Received May 9, 2019, accepted June 23, 2019, date of publication June 28, 2019, date of current version July 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2925623

Mobile Robot Path Planning in Dynamic Environment Using Voronoi Diagram and Computation Geometry Technique

BEN BEKLISI KWAME AYAWLI[1,2], XUE MEI[1], MOUQUAN SHEN[1], ALBERT YAW APPIAH[1,3], AND FRIMPONG KYEREMEH[1,3]

[1]College of Electrical Engineering and Control Science, Nanjing Tech University, Nanjing 214000, China
[2]Computer Science Department, Sunyani Technical University, Sunyani 27091, Ghana
[3]Electrical and Electronic Engineering Department, Sunyani Technical University, Sunyani 27091, Ghana

Corresponding author: Xue Mei (seraph_mx@163.com)

This work was supported in part by the Beijing Advanced Innovation Center for Intelligent Robots and Systems under Grant 2018IRS20.

ABSTRACT This paper presents a novel path planning method for mobile robots in complex and dynamic environments using Voronoi diagram (VD) and computation geometry technique (CGT) termed, VD-CGT. An algorithm to categorize moving obstacles based on their positions, velocities, distances, and directions to ascertain their collision threat level and possible replanning decision is introduced. The initial path computation is done using morphological dilation, VD, A-star, and cubic spline algorithms. Instead of considering the entire map of the environment, CGT is used to compute a small rectangular region estimated to enclose a detected collision-threat obstacle and the current position of the robot. The roadmap is computed in the geometrical shape using VD and nodes are added to the initial roadmap nodes to compute a new path for replanning. To avoid increasing time and space requirements, these nodes are discarded before subsequent replanning is done. The results indicate better path replanning performance in complex and dynamic environments in terms of success path computation rate, path cost, time, and the number of replanning computations compared with other five popular related path planning approaches. The proposed method is efficient, and it computes safe and shortest replan path to goal with low computation time requirement. Unnecessary replanning computations are avoided which aid in reducing time and distance to get to the goal. With the performance results, the proposed method is a promising method for achieving safe, less path cost, and time in path replanning computations in complex and dynamic environments.

INDEX TERMS Mobile robot, motion planning, path planning, unmanned autonomous vehicles, Voronoi diagram.

I. INTRODUCTION

Research on mobile robot path planning in static environment has witnessed much progress [1]–[4]. However, path planning in complex and dynamic environment is still a problem [4], [5]. Safe and efficient path planning in complex and dynamic environments is crucial for safe navigation of autonomous mobile vehicles.

To achieve autonomous navigation in complex environment, mobile robots are expected to have accurate knowledge of their environment to take wise navigation decisions. This knowledge can be obtained by building a roadmap in the environment of the robot. Voronoi diagram (VD) based algorithms are popular for building such roadmaps. VD is noted for not compromising path safety by computing roadmaps which are equidistant between obstacles. Arguably, VD based path planning has attracted the attention of many researchers [6]–[11]. Based on generalized VD, electric circuit-based path planning method for multiple robot path planning was presented in [11] to address traffic jam problems. Moreover, a motion planning approach with VD through a line-segment map was presented in [9]. The VD was used to obtain the way points from the line-segment-based map while Dijkstra's algorithm was employed to obtain the shortest path on the roadmap. In [10], a modified ant colony optimization (M-ACO) algorithm was combined with VD to propose

The associate editor coordinating the review of this manuscript and approving it for publication was Rui-Jun Yan.

a path planning method for safe and shortest path by point-to-point motion planning. The VD was employed to generate the roadmap while the M-ACO was used to generate the path. The technique to provide safety for the robot was however not given. Reference [8] proposed centroidal voronoi tessellation based intelligent control algorithm for self-assembly for swarm robots. There is however the challenge of ensuring agents identifying their positions and orientations. Although the aforementioned VD-based path planning methods have collectively advanced mobile robot path planning, they are limited to static environments.

Fortunately, the problem of mobile robot path planning in dynamic environments has not been left unattended: A couple of algorithms have been proposed. Reference [12] proposed a two-way D$*$ algorithm based on Witkowski's algorithm to compute path in weighted occupancy grid maps. Another path replanning method for vehicles on a road was presented by generating path candidates from which a path is selected for navigation [13]. In [14], firefly algorithm was used to present mobile robot navigation method in uncertain environment involving static and dynamic environment. Again, membrane evolution artificial potential field approach was presented for mobile robot path planning in static and dynamic environments [15]. Known static environments and partially known dynamic environments were considered. Also, Bezier curve-based technique using modified genetic algorithm was proposed to improve path computation using the classical genetic algorithm method [16]. Less complex environment was used to evaluate the method. Additionally, based on A* algorithm, a path planning method involving static and moving obstacles was proposed for unmanned surface vehicles [17]. There are other path planning methods for dynamic environments presented in [13], [18]–[26].

Generally, these path replanning methods can be categorized into two: incremental replanning and replanning that considers the entire environment (full map) of the robot when collision threat is detected [27]. The incremental replanning approaches compute new path for navigation at every incremental distance or time during navigation of the robot. Time is spent on replanning even if there is no collision threat. Incremental replanning is time consuming and may also results in the computation of inefficient path. On the other hand, path replanning computation that considers the entire environment when collision threat is detected can produce efficient path in terms of path length, but it is time consuming. It should be noted that, execution time is very crucial for robots in motion to take quick and intelligent decision before they collide with obstacles especially in situations where the speed of a vehicle is high and the environment is complex and dynamic. Moreover, most replanning algorithms consider replanning when obstacles are threat to the path without considering the dynamics of the specific obstacles to determine the threat level to the vehicle and the needed replanning response [28]. It is obvious that not all obstacles appearing in the replanning zone of a vehicle are threat to the vehicle to require replanning. Obstacles crossing the path of the vehicle or those in the same direction with the vehicle may not be a threat for collision. The position, distance, velocity and direction of these moving obstacles play a role in determining the level of threat to ensure safe replanning.

This paper presents a novel path planning algorithm, VD-CGT for mobile robots in complex and dynamic environment based on VD and computational geometry technique (CGT) for path replanning with low time computation requirement. To provide intelligent replanning, the position, distance, velocity, and direction of moving obstacles and that of the robots are considered in determining the level of collision threat. When an obstacle is detected during navigation, the proposed algorithm analyses the behavior of the obstacle from the robot to determine the level of collision threat and a possible replanning decision before it reaches the replanning zone of the robot. However, if no decision is taken before it reaches the critical threat zone (CTZ) of the robot, a reactive collision avoidance is done using a default replan decision to replan the path. Instead of using the entire map, a CGT is used to compute a small rectangular region of the environment estimated to enclose a detected collision-threat obstacle and the current position of the robot. A VD roadmap is then computed in the rectangular region to obtain new freeway nodes which are added to the existing nodes used for the initial path to compute new path. After each replanning, the new nodes are discarded when subsequent obstacles are detected to avoid increasing computation time and space.

The main contributions of this paper include:

- An algorithm to categorize detected dynamic obstacles based on their positions, distances, orientations, and velocities to determine a possible intelligent replanning decision. This algorithm helps to reduce unnecessary replans and number of replanning computations to reach goal. This reduction leads to a reduction in path cost to reach goal.
- An algorithm to reduce path cost and time requirement for path replanning computation by introducing a CGT to compute and select a small section of the robot's workspace that includes a detected collision-threat obstacle and a current position of the robot for path replanning computation.

II. PROBLEM FORMULATION AND ASSUMPTIONS

Workspace, W, of a mobile robot comprises sections termed as freeway space, W_{free}, and objects other than the vehicle itself known as obstacles, W_{obs}. The obstacles are made up of static and moving objects. The moving obstacles navigate at different velocities and have different orientations. The positions, distances, velocities, and orientations of the dynamic obstacles in the workspace of the robot occur randomly. Assume the task of a mobile robot is to navigate from a given start position to goal in a cluttered and dynamic environment with these obstacles requiring low space and time usage. To achieve this task, a robust replanning algorithm is required to compute and replan the path efficiently by reducing the number of replanning computations and aid the vehicle to navigate from the start to goal at a shorter time. This is the objective of this paper.

The path replanning method proposed in this paper considers kinematics of a car-like robot installed with a laser range sensor (S) (see Fig. 1). Computations to determine the threat levels of dynamic obstacles and its subsequent replan decisions are based on this kinematics. Fig. 1 illustrates a car-like mobile robot with front and rear wheels with a laser range sensor. The base frame representing the workspace of the robot's environment is represented as (X_W, Y_W) while the robot frame is represented as (X_R, Y_R). The rear wheels are aligned with the car but the front wheels spin about the vertical axes. Assuming the wheels do not slip, system constrains do occur as the wheels roll and spin [29]. From Fig. 1, the pose configuration of the robot ξ_W with respect to the base frame can be represented as $[X_A, Y_A, \theta_r, \varphi_r]^T$ with (X_A, Y_A) indicating the location of the midpoint A of the rear wheel axis. The orientation angle of the robot and the steering angle of the wheels with respect to the body of the robot is indicated by θ_r and φ_r, respectively.

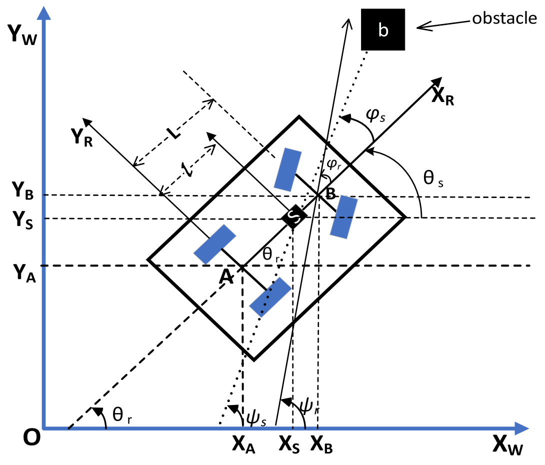


FIGURE 1. Kinematics model of a car-like mobile robot.

