Sensor-Based Motion Planning of Wheeled Mobile Robots in Unknown Dynamic Environments

Ellips Masehian

Journal of Intelligent & Robotic Systems

J Intell Robot Syst
DOI 10.1007/s10846-013-9837-3

Sensor-Based Motion Planning of Wheeled Mobile Robots in Unknown Dynamic Environments

Ellips Masehian · Yalda Katebi

Received: 29 August 2012 / Accepted: 26 April 2013

Abstract This paper presents a new sensor-based online method for generating collision-free paths for differential-drive wheeled mobile robots pursuing a moving target amidst dynamic and static obstacles. At each iteration, the set of all collision-free directions are calculated using velocity vectors of the robot relative to each obstacle, forming the Directive Circle (DC), which is the fundamental concept of our proposed method. Then, the best feasible direction close to the optimal direction to the target is selected from the DC, which prevents the robot from being trapped in local minima. Local movements of the robot are governed by the exponential stabilizing control scheme that provides a smooth motion at each step, while considering the robot's kinematic constraints. The robot is able to catch the target at a desired orientation. Extensive simulations demonstrated the efficiency of the proposed method and its success in coping with complex and highly dynamic environments with arbitrary obstacle shapes.

Keywords Robot motion planning · Dynamic environment · Moving target · Directive Circle · Obstacle avoidance

E. Masehian (✉) · Y. Katebi
Faculty of Engineering, Tarbiat Modares University, Tehran, Iran
e-mail: masehian@modares.ac.ir

1 Introduction

Starting from mid 1970's, early robot motion planning researches were mainly dedicated to solving the fundamental 'piano movers' problem, which consists of generating a collision-free path between start and goal positions of a robot navigating in a static and completely known environment, where there could be obstacles [1]. There is strong evidence that a 'complete' planner (i.e. one that finds a path whenever one exists and reports that no path exists otherwise) will take time exponential in the number of degrees of freedom (DOF) of the robot, and that the algorithm belongs to the NP-Complete class of problems [2]. In recent years, a class of motion planning problems that has received more attention is sensor-based motion planning of mobile robots in unknown dynamic environments. For such environments, online planning algorithms are needed, which typically suffer from lack of generality in the knowledge of the space in which they are executed. It was shown that dynamic motion planning for a point in the plane, with bounded velocity and arbitrary many obstacles is intractable and NP-Hard [3].

Mobile robot motion planning in dynamic environments has been studied extensively in [4]. A number of works exist on motion planning in dynamic environments with moving obstacles [5–8]. These studies, in terms of the knowledge of the

Published online: 15 May 2013

movement of obstacles, can be classified into two categories: In the first category, movements of obstacles are completely unknown to the robot [9], and in the second category, movements of obstacles are completely known. Our studied problem in this paper belongs to the first category. However, despite the lack of robot's knowledge of the environment, we try to optimize the robot's motion and conduct it toward the target in a safe path.

1.1 Related Work

Some methods are presented for optimal motion planning where a start to goal trajectory is computed at discrete time intervals by searching a tree of feasible avoidance maneuvers [10]. In [11] a method is presented for motion planning in dynamic environments by finding a trajectory for a robot in a scene consisting of both static and dynamic moving obstacles. An open problem in their presented method is the creation of smooth trajectories.

In addition to the mentioned two categories, some works introduce the factor of uncertainty in obstacles' motion. In this realm a method was proposed in [12] to predict the motion of an obstacle and its uncertainty derived from the history of its movement. Also in [13] a probabilistic model of the uncertainty is used to select the motion that minimizes the expected time of reaching the destination.

When the future location of obstacles is unknown, there are some algorithms that try to solve the problem locally [10, 14, 15]. Also there are global planners that direct the robot towards the goal while a local planner is used for avoiding obstacles [16]. An extension of a local planner to uncertain environments is presented in [17]. Another procedure which collects information during the planning process in dynamic, cluttered and uncertain environments is developed in [5].

An efficient neural network method is presented in [18] for real-time motion planning of a mobile robot, or a multi-joint robot manipulator, with safety considerations in a non-static environment. A data-driven fuzzy approach was developed in [19] for solving the motion planning of a mobile robot in the presence of moving obstacles. The approach consists of devising a general method for the derivation of input-output data to construct a Fuzzy Logic Controller in off-line mode. The work presented in [20] proposes an on-line planner for dynamic environments that is based on the concept of Velocity Obstacles (VO). It addresses the issue of motion safety, i.e. avoiding states of inevitable collisions, by selecting a proper time horizon for the velocity obstacle.

