \in W_{free}$ to some $V' \in W_{free}$, and there is a path from some $V'' \in W_{free}$ to $V_{goal} \in W_{free}$. There exists a path in RM between v' and v''. RM path-planning involves using algorithms to build RM in the W_{free} of the workspace of the robot after which a path is computed between the initial point of the robot to the goal point. Search algorithms including Dijkstra's, theta*, D*, and A* are then used to compute an efficient path from the initial position to the goal position using the graph generated by the RM method. Given $\mathcal{G}(V, E)$ of the RM with the initial point s, and goal point g, Algorithms 1 and 8 outline the Dijkstra's and A* algorithms that could be used to generate the path for RM path-planning methods, respectively.

RM path-planning methods can be classified into two namely: geometry-based and sampling-based (SB) methods. These methods are discussed in the next sections.

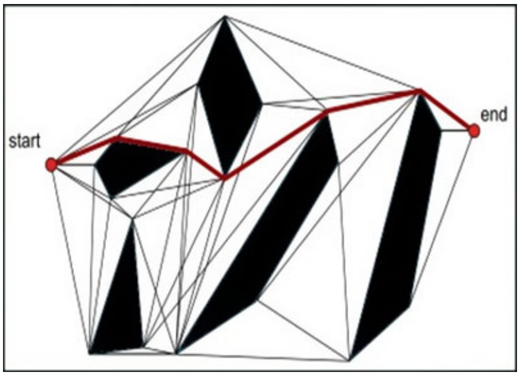

Figure 2. *Visibility graph with the path from the start to the goal points.*

3. Geometry-based roadmap path-planning methods

Geometry-based roadmap (GBRM) path-planning methods employ geometry computation techniques to compute the vertex and edges of the RM based on the $W_{obs}$and W_{free} of the workspace of the robot. Other constraints are also considered in generating the RM to ensure the safety and quality of the RM. The popular GBRM methods used by researchers over the years to address path-planning problems include the VG [23] and the VD [24]. The most popular of these methods is the VD.

3.1. VG path-planning method

In a polygonal configuration space, a VG is used to generate points and edges in W_{free} where connections are made to join all visible points to form a RM. The initial and goal positions are included as points for the connection [22]. Two points on a VG are connected, provided visibility can be established between them. Considering two arbitrary data values (x_i, y_i) and (x_j, y_j) to have visibility between them, a connection between these points can be established if any other data point (x_k, y_k) inserted between (x_i, y_i) and (x_j, y_j) satisfy Eq. (1) [25].

$$y_k < y_j + \left(y_i - y_j\right) \frac{x_j - x_k}{x_j - x_i} \tag{1}$$

Given point p, obstacles W_{obs}, and v_i as the vertices of all the polygonal obstacles in the workspace of the robot, we can obtain the visible vertices for the graph using Algorithm 2 provided in [26]. Algorithm 3 outlines the algorithm to build the VG RM after identifying the visible vertices using Algorithm 2.

After the RM is generated, the path planner tries to obtain a path from the initial position to the goal position on the RM. A sample VG with a path from the initial position to the goal is presented in Fig. 2.

VG has been used in path-planning research over the years to address various path-planning problems. Li et al. [23] present a VG-based path-planning algorithm with quadtree representation that efficiently reduces search space and computation time but faces challenges in maintaining accuracy in complex environments due to the quadtree's simplified representation. In ref. [27], the authors propose a long-voyage route planning method for autonomous ships using a multiscale VG that successfully balances computational efficiency and route safety over long distances but faces challenges in handling dynamic environmental changes and integrating real-time data. The study by Blasi et al. [28] introduces a path-planning approach with real-time collision avoidance using an essential VG, which effectively navigates dynamic environments but faces challenges in keeping the graph updated and accurate in highly dynamic settings. Also, in ref. [29], the authors proposed a VG-based path-planning method that effectively navigates dynamic environments with moving obstacles, but it faces significant challenges in managing computational overhead during real-time scenarios with frequent and unpredictable changes.

Algorithm 2. Determining Visible vertices for the visibility graph

```
Input: Set of polygonal obstacles $W_{obs}$, point $p$ in $W_{free}$ and vertices of obstacles $v_i$
output: Set of visible vertices from $p$
1:  Sort obstacles in the clockwise direction where half-line $\rho$ from $p$ to each vertex $v_i$ with the
2:   positive x-axis. The vertices should be sorted in order of ascendency
    Obtain $C_{obs}$ edges $e$ that intersect $\rho$ and store in the balanced search tree $\tau$ in order.
3:  $V \leftarrow \varnothing$;
4:  for $i \leftarrow 1$ to $n$
5:      if $visible(v_i)$
6:        Add $v_i$ to $V$;
7:        if $e$ incident to $v_i$ and clockwise direction of the half-line from $p$ to $v_i$
8:          Add $e$ to $\tau$;
9:        else
10:         Delete $e$ from $\tau$;
11:       end if
12:     end if
13:  end for
14:  return $V$;
```

Algorithm 3. Visibility graph roadmap

```
Input: A set of disjoint of polygonal obstacles $W_{obs}$, set of visible vertices $v_i$
output: Visibility graph
1:  $E = \varnothing$;
2:  $V \leftarrow \varnothing$;
3:  For all $v \in V$
4:      $V \leftarrow VisibleVertices(v_i, W_{obs})$;
5:        for each $v_i \in V'$
6:          $E \leftarrow (v_i, v_{i+1})$;
7:        end for
8:  end for
9:  return $\mathcal{G}(V, E)$;
10:  Delete $e$ from $\tau$;
```

The VG method requires the environment and the vertices of the obstacles to be well-defined. To mimic the real environment for robot navigation, the environment and obstacles used in this study are not well-defined. Hence, the VG was not included in the comparative analysis.

3.2 Voronoi diagram path-planning method

Another popular GBRM path-planning method considered by most researchers is the VD. VD was named after Georgy Fedosievyeh Voronoy, a Russian mathematician who is believed to have first defined VD. VD is also called Voronoi tessellation, Voronoi partition, Voronoi decomposition, Dirichlet tessellation or Thiessen polygons [30]. The VD method is applicable in many fields of study especially in science and technology [31, 32]. It has also been considered in path-planning research over the years [5, 24, 33].

In robot path planning, VD partitions the workspace of the robot into several regions whose edges form a RM. Each Voronoi region is closer to its generating node than to any other generating node. Any node in each region is closer to the generating obstacles than to any other obstacle [22]. The nearest

Algorithm 4. Voronoi diagram roadmap

```
Input: points in the plane $S(n)$ where $n \geq 1$
output: Edges and vertices of the VD
1:  $Q = S(n)$; //Initialize the priority queue of points in the plane with all sites
2:  $L \leftarrow$the list of regions (labelled by sites) and boundaries (labelled by pair of sites) containing
    convex and unbounded polygon with point $p$, $R_p$
3:  while $Q \neq \varnothing$
4:    $p \leftarrow \min(Q)$;
5:    if $p$ is site
6:      Find occurrence of region $R_q$ on $L$ containing $p$
7:      Create bisector $B_{pq}$
8:      Update $L \leftarrow R_q, C^-_{pq,} R_{p,} C^+_{pq,} R_q$;
9:      Delete from $Q$ the vertex between then left and right boundary of $R_q$, if any exist
10:     Insert into $Q$ the vertex between $C^-_{pq}$ and its neighbor to the left on $L$, if any and the vertex
        between $C^+_{pq}$ and its neighbor to the right if any
11v   else
12:     $p \leftarrow$ vertex of boundaries $C_{qr}$ and $C_{rs}$
13:     Obtain the bisector $B_{qs}$
14:     Update $L \leftarrow C_{qS} = C^-_{qs} or C^+_{qs}$;
15:     Delete from $Q$ any vertex between $C_{qr}$ and its neighbors to the left and between $C_{rs}$and its
        neighbor to the right.
16:     Insert any vertex between $C_{qs}$ and its neighbors to the left or right into $Q$
17:     Mark $p$ as a vertex and as an endpoint of $B_{qr,} B_{rs}$ and $B_{qs}$
18:   end if
19: end while
```

neighbor rule is used to obtain edges equidistant from the nodes. Considering a configuration space, S with $S=\{s_1,s_2,s_3,\ldots,s_i\}$ nodes, d as the distances function, and N indicating a set of nodes in S, the Voronoi region is made up of a set of nodes N_i in S with distances less or equal to the distances to other nodes N_j where $N_i \neq N_j$. The VD can, therefore, be computed using Eq. (2).

$$V_d = \{s\epsilon S | d\,(x, N_i) \leq d\left(x, N_j\right) \forall_{i\neq j}\} \tag{2}$$

The *VD* could be computed using Fortune's VD algorithm given in Algorithm 4 [5]. In the algorithm, L represents a sequence of regions and boundaries, and Q is a priority queue of nodes in the plane. Every node is labeled as a site or as the vertex of a pair of boundaries of given regions. It should be noted that elements of Q may not be unique. Details of the algorithm are given in ref. [5].

After the VD RM is generated, a search algorithm is used to compute a path from the initial position to the goal. Figure 3 demonstrates a generated VD RM and a path connecting the start position to the goal using the A* search algorithm.

