Robotica: (2019) volume 37, pp. 1438–1456. © Cambridge University Press 2019
doi:10.1017/S0263574719000043

Dynamic Window with Virtual Goal (DW-VG): a New Reactive Obstacle Avoidance Approach Based on Motion Prediction

Yu Xinyi, Zhu Yichen, Lu Liang and Ou Linlin*

Department of Automation, College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China.
E-mails: yuxinyinet@163.com, zhuyic@yeah.net, luliangeric@126.com

(Accepted January 1, 2019. First published online: March 4, 2019)

SUMMARY
This paper proposes a dynamic window with virtual goal (DW-VG) method for local collision avoidance in dynamic environments. Firstly, the debounce filter and polynomial curve-fitting algorithm are combined to predict the trajectory of the obstacles with timestamps. Based on the motion prediction of the obstacles, the virtual goal is proposed to replace the real goal, so that the robot can escape from the concave trap and avoid the dynamic obstacles. According to the timestamps and virtual goal, the optimal linear and angular velocities are selected from the dynamic window, which drive the robot toward its real goal. The simulation and experimental results show that the DW-VG method can not only escape the local minima and avoid dynamic obstacles but also is applicable to the dense environment. Furthermore, the simulation results also verify that the DW-VG method drives the robot to reach its goal faster and safer than other reactive obstacle avoidance methods.

KEYWORDS: Mobile robots; Motion prediction; Obstacle avoidance; Dynamic window; Reactive navigation.

1. Introduction

The increasing need for fully autonomous mobile robots in high-risk areas has attracted the attention of researchers.[1] The main objective of these robots is to execute the tasks in places where the people condition is dangerous or the tasks are impossible to be executed by human.[2] However, the environment where the robot executes tasks are partially or completely unknown. Besides, dynamic obstacles may collide with the robot when the robot performs tasks. Usually, for the self-navigating robot to execute a task, a global path planning is generated firstly, and then the robot carries out a reactive collision avoidance method employing its sensing devices when the obstacles are detected until the task is finished.

Early work in the reactive collision avoidance includes the BUG algorithms, which the robot avoids the obstacle only by following its edge.[3] The DistBug algorithm allows the robot to abandon following the obstacle boundaries as soon as it can reach the goal directly.[4] The TangentBug algorithm uses the local tangent graph (LTG) to choose the locally optimal direction while moving along an obstacle boundary.[5] The advantage of the above-mentioned methods are that the robot only needs collision sensors, which is well used in the sweeping robots. However, these algorithms may fail when there exist dynamic obstacles.[6]

The potential field (PF) approach was proposed to avoid dynamic obstacles.[7] The basic concept of the PF was that the movement of the robot is determined by the repulsive force of obstacles and the

*Corresponding author. E-mail: linlinou@zjut.edu.cn

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

attraction of the goal. Although the PF approach is considered to be computationally efficient,[8] it is easy to be stuck when it drives the robot to pass through a narrow passage.[9] The vector field histogram (VFH) approach transforms the two-dimensional environment map into one-dimensional histogram which represents the collision-free direction of the robot.[10] Compared with the PF approach, the VFH approach eliminates the oscillation when the robot passes through the dense area of the obstacles.[11] However, for the original VFH method, the robot is considered as a particle, which neglects its kinematics. For the VFH+ method, the kinematics of the robot is considered, which makes the motion of the robot smoother and safer.[12] Unfortunately, it is a significant problem of the PF and VFH methods for the robot to escape the local minima. Based on the PF method, a modified Newton method was proposed to eliminate the oscillation when the robot passes a narrow passage.[13] The Tangential Escape (TE) is used for the robot to escape from the local minima.[14] Although the TE method overcomes the shortcoming of the PF and VFH methods in some cases, its effectiveness decreases as the density of the obstacles increases.[15]

The linear and angular velocities which can be reached in the next time interval according to the kinematic parameters of the robot are selected by using the dynamic window (DW) approach.[16,17] The DW approach is well used in the unstructured static environment but may fail in the dynamic obstacles avoidance. By taking into account the velocity of obstacles, the velocity obstacles (VO) method is applied in the uncertain dynamic environment.[18,19] Nevertheless, for the DW and VO methods, there exits the possibility that the robot traps in concave areas.

A search graph method was proposed to find a free-collision path in the complex environment where the robot needs to avoid the obstacles, and follows traffic restrictions.[20] However, for the search graph method, the obstacles are supposed to be stationary. Goal-seeking module (GSM), static and dynamic obstacle avoidance module (SDOAM), emergency module (EM), and robot-setting module (RSM) are combined to steer the robot working in unknown unstructured environments.[21] Rapidly-Exploring Random Tree (RRT) can quickly explore the unknown regions through random sampling in the configuration space.[22] However, the efficiency of the RRT algorithm will be reduced dramatically when a large amount of obstacles scatter in the environment.[23,24] A new navigation algorithm is designed based on future prediction and priority behavior of the robot.[25] Lots of simulation experiments prove the performance of the new navigation algorithm is better than the state-of-the-art algorithms. However, the method is more reasonable if the kinematic parameters of the robot are considered.

Therefore, in order to deal with these existing problems, in this paper, we propose the dynamic window with virtual goal (DW-VG) method. The debounce filter (DF) and polynomial curve-fitting algorithm (PCF) are combined to predict the trajectory of the obstacles, and the virtual goal is proposed to replace the real goal, so that the robot can escape from the concave trap and avoid the dynamic obstacles. Furthermore, several experiments are provided to verify the effectiveness of the DW-VG method. We also introduce a performance evaluation to quantitatively estimate the power of the proposed method, and compare it with the traditional reactive obstacle avoidance methods such as the DW, BUG, PF, and VFH when kinematics parameters of the robot are considered.

The remainder of this paper is organized as follows. After putting forward the problem in Section 2, we present the DW-VG method design in Section 3. The simulation and experimental results are discussed in Sections 4 and 5, respectively. Section 6 highlights conclusion and future work.

2. Problem Formulation

In this paper, we focus on designing a reactive collision avoidance method when the status of the dynamic obstacles are not known beforehand. The environment where the robot executes tasks as shown in Fig. 1. The robot needs to avoid these obstacles and reaches the goal safely.

3. The Reactive Obstacle Avoidance Method Design

The DW-VG method for reactive obstacle avoidance will be presented in this section. Firstly, the position of the obstacles in previous moments is employed to predict the motion of the obstacles in future moments with timestamps. Based on the motion prediction of the obstacles, the virtual goal replaces the real goal, which helps the robot to escape from the concave trap and to avoid the dynamic

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

Table I. Notations about the motion prediction of the obstacles.

Symbol	Description
i	Data frame
D	New data
R	Raw data for prediction
$\mathbf{Q}$	Quadratic polynomial
P	Prediction of the obstacles
s	Threshold for state change
n	Number of state change
i_{state}	Determining state change
Q2	Temporary storage **Q**

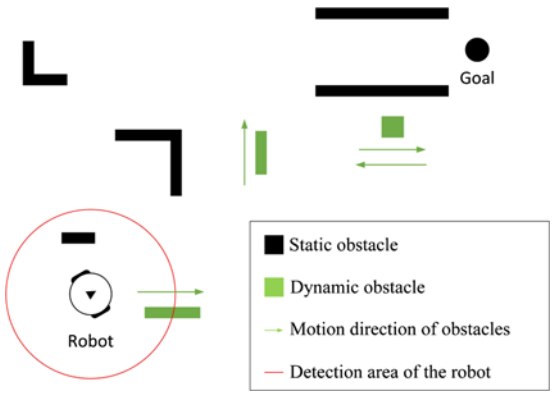


Fig. 1. The environment where the robot executes tasks.

obstacles. According to the motion prediction of the obstacles with timestamps and virtual goal, the optimal linear velocity and angular velocity are selected from the DW, which drive the robot toward its real goal.

3.1. Motion prediction of the obstacles

The DF and PCF algorithms are combined to predict the trajectory of the obstacles. Firstly, the position of obstacles is decomposed into X-axis and Y-axis, respectively. Then the position of each obstacle in previous moments is used to predict its position in future moments based on PCF algorithm. In order to increase the reliability of the prediction, the DF is applied to distinguish different movement states of the obstacle. Finally, the motion prediction of all the obstacles can be obtained according to the motion prediction of each obstacle on X-axis and Y-axis. For easy reference, the important notations are listed in Table I.