Although there are a bunch of algorithms for motion planning in the presence of dynamic obstacles, few researches have dealt with the problem of tracking a moving target by a mobile robot in online mode. In [21] the authors consider the problem of planning the motions of a mobile robot equipped with a visual sensor for tracking an unpredictable moving target in a workspace cluttered by obstacles. Another reactive control-based method for target tracking in the presence of moving obstacles with limited sensing range was proposed for point-mass vehicles. The method is based on composite potential fields that is time-varying and planned to consider moving obstacles as well [22].

In [23] a new potential function was defined as a weighted sum of the quadratic functions of relative positions and velocities of the robot with respect to the target and obstacles. In their proposed method, velocity and position are different physical terms with different units, and the potential function formed by combining them loses the physical meaning of 'potential'. In [24] a different approach is adopted to extend the potential fields method for velocity planning of a mobile robot in tracking a moving target among moving obstacles. The potential function is kept in its traditional form, and the robot's velocity is derived stage by stage as an explicit function of the velocity of the target or obstacles and the relative positions of the target, obstacles, and the robot itself. An improved potential field method has been proposed in [8] for mobile robot motion planning in a dynamic environment where both the target and obstacles are moving. The method takes the position, velocity, acceleration, and physical size of the robot into account, and the relative factor of the distance and the acceleration is introduced in the repulsive potential function.

Several applications require continuous monitoring of a moving target by a controllable vision

system. In applications which involve automated processes that are needed to be monitored, as in an assembly workcell, parts or subassemblies might need to be verified for accuracy, or must be in correct configuration. Visual monitoring tasks are also suitable for mobile robot applications. The problem of computing robot motion strategies that maintain visibility of a moving has been introduced in [25], where the robot and its target are in a bounded, Euclidean workspace cluttered with static obstacles. A novel rendezvous-guidance method is proposed for autonomous robotic interception of moving targets in a dynamic environment with static and/or moving obstacles [26]. In [27] a fuzzy potential field approach is presented for mobile robot motion planning in dynamic environments, which has the capability of providing a collision-free path for the robot and solving the local minimum problem effectively, better than other potential field based methods.

In this paper, which is an extended and enhanced version of our previous work [28], both the moving obstacles and moving target issues are addressed. It is assumed that the target's velocity is known, but no information is available about the positions, shapes, and velocities of static or dynamic obstacles. Those are to be calculated online, through rangefinder sensors. Through a new concept called 'Directive Circle' (DC) the robot identifies the free directions that guide it toward the target and keep it away from collision with obstacles.

After presenting the assumptions and notations in Section 2, the target-following procedure is described in Section 3, and an online obstacle avoidance method for circular robots is presented in Section 4. In Section 5 the algorithm is modified for online motion planning of holonomic polygonal robots. Section 6 presents simulations, computational results and comparisons, and conclusions and future works are presented in Section 7.

2 Assumptions

In this section the assumptions on which the model is constructed is briefly reviewed. Three components are important to the model: the robot, obstacles (including the border), and the target.

2.1 Robot

The robot is assumed to be a Wheeled Mobile Robot (WMR), either circular-shaped or convex polygonal-shaped. It has two independently-actuated fixed standard rear wheels and a castor or spherical front wheel, as depicted in Fig. 1. The configuration of a circular robot at time t is shown by $q_R(t) = (x_R(t), y_R(t))$, denoting the coordinates of the robot's origin at its center. For a polygonal robot each unique configuration is specified by $q_R(t) = (x_R(t), y_R(t), \varphi_R(t))$, the first two of which indicate the coordinates of the center of robot which the robot rotates (Origin), and $\varphi_R(t)$ shows the direction of the robot measured by its angle relative to the positive x-axis. The robot's rotation about its origin is limited by the angle $\theta_{\max}$ at each step, thus avoiding abrupt motions or collisions thereby.

The robot is equipped with rangefinder sensors with angular span of 360 degrees and sensing range of r_s, through which it performs a visibility scan and detects obstacle edges. It then records the positions of obstacles' visible edges in two successive iterations (i.e. time-intervals or time-steps) to determine the velocity vector of each obstacle.