The mapping, $\dot{\xi}_R$, between the base and the robot frame can be represented as in (1).

$$\dot{\xi}_R = R(\theta) \cdot [\dot{X}_A, \dot{Y}_A, \dot{\theta}_r]^T \tag{1}$$

where $R(\theta)$ is given as in (2)

$$R(\theta) = \begin{bmatrix} \cos(\theta_r) & \sin(\theta_r) & 0 \\ -\sin(\theta_r) & \cos(\theta_r) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Two nonholonomic constraints can be obtained as given in (3) and (4) [29].

$$-\dot{X}_A \sin\theta_r + \dot{Y}_A \cos\theta_r = 0 \tag{3}$$

$$-\dot{X}_B \sin(\theta_r + \varphi_r) + \dot{Y}_B \cos(\theta_r + \varphi_r) = 0 \tag{4}$$

X_B and Y_B represent the location coordinates of the front wheel midpoints B. Hence X_B and Y_B can be computed as in (5) and (6).

$$X_B = X_A + L\cos\theta_r \tag{5}$$

$$Y_B = Y_A + L\sin\theta_r \tag{6}$$

where L denotes the distance from midpoint A of the rear wheel axis to midpoint B of the front wheel axis. Based on (5) and (6), the nonholonomic constraints given in (4) can be written as in (7).

$$-\dot{X}_B \sin(\theta_r + \varphi_r) + \dot{Y}_B \cos(\theta_r + \varphi_r) + L(\cos\varphi_r)\dot{\theta}_r \tag{7}$$

Based on (5) and (6), we can obtain the pose of the sensor $S = [x_s, y_s, \theta_s]^T$ using (8).

$$S = \begin{bmatrix} x_s \\ y_s \\ \theta_s \end{bmatrix} = \begin{bmatrix} X_A + l\cos\theta_r \\ X_A + l\sin\theta_r \\ \theta_r \end{bmatrix} \tag{8}$$

where l represents the perpendicular distance between the midpoint of the sensor S and the midpoint A of the rear wheels. Given a driving velocity v_1 and a steering velocity v_α, the kinematic equations of a control system $[\dot{X}_A \;\; \dot{Y}_A \;\; \dot{\theta}_r \;\; \dot{\varphi}_r]$ can be represented as in (9).

$$\begin{bmatrix} \dot{X}_A \\ \dot{Y}_A \\ \dot{\theta}_r \\ \dot{\varphi}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r \\ \sin\theta_r \\ (1/L)\tan\dot{\varphi}_r \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_\alpha \tag{9}$$

where $v_\alpha = \dot{\varphi}_r$. Given a distance d from the sensor S to an obstacle b, the position of obstacle $b = [x_b, y_b]^T$ with respect to the configuration space of the robot can be computed using (10).

$$\begin{bmatrix} x_b \\ y_b \end{bmatrix} = \begin{bmatrix} x_s + d\cos\psi_s \\ y_s + d\sin\psi_s \end{bmatrix} \tag{10}$$

where $\psi_s = \theta_s + \varphi_s$ and φ_s represents the angle of the sensor S to the obstacle b with respect to the body of the robot. At subsequent distance readings, the distance $d_o = dist(b, b')$ between the initial b and current b' position sensor readings of an obstacle can be computed using (11).

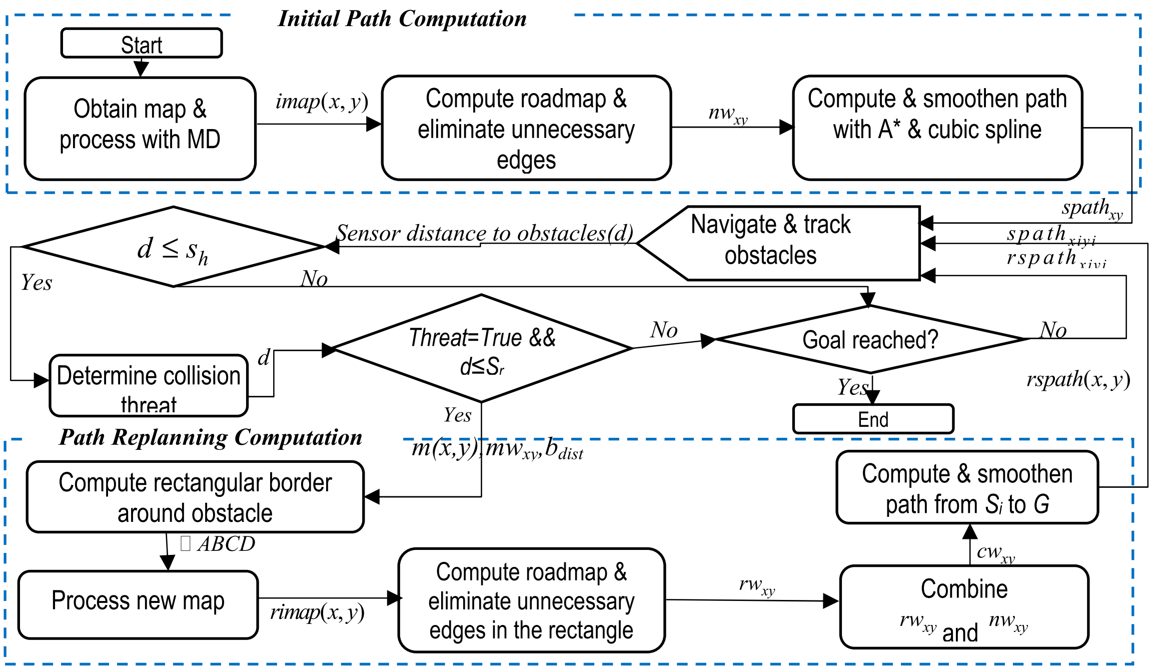
$$d_o = dist(b, b') = \sqrt{(b_x' - b_x)^2 + (b_y' - b_y)^2} \tag{11}$$

where (b_x, b_y) and (b_x', b_y') are the initial and current positions of the obstacle. With φ_s being the orientation of the sensor S to the obstacle, the estimated sensor direction to different positions of a dynamic obstacle with respect to the robot frame can be represented as $\varphi_{s(i)}$ where $i = \{1, 2, 3, \ldots, n\}$ with n representing the last orientation value.

Using this kinematics, detected obstacles are analyzed to determine their collision threat levels and the required path replanning decision. Based on the replanning decision, a replanned path is computed using CGT, morphological dilation (MD), VD, A*, and cubic spline interpolation (CSI). Fig. 2 shows step-by-step processes required in implementing the method presented in this paper.

III. INITIAL PATH COMPUTATION

The initial path computation (shown in Fig. 2) is the first path computed from the initial position of the mobile robot to the goal. This includes obtaining and processing a map using MD, computing the roadmap, shortest path, and path smoothening.


Initial Path Computation
Start
Obtain map & process with MD
imap(x, y)
Compute roadmap & eliminate unnecessary edges
nw_xy
Compute & smoothen path with A* & cubic spline
spath_xy
Navigate & track obstacles
spath_xiyi
rspath_xiyi
Sensor distance to obstacles(d)
d ≤ s_h
Yes
No
Determine collision threat
d
Threat=True && d≤S_r
No
Goal reached?
No
Yes
End
rspath(x, y)
Yes
m(x,y), mw_xy, b_dist
Path Replanning Computation
Compute rectangular border around obstacle
ABCD
Process new map
rimap(x, y)
Compute roadmap & eliminate unnecessary edges in the rectangle
rw_xy
Combine rw_xy and nw_xy
cw_xy
Compute & smoothen path from S_i to G

imap(x,y)=map with inflated obstacles, nw_{xy}=computed roadmap points, $spath_{xy}$=computed and smoothened path nodes, *d*=sensor distance from an obstacle, s_h=obstacle detection distance threshold, s_r=replanning zone distance threshold, rs_{path}=re-computed path nodes for replanning, □*ABCD*=computed replanning region, *m(x,y)*=estimated midpoint for □*ABCD*, b_{dist}=distance from *m(x,y)* to the boarder of □*ABCD*, *rimap(x,y)*=inflated obstacles on map with detected obstacle, rw_{xy}=newly computed roadmap nodes for replanning, cw_{xy}=column concatenation of nw_{xy} and rw_{xy}.

FIGURE 2. Workflow diagram of the proposed VD-CGT method.

Safety of vehicles in navigation is very important. MD is employed to inflate obstacles on the map using dilation rate based on the dimension of the vehicle and a minimum safety distance required for safety. This technique takes care of noise and other uncertainties due to hardware kinematics imbalances that could cause the robot to deviate from its computed path during navigation. Based on MD algorithm presented in [30], obstacles in a workspace of the robot in this paper are inflated using (12).

$$imap(x, y) = (map \oplus q_s) = \max\{map(t_r - x) + q_s(x)|(t_r - x) \in d_{map(x,y)} \wedge \in d_z\}, \quad (12)$$

where *imap(x, y)* is the results of the algorithm, map(x, y) represents the map of the workspace, q_s denotes the structuring element of the dilation, t_r is the translated radius representing the required safety distance, and d_z symbolizes the dimension of the structuring element. Before computing a roadmap and a path, obstacles on the map are inflated using (12). Given a map in Fig. 3a, Fig. 3b shows a map with inflated obstacles obtained using (12) with $t_r = 3$.

The safety distance is chosen based on the dimension of the vehicle and other necessary safety distance considerations. The safe zone, S_{zone}, to compute the roadmap for subsequent path computation can therefore be represented using (13).

$$s_{zone} = \{map(x, y) \in W_{free} \cap map(x, y) \notin (map \oplus q)t_r\} \quad (13)$$

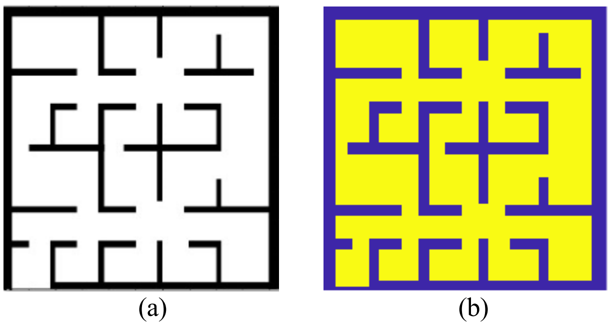


FIGURE 3. (a). Original map (b). Map with Inflated obstacles using MD with tr = 3.