Assume that the motion state of the obstacle is one of the static, uniform motion, and uniformly accelerated motion states. The following quadratic polynomial is introduced to represent the motion of the obstacle:

$$y(x, w) = w_0 + w_1 x + w_2 x^2 \tag{1}$$

Then, a fit can be obtained by minimizing the following error function:

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2 \tag{2}$$

where x_n is the data frame, t_n is the position of the obstacle which is obtained from the sensor, and N is the length of P:

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

$$\frac{\partial E(w)}{\partial w_i} = \sum_{n=1}^{N} \{y(x_n, w) - t_n\} x_n^i$$
$$= \sum_{n=1}^{N} \left\{ \sum_{j=0}^{2} w_j x_n^j - t_n \right\} x_n^i \tag{3}$$

Setting Eq. (3) to zero, it is obtained that

$$\sum_{n=1}^{N} \sum_{j=0}^{2} w_j x_n^{i+j} = \sum_{n=1}^{N} t_n x_n^i \tag{4}$$

Equation (2) is a quadratic in the coefficients w, and the derivative with respect to coefficients will be linear in elements of w. Thus, Eq. (2) has a unique solution w^* by solving Eq. (4). The resulting quadratic polynomial is defined as

$$\mathbf{Q} = y(x, w^*) \tag{5}$$

The performance of the motion prediction based on the PCF algorithm decreases when the state of obstacles changes. Therefore, the DF is introduced to distinguish different movement states of the obstacle firstly, and then the PCF algorithm is applied to predict the motion of the obstacle. The motion prediction of the obstacle based on the PCF and the PCF+DF algorithms are compared with the reference data that are shown in Fig. 2. In Fig. 2(a), the obstacle does not move when the data frame is 0–7, and then it changes into the uniformly accelerated motion state when the data frame is 8–15; in Fig. 2(b), the obstacle moves toward $+Y$ in uniform motion state when the data frame is 0–7, and then it moves toward $-y$ in uniformly accelerated motion state when the data frame is 8–15. It is obvious that the prediction data based on the PCF+DF algorithm are closer to the reference data compared with the PCF algorithm. The details of motion prediction of the obstacle based on PCF+DF algorithm are as follows. Firstly, R will be filled by D without the motion prediction. After R is filled, $\mathbf{Q}$ can be deduced according to Eqs. (1)–(5). The motion prediction of obstacle P on X-axis or Y-axis in future moments can be obtained according to $\mathbf{Q}$. The distance d between D and $P[n]$ is obtained to judge if the state of obstacle changes:

$$d = abs(D - P[n]) \tag{6}$$

$$d > s \tag{7}$$

where the n represents the number of Eq. (7) is satisfied continuously. Once Eq. (7) is satisfied, caching the $\mathbf{Q}$ to $\mathbf{Q2}$, and $i_{\text{state}} = i_{\text{state}} + 1$. If the following equation is satisfied,

$$n = i_{\text{state}} \tag{8}$$

which represents the state of obstacle changes, then update the $\mathbf{Q}$ and discard $\mathbf{Q2}$. If Eq. (7) is satisfied and Eq. (8) is not satisfied, which represents that the state of obstacle may change, then caching the $\mathbf{Q}$ to $\mathbf{Q2}$. Because if the new $\mathbf{Q}$ is used to judge the change of state, the overfitting will happen and the motion prediction may fail.

If Eq. (7) is not satisfied continuously, which means the noise data are obtained. Then discard D and update $\mathbf{Q2}$ to $\mathbf{Q}$. Finally, according to X-axis and Y-axis prediction position of each obstacle, the prediction positions of all the obstacles at $t = t_{i+1}, t_{i+2}, \ldots$ are obtained. Figure 3 shows the motion prediction of the obstacles based on PCF+DF algorithm, where the black solid blocks represent the obstacles position in previous moments, and the black hollow blocks represent the obstacles position in future moments. The PCF+DF algorithm is depicted in Algorithm 1.

3.2. Generation of the virtual goal

By replacing the real goal with a virtual goal, the robot can avoid trapping in the local minimum and work in dynamic environments. The candidate virtual goals are set on the boundary of the detection area of the robot (see Fig. 4: the solid squares represent the candidate virtual goals). When $t = t_i$, the algorithm of generating an optimal virtual goal is described in Algorithm 2.

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

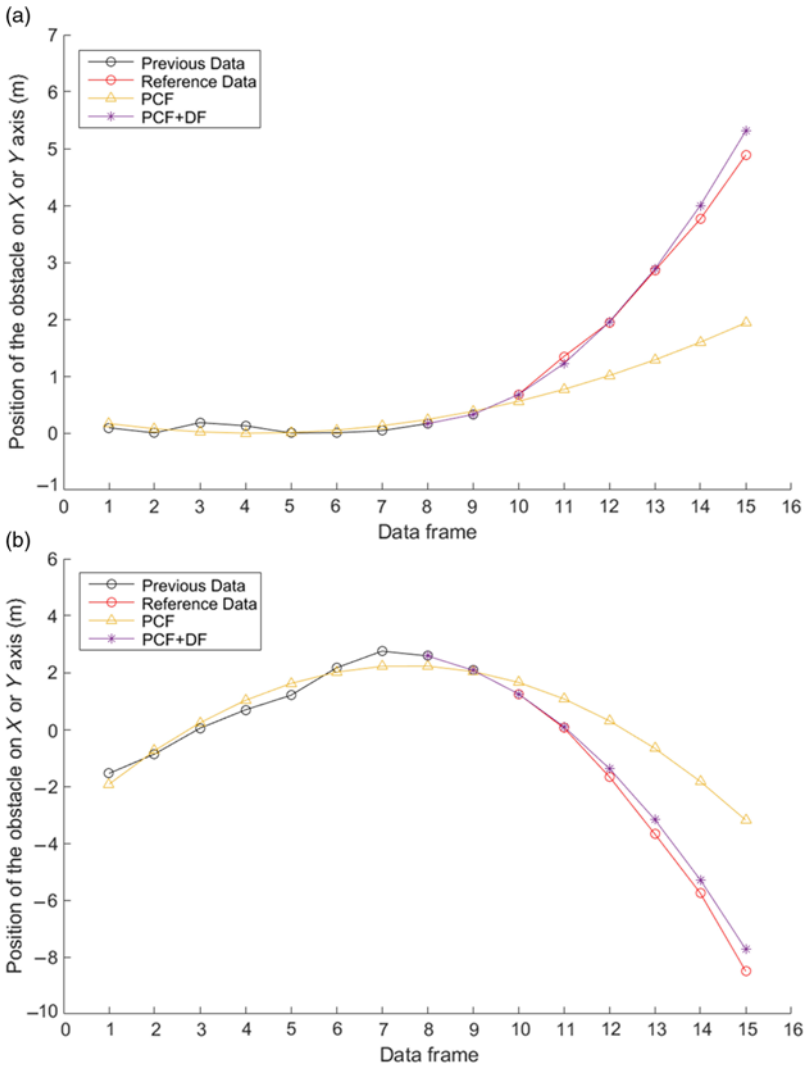


Fig. 2. The prediction of the obstacle position based on PCF and PCF+DF algorithms. (a) The state of the obstacle from static to motion state. (b) The motion state of the obstacle changes.

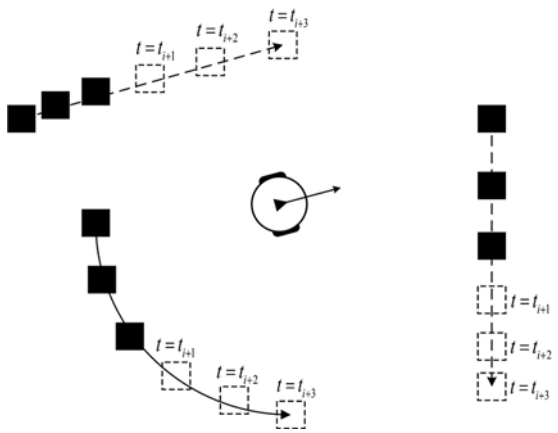


Fig. 3. The motion prediction of the obstacles based on PCF+DF algorithm.

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

Algorithm 1: Motion prediction of the obstacle

Input: i, D
Output: P
while *the robot does not reach its goal* **do**
  if $i \leq length(R)$ **then**
    $R[i] = D$
  else
    $\mathbf{Q} = \text{polyfit}(R)$;
    if $n \neq 0$ **then**
      Q=**Q2**;
    end
    $P = \text{polyval}(\mathbf{Q})$;
    d = abs($D - P[n]$);
    if $d > s$ **then**
      n=n+1;
      if n=*1* **then**
        $\mathbf{Q2} = \mathbf{Q}$;
      end
      if $n = i_{state}$ **then**
        n=0;
      end
    else
      n=0;
    end
    Discard the noise data;
    Updata the R;
  end
  i=i+1;
end

Algorithm 2: Generate a virtual goal

Input: Goal, robotic state, evaluation function
Output: Optimal virtual goal
while *the robot does not reach its goal* **do**
  if *the real goal enter robotic detection area* ***and*** *no obstacles are in the robotic detection area* **then**
    virtual goal=goal;
  else
    Generate candidate virtual goals;
    Delete candidate targets;
    Calculate the evaluation function;
    Get an optimal virtual goal;
  end
end

The details of Algorithm 2 are as follows: If the goal is in the detection area of the robot and there are no obstacles between the robot and the goal, the virtual goal coincides with the real goal, or else the robot generates an optimal virtual goal to replace the real goal. First, the robot generates candidate virtual goals which distribute averagely on the boundary of the detection area of the robot as shown in Fig. 4. When $t = t_i$, according to the position and detection area of the robot, the position of the candidate virtual goals is given by

$$VG_j(t_i) = (x_r(t_i), y_r(t_i)) + r_r \left(\cos\left(j\frac{2\pi}{m}\right), \sin\left(j\frac{2\pi}{m}\right) \right) \tag{9}$$

where j and m are the index and number of the candidate virtual goals, respectively.