Recently, VD has been combined with other techniques to propose path-planning methods. In ref. [5], the authors propose a VD combined with computational geometry for mobile robot path planning in dynamic environments, achieving real-time adaptability, though the method struggles with computational complexity in rapidly changing scenarios. Ayawli et al. [5] introduced a morphological dilation VD RM algorithm for mobile robot path planning, aiming to enhance obstacle avoidance and efficiency, with results showing improved path smoothness but challenges in handling highly cluttered environments. Also, in ref. [34], the author combines the VD with ant colony optimization for autonomous mobile robot path planning, resulting in efficient and adaptive paths, but faces challenges with convergence speed in complex environments. Kim et al. [35] presented a real-time path-planning

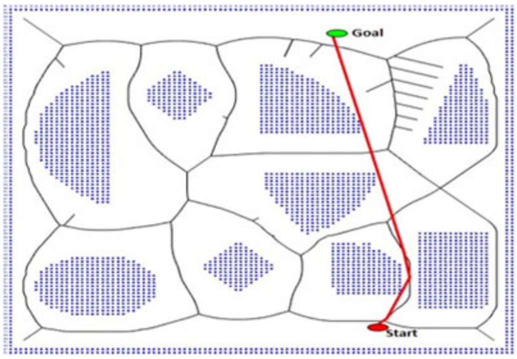

Figure 3. *Voronoi diagram roadmap with path from start to the goal using A* algorithm.*

method for unmanned aerial vehicles (UAVs) using a compensated VD to improve navigation accuracy, achieving reliable obstacle avoidance but encountering difficulties in highly dynamic airspaces. Moreover, Xu in ref. [36] proposed a Voronoi graph-based path inference method for unmanned maritime vehicles, focusing on efficient navigation in maritime environments, with positive results in route optimization but challenges in handling unpredictable maritime conditions. Additionally, Su et al., [37] enhanced multi-UAV path planning using Voronoi-based obstacle modeling and Q-learning, achieving improved navigation in complex environments, though the method is challenged by the need for extensive computational resources for real-time applications.

4. Sampling-based RM path-planning methods

In generating a RM using SB methods, random points are generated and sampled after which collision detections are done to ensure that sample points are in the W_{free} [38, 39]. Connected edges among these points form the RM. Search algorithms are then applied to connect the initial and goal positions to obtain the nodes for the path. SB methods have been employed successfully for robot path planning [40]. The well-known SB path-planning methods are PRM [41] and RRT [42]. These methods are described as possessing the ability to address trajectory-shaping problems of traditional path-planning methods in complex environments [43] with SB methods. There is no need to consider approximations even in complex systems [44]. PRM and RRT methods are discussed in the next sections.

4.1. Probabilistic RM path planning

PRM generates a RM of the environment by randomly sampling the space and connecting the samples to form a graph. PRM path is computed by generating random points in W_{free} of the workspace of the robot and a collision-free path is computed to link the start and the goal positions [22, 45, 46]. Path computation using PRM is made up of two stages: learning and query stages [41].

At the learning stage, the RM in the form of an undirected graph $\mathcal{G}(V, E)$ is constructed with V representing the generated random nodes and E representing the edges of the connected nodes. Algorithm 5 outlines how the RM using the PRM method is computed [47]. Figure 4 demonstrates a sample RM and path generated from point S to G using the PRM method.

At the query stage, a path is computed to connect the initial position of the robot and the goal positions based on the computed RM at the learning stage. Search algorithms including Dijkstrar's and A* algorithms are mostly used to obtain the shortest path on the RMs. Algorithm 6 outlines how to construct a path based on PRM using A* heuristic algorithm [47].

PRM is based on the assertion that a few numbers of points and a RM are enough to provide complete connectivity in the W_{free} by reducing the time of the path-planning process [47]. This makes the PRM method simple to implement and supports fast path queries. PRM is also probabilistically complete such that once there is a path on the RM, it can be computed. There are variants of PRM methods proposed

Algorithm 5. PRM roadmap learning stage algorithm

```
Input: Configuration space with defined $W_{free}$
output: PRM roadmap $\mathcal{G}(V, E)$
1:  Vertex $V \leftarrow \{\}$;
2:  Edge$E \leftarrow \{\}$;
3:  loop
4:      $p \leftarrow$randomly select the useful configuration from $W_{free}$
5:      $V \leftarrow V \cup \{p\}$;
6:      $V_p \leftarrow$A set of adjacent nodes $p$ selected from V;
7:      for all $p_{next} \in V_p$ in order of increasing$distance(p, p_{next})$
8:          if $not_{connected(p,p_{next})}$
9:              if $path_{found(p,p_{next})}$
10:                 $E \leftarrow E \cup \{(p, p_{next})\}$ //Add edge
11:             end if
12:         end if
13:     end for
14: end loop
```

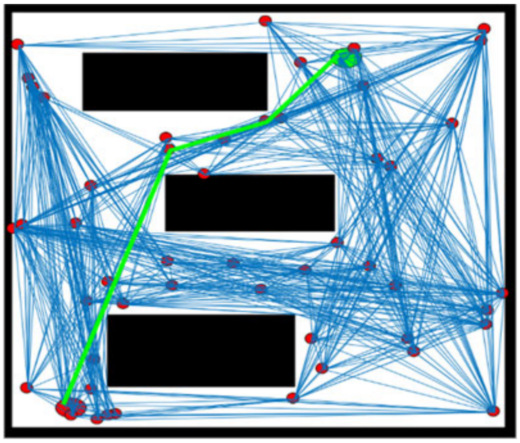

Figure 4. *Road map and path from point S to G using PRM method.*

over the years to improve its performance. Some of these variants include obstacle-based PRM [48], Bridge Test PRM [49], Gaussian PRM [50], medial axis PRM [51], Lazy toggle PRM [52], T-PRM, [53] Dancing PRM [54], and LD-PRM [55].

Recently, path-planning research using PRM combined with other techniques is becoming popular. For instance, Singh [56] modified PRM using a decision-making strategy for optimizing unmanned vehicle trajectories in unstructured environments, achieving time-efficient navigation but facing challenges related to unpredictable environmental complexity and dynamic obstacles. In ref. [57], Fan et al. presented a hierarchical path planner that combines PRMs with deep deterministic policy gradients to navigate unmanned ground vehicles with nonholonomic constraints, leading to enhanced navigation accuracy but encountering difficulties in real-time adaptability. Also, in ref. [58], the authors integrated belief space planning with PRMs to ensure reachability and consistency, offering a robust planning framework but confronting the challenge of computational cost in high-dimensional belief spaces. In ref. [59], an improved PRM method for robot path planning in narrow passages was presented, yielding more efficient pathfinding but facing obstacles in balancing computational efficiency and optimality in highly constrained environments.

Algorithm 6. PRM path computation at the query stage

```
Input: Roadmap using PRM method $\mathcal{G}(V, E)$
output: PRM Path
1:  $Open \leftarrow \varnothing$; //Path is empty
2:  $Closed \leftarrow \varnothing$; //Nodes visited
3:  $Open \leftarrow Open \cup p_s$; //Add the initial position to the path list
4:  $prev \leftarrow p_s$;
5:  $g(p_s) \leftarrow 0$;//cost from source to source
6:  $f(p) \leftarrow h(p_s, p_g)$; //heuristic cost estimate
7:  while $Closed \neq \varnothing$
8:      $p_{0 \leftarrow} \underset{p \in V}{\arg\min} f(p)$
9:      if $p_0 == p_g$
10:       $return\ RECONSTRUCT(prev, p_g, Closed)$;
11:       end if
12:       $Closed \leftarrow Closed - p_0$; //Remove current node from list
13:       $Open \leftarrow Open \cup \{p_0\}$; //Add current node to path node list
14:       $N \leftarrow NEIGHBORS(p_0)$; //Get the neighbors of $p_0$
15:       for all $p_{next} \in N$
16:           if $p_{next} \in Open$
17:             $g \leftarrow g[p_{next}] + dist(p_0, p_{next})$;
18:           end if
19:           if $p_{next} \in Closed$
20:             $Closed \leftarrow Closed \cup \{p_{next}\}$;
21:           else if
22:             $g \geq g[p_{next}]$;
23:             $continue$;
24:           end if
25:           $prev \leftarrow p_{next}$;
27:           $f[p_{next}] \leftarrow g[p_{next}] + h(p_{next}, p_g)$;
28:           $returnprev, f$;
29:     end for
30:   $Open \leftarrow p$;
31:   end while
```

4.2. Rapidly exploring random trees path-planning method

RRT path-planning method is noted to be suitable for path planning in high-dimensional configuration space, and it involves a random building of a tree from a given point termed as the root [38, 60]. The tree grows with random samples from the root toward the unsearched W_{free}, until the target or the given maximum iteration limit is reached [61]. Two main stages are followed alternatively to grow the tree. These include selection and propagation stages. During the selection stage, a sample, u_{rand} is generated at random and the nearest sample, u_{near} to u_{rand} is selected. The next stage is the propagation stage where an edge is extended from the u_{near} toward the u_{rand}. The ending point of the edge which is determined by the distance threshold to the u_{near} is added to the tree. The two stages are performed alternatively until the goal or the maximum iteration limit is reached [38, 62]. Algorithm 8 shows how to generate the RRT RM as given in ref. [63]. The growth of the tree could be computed to grow from two points towards each other to connect a branch of one to the other. The growth process ends when there is a connection between branches from each point. This type of RRT is termed as bidirectional RRT.