In this paper, VD is considered for the roadmap computation to generate candidate paths. Given P_{obs} as sets of obstacle points in a 2D geometrical plane, a voronoi region $V_r(P_{obs}(i))$ can be computed as in (14).

$$V_r(P_{obs(i)}) = \{x \in W|dist(x, P_{obs(i)}) \leq dist(x, P_{obs(j)})\forall i \neq j\}, \quad (14)$$

where *W* represents the workspace,$P_{obs}(i)$ and $P_{obs}(j)$ are VD sites. The VD comprises all the points with more than one nearest neighbor [31]. In this paper, Fortune's sweepline algorithm [32] is considered for the VD roadmap computation due to its efficiency as against other VD algorithms. While sweepline algorithm requires $O(n \log n)$ time [32], half-plane intersection, and divide and conquer methods require $O(n^3)$

time each, and incremental method requires $O(n^2)$ time [7]. To maximize the efficiency of sweepline algorithm to compute more efficient VD roadmap, a step-size ratio technique is used in this paper to reduce the nodes of the map for the computation of the roadmap. A step size is chosen such that the selected nodes for the computation of the roadmap does not affect the correct representation of the map. In this paper, a step size, sz, range of $sz \in Z : 6 \leq sz \leq 8$ is considered. Using the processed map *imap(x, y)* with inflated obstacles as input, the VD roadmap is computed. Navigable waypoints nw_{xy} are obtained by eliminating edges in W_{obs} using K-Nearest Neighbor (KNN) technique. Duplicate waypoints and points too close to each other are also removed to reduce the waypoints to facilitate fast path computation. The edges E and the intersection points, V of the edges form a graph, $\mathcal{G} = (E, V)$ representing the roadmap in the S_{zone}. Fig. 7b demonstrates a computed VD with discarded roadmaps in W_{obs}.

The shortest path from the start to goal positions is computed using A* heuristic function. A* is a search algorithm good at finding a path on a graph with least cost provided a path exist [33]. The A* algorithm used in this paper is represented as in (15)

$$f(q) = g(q) + h(q) \tag{15}$$

where $q = nw_{xy}$ represents the waypoints of the roadmap, $f(q)$ is the estimated cost to reach the target, $g(q) = c(q, q')$ is the actual cost from point q to q' and $h(q)$ is the heuristic function to estimate the cheapest cost from node q to the goal. $h(q)$ is obtained using Euclidean distance metric. A* search algorithm is known to be complete and optimal; hence, its choice to compute optimal path from a given start position to goal in this paper. It is complete because once a path exists in the s_{zone}, A* can find it. A* is admissible and consistent which are properties of optimality. Considering $g(q)$ as actual cost to get to node q', $f(q)$ would therefore not overestimate the cost to reach the target. This makes A* admissible. With *c(q, q')* as the cost from point q to q', consistency is achieved since $h(q) \leq c(q, q') + h(q)$ and admissibility is achieved since for all arcs of the path, $c(q, q') \geq \varepsilon > 0$.

The computed shortest path $apath_{xy}$ = f(nw$_{xy}$) from the nw_{xy} usually contains sharp curves that poses difficulty in smooth navigation. To eliminate these sharp curves, CSI algorithm is adopted to smoothen the path to provide more navigable path, $spath_{xy}$. The CSI equation used is given in (16).

$$spath_{xy} = \begin{cases} S_i(x) = a_i + b_i(x - x_i) + \dots & \\ c_i(x - x_i)^2 + d_i(x - x_i)^3 & for\ x_i \leq x \leq x_{i+1} \\ S_i(y) = a_i + b_i(y - y_i) + \dots & \\ c_i(y - y_i)^2 + d_i(y - y_i)^3 & for\ y_i \leq y \leq y_{i+1} \end{cases} \tag{16}$$

where $S_i(x)$ and $S_i(y)$ symbolize the CSI for each i node of the path $apath_{xy}$. a_i, b_i, c_i *and* d_i are the coefficients to be computed for each i. The number of intervals between points on $apath_{xy}$ determines the number of coefficients to be computed.

The algorithm to compute the initial path is given in Algorithm 1.

Algorithm 1 Initial Path Computation

```
Input: $map(x, y)$, safety space $s$, scale factor $f$, constant value $cv$
Output: $spath_{xy}$
1:  begin
2:   Initialize $t_r$, $s,f,cv$, $counter = 1$;
3:  $imap(x, y) \leftarrow (map \oplus q)$// Inflate obstacles (12)
4:   for $i = 1$ to $size(imap(x))$, interval $f$
5:     for $j = 1$ to $size(imap(y))$, interval $f$
6:       if $imap(x, y) == 0$;
7:         xx ← j;
8:         yy ← i;
9:         counter ++;
10:      end if
11:    end for
12:  end for
13: $dt = DT(xx, yy)$; //Delaunay triangulation
14: $[vx, vy] = voronoi(dt)$//compute VD [34]
15: $[A, D] \leftarrow KNN([xx, yy], [vx, vy])$;
16:  $inobs = (D, size(vx) < cv)$//Eliminate roadmaps in obstacles
17: $nw_{xy} = [vx(\sim inobs), vy(\sim inobs)]$; //safe waypoints
18: $nw_{xy} \leftarrow$ remove duplicates and points too close
19: $apath_{xy} \leftarrow f(nw_{xy})$;//compute shortest path (15)
20: $spath_{xy} \leftarrow CSI(apath_{xy})$//compute smooth path (16)
21: End
```

IV. PATH REPLANING

This section introduces the proposed path replanning method, VD-CGT, using CGT combined with VD. The proposed method comprises obstacle tracking, categorization and threat determination, computation of replanning area, and new path computation.

A. OBSTACLE TRACKING, CATEGORIZATION AND THREAT DETERMINATION

Range sensors are required to track obstacles in the obstacle tracking and threat categorization zone, s_h at a given distance threshold, h. The category of an obstacle and its level of threat to the robot is determined before it reaches the path replanning zone, s_r, where path replanning occurs. Decision is taken by the vehicle before the obstacle reaches the CTZ distance threshold, s_c. CTZ is a distance between the robot and the obstacle in the sensor spectrum which requires the robot to take immediate obstacle avoidance decision to prevent collision. Fig. 4 indicates the distance thresholds considered for path replanning in this paper.

In Fig. 4, S is the origin of the sensor, c represents the CTZ distance limit from S, the replanning distance limit from the sensor is denoted by r, and h represents the maximum

TABLE 1. Computations to determine obstacle categorization, collision threat and replan decisions.

Condition (*if*(….))	Obstacle category	Threat	Decision
$(\varphi_{s(i)} \approx \varphi_{s(i+1)}) \wedge (d_v + d_2 \approx d_1) \wedge (d_1 > d_2)$	Static with $r_{sp} > o_{sp}$	Yes	Re-plan at s_r
$(\varphi_{s(i)} \approx \varphi_{s(i+1)}) \wedge (d_v + d_2 > d_1) \wedge (d_1 \approx d_2)$	Obstacle moving in the same direction as robot with $r_{sp} = o_{sp}$	No	No re-plan
$(\varphi_{s(i)} \approx \varphi_{s(i+1)}) \wedge (d_v + d_2 > d_1) \wedge (d_1 < d_2)$	Obstacle moving in the same direction as robot with $r_{sp} < o_{sp}$	No	No re-plan
$(\varphi_{s(i)} \approx \varphi_{s(i+1)}) \wedge (d_v + d_2 > d_1) \wedge (d_1 > d_2)$	Obstacle moving in the same direction as robot with $r_{sp} > o_{sp}$	Yes (low)	Re-plan at $s_r = s_r - \left(\frac{d_o}{d_v} s_r\right)$
$(\varphi_{s(i)} \approx \varphi_{s(i+1)}) \wedge (d_v + d_2 < d_1)$	Obstacle moving in the opposite direction as robot at almost the same orientation	Yes (high)	Re-plan at $s_r = s_r + \left(\frac{d_o}{d_v} s_r\right)$
$(\varphi_{s(i)} > \varphi_{s(i+1)}) \wedge (d_v + d_2 \approx d_1) \wedge (d_{op(i)} \neq d_{op(i+1)})$	Obstacle moving from right to left across the robot.	Yes (high at left)	Stop for clearance at s_c or re-plan towards right at s_r
$(\varphi_{s(i)} < \varphi_{s(i+1)}) \wedge (d_v + d_2 \approx d_1) \wedge (d_{op(i)} \neq d_{op(i+1)})$	Obstacle moving from left to right across the robot.	Yes (high at right)	Stop for clearance at s_c or re-plan towards left at s_r
$(\varphi_{s(i)} < \varphi_{s(i+1)}) \wedge (d_v + d_2 > d_1) \wedge (d_{op(i)} \neq d_{op(i+1)})$	Obstacle moving diagonally from left to right away from the robot.	Yes (low)	Stop for clearance at s_c
$(\varphi_{s(i)} < \varphi_{s(i+1)}) \wedge (d_v + d_2 < d_1) \wedge (d_{op(i)} \neq d_{op(i+1)})$	Obstacle moving diagonally from left to right towards the robot.	Yes (high)	Re-plan towards left at $s_r = s_r + \left(\frac{d_o}{d_v} s_r\right)$
$(\varphi_{s(i)} > \varphi_{s(i+1)}) \wedge (d_v + d_2 > d_1) \wedge (d_{op(i)} \neq d_{op(i+1)})$	Obstacle moving diagonally from right to left away from the robot.	Yes (low)	Stop for clearance at s_c
$(\varphi_{s(i)} > \varphi_{s(i+1)}) \wedge (d_v + d_2 < d_1) \wedge (d_{op(i)} \neq d_{op(i+1)})$	Obstacle moving diagonally from right to left towards the robot.	Yes (high)	Re-plan towards right at $s_r = s_r + \left(\frac{d_o}{d_v} s_r\right)$

φs(i)=the direction of an obstacle to the sensor position at time *t*, d_1=the initial distance between the sensor and an obstacle at time *t*, d_2=the next distance between the sensor and an obstacle at time *t+1*, d_v=robot covered distance from a position at time *t* to the next position at time *t+1*, d_{op}=minimum distance threshold between an obstacle and available unnavigated path of the robot, d_o=estimated distance between two positions of an obstacle, s_r=the replanning zone distance threshold between the sensor and obstacles, r_{sp}=robot speed, o_{sp}=obstacle speed

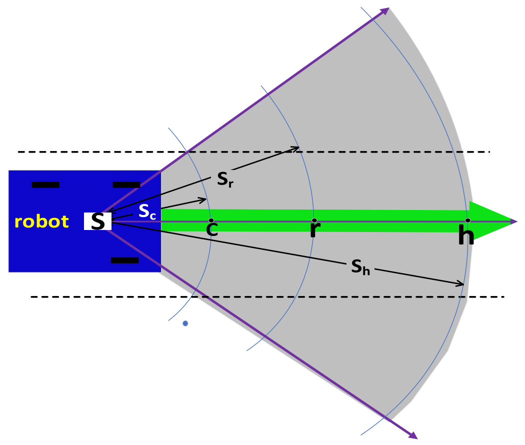


FIGURE 4. Distance threshold allocation structure.

distance from the sensor for obstacle tracking and threat categorization. Upon detecting an obstacle in the s_h, the category of the obstacle and its threat to the robot is determined.