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

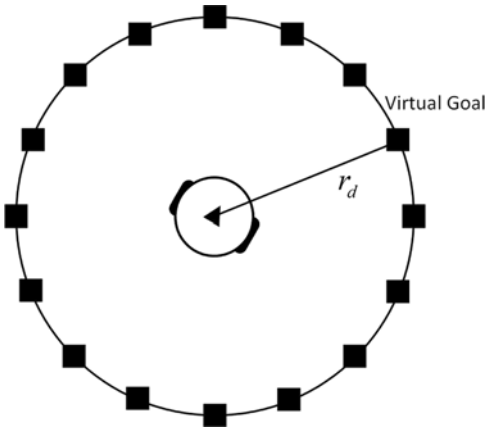


Fig. 4. The position of the candidate virtual goals.

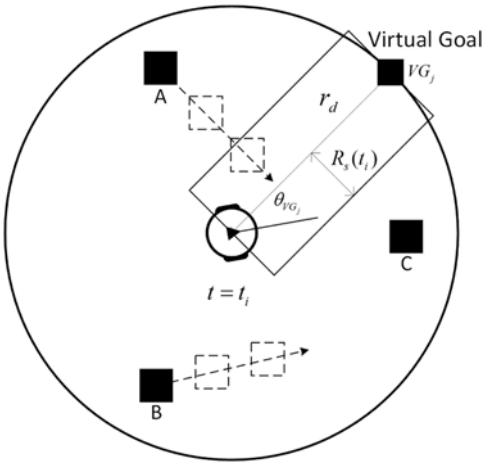


Fig. 5. The selection of the candidate virtual goals.

Then a part of candidate virtual goals should be deleted according to the safe area between the robot and it. The safe area is defined as a rectangular area, as shown in Fig. 5. The width and length of the safe area is $2R_s(t_i)$ and r_d, where $R_s(t_i)$ is the dynamic safe distance of the robot and is defined as

$$R_s(t_i) = r_r + r_s(t_i) \tag{10}$$

Here, r_r is the radius of the robot, and $r_s(t_i)$ is the braking distance with respect to the velocity of the robot

$$r_s(t_i) = \frac{\upsilon_r(t_i)^2}{2\dot{\upsilon}_{\max}} \quad (i \geq 2) \tag{11}$$

If there exist predicted obstacles in the safe area between the robot and the virtual goal, this virtual goal will be deleted from $VG_j(t_i)$.

Finally, an optimal virtual goal is selected by minimizing the evaluation function:

$$G_1(VG_j(t_i)) = \alpha_1 head_1(VG_j(t_i)) + \beta_1 dist_1(VG_j(t_i)) \tag{12}$$

where $head_1(VG_j(t_i))$ is given by $\theta_{VG_j}(t_i)$, which represents the angle between the direction of robot and the direction of robot toward to virtual goal. $dist_1(VG_j(t_i))$ denotes the distance between the robot and the virtual goal. $head_1(VG_j(t_i))$ and $dist_1(VG_j(t_i))$ are normalized to (0, 1):

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

$$\frac{dist_1(VG_j(t_i))}{\sum_{i=1}^{n} dist_1(VG_j(t_i))}, \frac{head_1(VG_j(t_i))}{\sum_{i=1}^{n} head_1(VG_j(t_i))} \tag{13}$$

3.3. Dynamic window obstacle avoidance with timestamps

In order to achieve accurate navigation where the kinematics parameters of the robot are fully considered, DW approach with timestamps is presented to avoid obstacles. Firstly, a DW is generated for the robot according to the kinematics parameters of the robot. Subsequently, a part of the DW that causes collision will be deleted according to the motion of the obstacles with timestamps. Finally, the optimal linear velocity and angular velocity from the DW are selected to drive the robot.

The kinematics parameters of the robot are denoted by $\upsilon_{\max}$, $\omega_{\max}$, $\dot{\upsilon}_{\max}$, $\dot{\omega}_{\max}$, acc_υ and acc_ω, where $\upsilon_{\max}$ and $\omega_{\max}$ are the maximum linear velocity and angular velocity, respectively, $\dot{\upsilon}_{\max}$ and $\dot{\omega}_{\max}$ are the maximum linear acceleration and angular acceleration, respectively, acc_υ and acc_ω are the resolution of the linear velocity and angular velocity, respectively. Therefore, when $t = t_i$, the DW of the robot which contains all the reachable linear velocity and angular velocity within the next sampling interval Δt are denoted by

$$V_D(i+1) = \{(\upsilon_r(i+1), \omega_r(i+1))\} \tag{14}$$

where

$$\upsilon_r(i+1)\epsilon[\max(\upsilon_{r_\min}, 0), \min(\upsilon_{r_\max}, \upsilon_{\max})]$$
$$\omega_r(i+1)\epsilon[\max(\omega_{r_\min}, -\omega_{\max}), \min(\omega_{r_\max}, \omega_{\max})]$$

Here

$$\upsilon_{r_\min} = \upsilon_r(t_i) - \dot{\upsilon}_{\max}\Delta t \qquad \upsilon_{r_\max} = \upsilon_r(t_i) + \dot{\upsilon}_{\max}\Delta t$$
$$\omega_{r_\min} = \omega_r(t_i) - \dot{\omega}_{\max}\Delta t \qquad \omega_{r_\max} = \omega_r(t_i) + \dot{\omega}_{\max}\Delta t$$

The intervals of the υ_r and ω_r are acc_υ and acc_ω, respectively. The motion of the robot is considered as an arc. The radius of the arc $r(t_i)$ is determined by the linear velocity and angular velocity of the robot:

$$r(t_i) = \frac{\upsilon_r(t_{i+1})}{\omega_r(t_{i+1})} \tag{15}$$

If $\omega_r(t_i) \neq 0$, the position of the robot is updated to

$$\begin{aligned} x_r(t_{i+k}) &= x_r(t_i) - r(t_i)\sin(\theta(t_i)) \\ &\quad - r(t_i)\sin(\theta(t_i) + k\omega_r(t_i)\Delta t) \\ y_r(t_{i+k}) &= y_r(t_i) - r(t_i)\cos(\theta(t_i)) \\ &\quad + r(t_i)\cos(\theta(t_i) + k\omega_r(t_i)\Delta t) \\ \theta(t_{i+k}) &= \theta(t_i) + k\omega(t_i)\Delta t \end{aligned} \tag{16}$$

Otherwise, the position of the robot is updated to

$$\begin{aligned} x_r(t_{i+k}) &= x_r(t_i) + k\upsilon_r(t_i)\cos(\theta(t_i))\Delta t \\ y_r(t_{i+k}) &= y_r(t_i) + k\upsilon_r(t_i)\sin(\theta(t_i))\Delta t \\ \theta(t_{i+k}) &= \theta(t_{i+k}) \end{aligned} \tag{17}$$

According to Eqs. (14)–(17), the robot gets its trajectory during the predicted time t_p. In order to delete the velocity command that causes the collision between the robot and the obstacles, the concept of time tags is proposed. When $t = t_{i+k}$, the positions of the robot and the obstacles are marked by timestamps t_{i+k}. The minimal distance between the robot and the obstacles marked by the same timestamps is denoted by $\rho_{\min}(t_{i+k})$, which is shown in Fig. 6. In Fig. 6, the area of the circles around the robot is its dynamic safe area. The lines without arrow represent the predicted trajectory of the robot according to the DW. The dotted lines without arrow represent the minimal distance between the robot and the obstacles which have the same timestamps. Once Eq. (18) cannot

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

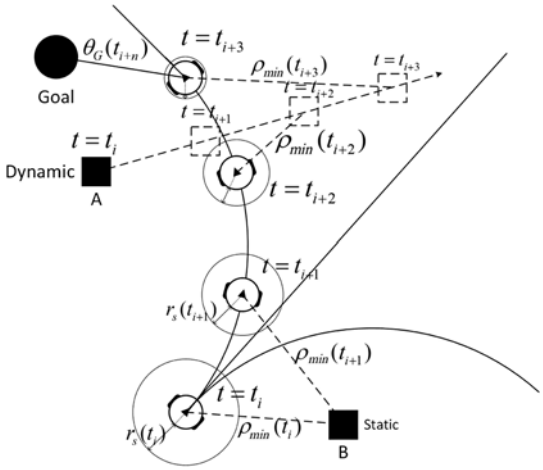


Fig. 6. Dynamic window obstacle avoidance with timestamps.