Algorithm 7. RRT roadmap algorithm

```
Input: Configuration space with defined $W_{free}$
output: RRT roadmap
1C   $T.Init(q_{init})$; //Start the tree from the initial position (root)
2:   for all $k = 1 to K$; //Setting the iteration limit K
3:       $q_{rand} \leftarrow RandomConfig()$;
4:       $q_{near} \leftarrow T.\ NearestNeighbor(q_{rand})$;
5:       if $q_{new} \leftarrow T.\ Connect(q_{rand}, q_{near})$;
6:           $T.\ AddVertex(q_{new})$;
7:           $T.\ AddEdge(q_{near}, q_{new})$;
8:       end if
9:   end for
10:  return $T$
```

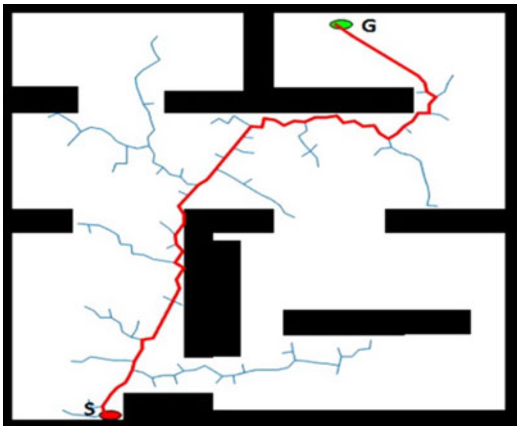

Figure 5. *Road map and path from point S to G using RRT.*

Just like other RM methods, the generation of the tree forming the RM as indicated in Algorithm 7 is done at the learning stage. Other algorithms are applied at the query stage to obtain the path from the initial position to the goal position. Figure 5 is a sample RM and path computed using the RRT method.

The RRT method is simple to implement. It is also probabilistic complete. However, it is believed to produce suboptimal solutions with poor path quality [60].

RRT method is believed to be a fast path-planning method that performs efficiently in complex and high-dimensional workspace [64, 65].

To improve the RRT method, variants of these methods were proposed. Some of these include RRT* [66], RRT^{x,} TG-RRT* [67], RRT-A* [68], parallel RRT*, RRT-Edge, Liveness-based RRT, Theta*-RRT, human-guided RRT*, ORRT-A* [11], ELMs-improved RRT (ELM-IRRT) [15], continuum-RRT* [69], and continuous bidirectional Quick-RRT* [70] planning methods. Each of these variants tried to address a particular path-planning problem using RRT with other techniques.

Path-planning research involving the RRT algorithm is gaining attention. Yu in ref. [71] introduced the RDT-RRT algorithm that combines real-time double-tree and rapidly exploring random tree (RRT) methods for autonomous vehicle path planning. The proposed algorithm improved computational speed and efficiency, but faces challenges with complex dynamic environments. The Improved Rapidly Exploring Random Trees Star (RRT*) algorithm was presented by Wang and Zheng [72]. It enhanced robot path planning by increasing exploration speed and solution quality, though it struggles with high dimensionality. Also, in ref. [73], a hybrid RRT* algorithm was proposed by Ganesan et al for mobile navigation. The method was good at merging SB methods with optimization for smoother path

Algorithm 8. A star algorithm

Input: $\mathcal{G}(V, E)$, initial position s, goal position g
output: vertices and distance of the shortest path

```
1:  $S = \{\}$; //nodes visited
2:  $Q = \{\}$; //nodes not visited
3:  $prev \leftarrow \varnothing$; //previous node visited
4:  $gCost \leftarrow \infty$; //cost from start node to another node
5:  $gCost[s] \leftarrow 0$; //cost from source to source
6:  $fCost \leftarrow \infty$; //cost from source to goal
7:  $fCost[s] \leftarrow h(s, g)$; //heuristic cost estimate
8:  while $Q \neq \varnothing$
9:      $v \leftarrow mindist(Q, fCost)$; //current node is the node with the lowest cost
10:     if $v == goal$
11:         return $path(prev, v)$;
12:     end if
13:     $Q \leftarrow Q - \{v\}$; //remove current node from list
14:     $S \leftarrow S \cup \{v\}$; /add current node to nodes visited
15:     for all $u of v$ //for each neighbor of current node
16:         if $u \in S$
17:             $prov_{gCost} = gCost[v] + dist(v, u)$;//provisional gCost
18:         end if
19:         if $u \notin Q$
20:             $Q \leftarrow Q \cup \{u\}$;
21:         else if $prov_{gCost} \geq gCost[u]$
22:             continue;
23:         end if
24:         $prev[u] = u$;
25:         $gCost[u] = prov_{gCost}$;
26:         $fCost[u] = gCost[u] + h(u, g)$;
27:         return $prev, fCost$;
28:     end for
29: end while
```

planning, but it is limited by environmental complexity. Moreover, the RRT-A* hybrid algorithm presented by Ayawli et al. [11] optimizes path planning in partially known environments for mobile robots, achieving reduced computational time but facing obstacles in highly dynamic settings.

In summary, RM path-planning methods, including PRM, RRT, and VG, offer unique advantages and challenges in navigating complex environments. Details of the advantages and challenges of each of these methods are essential for making the appropriate integration of other modern methods. The next section will describe the methodology employed to evaluate these methods, focusing on their performance in various scenarios.

5. Methodology

This section describes the methodology used for the comparative analysis of PRM, RRT, and VD path-planning methods.

A comparative research methodology was adopted in this research to determine the performance of popular RM path-planning methods (VD, RRT, and PRM) in terms of path optimality, computation time, path safety, and the stability of path computation. Therefore, the metrics considered for the comparison include computed path length, path computation time, safety of the computed path, and the consistency of path computation for each of the methods under consideration. An optimal path is achieved if a shorter path is computed to enable a robot to reach its goal as fast as possible. The time required to compute a path is very important and it is expected that a chosen path-planning method computes a path in the shortest time to enable the robot to make the necessary decisions during navigation to avoid colliding with obstacles or missing its goal. The safety of the computed path is also very important, hence its consideration as a metric in the analysis. The safety of the computed path in this research is determined by the distance between the computed path and obstacles. This is required to ensure that the systemic and nonsystemic conditions of the robot during navigation do not negatively affect the safe navigation of the robot. In as much as we require the shortest path and the shortest time, a computed path should make the provision for a robot to navigate freely from its start point to the goal point without colliding with obstacles considered during the global path computation despite the systemic and nonsystemic challenges of the robot. Hence, the higher the average distance between the path and the obstacles, the safer the computed path and vice versa.

Although VG, a geometric-based RM has also been used in the past for global path planning, it has not been included in this comparison because VG requires that the dimensions (vertices) of obstacles positions in the environment are known before path computation [32]. In real situations with multiple and complex environments with obstacles with different shapes, it is difficult to determine the dimensions of obstacle positions in the environment before global path computation. Hence, VG cannot be used in a complex environment with different shapes and a higher occupancy rate of obstacles.

This research considers 100 maps of different complexities with obstacle occupancy rates ranging from 50.95% to 78.42% to evaluate the performance of the RM path-planning methods. The obstacle occupancy rates of the maps were obtained using Eq. (3)

$$OO_{rate} = \left(\frac{obs_{px}}{map_{px}}\right) * 100, \tag{3}$$

where OO_{rate} represents obstacle occupancy rate, obs_{px} indicates the map obstacle pixels, and $map_{px} =$ the total pixel of the map represents the environment of the robot. Each of the maps considered was 500X500 pixels in size and was used in previous research work [5, 11].

A RM path-planning method usually relies on search algorithms to achieve efficient path computation. In this research, A* algorithm indicated in Algorithm 8 was used for each of the three methods for the path computation.

To ensure fairness in the comparison, each of the three methods was implemented on the same platform and datasets. Simulations were conducted on a machine with an Intel Core(TM) i7-7700HQ CPU @ 2.80 GHz, 8 GB RAM, running Windows 10. The path-planning algorithms were implemented and tested using MATLAB R2018b.

Unlike VD and RRT methods, PRM requires the initialization of nodes to be used for the path computation. It is believed that the higher the number of nodes, the better the path length to be obtained but the computation time increases. To ensure fairness in the comparison, the computed average nodes of VD and RRT were used for computing the path as indicated in Eq. (4).

$$PRM_n = \frac{VD_n + RRT_n}{2}, \tag{4}$$

where PRM$_n$ represents the nodes to be used by the PRM method for the path computation. VD$_n$ and RRT$_n$ represent the nodes used by the VD and the RRT in the path computation, respectively.

The methodology outlined provides a robust framework for evaluating the selected RM path-planning methods. The next section presents the results of the evaluation, highlighting key differences in performance among PRM, RRT, and VG.

Table I demonstrates the performance of VD, RRT, and PRM in terms of the average nodes (*aNode*) used for the path computation, average path length (*aLength*) obtained, average computation time

Table I. *Results of path computation performance of VD, RRT, and PRM of 10 trials each of 100 different maps.*

Trials	Method	aNode	aLength	aTime	*p–o* Distance
1	VD	1491	503.9087	57.009	**15.2401**
	RRT	100	528.7141	**0.1806**	1.4562
	PRM	796	**490.9773**	19.1926	4.968
2	VD	1491	503.9087	56.89	**15.2401**
	RRT	100	536.3141	**0.1516**	1.4488
	PRM	796	**491.7518**	19.1699	5.052
3	VD	1491	503.9087	56.7105	**15.2401**
	RRT	98	528.8236	**0.1343**	1.1597
	PRM	795	**491.3648**	19.0378	4.4513
4	VD	1491	503.9087	56.763	**15.2401**
	RRT	97	526.2076	**0.1439**	1.719
	PRM	794	**492.0318**	19.1842	4.5217
5	VD	1491	503.9087	57.1313	**15.2401**
	RRT	102	539.1652	**0.1572**	1.5294
	PRM	797	**492.0633**	19.3196	5.1045
6	VD	1491	503.9087	56.9904	**15.2401**
	RRT	101	536.4024	**0.1572**	1.3558
	PRM	796	**490.8728**	19.2221	5.3659
7	VD	1491	503.9087	56.7078	15.2401
	RRT	100	534.3364	**0.1537**	**1.0579**
	PRM	796	**491.5721**	19.1773	5.0958
8	VD	1491	503.9087	57.0134	**15.2401**
	RRT	101	535.7263	**0.1527**	1.2844
	PRM	796	**491.9498**	19.2442	4.6622
9	VD	1491	503.9087	56.6632	**15.2401**
	RRT	**97**	524.2242	**0.1471**	1.6503
	PRM	794	**491.0625**	19.1065	5.0132
10	VD	1491	503.9087	57.0898	**15.2401**
	RRT	99	525.9661	**0.149**	1.2914
	PRM	795	**490.9949**	19.2214	4.786

(*aTime*), and the average path–obstacle (*p–o*) distance for each trial of the 100 maps used in the simulation.