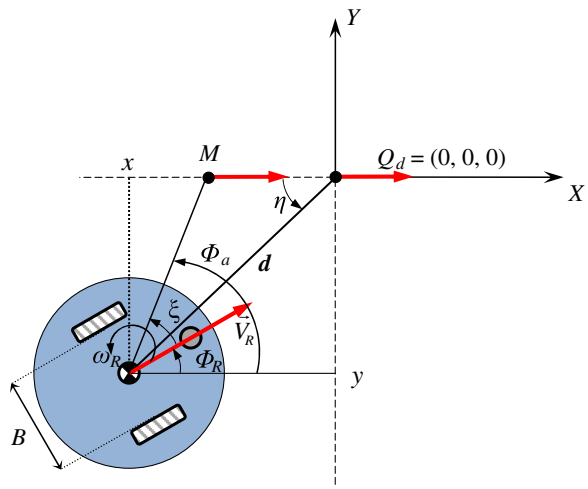

Fig. 1 The exponential stabilizing control strategy

2.1.1 Kinematic Model

The robot is assumed to have differential-drive kinematics that moves at translational velocity $\vec{V}_R$ and rotational velocity ω_R, with no slide. The nonholonomic constraint of the robot is expressed as:

$$\dot{x}_R \sin\varphi_R - \dot{y}_R \cos\varphi_R = 0, \tag{1}$$

where $q_R = (x_R, y_R, \varphi_R)$ is the configuration vector. The velocities of left and right wheels are calculated by:

$$v_L = \vec{V}_R - \frac{B\cdot\omega_R}{2}, \quad v_R = \vec{V}_R + \frac{B\cdot\omega_R}{2} \tag{2}$$

in which B is the distance of the two rear wheels.

In order to drive the robot from its current configuration to a desired configuration, we use an exponential stabilizing controller as proposed in [29]. In that work, it is assumed that the desired configuration is $Q_d = (X_d, Y_d, \Phi_d) = (0, 0, 0)$, and the robot reaches the target through an auxiliary target point M on the X-axis and on the same side of the plane as the robot (Fig. 1). The robot's heading angle Φ_R approaches the auxiliary direction Φ_a, which itself approaches the desired orientation $\Phi_d = 0$ as the robot moves toward the desired configuration. In other words, the robot approaches the auxiliary point M, while M approaches to the target Q_d.

As the robot's distance to its target (d) gradually decreases, the piecewise continuous control vector is updated as:

$$\vec{V}_R = -c_1 s_x \frac{\psi_1}{\psi_1^2+\xi^2} d,$$
$$\omega_R = c_2\cdot\xi\frac{c_1}{\psi_1^2+\xi^2}\left(\frac{2}{b}\psi_1\psi_2 + \xi d^2\right), \tag{3}$$

in which $\psi_1 = \cos(\eta - \Phi_R)$, $\psi_2 = \sin(\eta - \Phi_R)$, c_1 and c_2 are arbitrary positive constants, $\xi = \Phi_a - \Phi_R$, $d = \sqrt{x^2+y^2}$, and $\Phi_a = (2/b)\,\eta$, for $1 < b < 2$ and $-\pi/2 \le \eta \le \pi/2$. Also, $\eta = s_x \tan^{-1}\left(y/|x|\right)$, where $s_x = \text{sign}(x)$.

Since the target configuration in our problem is not generally $Q_d = (0, 0, 0)$, in order to reach an arbitrary configuration $q'_R = (x'_R, y'_R, \varphi'_R)$ from the current configuration $q_R = (x_R, y_R, \varphi_R)$ we first transform the robot's universal coordinates to the local coordinate system illustrated in Fig. 1 by:

$$\begin{aligned} x &= x_R\cos(\varphi'_R) + y_R\sin(\varphi'_R) \\ &\quad -x'_R\cos(\varphi'_R) - y'_R\sin(\varphi'_R), \\ y &= -x_R\sin(\varphi'_R) + y_R\cos(\varphi'_R) \\ &\quad +x'_R\sin(\varphi'_R) - y'_R\cos(\varphi'_R), \\ \Phi_R &= \varphi_R - \varphi'_R, \end{aligned} \tag{4}$$

and then locally plan the robot's trajectory by Eqs. 2 and 3, and transform back the local coordinates to the universal coordinates by:

$$\begin{aligned} x_R &= x\cos(\varphi'_R) - y\sin(\varphi'_R) + x'_R, \\ y_R &= x\sin(\varphi'_R) + y\cos(\varphi'_R) + y'_R, \\ \varphi_R &= \Phi_R + \varphi'_R. \end{aligned} \tag{5}$$

By this method we can control the position and orientation of the differential-drive wheeled robot in order to reach any desired position and orientation. Figure 2 shows a simulation of this method, in which the initial configuration is $q_R = (0, 0, 90)$ and desired configurations are located on a 20×20 square centered at (0, 0). Arrows in red show the instantaneous orientations of the robot. In this way we can control the position and orientation of the robot simultaneously.