be satisfied, the velocity needs to be deleted, because this velocity will drive the robot to collide with obstacles when $t = t_{i+k}$:

$$\rho_{\min}(t_{i+k}) \geq R_s(t_{i+k}) \tag{18}$$

By deleting the velocity command from $V_D(i+1)$, the robot can get a feasible DW $V'_D(i+1)$. When the $V'_D(i+1)$ is not empty, the evaluation function $G_2(\upsilon, \omega)$ is defined as

$$G_2(\upsilon, \omega) = \alpha_2 head_2(\upsilon, \omega) + \beta_2 dist_2(\upsilon, \omega) + \gamma vel(\upsilon, \omega) \tag{19}$$

where $head_2(\upsilon, \omega)$ is given by $180 - \theta_G(t_{i+n})$. $\theta_G(t_{i+n})$ is denoted by the angle between the direction of robot and the line which denotes by the robot and goal when $t = t_{i+n}$ as shown in Fig. 6. $dist_2(\upsilon, \omega)$ represents the minimum distance $\rho_{\min}(t_{i+k})$. $vel(\upsilon, \omega)$ is denoted by the linear velocity which is selected from $V'_D(i+1)$. $head_2(\upsilon, \omega)$, $dist_2(\upsilon, \omega)$ and $vel(\upsilon, \omega)$ are normalized to (0, 1). Calculating each velocity command from $V'_D(i+1)$ according to $G_2(\upsilon, \omega)$, the optimal linear velocity and angular velocity will be derived by minimizing the evaluation function as a moving command for the robot to reach its goal. On the other hand, when the $V'_D(i+1)$ is empty, which means the robot has no way to go, the robot stops and waits for next detecting loop.

After the robot updates its position, the distance d between the goal and the robot is obtained

$$d = \sqrt{(x_r(t_{i+1}) - x_g)^2 + (y_r(t_{i+1}) - y_g)^2} \tag{20}$$

If $d \leq d_{\text{reach}}$, the robot reaches its goal. Otherwise, the robot continues to move until $d \leq d_{\text{reach}}$. d_{reach} is used to judge whether the robot reaches its goal.

4. Simulation Results

Several simulation experiments are carried out to verify the performance of the DW-VG method. Before presenting these experiments, we first introduce the simulation settings. Subsequently, the performance measures are described to assess the behavior of the testing methods. Finally, the discussion and comparison with the DW-VG method are presented. All experiment data are presented in Table II.

The simulation settings are shown in Fig. 7. The maximal linear velocity and angular velocity of the robot are 1 m/s and 50°/s, respectively. Its maximal linear acceleration and angular acceleration are 1 m/s^2 and 50°/s^2, respectively. Its minimal accuracy of the linear velocity and angular velocity are 0.1 m/s and 2°/s, respectively.

In this section, we consider six different cases in static and dynamic environments where the location, shape, and velocity of the obstacles are unknown for the robot in advance. For the sake of comparison, these experiments are carried out using the DW-VG, DW, BUG, PF, and VFH methods. Note that we try several parameters settings including *the safe distance* (the minima safe distance

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

Table II. Performance evaluation of the DW, DW-VG, BUG, PF, and VFH methods for simulation Experiments 1–8.

Case	Method	TRG	PL	SD	RO (%)
1	DW-VG	8	7.47	0.3	82.8
	DW	7.5	7.30	0.3	83.3
	BUG	14	7.56	0.5	75.3
	PF	13.7	7.96	0.4	77.0
	VFH	10.4	7.36	0.3	81.4
2	DW-VG	9.3	6.51	0.3	72.5
	DW	19.4	10.28	0.3	58.3
	BUG	0	0	0	Fail
	PF	0	0	0	Fail
	VFH	9.8	6.22	0.4	69.1
3	DW-VG	6.2	5.73	0.3	83.7
	DW	5.9	5.67	0.3	84.0
	BUG	11.2	6.2	0.5	75.8
	PF	13.6	7.80	0.4	74.1
	VFH	7.9	5.94	0.3	82.2
4	DW-VG	6.5	6.08	0.3	72.3
	DW	0	0	0	Fail
	BUG	0	0	0	Fail
	PF	9.8	6.18	0.4	65.1
	VFH	8.9	6.02	0.55	62.5
5	DW-VG	7.5	6.74	0.3	76.9
	DW	0	0	0	Fail
	BUG	0	0	0	Fail
	PF	13.8	8.07	1.2	54.3
	VFH	10.7	6.40	0.65	68.7
6	DW-VG	26.5	15.97	0.3	100
	DW	0	0	0	Fail
	BUG	0	0	0	Fail
	PF	0	0	0	Fail
	VFH	0	0	0	Fail

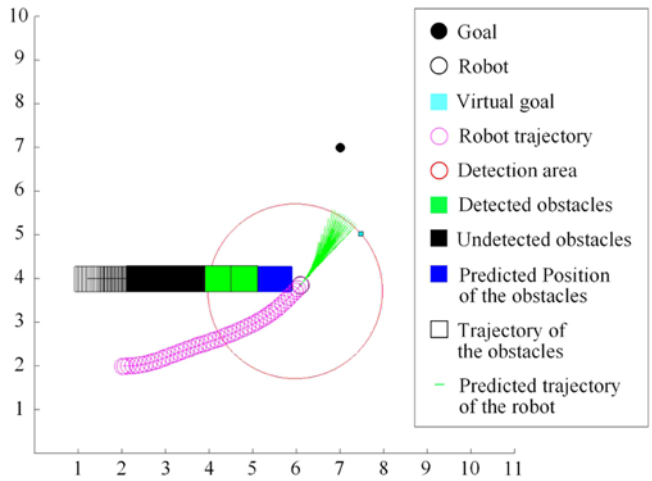


Fig. 7. The simulation settings. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of the robot is its radius. The radius of the robot is 0.3 in this paper. We control the robot by sending speed commands at a high-frequency rate. However, the DW, BUG, PF, and VFH methods neglect the kinematic parameters of the robot, which means its speed cannot immediately adjust to its speed command, even the speed command is not reachable according to its kinematics parameters.

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

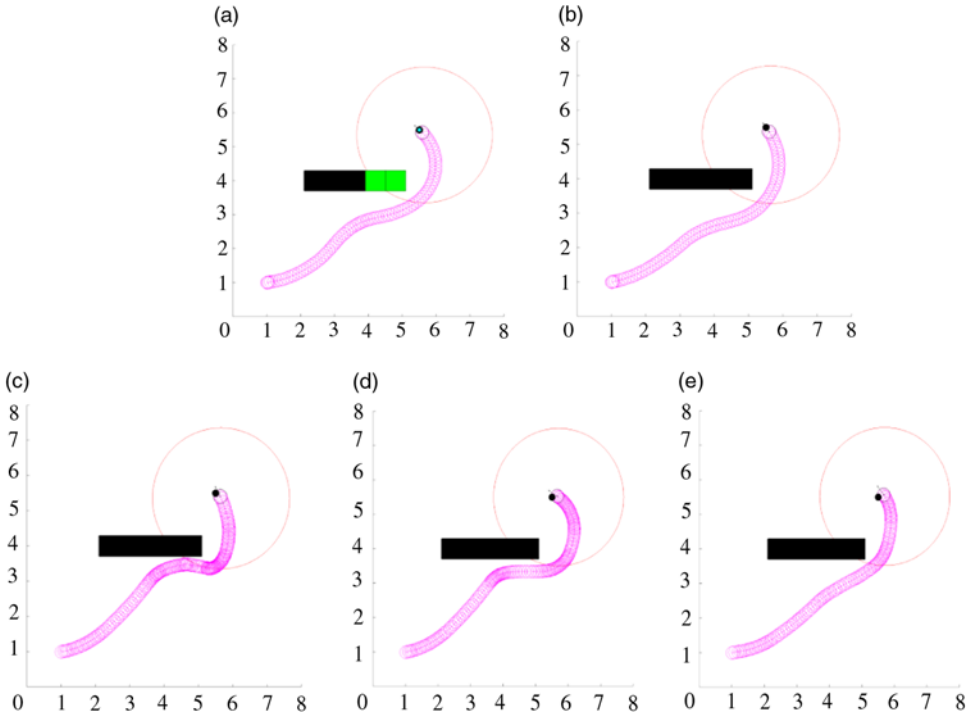


Fig. 8. The trajectories of the robot for Case 1. (a) DW-VG, (b) DW, (c) BUG, (d) PF, and (e) VFH.

Therefore, the actual trajectory of the robot deviates from the desired trajectory, the collision happen. Usually, in order to ensure the safety of the robot, its maximal speed and safe distance need to decrease and increase, respectively.) before selecting the most suitable values for each simulation experiment. All simulation experiments are implemented on MATLAB R2015b.