6. Results analysis

This section presents the results of our comparative analysis, based on the methodology described in Section 6. The findings provide insights into the relative strengths and weaknesses of PRM, RRT, and VG in different environments.

The experiments were performed on 100 maps of varying complexities. To validate the consistency of the results, each simulation was run ten times. The metrics used for the comparison included path length, computation time, and path safety, measured as the path–obstacle (p–o) distance.

Figures 6–10 present the results of the computed path for VD, RRT, and PRM. Figure 6 shows computed path lengths of VD, RRT, and PRM in a map with an occupancy rate of 55%. PRM obtained the shortest path length of 542.76 in 9.9 s while RRT obtained the longest path length of 618 in 0.2 s. It

Table II. *A summary of path computation performance of VD, RRT, and PRM.*

Method	Node	pLength	pLength%	Time	Time%	p–o dist	p–dist %
VD	1491	503.91	33	56.90	74.6	**15.24**	**70.75**
RRT	99	531.59	34.8	**0.15**	**0.2**	1.40	6.50
PRM	795	**491.46**	**32.2**	19.19	25.2	4.90	22.75

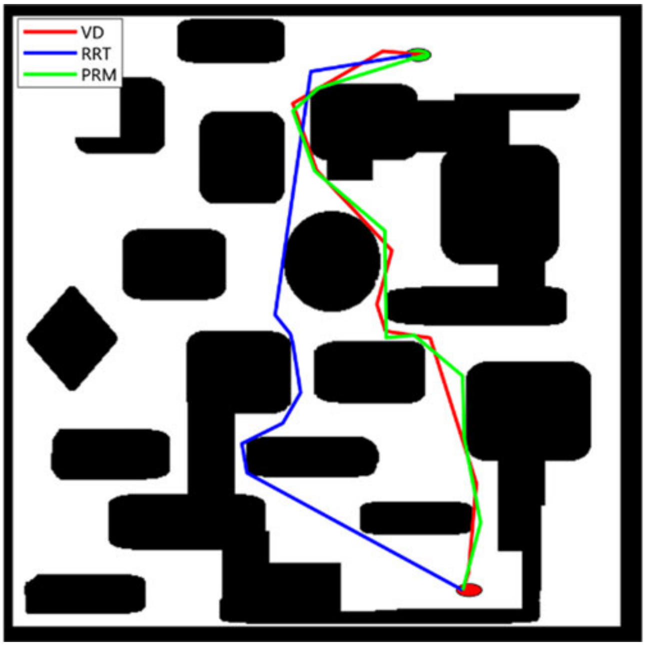

Figure 6. *Generated path in an environment with an occupancy rate of 55% with VD obtaining path length of 562, RRT 618.36, and PRM 542.76.*

can be noted that, though RRT obtained the longest path, it used the shortest time in its path computation. In Fig. 7, the shortest path length of 461.44 was computed by PRM in 13.89 s while RRT computed the longest path length of 518.91 in 0.1 s. Among the three methods, RRT computed its path in the shortest time. Figure 8 demonstrates the shortest path length of 485.57 computed by the VD in the longest time of 65.77 s while RRT recorded the longest path of 537.97 in the shortest path of 0.11 s. Figures 9 and 10 illustrate the shortest path length computed by PRM with path lengths of 500.81 and 440.63 in 31.64 and 30.71 s, respectively. RRT obtained the shortest computation time of 0.11 s but recorded the longest path length of 479.9 in Fig. 10. In Fig. 9, VD recorded the longest path length of 514.41.

The following observations can be derived from Table I.

- PRM obtained the shortest average path lengths of 490.98, 491.75, 491.36, 492.03, 492.06, 490.87, 491.57, 491.95, 491.06, and 490.99 for each of the 10 trials, respectively. RRT recorded the longest path lengths ranging from 524.22 to 536.31 for each of the trials.
- Concerning performance in computation time, RRT obtained the shortest computation time ranging from 0.13 to 0.18 s for each of the trials. Though the VD method performed better compared to RRT in terms of path length computation, it recorded the longest computation time ranging from 56.66 to 57.13 s.

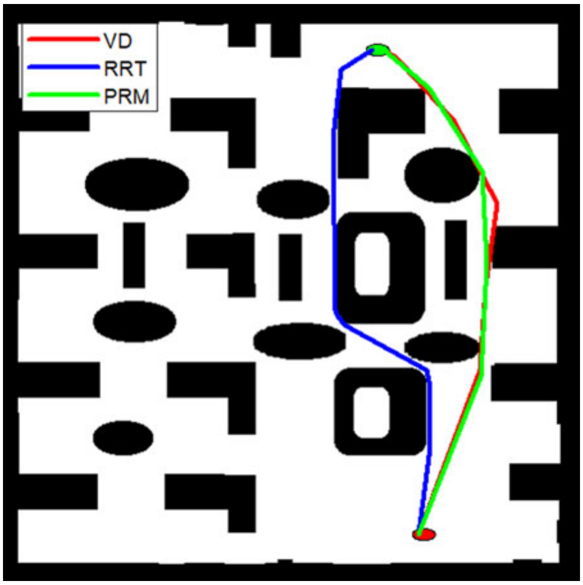

Figure 7. *Generated path in an environment with an occupancy rate of 42.9% with VD obtaining path length of 471.11, RRT 518.91, and PRM 461.44.*

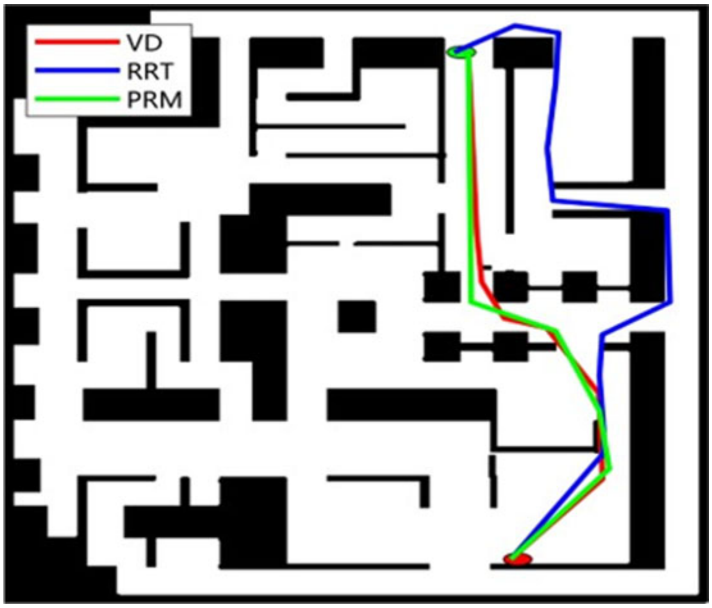

Figure 8. *Generated path in an environment with an occupancy rate of 43.2% with VD obtaining path length of 485.57, RRT 537.97, and PRM 488.21.*

- *p-o* distance determines the safety of the computed path and vice versa. In Table II, VD recorded a stable and the highest p–o distance value of 15.24 for each trial. However, RRT recorded the lowest distance ranging from 1.05 to 1.72 for each of the trials.

Figure 11 illustrates the path lengths obtained by VD, RRT, and PRM for each map and the 1000 trials. It can be deduced that PRM demonstrated the best performance in terms of shortest path computation, while RRT demonstrated the worst performance of the three methods.

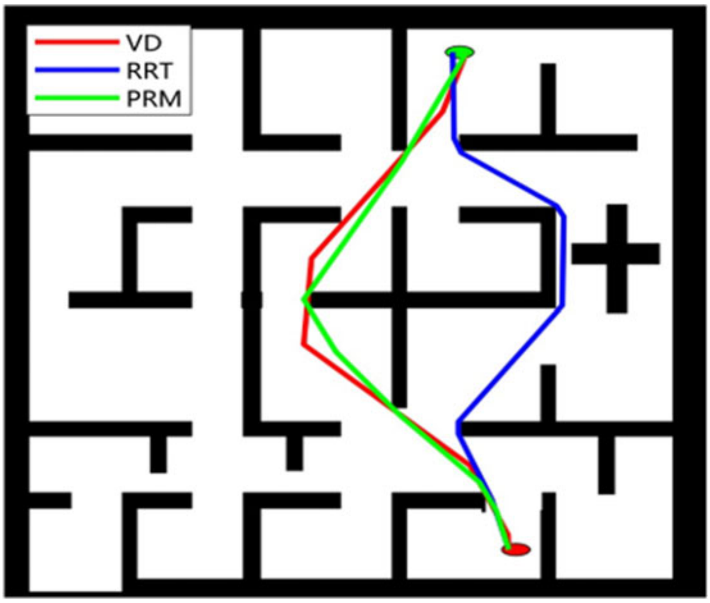

Figure 9. *Generated path in an environment with an occupancy rate of 33.8% with VD obtaining path length of 514.41, RRT 512.12, and PRM 500.81.*