The proposed algorithm (demonstrated in Table 1 based on the kinematics in Fig.1, 4 and 5) considers four main categorizations of dynamic obstacles for the replanning task which has been sub divided into eleven moving obstacle categories. These include (a) static obstacles on the path of the robot, (b) obstacles in front and moving in the same direction as the robot (three sub categories), (c) obstacles in front and moving in the opposite direction of the robot and (d) obstacles in front and moving across the path of the robot at different angles (six sub categories). Obstacle categories, obstacle threats to robot and their correspondent replan decisions are determined based on the satisfaction of the conditional statements in Table 1.

Fig. 5 demonstrates trajectories of two obstacles in terms of positions, distances, and directions to the robot during navigation. d_{op} is the minimum distance between an obstacle and the available unnavigable path by the robot. This can be computed using KNN.

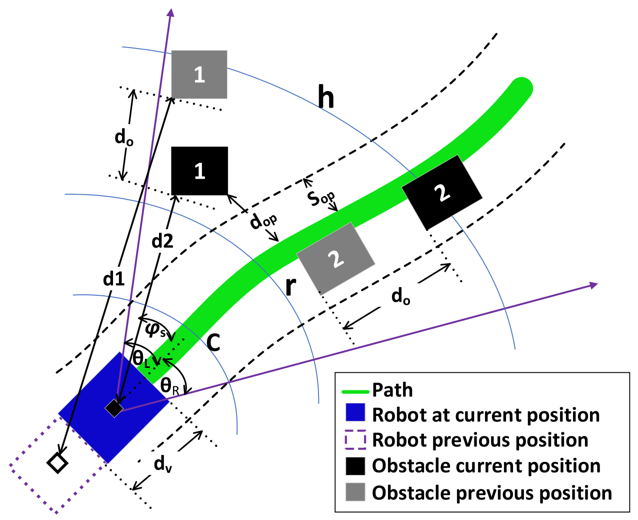


FIGURE 5. Trajectories of two obstacles and distance measurement model.

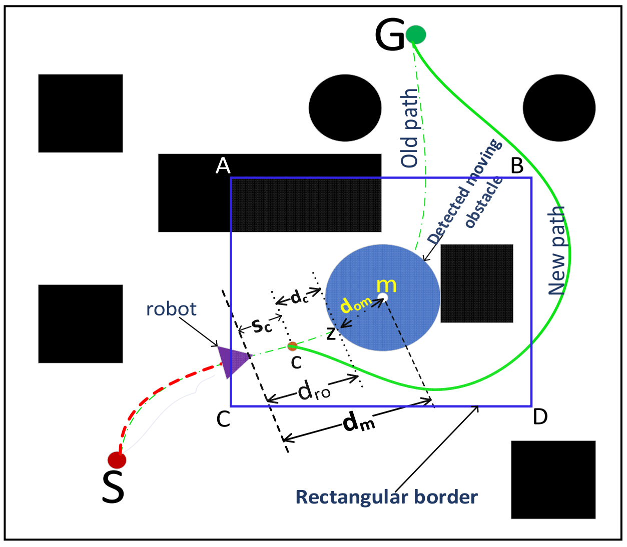


FIGURE 6. Computing the replanning area.

The distances $dist(o, p)$, between the vertices of the obstacles, o, and that of the path, p, are computed using (17).

$$dist(o, p) = \sqrt{\sum_{i=1}^{n}(o_i - p_i)^2} \tag{17}$$

d_1 represents the initial distance between the sensor and the obstacle at time$t$. d_2 is the next distance between the sensor, S, and an obstacle at time $t + 1$. The distance covered by the robot from one position at time t to the next position at time $t + 1$ is denoted by d_v. The direction of the obstacle to the sensor position at time t and $t + 1$ can be represented as $\varphi_{s(i)}$ and $\varphi_{s(i+1)}$, respectively. The estimated distance between two positions of an obstacle based on the distances and angles from the sensor to the initial and current positions of the obstacle is indicated as d_o which can be computed using (18a) or (18b).

$$d_o = \sqrt{d_1^2 + d_2^2 - 2d_1 d_2 \cos(\varphi_{s(i)} - \varphi_{s(i+1)})} \tag{18a}$$

$$d_o = d_1 - (d_v + d_2) \tag{18b}$$

The distance, d_{op}, is compared with safety-obstacle-to-path distance threshold, s_{op}, to determine the threat of an obstacle to the path used by the robot. This is done at the initial point of detection and just before a replanning decision is taken at s_r. However, it should be noted that a threat of an obstacle to a path may not pose a threat to the robot. As a result, additional conditions involving distance, velocity and direction between the robot and an obstacle are considered in this paper in determining collision-threat obstacles to the robot. Based on computation information with regard to Fig. 4 and 5, Table 1 shows conditional computation statements to determine obstacle category and the level of threat an obstacle poses to a robot and the necessary replanning decision to be taken.

B. COMPUTATION OF THE REPLANNING AREA

After an obstacle is detected and threat analysis is done, a decision is taken by the robot whether to replan or not (see Table 1). When a decision is made to replan, the computation of new path begins with the computation of the replanning area. Fig 6 shows a computed replanning area represented by $\square ABCD$. s_c represents the CTZ distance of the robot.

Position $c(x, y)$ along the path can be computed as in (19).

$$c(x, y) = [r_x + s_c \cos\varphi,\ \ r_y + s_c \sin\varphi] \tag{19}$$

where (r_x, r_y) represents the position of the robot and φ indicates the orientation of the robot to the obstacle. d_{ro} is the distance between the sensor position of the robot to the obstacle. d_{om} is an estimated distance from point z on the obstacle to position m. Position m is an estimated midpoint for the geometric boarder that represents the replanning area of the workspace of the robot. The estimated distance between the position of the robot and point m is represented by d_m. This is computed using (20).

$$d_m = d_{ro} + d_{om} \tag{20}$$

The position of the estimated origin $m(x, y)$ of the geometric boarder can be represented as in (21).

$$m(x, y) = \left[r_x + d_m \cos\varphi,\ \ r_y + d_m \sin\varphi\right] \tag{21}$$

The x and the y-axis for each vertex are computed using (22)-(25).

$$Ax_a = Bx_a = m(x) - b_{dist} \tag{22}$$

$$Cx_a = Dx_a = m(x) + b_{dist} \tag{23}$$

$$Ay_a = Dy_a = m(y) - b_{dist} \tag{24}$$

$$By_a = Cy_a = m(y) + b_{dist} \tag{25}$$

where $m(x)$ and $m(y)$ represent the x and y-axes of the origin of the generated rectangular replanning area estimated to enclose the obstacle. The estimated origin of the obstacle to the rectangular border is denoted as b_{dist} where $b_{dist} > d_m$ to ensure that at least the current position of the robot is included in the replanning area. To be certain that the computed vertices of the replanning area $\square ABCD$ are within the workspace of the robot, (26) is used to refine and recompute the vertices that fall outside the workspace of the robot.

$$A, B, C, D = \begin{cases} (w(x), 1) & x_a \geq w(x) \wedge y_a \leq 0 \\ (1, w(y)) & x_a \leq 0 \wedge y_a \geq w(x) \\ (w(x), w(y)) & x_a \leq w(x) \wedge y_a \geq w(y) \\ (x_a, w(x)) & x_a \leq w(x) \wedge y_a \geq w(y) \\ (w(x), y_a) & x_a \geq w(x) \wedge y_a \geq w(y) \\ (1, y_a) & x_a \leq 0 \wedge y_a \geq 0 \\ (x_a, 1) & x_a \geq 0 \wedge y_a \leq 0 \\ (1, 1) & x_a \leq 0 \wedge y_a \leq 0 \\ (x_a, y_a) & otherwise \end{cases} \tag{26}$$

where $w(x)$ and $w(y)$ are the maximum size of the x and the y axes of the workspace of the vehicle, x_a and y_a are the computed x and y-axes for the vertices of $\square ABCD(Ax_a, Ay_a, Bx_a, By_a, Cx_a, Cy_a, Dx_a$ and $Dy_a)$.

The given computations to obtain the replanning region is based on the position of the robot at the time of obstacle detection in the workspace. To address pose errors of the robot during navigation, the pose of the robot is checked against the computed path using path tracking algorithm presented in [34]. This check ensures that the angle of the robot, θ_r, and that of the path, θ_p, are the same. The error, θ_ε, can be computed as $\theta_\varepsilon = \theta_r - \theta_p$. Using this information, the error of determining the current pose of the robot to serve as input to compute the replanning area and the path for replanning is controlled. The replanning area is computed such that a feasible path can be recomputed from the current position of the robot to goal.

C. NEW PATH COMPUTATION

The replanning is done by computing a roadmap in the rectangular region of the replanning area. Before the computation of the roadmap, the newly detected and other obstacles within the rectangular region based on the original map are inflated using MD to ensure safe path computation. The roadmap is computed using VD. The computed feasible waypoints are added to the initial computed freeway points nw_{xy} to obtain the nodes for the path computation. As demonstrated in (27)-(29), rw_{xy} represents the newly computed roadmap nodes in the rectangular replanning area and cw_{xy} is the column concatenation of nw_{xy} and rw_{xy}.

$$nw_{xy} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} \in \mathbb{R}^{nx2} \tag{27}$$

$$rw_{xy} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_k & y_k \end{bmatrix} \in \mathbb{R}^{kx2} \tag{28}$$

$$cw_{xy} = \begin{bmatrix} nw_{xy}(x, y) \\ rw_{xy}(x, y) \end{bmatrix} \in \mathbb{R}^{(n+k)x2} \tag{29}$$

To avoid incremental increase in space requirement, computed replanning roadmap nodes rw_{xy} are discarded whenever new replanning roadmap is computed. This helps to maintain $\mathbb{R}^{(n+k)x2}$ nodes for replanned path computation.

Using cw_{xy} as input, a new path $rpath(x, y)$ is computed using A* heuristic algorithm used in computing the initial path ensuring that $rpath(x, y) \in s_{zone}$. Just as with the initial path, CSI is used to compute a smooth path *rspath(x,y)* to avoid sharp curves and edges using *rpath(x,y)* as inputs.