Case 1: There exists a rectangular obstacle on the way of the robot, the initial state of the robot is $r_{t_0} = (1, 1, 0)$. The position of the goal is (5.5, 5.5). Using the BUG and PF methods, the robot navigates close to the obstacles (see Fig. 8(c) and (d)). In addition, the robot runs faster and smoother using the DW-VG and DW methods than the VFH method (see Fig. 8(a), (b), (e) and Table II).

Case 2: The robot detects a concave trap. The initial state of the robot is $r_{t_0} = (1, 1, \pi/5)$. The position of the goal is (4.5, 4.5). For the DW method, the robot escapes the concave trap by struggling for a long time (see Fig. 9(b)). Using the BUG and PF methods, the robot collides with the obstacles (see Fig. 9(c) and (d)). By increasing the safe distance of the VFH method to 0.4, the robot reaches the goal safely (see Fig. 9(e)). The DW-VG method also drives the robot to reach the goal and faster than the VFH method (see Fig. 9(a)).

Case 3: In the experiment, the robot meets the dynamic obstacles. The initial state of the robot is $r_{t_0} = (2, 1.8, 0)$. The obstacles is toward $+Y$, its speed is 0.2 m/s as shown in Fig. 10. By increasing the safe distance of the BUG and PF methods to 0.5 and 0.4, respectively, the robot avoids the dynamic obstacles safely (see Fig. 10(c) and (d)). The experiment result shows that the DW-VG, DW, and VFH methods have similarly performance when they deal with this situation (see Fig. 10(a), (b), (e) and Table II). This experiment result also demonstrates that the DW-VG, DW, BUG, PF, and VFH methods have the ability to avoid dynamic obstacles in some cases.

Case 4: We continue to test the dynamic obstacle avoidance performance of the DW-VG, DW, BUG, PF, and VFH methods. Different from Case 3, the dynamic obstacles have the same movement direction as the robot. The initial state of the robot is $r_{t_0} = (2, 1.8, 0)$. The obstacle is toward $+X$ and its speed is 0.25 m/s. When the robot uses the DW and BUG methods, it is stuck on the right side of the obstacle, and never reaches its goal (see Fig. 11(b) and (c)). The reason is that, for the robot, there always exists a obstacle on its left side. By increasing the safe distance of the PF and VFH methods to 0.4 and 0.55, respectively, the robot arrives the goal safely (see Fig. 11(d) and (e)). The DW-VG method drives the robot smoother and faster than the PF and VFH method (see Fig. 11(a) and Table II).

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

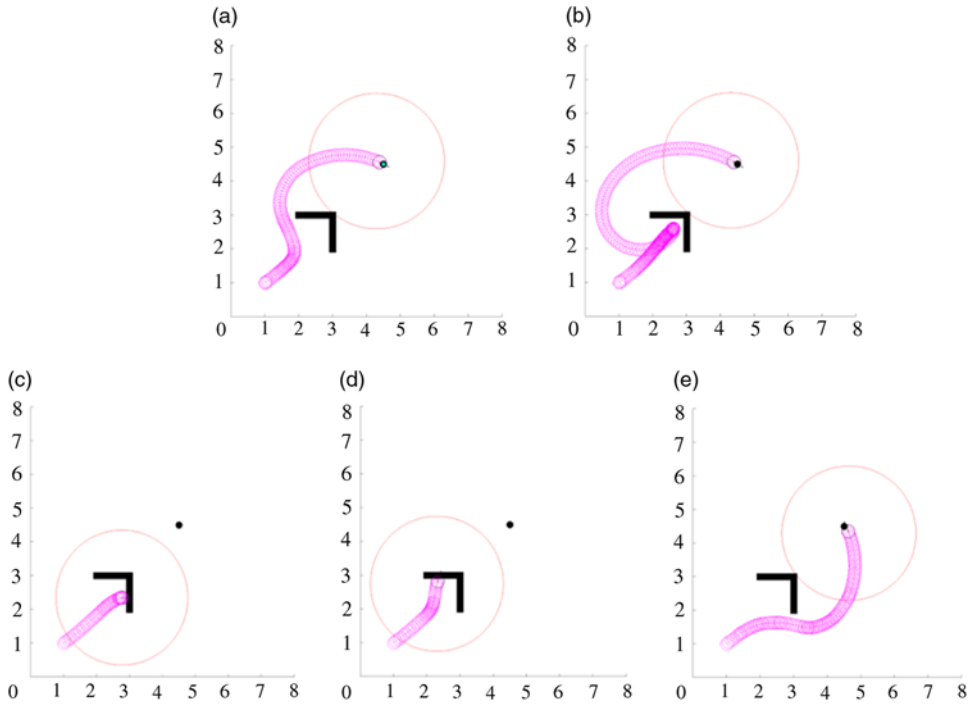


Fig. 9. The trajectories of the robot for Case 2. (a) DW-VG, (b) DW, (c) BUG, (d) PF, and (e) VFH.

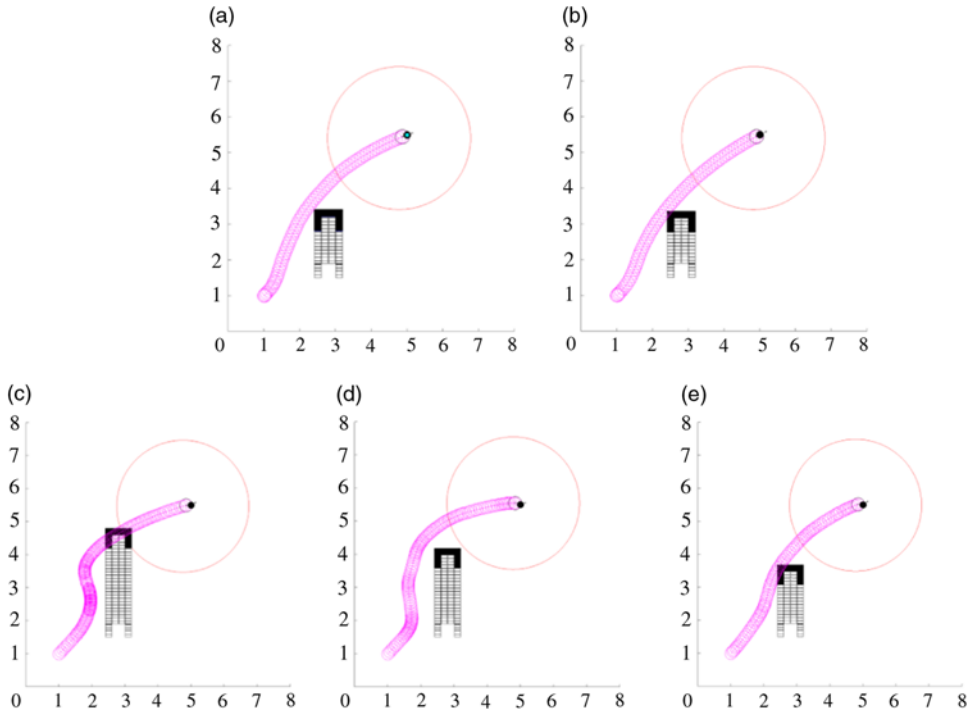


Fig. 10. The trajectories of the robot for Case 3. (a) DW-VG, (b) DW, (c) BUG, (d) PF, and (e) VFH.

Case 5: The moving direction of the dynamic obstacle is the same as the initial moving direction of the robot. The initial state of the robot is $r_{t_0} = (3.5, 1.3, \pi/3)$. The velocity of the obstacle is 0.2 m/s. The results of Case 5 are similar to those of Case 4, but the safe distance of the PF and VPF methods is increasing to 1.2 and 0.65 in Case 5, which are larger than 0.4 and 0.55, respectively, in Case 4. However, the safe distance of the DW-VG method is 0.3 in all cases, which means a shorter

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

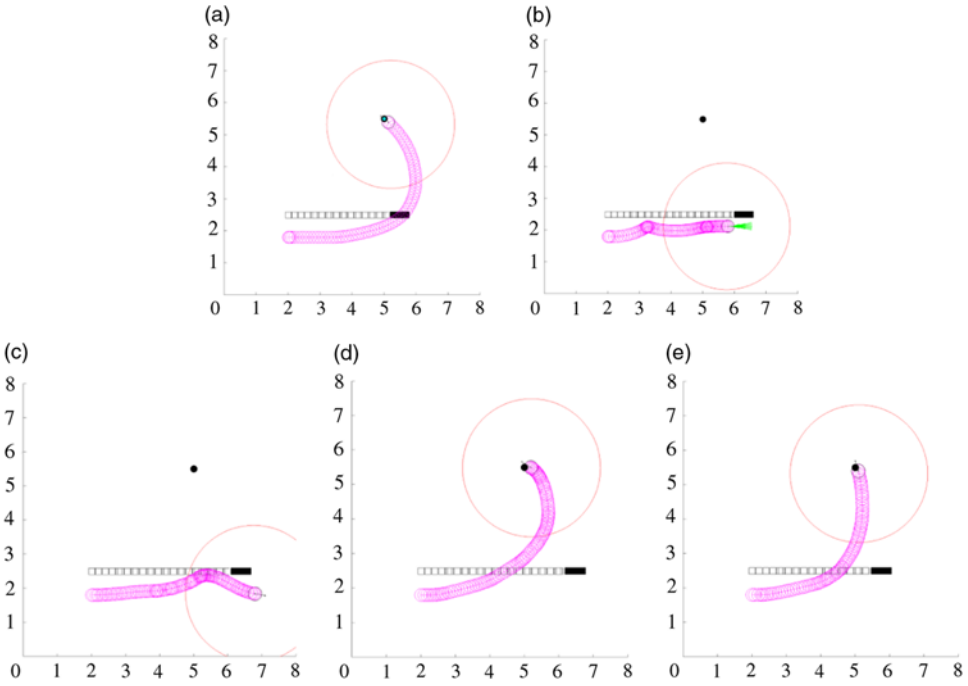


Fig. 11. The trajectories of the robot for Case 4. (a) DW-VG, (b) DW, (c) BUG, (d) PF, and (e) VFH.