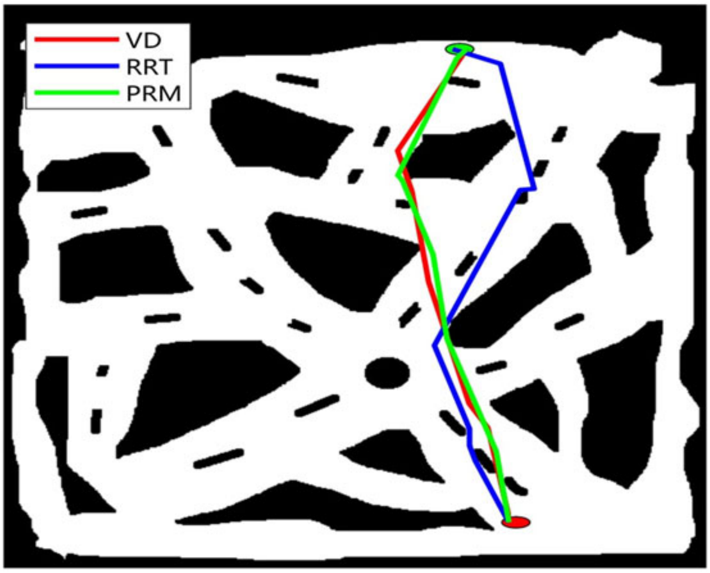

Figure 10. *Generated path in an environment with an occupancy rate of 44.9% with VD obtaining path length of 453.12, RRT 479.9, and PRM 440.63.*

Figure 12 shows the computation time recorded by VD, RRT, and PRM for each map and each trial. Figure 13 demonstrates the expanded chart to reveal the detailed computation time obtained by RRT for each of the trials as shown in Fig. 12. Results demonstrate the best performance of RRT recording less than 2.5 s for each of the trials on each map. However, VD obtained the highest computation time, with some recording computation time of more than 100 s.

Detailed *p–o* distances to determine the safety of the computed path is shown in Fig. 14. It can be observed that VD recorded the highest and most consistent *p–o* distance for each of the maps and trials.

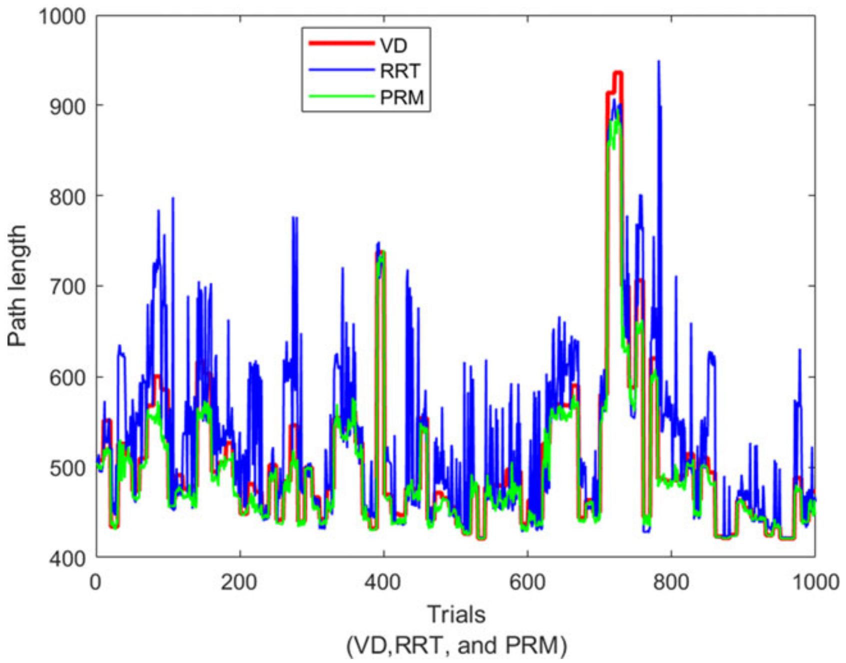

Figure 11. *Detailed path length obtained by VD, RRT, and PRM for each of the 1000 trials.*

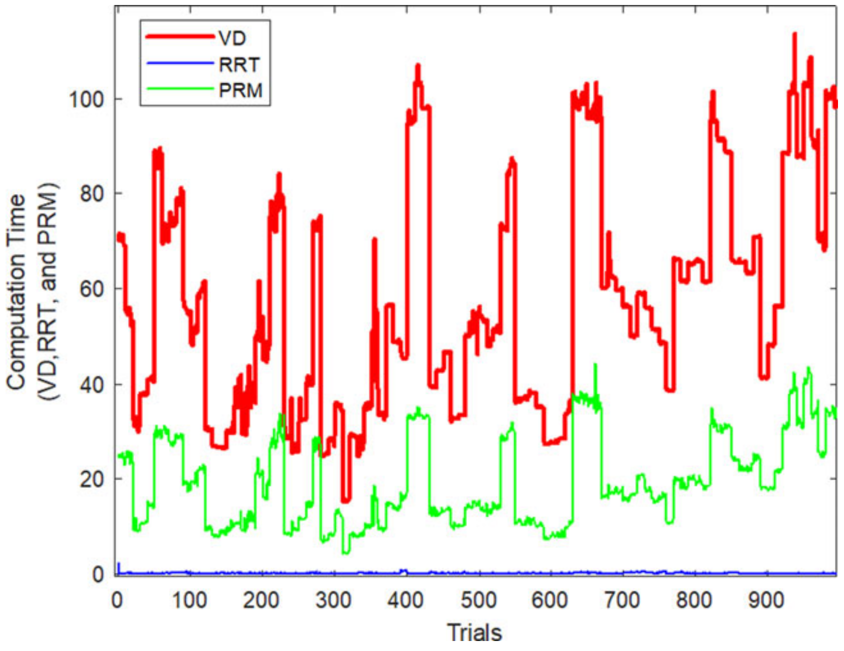

Figure 12. *Computation time by VD, RRT, and PRM for each of the 1000 trials.*

The lowest *p–o* distance value was recorded by the RRT method. It could be noted that while the *p–o* distance of VD was consistent, RRT and PRM were inconsistent.

To determine the stability in path computation of VD, RRT, and PRM, the standard deviation of the computed path lengths and computation times for the ten trials for each of the 100 maps were computed for comparison. The standard deviation of the computed path lengths is demonstrated in Fig. 15. It can be observed that while the variation of path computation remains 0 for VD demonstrating its path

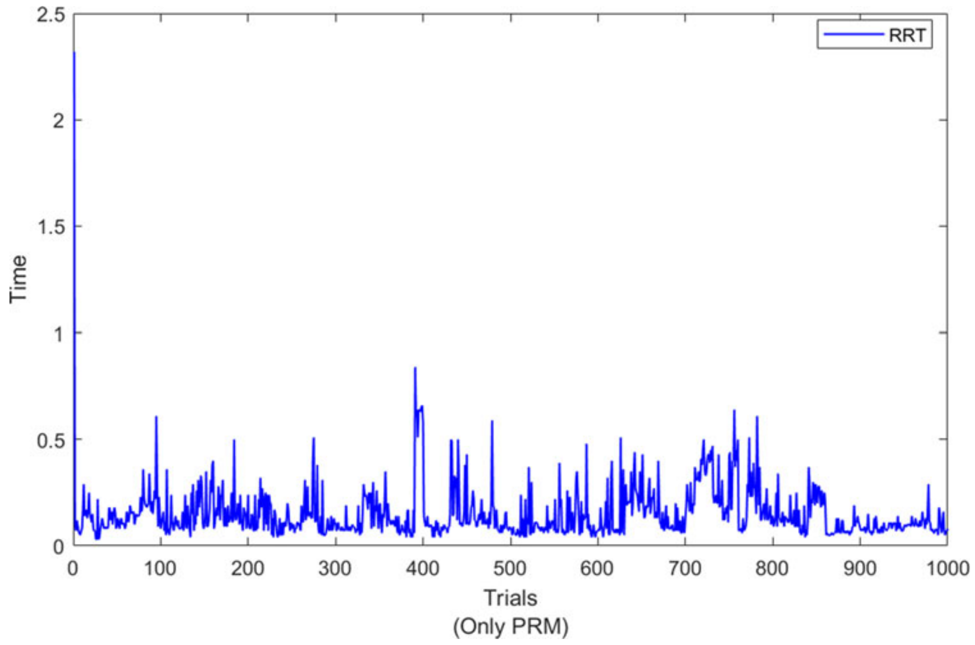

Figure 13. *Expanded chart showing the computation time by RRT for each of the 1000 trials.*

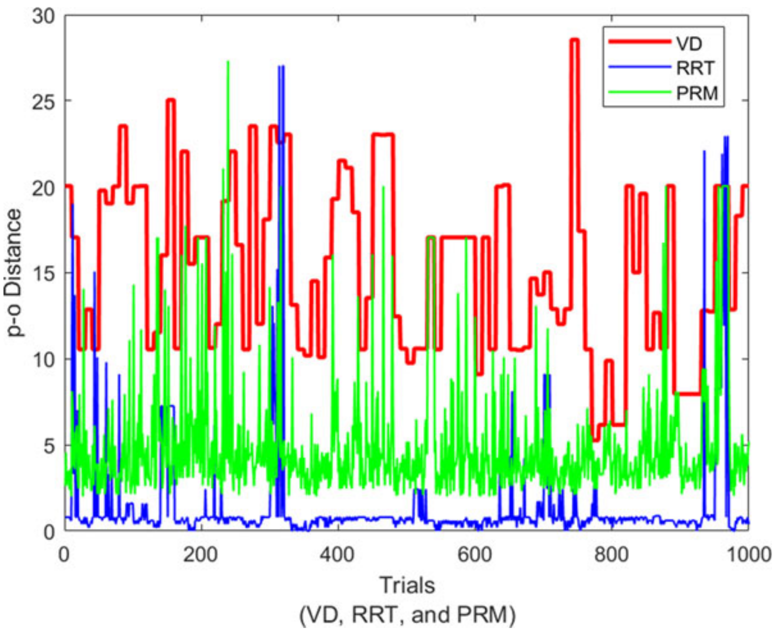

Figure 14. *Average p–o distance of VD, RRT, and PRM for each trial.*

computation consistency, RRT and PRM have varied values demonstrating their inconsistencies in path computation. The method with the highest instability on path length computation was RRT. In terms of variations in computation time, Fig. 16 demonstrates RRT to be the most consistent while VD is the most inconsistent. The standard deviation of path computation time demonstrated in Fig. 16 shows that VD obtained the highest computation time variation indicating time instability in path computation while RRT showed the highest stability in its computation time.