Navigation continues with the newly computed path until a new collision-threat obstacle is detected that requires replanning. The algorithm for the path replanning is given in Algorithm 2.

Algorithm 2 Path Replanning

Input: *map(x, y)*, safety space s, scale factor f, constant value c,nw_{xy}
Output: $rspath_{xy}$

```
1:  begin
2:     Initialize $t_r, c_r, s_r, h_r, s_{op}, new_{sr} = s_r$
3:    while goal not reached
4:     Track robot-obstacle distance $d_{ro}$
5:     Track obstacle-path distance $d_{op}$
6:     if $d_{ro} \leq c_r \wedge d_{op} \leq s_{op}$
7:       Use default replan decision to avoid obstacle;
8:     end
9:     elseif $d_{ro} \leq h_r \wedge d_{op} \leq s_{op}$
10:       Determine collision threat //Table 1
11:        if $threat = True \wedge rdecision = True$
12:          Compute $new_{sr}$; //Table 1
13:           if $d_{ro} \leq new_{sr}$
14:          Compute replanning area □ABCD;//(19)-(26)
15:            $rimap(x,y) \leftarrow$ Inflate $W_{obs}$ in □ABCD;//(12)
16:             $rwxy \leftarrow$ Compute roadmap//[34]
17:            $cw_{xy} = (new_{xy};\ rw_{xy})$ ; //(29)
18:            Compute shortest path $f(cw_{xy})$f(cw$_{xy}$);//(15)
19:        Compute $rspath_{xy}$; //(16)
20:      end if
21:       Navigate on $rspath_{xy}$;
22:     end if
23:   end if
24:   end while
25: End
```

V. EXPERIMENTAL RESULTS AND DISCUSSION

This section presents environmental setup for the experiments to validate the proposed method, results of the proposed method, and comparison of the proposed path planning methods with other popular related methods.

A. ENVIRONMENTAL SETUP FOR THE EXPERIMENT

For effective validation of the performance of the proposed method, simulation is carried out in 800 randomly generated dynamic environment maps of 500x500 grid size each. Each map is made up of 23 static and varied number of

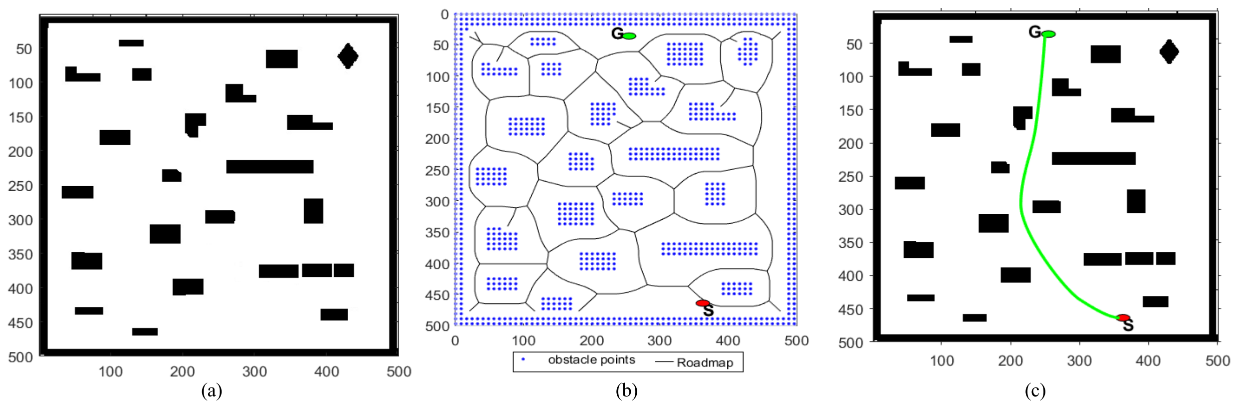


FIGURE 7. Stage1: Initial path computation (a) Initial map with static obstacles (b) Initial computed VD roadmap (c) Final initial path from S to G with static obstacles only.

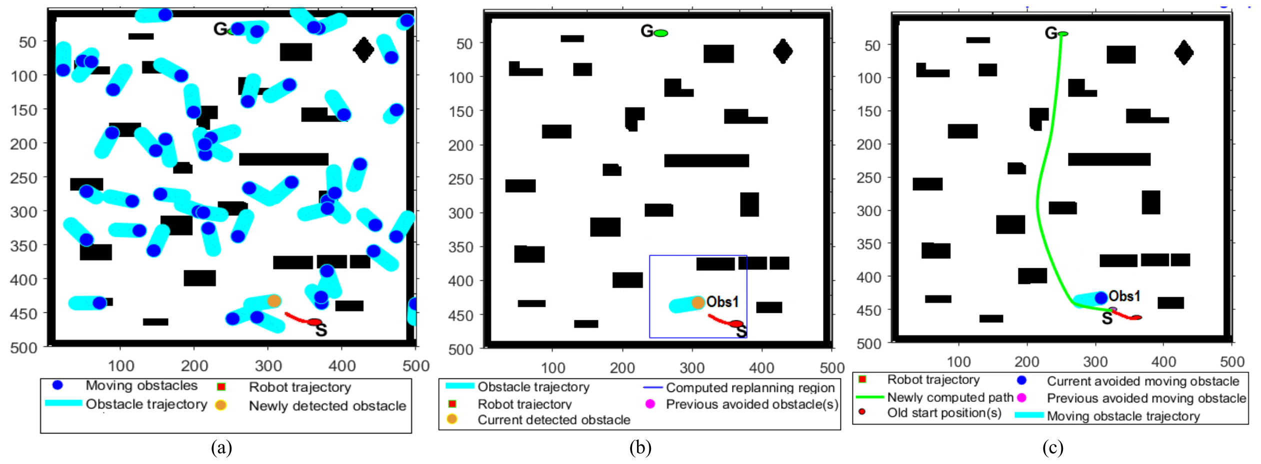


FIGURE 8. Stage 2: First replanning computation to avoid a moving obstacle (a) Navigated path in a dynamic environment towards the first collision-threat moving obstacle (b) First moving obstacle (*obs1*) detected with a computed rectangular region (c) Previous navigated path and the current computed path from S to G that avoids *Obs1.*

moving obstacles. The maps considered in the simulation are categorized into four with each category having 200 different environmental setups in terms of obstacles positions, distances, velocities and directions of movement but with the same number of moving obstacles. In addition to the 23 static obstacles for each map, each category has 20,50,100, and 150 moving obstacles, respectively. There is no control over the movement of obstacles in the experiment. They can run over each other. The random generation guarantees unpredictable obstacles for effective validation of the proposed method. Simulations were carried out using MATLAB.

B. RESULTS OF THE PROPOSED METHOD

This section presents path planning simulation results of the proposed method with 23 static and 50 moving obstacles. Fig. 7-11 present results of path computation at different stages for replanning to avoid obstacles from the initial path computation to the final path computation to reach goal. Obstacle trajectories already covered by moving obstacles are part of the W_{free}. The goal is represented by G while S denotes the current starting point of the robot. Obstacle detection is based on distance measurement compared to a distance threshold based on the proposed algorithm. For real implementation of this method, range sensors are required to detect and provide distance measurements from the obstacles. Observations in each stage shown in Fig. 7-11 are as follows:

- At the first stage, Fig. 7 shows the initial path computation without moving obstacles. The original map, the computed VD roadmap and the final initial path are shown in Fig. 7(a) to 7(c), respectively.
- At the second stage, Fig. 8 presents the first replanning to avoid a moving obstacle (*obs1*). It can be observed from Fig. 8(a) that an obstacle was moving across the robot path from left to the right of the map with velocity almost the same as the robot. It was detected as a collision threat to the robot. Hence, upon reaching the replanning zone, intelligent replanning was done through the left side of the obstacle (Fig. 8(b)-8(c)). Though computing a path through the right side of *obs1* would be shorter, there is danger of colliding with *obs1* which is moving to the right.

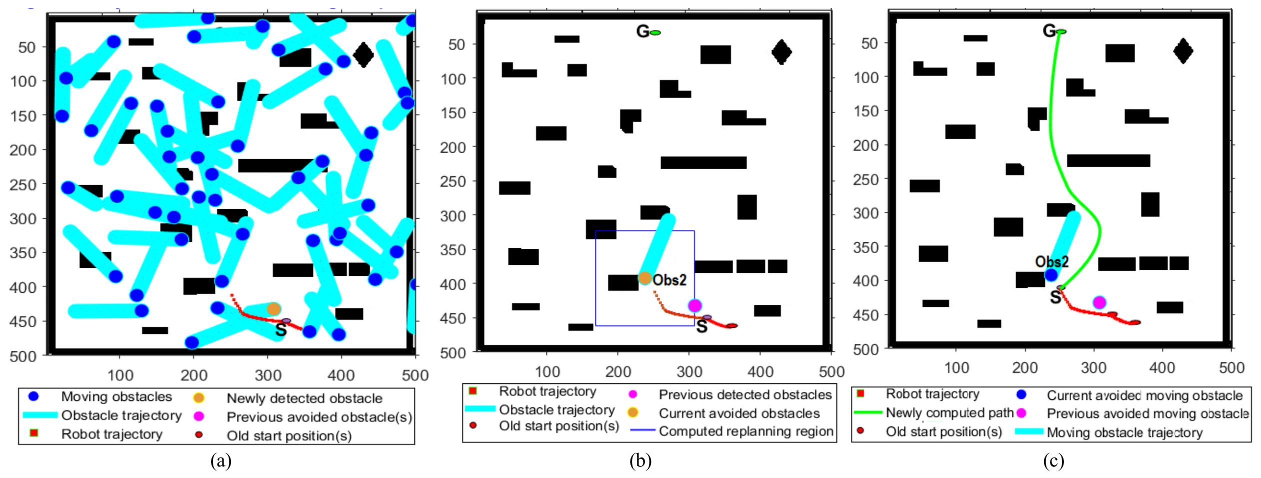


FIGURE 9. Stage 3: Second replanning computation to avoid a moving obstacle (a) Navigated path in a dynamic environment towards the second collision-threat moving obstacle (b) Second moving obstacle (*obs2)* detected with a computed rectangular region (c) Previous navigated path and the current computed path from S to G that avoids *Obs2.*