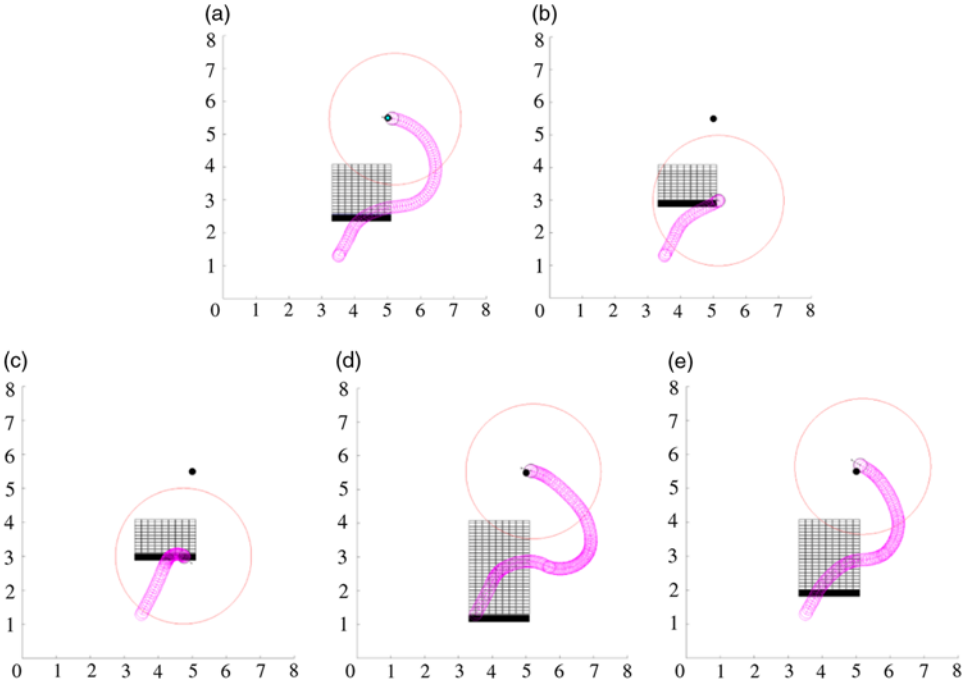


Fig. 12. The trajectories of the robot for Case 5. (a) DW-VG, (b) DW, (c) BUG, (d) PF, and (e) VFH.

path length and less running time (see Fig. 12). Moreover, a shorter safe distance has the benefit for the robot to pass a narrow passage as shown in Case 6.

Case 6: The robot needs to work in the dense environment which contains a narrow passage, static obstacles, and dynamic obstacles as shown in Fig. 13. The initial state of the robot is $r_{t_0} = (0.5, 0.5, \pi/4)$. The position of the goal is (8, 8). The dynamic obstacles C and E moved periodically at 0.2 m/s. The robot needs to pass the narrow passage A, to escape the concave trap D, to avoid the static obstacle B and dynamic obstacles C and E. In order to pass the narrow passage A, the robot

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

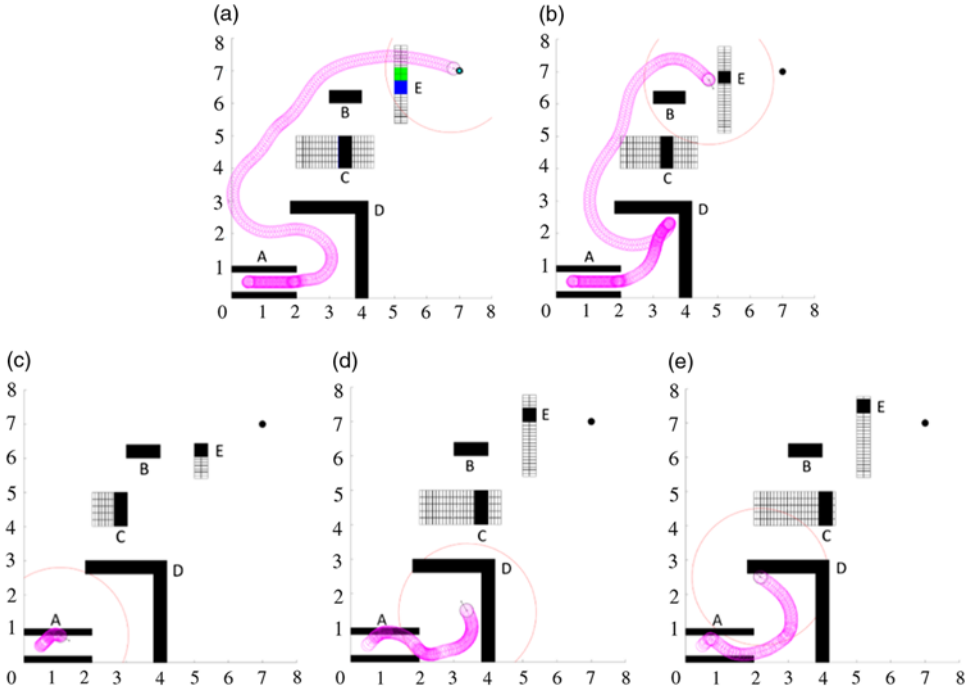


Fig. 13. The trajectories of the robot for Case 6. (a) DW-VG, (b) DW, (c) BUG, (d) PF, and (e)VFH.

needs to decrease its safe distance. On the other hand, with the decrease of the safe distance, the probability of the robot collides with obstacles increases as explained in Cases 2–6. Therefore, the robot collided with the obstacles using the BUG, PF, and VFH methods (see Fig. 13(c)–(e)). The DW method fails, because the DW is empty when the dynamic obstacle E appears in front of the robot (see Fig. 13(b)). Only the DW-VG method drives the robot to its goal safely as shown in Fig. 13(a).

In order to evaluate the effectiveness of the proposed DW-VG method, and to compare the performance of the DW-VG with DW, BUG, PF, and VFH methods, the following metrics are considered:

Time to reach the goal (TRG): The total time that the robot completes the mission. For high performance, it is preferable to have a low TRG.

Path length (PL): The total distance traveling from the starting to the goal location. Shorter path length means the better performance. If the robot reaches the goal at time $t = t_n$, the path length (PL) is defined as

$$PL = \sum_{i=0}^{n} \sqrt{(x_r(t_{i+1}) - x_r(t_i))^2 + (y_r(t_{i+1}) - y_r(t_i))^2} \tag{21}$$

Safe distance of the robot (SD): The SD is defined as

$$SD = r_r + x \tag{22}$$

where the $r_r = 0.3$ is the radius of the robot. x is the extra safe distance that ensures the safety when the robot works in the uncertain environment. Shorter SD means better performance.

Relative optimum (RO): The relative optimal normalizes the TRG, PL, and SD to evaluate the effectiveness of the algorithms. Higher RO means better performance. If the method M drives the robot to reach its goal successfully, the RO^M can be defined as

$$RO^M = 1 - \left(RO^M_{TRG} + RO^M_{PL} + RO^M_{SD}\right)/3 \tag{23}$$

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

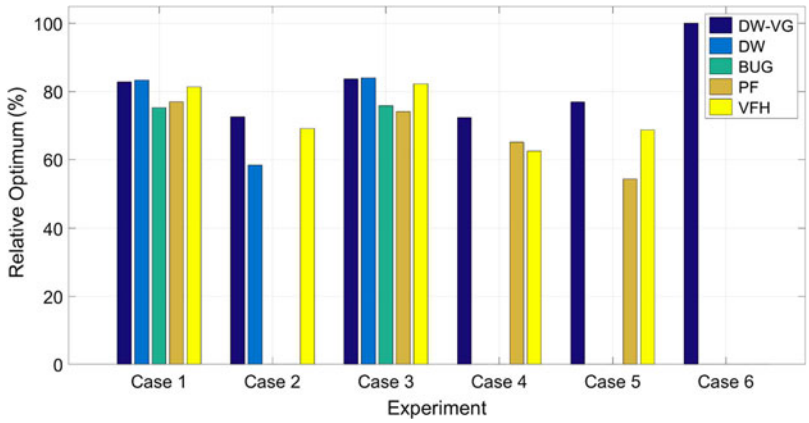


Fig. 14. The Relative Optimal (RO) of the DW-VG, DW, BUG, PF, and VFH methods. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where

$$RO^{M}_{TRG} = \frac{1}{TRG_M} \sum_{m=1}^{5} TRG_m$$

$$RO^{M}_{PL} = \frac{1}{PL_M} \sum_{m=1}^{5} PL_m$$

$$RO^{M}_{SD} = \frac{1}{SD_M} \sum_{m=1}^{5} SD_m$$

$m = 1, 2, 3, 4$, and 5 represent the DW, DW-VG, BUG, PF, and VFH, respectively. The RO^{M}_{TRG}, RO^{M}_{PL}, and RO^{M}_{SD} denote the TRG, PL, and SD of the method M (M represents the DW, DW-VG, BUG, PF, and VFH).