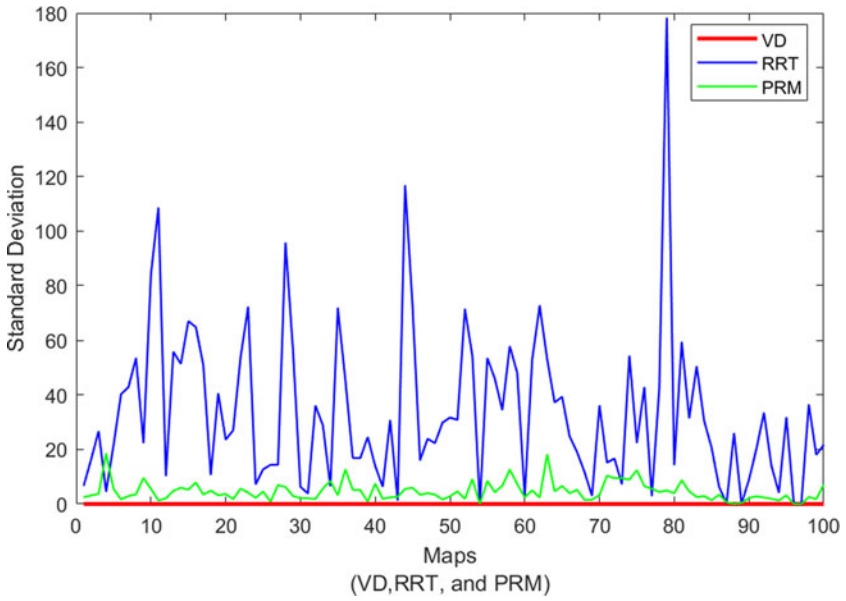

Figure 15. *Path length computation variation of VD, RRT, and PRM.*

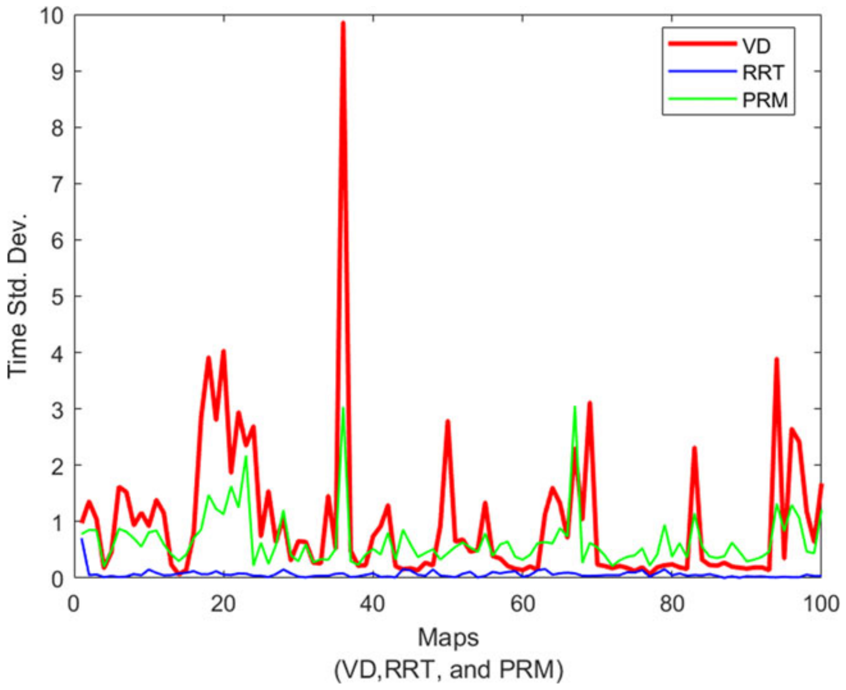

Figure 16. *Computation time variation of VD, RRT, and PRM.*

In summary, Table II demonstrates PRM to be the most efficient RM planning method in terms of path length computation by recording the overall average value of 491.46 (32.2%). The worst performance among the three methods was RRT which recorded a path length of 531.59 (34.8%). RRT recorded the best overall average computation time of 0.15 (0.2%). However, VD obtained the worst overall average computation time of 56.9 (74.6%). Regarding path safety, VD recorded the best overall average p–o distance of 15.24 (70.75%) while RRT recorded the worst *p–o* distance of 1.40 (6.50%).

Table III. *The advantages and disadvantages of PRM, RRT, and VD methods.*

Method	**Advantages**	**Disadvantages**
PRM	Shortest path length, effective in varied environments	Inconsistent path computation time, fixed node initialization
RRT	Fast computation time, adaptable to dynamic environments	Poor path safety, longest path length
VD	Highest path safety, stable performance	Long computation

Table IV. *The concise tradeoffs associated with PRM, RRT, and VD path planning methods.*

Criterion	**PRM**	**RRT**	**VD**
Path length	Shortest paths but inconsistent	Longest paths due to random exploration	Moderate paths, more consistent than RRT
Computation time	Moderate, varies with environment complexity	Fastest, ideal for real-time applications	Longest due to exhaustive search
Path safety	Moderate depends on node placement	Lowest paths often near obstacles	Highest consistently avoids obstacles
Stability	Dependent on environment and node distribution	High in time, low in path quality	Stable in safety, less so in computation time
Scalability	Scales well but may need tuning	Highly scalable, especially in large spaces	Less scalable demands high computation
Best application	Efficient paths where predictability is less critical	Real-time navigation with speed priority	Safety-critical scenarios, for example, avoiding collisions

The results show that the PRM method consistently computes the shortest paths, which is crucial for applications where path efficiency is paramount, such as autonomous delivery robots navigating complex urban environments. However, the variation in PRM's computation time suggests that it may not be suitable for time-sensitive operations where predictability and consistency are critical.

On the other hand, the RRT method, despite generating longer paths, excels in computation speed. This makes it highly suitable for applications requiring real-time decision-making, such as UAVs performing dynamic obstacle avoidance in rapidly changing environments. However, its lower safety margin indicates a potential risk in high-stakes scenarios, such as search and rescue operations in disaster-stricken areas, where collision avoidance is critical.

The VD method, which consistently produces the safest paths, demonstrates its utility in environments where safety is the primary concern. For instance, industrial robots operating in close proximity to humans could benefit from VD's higher p–o distance, ensuring safer operations even in densely populated or cluttered spaces.

After analyzing the performance of PRM, RRT, and VD across different environments, Table III summarizes the key advantages and disadvantages of each method. The concise tradeoffs associated with PRM, RRT, and VD methods using the criteria; path length, computation time, path safety, stability, scalability, and best application are illustrated in Table IV. This comparative overview highlights the strengths and weaknesses of each approach, providing a clear perspective on their suitability for various robotic path-planning scenarios.

Table IV provides a concise overview of the tradeoffs associated with each path-planning method. As shown, PRM is most effective in scenarios where path length efficiency is crucial, though it requires careful consideration of node initialization to ensure consistent results. RRT is distinguished by its rapid computation time, making it ideal for real-time applications, albeit at the expense of path safety. VD, while slower, excels in ensuring path safety, making it the preferred choice for environments where avoiding collisions is the top priority. These insights are critical for selecting the most appropriate path-planning method based on the specific requirements of a given application.

The comparative analysis reveals that each path-planning method has distinct advantages and limitations depending on the application context. PRM is ideal for scenarios requiring efficient path optimization but may falter in time-critical tasks. RRT's strength lies in its rapid computation, making it suitable for dynamic and unpredictable environments, though its path safety is a concern. VD provides the safest paths, which is crucial for applications where minimizing risk is essential, albeit at the cost of longer computation times. Using the VD method, the effects of systematic and nonsystematic noise of the robot would be minimal and would also avoid local minimal challenges. These findings highlight the importance of selecting the appropriate path-planning algorithm based on the specific needs of the application, whether it be speed, safety, or path efficiency.

The results of this comparative analysis align with recent advancements in the field of path planning, particularly regarding the tradeoffs between computation time, path optimality, and safety. For example, the findings on RRT's rapid exploration but suboptimal paths are consistent with the work of Xiao et al. [12], who noted similar challenges with standard RRT and addressed them using heuristic optimizations. Similarly, the path safety observed in VD aligns with the improvements proposed by Ayawli et al. [5], whose enhanced VD algorithm focused on increasing clearance from obstacles in densely cluttered environments. While the integration of machine learning techniques such as Learning from Demonstration (LfD) with PRM in refs. [15, 74] shows potential for improving adaptability in dynamic environments, our analysis of the baseline PRM method highlights its limitations in computational efficiency in such scenarios. These comparisons demonstrate the continued relevance of PRM, RRT, and VD in contemporary path-planning applications, while also suggesting areas where further enhancement is necessary.

7. Conclusions

This article presents a comprehensive comparative analysis of popular RM path-planning methods, including VD, RRT, and PRM, to evaluate their performance in global path planning. Key metrics considered in the evaluation include path length, computation time, path safety, and consistency of path computation.