Since according to Eq. 3 in each instant the robot's velocity depends on its distant to the goal, as the distance decreases, the velocity reduces accordingly (with a negative acceleration). As a result, the robot approaches too slowly to the target when it is very near to it, which is unfavorable for real-time obstacle avoidance. So, we define a clearance threshold for reaching the target so that the robot stops approaching the target beyond the threshold. Now if the robot slips on the ground support at any instant during approaching the target, it will either accidentally end up near the target beyond the clearance threshold, or deviate from its normal track. In the first case the robot will stop and plan its next local target, and in the second case it will correct its trajectory by defining a new auxiliary point which will lead the robot toward the target.

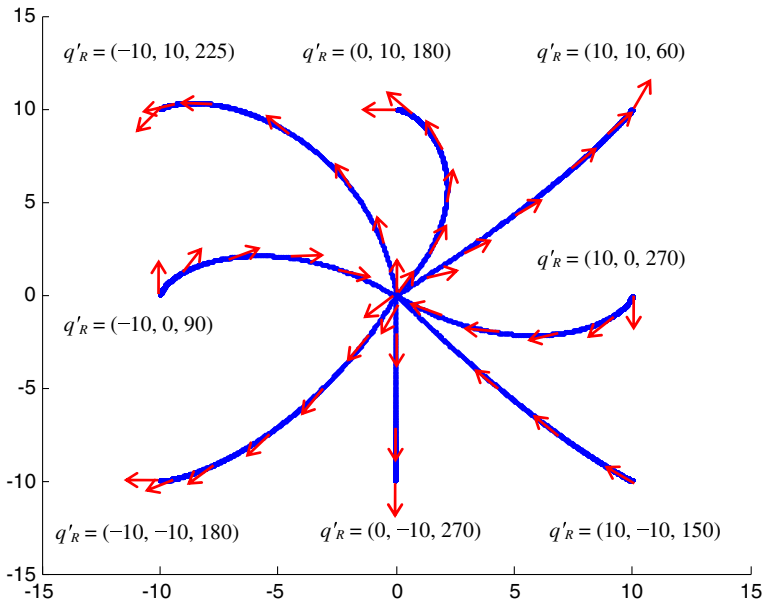

Fig. 2 Robot trajectory in x-y plane with initial condition $q_R = (0, 0, 90)$ and reaching various configurations

2.2 Obstacles

Obstacles are planar polygonal convex or concave objects which can be static or dynamic. The configuration of obstacle O_i at time t is denoted by $q_{O_i}(t) = \left(x_{O_i}(t), y_{O_i}(t)\right)$ which indicates the coordinates of the obstacle's origin point. The velocity of obstacle O_i is specified by $\vec{V}_{O_i}(t)$, with a speed proportional to the robot's velocity and a value in the interval $\left[0,\ \alpha \cdot \vec{V}_R\right]$ in which $0 < \alpha < 1$ is a regulating parameter.

The obstacles are identifiable by the robot but their instantaneous velocities are unknown to the robot. Obstacles move along arbitrary trajectories within a predefined rectangular border, without rotating about an internal axis. It is assumed that when obstacles collide with the border or other obstacles, they bounce back along the reverse direction of their pre-collision route.

2.3 Target

It is assumed that only one target exists in the environment which moves within the defined border for both circular and polygonal robots the position and direction of the target is important. As a result, the robot must regulate its translation and rotation in a way that it finally catches the target at the right direction. The configuration of the target at time t is shown by $q_T(t) = (x_T(t), y_T(t), \varphi_T(t))$ and its velocity is shown by $\vec{V}_T(t)$, respectively, and it is assumed that $q_T(t)$ is known to the robot at time t.

3 Target Following

One of the earliest examples of target following dates back to 1870s, when a missile guidance technique was used by the Prussian Army to destroy enemy vessels by guided torpedoes. The missile guidance discipline has evolved ever since into five main approaches: (1) Line-Of-Sight; (2) Pure Pursuit; (3) Proportional Navigation Guidance; (4) Optimal-control Guidance; and (5) other methods including differential game theory [30]. In the above methods, it is generally assumed that the future trajectory of the target is completely known to the follower [31].