Table II shows the obtained results, where it can be seen that the proposed DW-VG method can drive the robot to work well in the experiments environment. More importantly, Fig. 14 shows better performance of the DW-VG method in comparison with the other methods.

The common drawback of the DW, BUG, PF, and VFH methods can be observed in the simulation experiments. The DW method inclines to get trapped in a local minimum when the robot meets concave obstacles (see Fig. 9(b)), and fail to avoid the dynamic obstacles (see Figs. 11(b), 12(b) and 13(b)). The BUG, PF, and VFH methods neglect the kinematics of the robot when they send the velocity command to the robot. Accordingly, in order to ensure the safety of the robot, its safe distance needs to increase, which means the longer path length and running time (See Cases 2–5). Moreover, with increase of the safe distance, the ability of the robot to pass through the dense obstacles decreases as explained in Case 6. However, according to Table II and Fig. 14, the DW-VG method does not need to increase its minimal safe distance, and drives the robot to reach its goal with shorter path length and less running time in comparison with the DW, BUG, PF, and VFH methods.

5. Experimental Verification

In this section, several experiments were carried out to verify the reliability of the proposed DW-VG method. The experimental environment is shown in Fig. 15. The details of the experimental setup are as follows: The EAI mobile car in Fig. 15-A was chosen as the real robot, which is equipped with an on-board computer having a 1.2 GHz 64-bit quad-core ARM Cortex-A53 CPU. The maximum linear velocity of EAI is 1 m/s and its maximal angular velocity is 2 rad/s. Figure 15-B is a SLAMTEC mobile car. Its maximal linear velocity is 0.5 m/s and the maximal angular velocity is 1 rad/s. In order to control the motion of the dynamic obstacles, the box is putted over the SLAMTEC mobile car as shown in Fig. 15-C. The boxes D and E in Fig. 15 and represent the static obstacles. A gray square carpet represents the goal location. The position of the obstacles and the robot are obtained by the Vicon–Vantage motion capture systems in the frequency of 50 Hz. The algorithm was programmed in

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

Table III. The experimental parameters of the DW-VG method.

Description	**Value**
Threshold for state changes: s	0.05 m
Length of prediction of obstacles: P	25
Determining state changes: i_{state}	10
α_1 of G_1	3
β_1 of G_1	1
α_2 of G_2	0.5
β_2 of G_2	2
γ_2 of G_2	1
Maximal linear velocity of robot	1 m/s
Maximal angular velocity of robot	2 rad/s
Maximal linear acceleration of robot	1 m/s^2
Maximal angular acceleration of robot	1 rad/s^2
Accuracy of linear velocity	0.1 m/s
Accuracy of angular velocity	0.2 rad/s
Radius of robot	0.21 m
Radius of obstacles	0.16 m



Fig. 15. The experimental environment.

C++ language under the Linux ubuntu robot-operating system (ROS). The coordinate origin, X-axis and Y-axis are shown in Fig. 15. The experimental parameters of the DW-VG method in experiments are listed in Table III.

The details of the experimental process are as follows. Before the experiments are executed, the mark balls are putted on every obstacle and the EAI mobile car, which can help us to get the obstacles and EAI position based on Vicon–Vantage system. At each control cycle of EAI, firstly, update the positions of the obstacles and EAI. According to the proposed PCF+DF method, the obstacles position in the future moments are obtained. Based on the motion prediction of the obstacles, the virtual goal is generated. Then, the optimal linear velocity and angular velocity are deduced based on DW according to the motion prediction of the obstacles with timestamps, position of the virtual goal and the kinematic parameters of the EAI. Finally, under the ROS system, publish the optimal

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

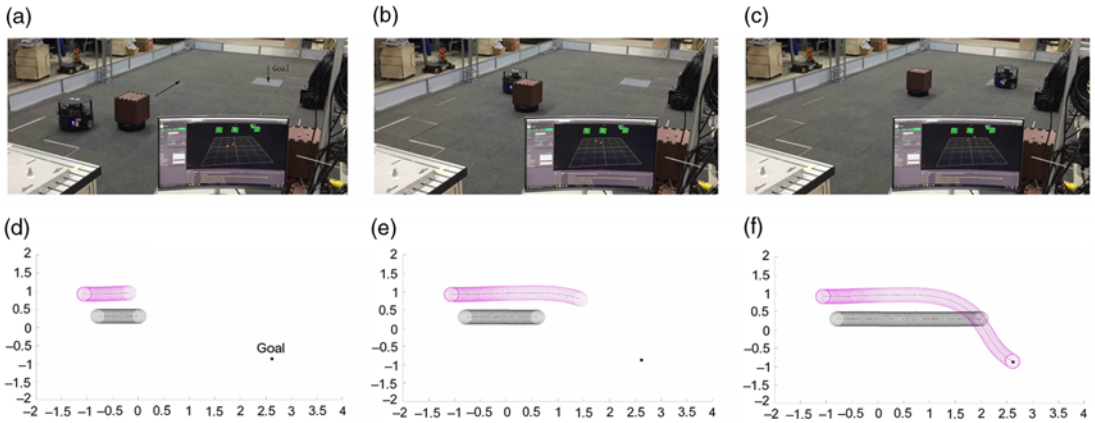


Fig. 16. Experiment 1. (d)–(f) show the trajectory of the robot and the obstacles correspond to (a)–(c), respectively.

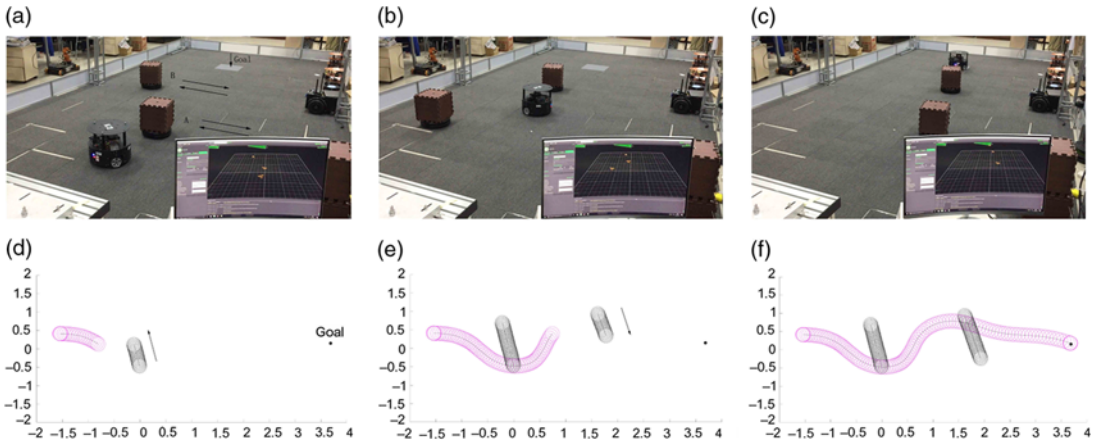


Fig. 17. Experiment 2. (d)–(f) show the trajectory of the robot and the obstacle correspond to (a)–(c), respectively.

linear velocity and angular velocity to the *cmd_vel* topic. By subscribing the *cmd_vel* topic, the EAI can work in the dynamic environments.

The motion prediction of obstacles and experiments are fully reported in the video which can be obtained from https://youtu.be/32ZYIHKiiL0. The video consists of four parts. The part one shows the motion prediction of the obstacles based on PCF+DF method. The black solid blocks and the black lines represent the obstacles and the motion prediction of the obstacles in next 1 s, respectively. Part 2, Part 3, and Part 4 are corresponding to Experiment 1, Experiment 2, and Experiment 3. The purpose of these experiments is to verify the performance of the proposed DW-VG method when avoid the dynamic obstacles and pass through the dense obstacles.