The results demonstrate that PRM excels in computing the shortest path length, although its performance is hindered by the need for fixed node initialization and inconsistency in path length and computation time. RRT stands out as the fastest method for path computation, making it ideal for time-critical applications like mobile robots and UAVs, but it performs poorly in path safety. VD, despite having the longest computation time, is the most reliable for ensuring safe and consistent paths, making it suitable for applications where safety is paramount.

Each method has its limitations, underscoring the need for further improvements. Future research should focus on addressing PRM's fixed node initialization challenge by developing adaptive node generation algorithms, enhancing RRT's path safety and length without compromising speed, and reducing VD's computation time to make it more competitive. If these challenges are resolved, VD could emerge as the most efficient method due to its superior path safety and consistency. This study provides practical guidance for selecting RM methods based on application-specific priorities in robotics path planning.

Author contributions. All authors contributed equally to this article.

Financial support. No financial support.

Conflicts of interest. The authors declare no conflicts of interest exist.

Ethical approval. Not applicable.

References

[1] P. J. Bentley, *Artificial Intelligence and Robotics: Ten Short Lessons* (Johns Hopkins University Press, Baltimore, 2020). doi:10.1353/book.99599.

[2] P. Boscariol, A. Gasparetto and L. Scalera, "Path Planning for Special Robotic Operations," In: *Robot Design. Mechanisms and Machine Science* (G. Carbone and M. A. Laribi, eds.), vol. 123 (Springer, Cham, 2023). doi:10.1007/978-3-031-11128-0_4.

[3] U. Orozco-Rosas, K.Picos, J.J. Pantrigo, A. S. Montemayor and A. Cuesta-Infante, "Mobile robot path planning using a QAPF learning algorithm for known and unknown environments," *IEEE Access* **10**, 84648–84663 (2022). doi: 10.1109/ACCESS.2022.3197628.

[4] B. B. K. Ayawli, R. Chellali, A. Y. Appiah and F. Kyeremeh, "An overview of nature-inspired, conventional, and hybrid methods of autonomous vehicle path planning," *J. Adv. Transp.* **2018**, 1–27 (2018). doi: 10.1155/2018/8269698.

[5] B. B. K. Ayawli, A. Y. Appiah, IK. Nti, F. Kyeremeh and E. I. Ayawli, "Path planning for mobile robots using morphological dilation voronoi diagram roadmap algorithm," *Sci. Afr.* **12**, e00745 (2021). doi: 10.1016/j.sciaf.2021.e00745.

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.* **30**(7), 846–894 (2011). doi: 10.1177/0278364911406761.

[7] R. M. C. Santiago, A. L. de Ocampo, A. T. Ubando, A. A. Bandala and E. P. Dadios, "Path planning for Mobile Robots Using Genetic Algorithm and Probabilistic Roadmap," **In:** *IEEE 9th International Conference on Humanoid Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), Manila* (2017), pp. 1–5. doi: 10.1109/HNICEM.2017.8269498.

[8] W. G. Aguilar and S. G. Morales, "3D environment mapping using the kinect V2 and path planning based on RRT algorithms," *Electronics* **5**(4), 70 (2016). doi: 10.3390/electronics5040070.

[9] A. de Santana Correia, S. Kamarry, L. Molina, E.Á.N. Carvalho and E. O. Freire, "Wheeled Mobile Robotics From Fundamentals Towards Autonomous Systems," **In:** *Latin American Robotics Symposium (LARS) and Brazilian Symposium on Robotics (SBR), Curitiba* (2017) pp.1–6. doi: 10.1109/SBR-LARS-R.2017.8215282.

[10] Y. Tusi and H. Y. Chung, "Using ABC and RRT Algorithms to Improve Mobile Robot Path Planning with Danger Degree," **In:** *5th International Conference on Future Generation Communication Technologies (FGCT), Luton* (2016) pp. 21–26. doi: 10.1109/FGCT.2016.7605068.

[11] B. B. K. Ayawli, X. Mei, M. Shen, A. Y. Appiah and F. Kyeremeh, "Optimized RRT-A* path planning method for mobile robots in partially known environment," *Inf. Technol. Control* **48**(2), 179–194 (2019). doi: 10.5755/j01.itc.48.2.21390.

[12] G. Xiao, L. Zhang, T. Wu, Y. Han, Y. Ding and C. Han, "FBi-RRT: A path planning algorithm for manipulators with heuristic node expansion," *Robotica* **42**(3), 644–659 (2024). doi: 10.1017/S0263574723001674.

[13] S. Katiyar and A. Dutta, "Dynamic path planning over CG-Space of 1*0*DOF Rover with static and randomly moving obstacles using RRT* rewiring," *Robotica* **4**(8), 2610–2629 (2022). doi: 10.1017/S0263574721001843.

[14] K. Katona, H. A. Neamah and P. Korondi, "Obstacle avoidance and path planning methods for autonomous navigation of mobile robot," *Ah S Sens.* **24**(11), 3573 (2024).

[15] Z. Meng, S. Li, Y. Zhang and Y. Sun, "Path planning for robots in preform weaving based on learning from demonstration," *Robotica* **42**(4), 1153–1171 (2024). doi: 10.1017/S0263574724000146.

[16] L. Qiao, X. Luo and Q. Luo, "An optimized probabilistic roadmap algorithm for path planning of mobile robots in complex environments with narrow channels," *Ah S Sens.* **22**(22), 8983 (2022). doi: 10.3390/s22228983.

[17] B. Tao and J. H. Kim, "Mobile robot path planning based on bi-population particle swarm optimization with random perturbation strategy," *J. King Saud Univ.-Comput. Inf. Sci.* **36**(2), 101974 (2024). doi: 10.1016/j.jksuci.2024.101974.

[18] M. R. Rezaee, N. A. W. A. Hamid, M. Hussin and Z. A. Zukarnain, "Comprehensive review of drones collision avoidance schemes: Challenges and open issues," *IEEE Trans. Intell. Transp.* **25**(7), 6397–6426 (2024). doi: 10.1109/TITS.2024.3375893.

[19] J. Qi, H. Yang and H. Sun, "MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment," *IEEE Trans. Indust. Electron* **68**(8), 7244–7251 (2020). doi: 10.1109/TIE.2020.2998740.

[20] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi and I. Ahmedy, "RRT hybrid a semi-dual-tree RRT-based motion planner," *IEEE Access* **8**, 18658–18668 (2020). doi: 10.1109/ACCESS.2020.2968471.

[21] A. Francis, A. Faust, H. L. Chiang, J. Hsu, J. C. Kew, M. Fiser and T. E. Lee, "Long-range indoor navigation with PRM-RL," *IEEE Trans. Rob.* **1-20**(4), 1115–1134 (2020). doi: 10.1109/TRO.2020.2975428.

[22] G. Klančer, A. Zdešar, S. Blažič and I. Skrjanc, *Wheeled Mobile Robotics From Fundamentals Towards Autonomous Systems (Elsivier Inc* (Heinemann, Butterworth, 2017) pp. 161–206.

[23] W. Lee, G. H. Choi and T. W. Kim, "Visibility graph-based path-planning algorithm with quadtree representation," *Appl. Ocean Res.* **117**, 102887 (2021). doi: 10.1016/j.apor.2021.102887.

[24] M. Candeloro, A. M. Lekkas and A. J. Sørensen, "A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels," *Control Eng. Pract.* **61**, 41–54 (2017). doi: 10.1016/j.conengprac.2017.01.007.

[25] L. Lacasa, B. F.Ballesteros, J. Luque and J. C. Nuno, "From time series to complex networks: The visibility GRAPH," *Proc. Natl. Acad. Sci.* **105**(13), 4972–4975 (2008). doi: 10.1073/pnas.0709247105.

[26] D. B. Mark, C. Otfried, V. K. Marc and O. Mark. *Computational Geometry Algorithms and Applications"*, 3rd ed. (Springer, Berlin, Heidelberg, 2008).

[27] G. Wu, I. Atilla, T. Tahsin, M. Terziev and L. Wang, "Long-voyage route planning method based on multi-scale visibility graph for autonomous ships," *Ocean Eng.* **219**, 108242 (2021). doi: 10.1016/j.oceaneng.2020.108242.

[28] L. Blasi, E. D'Amato, M. Mattei and I. Notaro, "Path planning and real-time collision avoidance based on the essential visibility graph," *Appl. Sci.* **10**(16), 5613 (2020). doi: 10.3390/app10165613.