In this section, we present our method of following a moving target, which is inspired by [26], where a Line-Of-Sight (LOS) was defined as the relative position vector $\vec{r}$ connecting the robot to the target, as shown in Fig. 3. Also, the 'Parallel Navigation' law was introduced which states that while the robot approaches the target, the direction of the LOS should remain constant relative to a non-rotating frame. That is, the relative velocity between the robot and the target should remain parallel to the LOS (i.e. $\vec{r}$) at all times. If

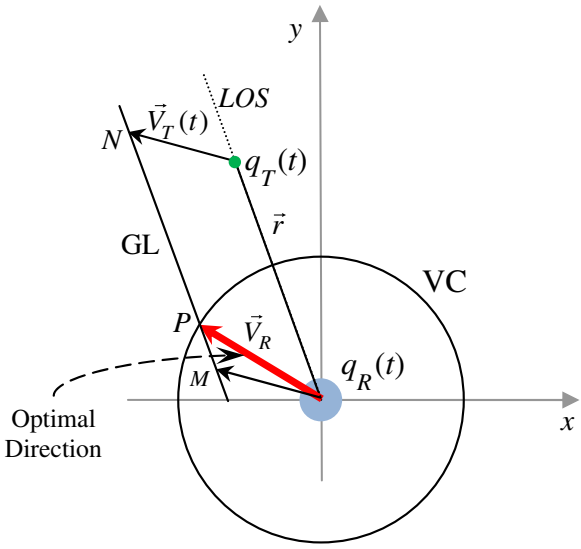

Fig. 3 Case C2: at the direction $\vec{V}_R$ the target will be intercepted in minimal time

this rule holds throughout the motion of the interceptor, the distance between the robot and the target would decrease until they collide. Furthermore, if the target moves with a constant velocity, parallel navigation results in global time-optimal interception.

The parallel navigation law is expressed by the following two relations:

$$\vec{r} \times \dot{r} = 0, \tag{6}$$

$$\vec{r} \cdot \dot{r} < 0. \tag{7}$$

Equation 6 guarantees that the LOS and relative velocity remain parallel, while Eq. 7 ensures that the interceptor is not moving away from the target.

In Fig. 3 suppose that the target T moves along $\vec{V}_T(t)$. In order to show the instantaneous relative positions of the target and the robot, the origin of the coordinates system was set to be on the center of the robot. The velocity of the target $\vec{V}_T(t)$ at time t and the direction of the LOS ($\vec{r}$) are known to the robot. The endpoints of the velocity vectors show the position of the target and the robot after that one time-interval has elapsed (i.e., at time $t+1$). A 'Guidance Line' (GL) is defined as a line parallel to the LOS and passing through the target's velocity vector (see Fig. 3).

According to the Parallel Navigation rule, in order to guarantee a positional matching between the robot and the moving target the direction of the LOS must remain constant. This can be achieved by selecting the robot's velocity vector $\vec{V}_R$ in a way that its endpoint lies on the GL. Therefore we propose the following steps for calculating the $\vec{V}_R$ (with reference to Fig. 3):

Step 1 Draw the LOS by connecting $q_R(t)$ and $q_T(t)$.

Step 2 Draw a vector parallel and equal to the $\vec{V}_T(t)$ from $q_R(t)$.

Step 3 Draw the Guidance line (GL) passing through the endpoint of this vector (i.e. point M in Fig. 3) and parallel to the LOS.

Step 4 Find the locus of points with a distance $\left\|\vec{V}_R\right\|$ from $q_R(t)$, which lie on the Velocity Circle $C = \left\{(x, y) \middle| \sqrt{(x - x_R(t))^2 + (y - y_R(t))^2} = \left\|\vec{V}_R\right\|\right\}$.

Step 5 Based on possible intersections of the line GL and the Velocity Circle VC, the following cases may occur:

(C1) The GL and VC do not intersect (Fig. 4a). I.e.: VC ∩ GL = ∅
This case means that the velocity of the robot is too small to reach the target. So, if the velocity of the target cannot be decreased or the velocity of the robot cannot be increased then the problem will have no feasible solution. Nevertheless, the target may be caught if the robot waits until the target (possibly) moves toward it.

(C2) The GL and VC have two intersection points, one of them lying on $\overline{MN}$ (Fig. 3). I.e.:

$$\exists\, p_1\,,\ p_2 \in \text{VC} \cap \text{GL};$$
$$p_1 \in \overline{MN}\,,\ p_2 \notin \overline{MN},$$

In this case the optimal direction $\vec{V}_R$ can be found as shown in Fig. 3.