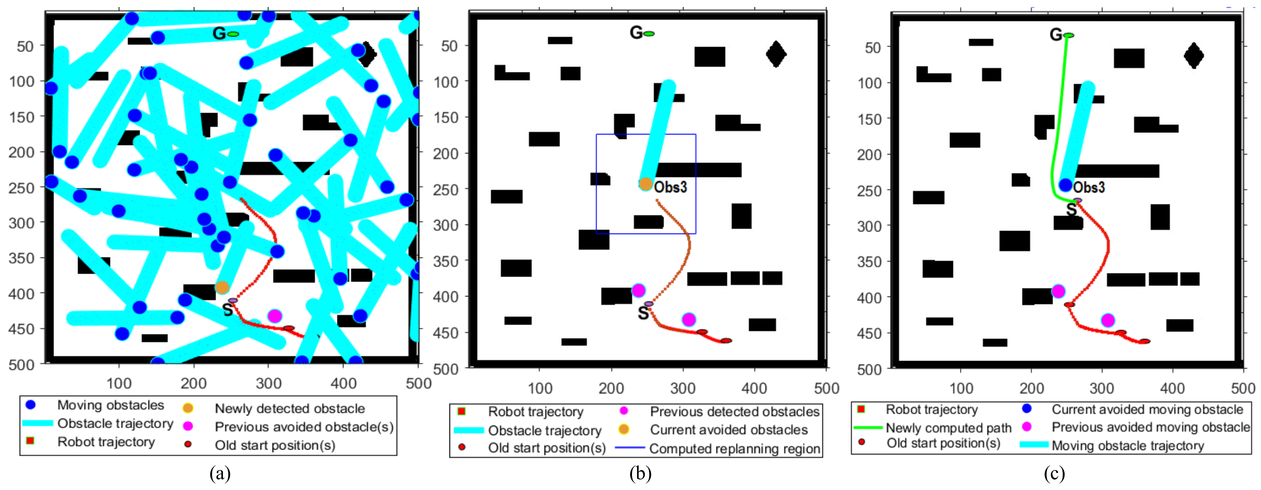


FIGURE 10. Stage 4: Third replanning computation to avoid a moving obstacle (a) Navigated path in a dynamic environment towards the third collision-threat moving obstacle (b) third moving obstacle (*obs3)* detected with a computed rectangular region (c) Previous navigated path and the current computed path from S to G that avoids *Obs3.*

- The third stage shown in Fig. 9 demonstrates the second replanning to avoid a second moving obstacle (*obs2*). In Fig. 9(a), an obstacle was moving at a higher velocity compared to the robot towards the replanning zone of the robot and it was detected as a collision threat to the robot. Considering its direction, the safest direction to replan is the right side of the obstacle. The algorithm computed a new path through the right side of *obs2* to goal (Fig. 9(b)-9(c)).
- Fig. 10 depicts the fourth stage of a third replanning to avoid a third obstacle (*obs3*). As indicated in Fig. 10(a), during navigation from *S*, an obstacle was detected but it was analyzed to have a lower velocity compared to that of the robot. The algorithm identifies it not to be a collision threat, hence no replanning was done. The robot moved and bypassed the obstacle before the obstacle got closer to the path and finally crossed over the robot's navigated path. In Fig. 10(a), another obstacle was detected to be on the path of the robot and moving at a velocity of almost zero. This posed a collision threat as it reaches the replanning zone of the robot. A new path is therefore computed to avoid *obs3*as shown in Fig. 10(b) and Fig.10(c).
- At the final stage as in Fig. 11, the final navigation from S to G is shown. In Fig. 11(a), during navigation from *S* to *G*, an obstacle was detected but was seen not to have posed a collision threat due to its low velocity compared to that of the robot. Hence, no new path was computed and the robot ended its navigation safely to goal (Fig. 11(b)).

The results presented in Fig. 7-11 indicate intelligent path replanning decisions of the proposed algorithm.

C. COMPARISON OF PATH PLANNING METHODS

To evaluate the performance of the proposed method, comparison was made with other popular related methods including RRT-based [35], PRM-based and A*-based [17] techniques. Most path replanning algorithms usually consider either

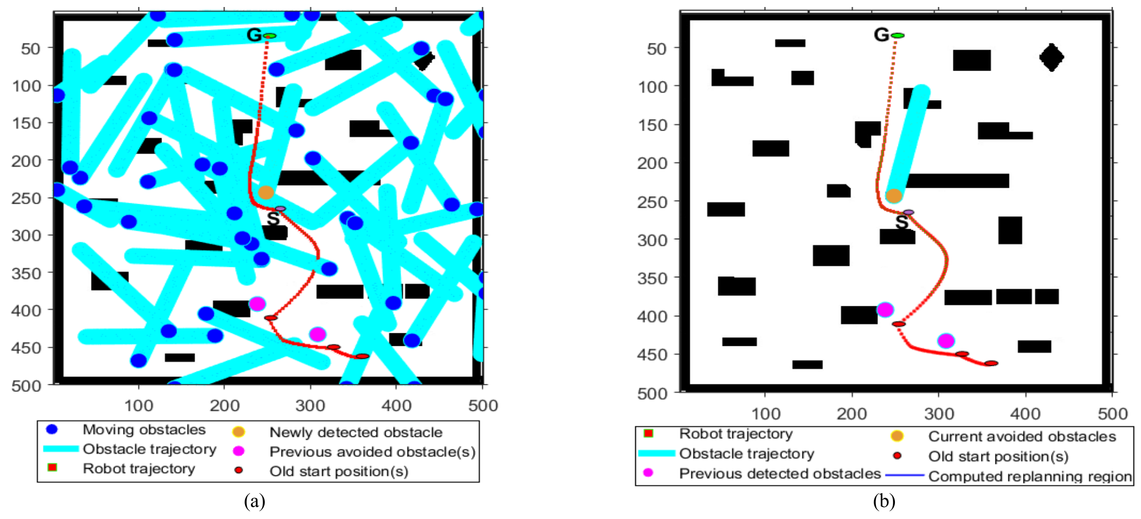


FIGURE 11. Stage 5: Final computed path (a) Navigated path in a dynamic environment towards goal without collision-threat moving obstacles.(b) Final navigated path from the initial position to G.

replanning at incremental distance or time interval, or execute path replanning when an obstacle is detected using the entire map [27]. Hence, the evaluation of the proposed method also consider VD based on incremental (VD-INC) and full map (VD-FM) in the comparison. The aforementioned five methods were implemented on the same platform and datasets (described in Subsection A of this Section) as the proposed method. Consideration was given to the number of environments with successful replanning computations to reach goal, computation time used, path cost in terms of length, and the number of path replanning computations. Table 2 shows results of the comparison. Since the moving obstacles are generated randomly and are not controlled, the inability to find a path may be due to obstacles running over the robot and preventing it from further movement. In addition, the high complexity of an environment may make it impossible to compute a path from start to a goal position. The results in Table 2 indicates that the proposed method outperforms the RRT-based, PRM-based, A*-based, VD-INC and VD-FM replanning approaches. The performance is measured in terms of number of successful path planning computations in complex and dynamic environments, total number of replanned path computations to reach goal, path cost and computation time required to compute and replan the path. The following observations can be made from Table 2.

- Considering path replanning performance in complex and dynamic environments with 20, 50, 100 and 150 moving obstacles, the proposed VD-CGT method succeeded in computing 200 (100%), 181(90.5%), 168 (84%), and 119 (59.5%) paths, respectively. Among the other methods, the highest successful path computations in environments with 20, 50 and 100 moving obstacles were recorded by VD-FM as 194 (97%), 165 (82.5%) and 138 (69%), respectively. With 150 moving obstacles, the highest success rate among the other five methods is 141(70.5%), recorded by A* method.
- In environments with 20, 50, 100 and 150 moving obstacles, the proposed method recorded 57, 152, 326 and 491 total number of path replanning computations to reach goal, respectively. Among the other five methods, the PRM-based method recorded the minimum total number of path replanning computations to reach goal for environments with 20, 50 and 100. The recorded path replanning computations to reach goal for the PRM were 308, 793, and 1407 for environments with 20, 50, and 100 moving obstacles, respectively. VD-INC obtained the lowest total number (1448) of path replanning computations to reach goal for the environment of 150 moving obstacles among the five other methods considered in the comparison.
- The average computation time (in seconds) recorded in environments with 20, 50, 100 and 150 moving obstacles by the proposed method were 4.18, 6.92, 10.14 and 13.81, respectively. The best average computation time recorded among the other five methods in environments with 20 moving obstacles was recorded by RRT method as 4.76s. In environments with 50, 100 and 150 moving obstacles, the best average computation time (in seconds) among the other five methods were recorded by PRM method as 11.33, 16.68 and 24.12, respectively.
- The average path length recorded in environments with 20, 50, 100 and 150 moving obstacles by the VD-CGT method were 493.84, 503.43, 512.90 and 520.86, respectively. The lowest average path length for environments with 20, 50, 100 and 150 moving obstacles among the other methods were obtained by A* as 494.56, 501.73, 509.95 and 511.70, respectively. This indicates that in an environment with 150 moving obstacles, A* recorded

TABLE 2. Path replanning results for 800 maps with different environmental configurations.

	Map	Mobs	ICost	ITime	Suc	NNPF	TNRep	Acost	Atime
VD (Incremental)	200	20	489.91	2.61	149	51	5335	559.16	128.71
	200	50	489.91	2.61	87	113	3449	553.78	92.62
	200	100	489.91	2.60	27	173	2038	518.35	57.99
	200	150	489.91	2.61	8	192	1488	586.13	47.52
VD (Full map)	200	20	489.92	2.53	194	6	670	502.17	11.40
	200	50	489.92	2.54	165	35	1363	513.99	21.90
	200	100	489.92	2.73	138	62	2420	529.43	42.76
	200	150	489.92	2.50	110	90	3250	543.08	54.12
RRT-based	200	20	561.9066	2.09	158	42	415	614.00	4.76
	200	50	573.2334	2.33	160	40	1065	657.32	12.57
	200	100	564.2005	2.01	95	105	1930	700.52	34.89
	200	150	564.5811	2.09	64	136	2512	730.71	59.57
PRM-based	200	20	504.3115	2.32	135	65	308	549.98	10.38
	200	50	504.7393	2.33	98	102	793	588.93	11.33
	200	100	504.7348	2.30	52	148	1407	625.18	16.68
	200	150	503.5795	2.28	31	169	2008	676.26	24.12
A*-based	200	20	462.99	30.18	125	75	344	494.56	62.60
	200	50	462.99	29.24	153	47	1047	501.73	117.39
	200	100	462.99	31.19	134	66	2112	509.95	254.13
	200	150	462.99	32.17	141	59	3055	511.70	359.49
Proposed	200	20	489.92	2.49	200	0	57	493.84	4.18
	200	50	489.92	2.60	181	19	152	503.43	6.92
	200	100	489.92	2.50	168	32	326	512.90	10.14
	200	150	489.92	2.53	119	81	491	520.86	13.81

Mobs=number of moving obstacles, *ICost* and, *ITime* represent average initial path length and time used to compute path with only static obstacles, respectively. *Suc* = number of configurations representing the maps that generate successful replanned path from start to goal. *TNRep* =total number of replanning computations recorded, *ACost* =average final path cost, *ATime* =average time used to reach goal, and *NNPF* =number of environmental configurations where the algorithms could not compute path to goal

shorter path length, which is 9.16 less than that of VD-CGT. However, A* used an average of 359.49s as against 13.81s for VD-CGT to achieve the given path length gain.