[29] H. Kaluđer, M. Brezak and I. Petrović, "A Visibility Graph-based Method for Path Planning in Dynamic Environments," **In:** *Proceedings of the 34th International Convention MIPRO* (2011) pp. 717–721.
[30] H. Moulinec, "A simple and fast algorithm for computing discrete Voronoi, Johnson-Mehl or Laguerre diagrams of points," *Adv. Eng. Softw.* **170**, 103150 (2022).
[31] Y. V. Pehlivanoglu, "A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV," *Aerospace Sci. Technol.* **16**(1), 47–55 (2012).
[32] M. Bełej and M. Figurska, "3D modeling of discontinuity in the spatial distribution of apartment prices using voronoi diagrams," *Remote Sens-BASEL* **12**(2), 229 (2020).
[33] X. Sun, K. Ishida, K. Makino, K. Shibayama and H. Terada, "Development of the Quad-SCARA platform and its collision avoidance based on Buffered Voronoi Cell," *Robotica* **41**(12), 3687–3701 (2023). doi: 10.1017/S0263574723001236.
[34] A. Tuncer, "Path planning of autonomous mobile robots based on Voronoi diagram and ant colony optimization," *J. Innovative Eng. Natural Sci.* **4**(1), 138–146 (2024).
[35] M. J. Kim, T. Y. Kang and C. K. Ryoo, "Path planning for unmanned aerial vehicles based on compensated voronoi diagram," *Int. J. Aeronaut. Space Sci.* **26**(1), 235–244 (2025).
[36] X. Xu, "Path inference based on voronoi graph for unmanned maritime vehicles," *Rob. Auton. Syst.* **173**, 104616 (2024).
[37] W. Su, M. Gao, X. Gao and Z. Xuan, "Enhanced multi-UAV path planning in complex environments with voronoi-based obstacle modelling and Q-learning," *Int. J. Aerospace Eng.* **2024**(1), 5114696 (2024).
[38] B. Li and B. Chen, "An adaptive rapidly-exploring random tree," *IEEE/CAA J. Autom. Sin.* **9**(2), 283–294 (2021).
[39] S. M. Lavalle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Rob. Res.* **20**(5), 278–400 (2001).
[40] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Rob. Autom. Lett.* **5**(2), 1500–1507 (2020).
[41] Z. Xu, D. Deng and K. Shimada, "Autonomous UAV exploration of dynamic environments via incremental sampling and probabilistic roadmap," *IEEE Rob. Autom. Lett.* **6**(2), 2729–2736 (2021).
[42] M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *Int. J. Rob. Res.* **35**(7), 797–822 (2016). doi: 10.1177/0278364915594679.
[43] P. Pharpatara, B. Hérissé, R. Pepy and Y. Bestaoui, "Sampling-based path planning: A new tool for missile guidance," *IFAC Proc.* **46**(19), 131–136 (2013).
[44] R. Pepy, A. Lambert and H. Mounier, "Reducing navigation errors by planning with realistic vehicle model," *IEEE Intell. Veh.*, 300–307 (2006).
[45] X. Zheng, J. Cao, B. Zhang, Y. Zhang, W. Chen, Y. Dai and J. Zhao, "Path planning of PRM based on artificial potential field in radiation environments," *Ann. Nucl. Energy* **208**, 110776 (2024).
[46] M. Novosad, R. Penicka and V. Vonasek, "CTOPPRM: Clustering topological PRM for planning multiple distinct paths in 3D environments," *IEEE Rob. Autom. Lett.* **8**(11), 7336–7343 (2023).
[47] J. Velagic, D. Delimustafic and D. Osmankovic, "Mobile Robot Navigation System Based on Probabilistic Road Map (PRM) with Halton Sampling of Configuration Space," **In:** *IEEE, 23rd International Symposium on Industrial Electronics (ISIE), Istanbul* (2014) pp. 1227–1232.
[48] S. Alarabi, C. Luo and M. Santora, "A PRM Approach to Path Planning with Obstacle Avoidance of an Autonomous Robot," **In:** *8th International Conference on Automation, Robotics and Applications (ICARA)* (2022) pp. 76–80.
[49] D. Hsu, T. Jiang, J. Reif and Z. Sun, "The Bridge Test for Sampling Narrow Passages with Probabilistic Roadmap Planners," **In:** *IEEE International Conference on Robotics and Automation (Cat. no. 03CH37422)*, vol. **3** (IEEE, 2003) pp. 4420–4426.
[50] Y. T. Lin, "The Gaussian PRM Sampling for Dynamic Configuration Spaces," **In:** *IEEE 9th International Conference on Control, Automation, Robotics and Vision* (IEEE, 2006) pp. 1–5.
[51] F. F. Arias, B. Ichter, A. Faust and N. M. Amato, "Avoidance Critical Probabilistic Roadmaps for Motion Planning in Dynamic Environments," **In:** *IEEE International Conference on Robotics and Automation (ICRA)* (2021) pp. 10264–10270.
[52] J. Denny, K. Shi and M. Amato, "Lazy Toggle PRM: A Single-Query Approach to Motion Planning," **In:** *IEEE International Conference on Robotics and Automation* (2013) pp. 2407–2414.
[53] M. Hüppi, L. Bartolomei, R. Mascaro and M. Chli, "T-PRM: Temporal Probabilistic Roadmap for Path Planning in Dynamic Environments," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2022) pp. 10320–10327.
[54] D. Kim, Y. Kwon and S. E. Yoon, "Dancing PRM: Simultaneous Planning of Sampling and Optimization with Configuration Free Space Approximation," **In:** *IEEE International Conference on Robotics and Automation (ICRA)* (2018) pp. 7071–7078.
[55] W. Khaksar, T. S. Hong, M. Khaksar and O. Motlagh, "A low dispersion probabilistic roadmaps (LD-PRM) algorithm for fast and efficient sampling-based motion planning," *Int. J. Adv. Rob. Syst.* **10**(11), 397 (2013).
[56] R. Singh, "Trajectory optimization with hybrid probabilistic roadmap approach to achieve time efficient navigation of unmanned vehicles in unstructured environment," *Robotic Intell. Autom.* **44**(1), 164–189 (2024).
[57] J. Fan, X. Zhang, K. Zheng, Y. Zou and N. Zhou, "Hierarchical path planner combining probabilistic roadmap and deep deterministic policy gradient for unmanned ground vehicles with non-holonomic constraints," *J. Frankl. Inst.* **361**(8), 106821 (2024).
[58] D. Zheng, J. Ridderhof, Z. Zhang, P. Tsiotras and A. Agha-Mohammadi, "CS-BRM: A probabilistic roadMap for consistent belief space planning with reachability guarantees," *IEEE Trans. Rob.* **40**, 1630–1649 (2024).
[59] Y. Huang, H. Wang, L. Han and Y. Xu, "Robot path planning in narrow passages based on improved PRM method," *Intell. Serv. Rob.* **1-12**(3), 609–620 (2024).
[60] D. Devaurs, T. Siméon and J. Cortés, "Cortés "Optimal path planning in complex cost spaces with sampling-based algorithms," *IEEE Trans. Autom. Sci. Eng.* **13**(2), 415–424 (2016).

[61] M. G. Mohanan and A. A. Salgoankar, “Survey of robotic motion planning in dynamic environments,” *Rob. Auton. Syst.* **100**, 171–185 (2018).
[62] M. S. Branicky, M. M. Curtiss, J. Levine and S. Morgan, “Sampling-based planning, control and verification of hybrid systems,” *IEE P-CONTR Theory Appl.* **153**(5), 575–590 (2006).
[63] A. Yershova, L. Jaillet, T. Siméon and S. M. LaValle, “Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain,” **In:** *Proceedings of the 2005 IEEE International conference on Robotics and Automation* (2005) pp. 3856–3861.
[64] R. Heβ, T. Lindeholz, D. Eck and K. Schilling, “RRTCAP* - RRT* controller and planner - simultaneous motion and planning,” *IFAC-PapersOnLine* **48**(10), 52–57 (2015).
[65] L. Yang, Z. Wei-guo, S. Jing-ping and L. Guang-wen, “A Path Planning Method Based on Improved RRT,” **In:** *IEEE Chinese Guidance, Navigation and Control Conference, Yantai* (2014) pp. 564–567.
[66] J. Li, S. Liu, B. Zhang and X. Zhao, “RT-A* Motion Planning Algorithm for Non-Holonomic Mobile Robot,” **In:** *Proceedings of the SICE Annual Conference, Sapporo* (2014) pp. 1833–1838. doi: 10.1109/SICE.2014.6935304.
[67] A. H. Qureshi, S. Mumtaz, K. F. Iqbal, Y. Ayaz, M. S. Muhammad, O. Hasan and M. Ra, “Triangular Geometry Based Optimal Motion Planning Using RRT*-Motion Planner,” **In:** *IEEE 13th International Workshop on Advanced Motion Control (AMC)* (2014) pp. 380–385.
[68] J. Li, S. Liu, B. Zhang and X. Zhao, “RRT-A* Motion Planning Algorithm for Non-Holonomic Mobile Robot,” **In:** *Proceedings of the SICE Annual Conference (SICE)* (IEEE, 2014) pp. 1833–1838.
[69] G. Gao, D. Li, K. Liu, Y. Ge and C. Song, “A study on path-planning algorithm for a multi-section continuum robot in confined multi-obstacle environments,” *Robotica* **1-24**(10), 3324–3347 (2024). doi: 10.1017/S0263574724001383.
[70] L. Ye, J. Li and P. Li, “Improving path planning for mobile robots in complex orchard environments: The continuous bidirectional Quick-RRT* algorithm,” *Front. Plant Sci.* **15**, 1337638 (2024).
[71] J. Yu, C. Chen, A. Arab, J. Yi, X. Pei and X. Guo, “RDT-RRT: Real-time double-tree rapidly-exploring random tree path planning for autonomous vehicles,” *Exp. Syst. Appl.* **240**, 122510 (2024).
[72] J. Wang and E. Zheng, “Path planning of a mobile robot based on the improved rapidly exploring random trees star algorithm,” *Electronics* **13**(12), 2340 (2024).
[73] S. Ganesan, B. Ramalingam and M. R. Elara, “A hybrid sampling-based RRT* path planning algorithm for autonomous mobile robot navigation,” *Exp. Syst. Appl.* (258), 125206 (2024). doi: 10.1016/j.eswa.2024.125206.
[74] Y. Wang, W. Li and Y. Liang, “A trajectory optimisation-based incremental learning strategy for learning from demonstration,” *Appl. Sci.* **14**(11), 4943 (2024).

Cite this article: B. B. K. Ayawli, J. K. Dawson, E. Badu, I. E. B. Ayawli and D. Lamusah, “Comparative analysis of popular mobile robot roadmap path-planning methods”, Robotica. https://doi.org/10.1017/S0263574725000281