Experiment 1: The goal was on the right of the robot, there exists a dynamic obstacle between the robot and its goal as shown in Fig. 16(a) and (d). The initial linear velocity of the dynamic obstacle is the same as the initial linear velocity of the robot at 0.2 m/s. By analyzing the position of the dynamic obstacle in previous moments, the robot predicted the motion of the dynamic obstacle in future moments. Therefore, the robot increased its speed to pass the dynamic obstacle as shown in Fig. 16(b) and (e). Finally, the robot arrived its goal.

Experiment 2: There were two dynamic obstacles moving back and forth at 0.2 m/s. Firstly, the robot needed to avoid the dynamic obstacle A which was moving at 0.2 m/s as shown in Fig. 17(a) and (d). By setting the virtual goal and predicting the motion of the dynamic obstacle A, the robot avoided the dynamic obstacle A quickly as shown in Fig. 17(b) and (e). Similarly, the robot avoided the dynamic obstacle B and arrived its goal as shown in Fig. 17(c) and (f).

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

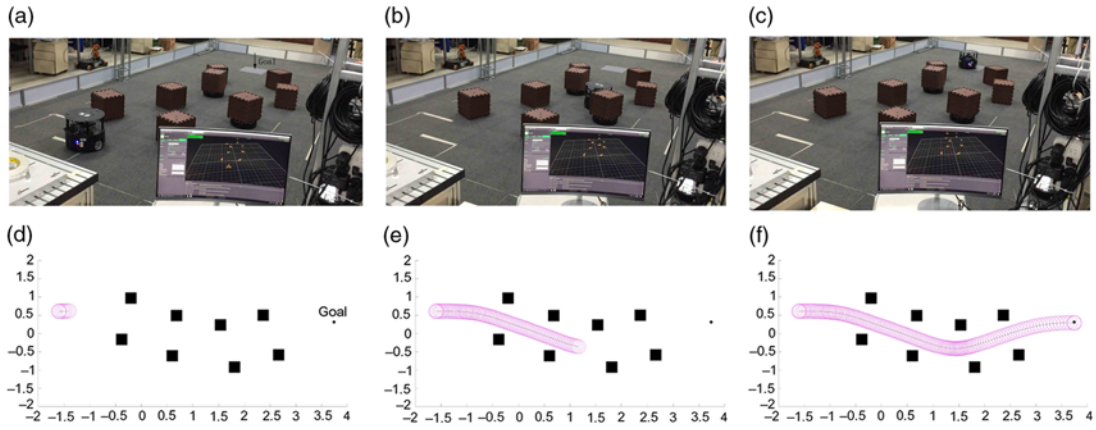


Fig. 18. Experiment 3. (d)–(f) show the trajectory of the robot and the position of the obstacles correspond to (a)–(c), respectively.

Experiment 3: In this experiment, eight static obstacles were putted between the robot and its goal as shown in Fig. 18(a) and (d). The robot needed to pass a narrow passageway without collision. Firstly, the robot entered the narrow passageway smoothly according to its DW as shown in Fig. 18(b) and (e). Subsequently, the robot changed its direction smoothly and decreased its speed according its DW as shown in Fig. 18(c) and (f). Finally, the robot arrived its goal safely.

6. Conclusions and Future Work

The DW-VG reactive obstacle avoidance method was presented for the dynamic environment. By predicting the motion of obstacles based on PCF+DF method with timestamps and setting the virtual goal, the DW-VG method drove the robot fast and smoothly with collision-free motion. The performance evaluation index values indicate that the performance of the DW-VG method is better than DW, BUG, PF, and VFH methods. Furthermore, the experimental results demonstrated that the proposed method is feasible when the motion of the obstacles were unknown previously.

As a reactive obstacle avoidance approach, the DW-VG method does not use the global information about entire environment. However, the purposed method can combine with the global path planning method by adjusting the parameters of its evaluation functions.

Acknowledgments

This paper was supported by National Key Research and Development Plan "Intelligent Robot" Key Project (2017YFB1303900), Robotics Enterprise Cooperation Project(KYY-HX-20180162), and the National Science Foundation of China (61473183; U1509211).

References

1. M. Mujahed, D. Fischer and B. Mertsching, "Tangential Gap Flow (TGF) navigation: A new reactive obstacle avoidance approach for highly cluttered environments," *Rob. Auton. Syst.* **84**, 15–30 (2016).
2. S. Hirose and E. F. Fukushima, "Snakes and strings: New robotic components for rescue operations," *Int. J. Rob. Res.* **23**, 341–349 (2004).
3. V. Lumelsky and A. Stepanov, "Dynamic path planning for a mobile automaton with limited information on the environment," *IEEE Trans. Autom. Control* **31**, 1058–1063 (1986).
4. I. Kamon and E. Rivlin, "Sensory-based motion planning with global proofs," *IEEE Trans. Rob. Autom.* **13**, 814–822 (1997).
5. I. Kamon, E. Rimon and E. Rivlin, "TangentBug: A range-sensor-based navigation algorithm," *Int. J. Rob. Res.* **17**, 934–953 (1998).
6. J. Ng and T. Bräunl, "Performance comparison of bug navigation algorithms," *J. Intell. Rob. Syst.* **50**, 73–84 (2007).
7. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Rob. Res.* **5**, 90–98 (1986).
8. J. Minguez, F. Lamiraux and J.-P. Laumond, "Motion Planning and Obstacle Avoidance," **In**: *Springer Handbook of Robotics* (Springer, Switzerland, 2016) pp. 1177–1202.
9. Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," (IEEE, Sacramento, CA, USA, 1991) pp. 1398–1404.

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press

10. J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. Rob. Autom.* **7**, 278–288 (1991).
11. A. Babinec, F. Duchoň, M. Dekan, P. Pásztó and M. Kelemen, "VFH* TDT (VFH* with Time Dependent Tree): A new laser rangefinder based obstacle avoidance method designed for environment with non-static obstacles," *Rob. Autonom. Syst.* **62**, 1098–1115 (2014).
12. I. Ulrich and J. Borenstein, "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots," (IEEE, Leuven, Belgium, 1998) pp. 1572–1577.
13. J. Ren, K. A. McIsaac and R. V. Patel, "Modified Newton's method applied to potential field-based navigation for mobile robots," *IEEE Trans. Rob.* **22**, 384–391 (2006).
14. A. Ferreira, F. G. Pereira, R. F. Vassallo, T. F. Bastos Filho and M. Sarcinelli Filho, "An approach to avoid obstacles in mobile robot navigation: The tangential escape," *Sba: Controle & Automação Sociedade Brasileira de Automatica* **19**, 395–405 (2008).
15. H.-T. Chiang, N. Malone, K. Lesser, M. Oishi and L. Tapia, "Path-Guided Artificial Potential Fields with Stochastic Reachable Sets for Motion Planning in Highly Dynamic Environments," (IEEE, Seattle, WA, USA, 2015) pp. 2347–2354.
16. D. Fox, W. Burgard and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Rob. Autom. Mag.* **4**, 23–33 (1997).
17. A. Maroti, D. Szalóki, D. Kiss and G. Tevesz, "Investigation of Dynamic Window Based Navigation Algorithms on a Real Robot," (IEEE, 2013), pp. 95–100.
18. P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Rob. Res.* **17**, 760–772 (1998).
19. F. Large, C. Laugier and Z. Shiller, "Navigation among moving obstacles using the NLVO: Principles and applications to intelligent vehicles," *Autonom. Rob.* **19**, 159–171 (2005).
20. M. A. H. Ali, M. Mailah and T. H. Hing, "A novel approach for visibility search graph based path planning," 13th International Conference on Robotics, Control and Manufacturing Systems (2013) pp. 44–49.
21. M. Faisal, K. Al-Mutib, R. Hedjar, H. Mathkour, M. Alsulaiman and E. Mattar, "Multi modules fuzzy logic for mobile robots navigation and obstacle avoidance in unknown indoor dynamic environment," **In**: Proceedings of the 2013 International Conference on Systems, Control and Informatics (2013) pp. 371–379.
22. S. M. Lavalle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," TR 98-11 (Computer Science Department, Iowa State University, 1998).
23. Q. Wang, W. Wang and Y. Li, "A Multi-RRT Based Hierarchical Path Planning Method," *2012 IEEE 14th International Conference on Communication Technology (ICCT)* (IEEE, 2012).
24. J. M. Pimentel, M. S. Alvim, M. F. M. Campos and D. G. Macharet, "Information-driven rapidly-exploring random tree for efficient environment exploration," *J. Intell. Rob. Syst.* **91**(2) 1–19 (2017).
25. F. Kamil, T. S. Hong, W. Khaksar, M. Y. Moghrabiah, N. Zulkifli and S. A. Ahmad, "New robot navigation algorithm for arbitrary unknown dynamic environments based on future prediction and priority behavior," *Expert Syst. Appl.* **86**, 274–291 (2017).

https://doi.org/10.1017/S0263574719000043 Published online by Cambridge University Press