Performance in terms of path cost and computation time for each map with 20 and 50 moving obstacles in addition to 23 static obstacles for the comparison based on the maps which all the six methods were successful in computing path to goal is demonstrated in Fig. 12 and Fig. 13. Fig. 14 shows the performance in terms of total number of replanning for each map for the comparisons where all the six methods were successful in computing path to goal as well as at least one path replanning computation. Charts for the environments for 100 and 150 were not included because they recorded less than five successful path computations common to all the six methods used in the comparison. Fig. 12-14 indicate better performance of the proposed method with respect to path cost, computation time used to reach goal, and the number of replanning computations for each map with both static and dynamic obstacles.

To give more insights based on the results of the simulation, Table 3 demonstrates a summary of the performance of each of the methods considered in the comparison including the proposed method. Out of the 800 environmental configurations, the proposed method obtained the highest success rate of 83.5% (668) whereas VD-FM, A*, RRT, PRM and VD-INC obtained 75.9% (607), 69.1% (553), 59.6% (477), 39.5% (316), and 33.9% (271) success rates, respectively. The proposed method had the efficient average replanning per an environment of 1.54. With respect to the average path cost, the A* method got the best path cost of 504.49 which is a difference of 3.27 better than the path cost obtained by the VD-CGT (507.76). However, A* used the worse average computation time per an environment of 198.4s as against

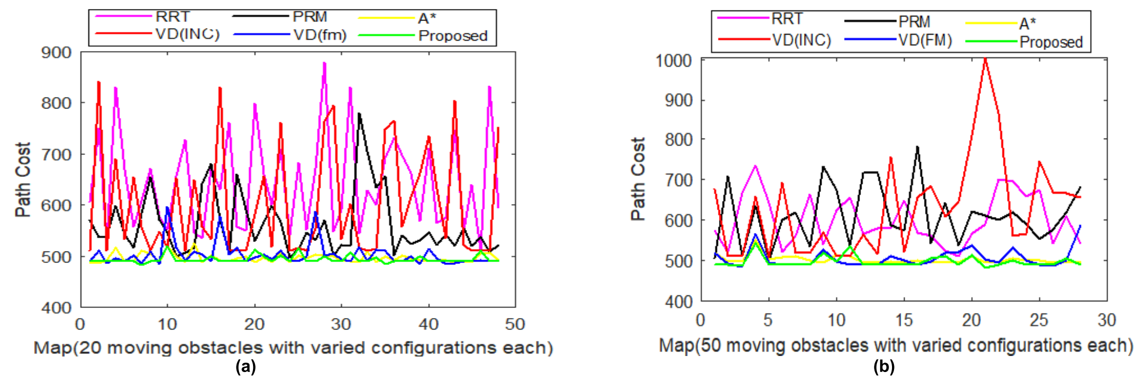


FIGURE 12. Path costs recorded for each common successful trial using the proposed and other five methods (a) Map with 23 static and 20 moving obstacles (b) Map with 23 static and 50 moving obstacles.

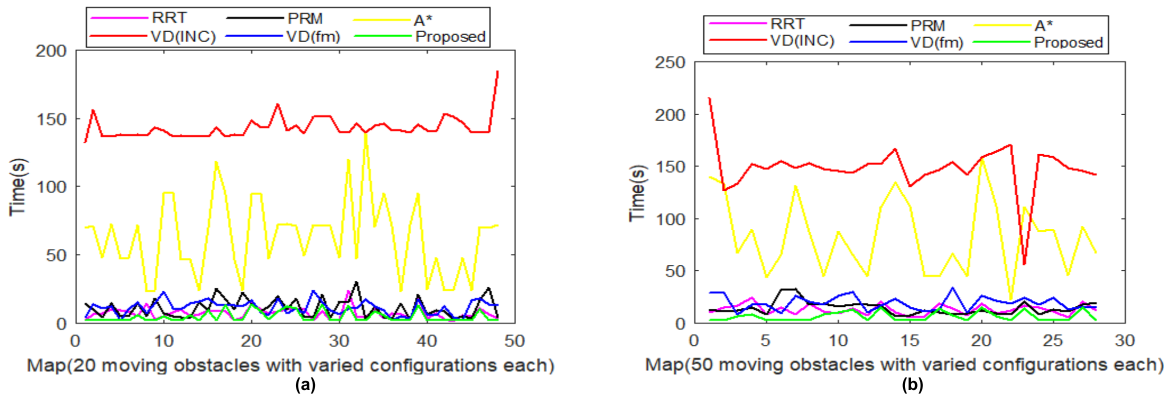


FIGURE 13. Execution time recorded to reach goal for each common successful trial using the proposed and other five methods (a) Map with 23 static and 20 moving obstacles (b) Map with 23 static and 50 moving obstacles.

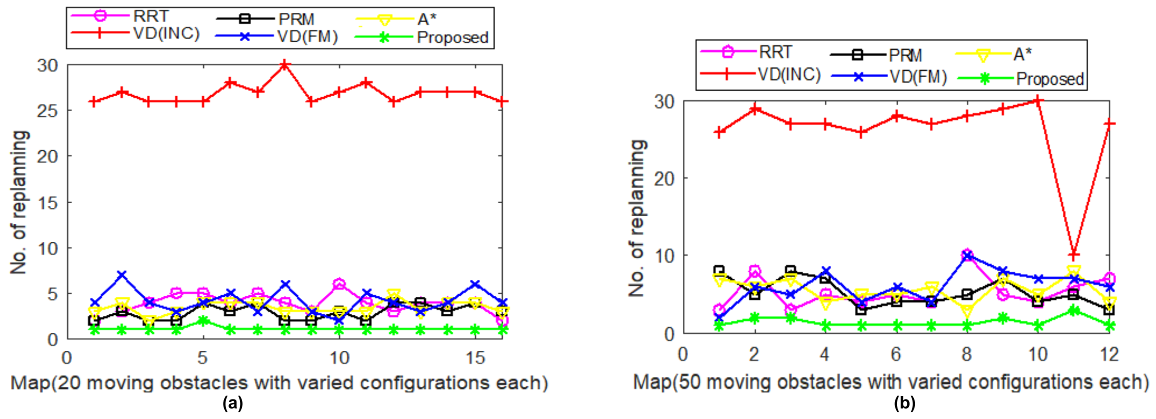


FIGURE 14. Number of replanning computations recorded to reach goal for each common successful trial using the proposed and other five methods (a) Map with 23 static and 20 moving obstacles (b) Map with 23 static and 50 moving obstacles.

8.76s for the proposed method. To sum it all, the best path computation speed (cost/time) of 57.96 was obtained by the proposed method as against 2.54 for A* method.

The performance of the proposed method is as a result of the introduction of the algorithm to categorize and determine collision threat level to help choose appropriate replan decisions. This also helps to record lower computation time. The lower computation time recorded is also due to the use of CGT to select small area of the environment as the replanning area for the path computation. It is noted from the results that the proposed method performed very well with lower computation time requirement compared to RRT-based, PRM-based, A*-based, VD-INC and VD-FM replanning approaches. Generally, computation time is

TABLE 3. Summary of path replanning results using 800 maps with different environmental configurations.

	Success (800 Maps)		Average Computations (800 Maps)			
	Total	%	Rep	Path Cost	Time	Speed (Cost/Time)
VD (Incremental)	271	33.88	45.42	554.36	81.71	6.780
VD (Full map)	607	75.88	12.69	522.17	32.55	16.04
RRT-based	477	59.63	12.42	675.64	27.95	24.18
PRM-based	316	39.5	145.68	610.09	15.63	39.04
A*-based	553	69.13	11.86	**504.49**	198.40	2.540
Proposed	**668**	**83.5**	**1.54**	507.76	**8.76**	**57.96**

expected to increase as the number of replannings to reach a goal increase. The higher time requirement to compute replanned path using VD-INC method is not surprising since incremental method recomputes path for replanning at a given time or distance intervals even if there are no collision threats. It is also noted from the results that; the proposed approach comparatively performs better by successfully computing path in the given complex and dynamic environmental configurations at highest success rate (83.5%) and highest computation speed (57.96) in the experiment.

VI. CONCLUSION

In this paper, a novel path planning algorithm for mobile robots in complex and dynamic environments is presented using VD and CGT. An algorithm to categorize moving obstacles based on their positions, velocities, distances and directions of movement to determine the collision threat level of obstacles to a vehicle with reference to its kinematics to provide possible intelligent replanning decision is given.

Comparative results against RRT-based, PRM-based, A*-based, VD-INC and VD-FM approaches demonstrate better performance of the proposed method in terms of success rate of path computation in complex and dynamic environments, lower number of replanning computations, low path cost and low computation time. It can be concluded from the simulation experimental results that the VD-CGT method is efficient in determining collision-threat moving obstacles and avoiding unnecessary replanning computations. This resulted in low number of path replanning computations to reach goal. Additionally, the method has good performance with regard to safe, smooth and low path length computation with low computation time for path replanning in complex and dynamic environments. The proposed method is therefore promising to providing low path replanning computation time required for robots in motion to take quick decision before they collide with obstacles.

Notwithstanding the positive results of the proposed method, the size and shape of obstacles can affect its performance. Situations where the computed rectangular region is not large enough to accommodate the size of detected obstacle may results in wrongful path computations. To address this challenge, our future work would employ obstacle size and shape algorithms similar to those in [36], [37] to include the size and shape of moving obstacles in computing the rectangular region for the path replanning. Future consideration is also given to implementing the algorithm to work with multiple robots in dynamic environments. Real implementation is under consideration in our future work.

REFERENCES

[1] Y. Chen and J. Sun, "Distributed optimal control for multi-agent systems with obstacle avoidance," *Neurocomputing*, vol. 173, pp. 2014–2021, Jan. 2016.

[2] A. S. Matveev, A. V. Savkin, M. Hoy, and C. Wang, "Biologically-inspired algorithm for safe navigation of a wheeled robot among moving obstacles," in *Safe Robot Navigation Among Moving and Steady Obstacles*, 1st ed. Oxford, U.K.: Butterworth-Heinemann, 2016, pp. 161–184. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128037300000081

[3] M. Wang, J. Luo, and U. Walter, "A non-linear model predictive controller with obstacle avoidance for a space robot," *Adv. Space Res.*, vol. 57, no. 8, pp. 1737–1746, 2016.

[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.*, vol. 2018, Jul. 2018, Art. no. 8269698.

[5] B. Kakillioglu, K. Ozcan, and S. Velipasalar, "Doorway detection for autonomous indoor navigation of unmanned vehicles," in *Proc. IEEE Int. Conf. Image Processing*, Phoenix, AZ, USA, Sep. 2016, pp. 3837–3841.

[6] M. Candeloro, A. M. Lekkas, and A. J. Sørensen, "A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels," *Control Eng. Pract.*, vol. 61, pp. 41–54, Apr. 2016.

[7] H. Dong, W. Li, J. Zhu, and S. Duan, "The path planning for mobile robot based on Voronoi diagram," in *Proc. 3rd Int. Conf. Intell. Netw. Intell. Syst.*, Shenyang, China, 2010, pp. 446–449.

[8] H.-X. Wei, Q. Mao, Y. Guan, and Y.-D. Li, "A centroidal Voronoi tessellation based intelligent control algorithm for the self-assembly path planning of swarm robots," *Expert Syst. Appl.*, vol. 85, pp. 261–269, Nov. 2017.

[9] Q. Hui and J. Cheng, "Motion planning for AmigoBot with line-segment-based map and Voronoi diagram," in *Proc. Annu. IEEE Syst. Conf. (SysCon)*, Orlando, FL, USA, Apr. 2016, pp. 1–8.

[10] N. Habib, D. Purwanto, and A. Soeprijanto, "Mobile robot motion planning by point to point based on modified ant colony optimization and Voronoi diagram," in *Proc. Int. Seminar Intell. Technol. Appl. (ISITIA)*, Lombok, Indonesia, 2016, pp. 613–618.

[11] Q. Wang, M. Langerwisch, and B. Wagner, "Wide range global path planning for a large number of networked mobile robots based on generalized Voronoi diagrams," in *Proc. IFAC Symp. Telematics Appl.*, Seoul, South Korea, 2013, pp. 107–112.

[12] M. Dakulović and I. Petrović, "Two-way D* algorithm for path planning and replanning," *Robot. Auto. Syst.*, vol. 59, no. 5, pp. 329–342, 2011.

[13] X. Hu, L. Chen, B. Tang, D. Cao, and H. Hee, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles," *Mech. Syst. Signal Process.*, vol. 100, pp. 482–500, Feb. 2018.

[14] B. K. Patle, A. Pandey, A. Jagadeesh, and D. R. Parhi, "Path planning in uncertain environment by using firefly algorithm," *Defence Technol.*, vol. 14, pp. 691–701, Dec. 2018.
[15] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Appl. Soft Comput. J.*, vol. 77, pp. 236–251, Apr. 2019.
[16] M. Elhoseny, A. Tharwat, and A. E. Hassanien, "Bezier curve based path planning in a dynamic field using modified genetic algorithm," *J. Comput. Sci.*, vol. 25, pp. 339–350, Mar. 2018.
[17] Y. Singh, S. Sharma, R. Sutton, D. Hatton, and A. Khan, "A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents," *Ocean Eng.*, vol. 169, pp. 187–201, Dec. 2018.
[18] Y. Zhang, L. Wu, and S. Wang, "UCAV path planning by fitness-scaling adaptive chaotic particle swarm optimization," *Math. Problems Eng.*, vol. 2013, Jun. 2013, Art. no. 705238.
[19] O. Montiel, U. Orozco-Rosas, and R. Sepúlveda, "Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles," *Expert Syst. Appl.*, vol. 42, no. 12, pp. 5177–5191, 2015.
[20] M. A. Hossain and I. Ferdous, "Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique," *Robot. Auto. Syst.*, vol. 64, pp. 137–141, Feb. 2015.
[21] P. Yao, H. Wang, and Z. Su, "Real-time path planning of unmanned aerial vehicle for target tracking and obstacle avoidance in complex dynamic environment," *Aerosp. Sci. Technol.*, vol. 47, pp. 269–279, Dec. 2015.
[22] W.-B. Xu, X.-B. Chen, J. Zhao, and X.-P. Liu, "Function-segment artificial moment method for sensor-based path planning of single robot in complex environments," *Inf. Sci.*, vol. 280, pp. 64–81, Oct. 2014.
[23] J. C. Mohanta, D. R. Parhi, and S. K. Patel, "Path planning strategy for autonomous mobile robot navigation using Petri-GA optimisation," *Comput. Elect. Eng.*, vol. 37, no. 6, pp. 1058–1070, 2011.
[24] M. A. P. Garcia, O. Montiel, R. Castillo, R. Sepúlveda, and P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation," *Appl. Soft Comput.*, vol. 9, no. 3, pp. 1102–1110, 2009.
[25] X. Zhang, Y. Zhao, N. Deng, and K. Guo, "Dynamic path planning algorithm for a mobile robot based on visible space and an improved genetic algorithm," *Int. J. Adv. Robot. Syst.*, vol. 13, no. 3, pp. 1–17, 2016.
[26] M. A. K. Jaradat, M. Al-Rousan, and L. Quadan, "Reinforcement based mobile robot navigation in dynamic environment," *Robot. Comput. Integr. Manuf.*, vol. 27, no. 1, pp. 135–149, Feb. 2011.
[27] J. Han and Y. Seo, "Path regeneration decisions in a dynamic environment," *Inf. Sci.*, vol. 450, pp. 39–52, Jun. 2018.
[28] A. Šelek, M. Seder, and I. Petrović, "Mobile robot navigation for complete coverage of an environment," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 512–517, 2018.
[29] S. G. Tzafestas, "Mobile robot kinematics," in *Introduction to Mobile Robot Control*. Amsterdam, The Netherlands: Elsevier, 2014, pp. 31–67.
[30] R.C. Gonzalez and R. E. Wood, *Digital Image Processing*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.
[31] Y.-J. Ho and J.-S Liu, "Collision-free curvature-bounded smooth path planning using composite Bezier curve based on Voronoi diagram," in *Proc. IEEE Int. Symp. CIRA*, Daejeon, South Korea, Dec. 2009, pp. 463–468.
[32] S. Fortune, "A sweepline algorithm for Voronoi diagrams," *Algorithmica*, vol. 2, pp. 153–174, 1987.
[33] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2010.
[34] R. Lenain, E. Lucet, C. Grand, B. Thuilot, and F. Ben Amar, "Accurate and stable mobile robot path tracking: An integrated solution for off-road and high speed context," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Taipei, Taiwan, Oct. 2010, pp. 196–201.
[35] D. Connell and H. M. La, "Extended rapidly exploring random tree–based dynamic path planning and replanning for mobile robots," *Int. J. Adv. Robot. Syst.*, vol. 15, no. 3, pp. 1–15, 2018.
[36] H. Saito, "Estimating target-object shape using location-unknown mobile fixed-direction distance sensors," Nov. 2017, *arXiv:1712.00382*. [Online]. Available: https://arxiv.org/abs/1712.00382
[37] A. Pieropan, N. Bergström, M. Ishikawa, D. Kragic, and H. Kjellström, "Robust tracking of unknown objects through adaptive size estimation and appearance learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Stockholm, Sweden, May 2016, pp. 559–566.

BEN BEKLISI KWAME AYAWLI received the B.Ed. degree in information technology from the University of Education, Winneba, Ghana, in 2008, and the M.Sc. degree in information technology from Sikkim Manipal University, India, in 2011. He is currently pursuing the Ph.D. degree in automation with the Nanjing Tech University, Nanjing, China.

He was a Lecturer with the Computer Science Department, Sunyani Technical University, Ghana. He was the ICT Director of Sunyani Technical University, from 2013 to 2016. He is also a Web Application Developer. His research interests include robot path planning and navigation, deep learning, data mining, and web applications.

XUE MEI received the B.S. degree in engineering from the Harbin Institute of Technology, Harbin, China, in 1998, the M.S. degree in engineering from Southeast University, Nanjing, China, in 2004, and the Ph.D. degree in engineering from Southeast University, Nanjing, China, in 2008.

She is currently an Associate Professor with the College of Electrical Engineering and Control Science, Nanjing Tech University. Her research interests include pattern recognition, robot motion planning, machine vision, and image processing.

MOUQUAN SHEN received the Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 2011. He is currently a Professor with the Department of Electrical Engineering and Control Science, Nanjing Technology University, Nanjing, China. His current research interests include Markov jump systems, robust control, adaptive control, iterative learning control, sliding mode control, and quantized control.

ALBERT YAW APPIAH received the B.Sc. and M.Sc. degrees from the University of Mines and Technology, Tarkwa, Ghana, in 2007 and 2014, respectively. He is currently pursuing the Ph.D. degree with Nanjing Tech University, Nanjing, China. He was with the Electrical/Electronic Engineering Department, Sunyani Technical University, as a Lecturer. His present research interests include solar photovoltaics, deep learning, reinforcement learning, robotics, and fault diagnosis.

FRIMPONG KYEREMEH received the bachelor's degree in technology education from the University of Education Winneba, Ghana, in 2005, and the M.Sc. degree in electrical engineering with power electronics from Bradford University, U.K., in 2008. He is currently pursuing the Ph.D. degree in electrical engineering and control science with Nanjing Tech University, China. He is currently a Lecturer with the Electrical/Electronic Engineering Department, Sunyani Technical University, where he taught courses on basic electronics, power electronics, engineering practice, and power systems. His current research interests include microgrid control, machine learning, and multi-agent systems.