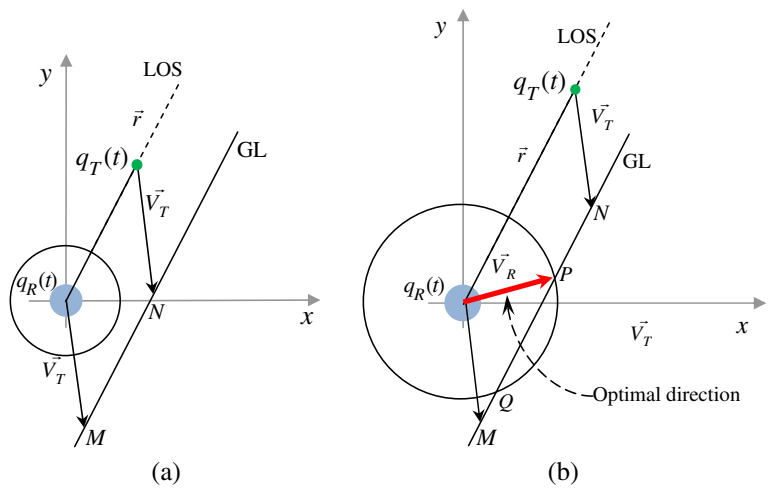

Fig. 4 **a** Case C1 with no feasible solution, **b** Case C3

(C3) The GL and VC have two intersection points, both of them lying on $\overline{MN}$ (Fig. 4b). I.e.:

$$\exists\, p_1\ ,\ p_2 \in \mathrm{VC} \cap \mathrm{GL}; \quad p_1, p_2 \in \overline{MN}.$$

This case is a generalization of the case when the GL and VC have one intersection point. In this case by selecting either of the points P or Q as the endpoint of the $\vec{V}_R$ the target can be caught, but since the point P is nearer to the endpoint of $\vec{V}_T(t)$, its selection will yield the optimal solution.

(C4) The GL and VC have two intersection points with the point N lying in between (Fig. 5). I.e.:

$$\exists\, p_1\ , p_2 \in \mathrm{VC} \cap \mathrm{GL}; \quad N \in \overline{p_1 p_2}.$$

This case occurs when the robot can catch the target in one time-interval (Fig. 5a), or the velocity of the robot is greater than that of the target (Fig. 5b). Accordingly, one of the following decisions can be made:

(1) The velocity of robot is reduced: as a result the optimum path will be along the vector that starts from the robot and ends at the point N, as shown in Fig. 5a. This solution is optimal and is used in this paper.
(2) The point Q is selected as the endpoint of the velocity vector of the robot: as a result the robot will be directed toward the target but its path will not be optimal. This solution is used when the velocity of the robot cannot be changed.

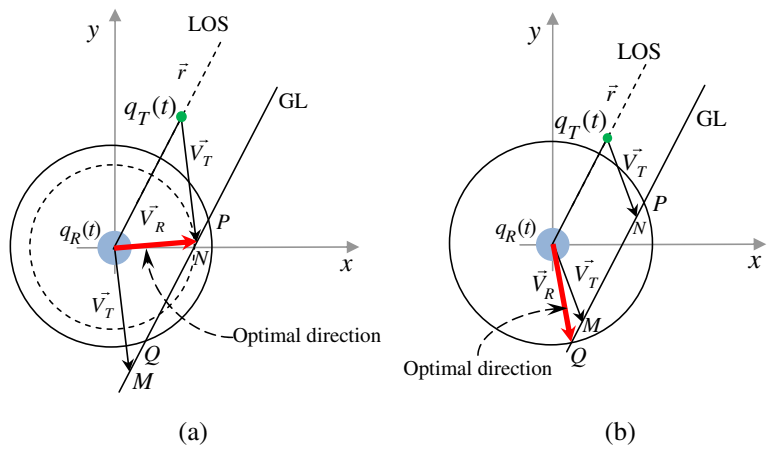

Fig. 5 Two possibilities for the Case C4

Through the abovementioned 5-step procedure the robot's optimal direction for intercepting the target as early as possible can be calculated. As long as the robot can freely follow this direction, no other direction needs to be calculated. However, in more realistic or complicated problems the robot should be able to maneuver through static or dynamic obstacles and cannot always move along the optimal direction toward the target. As a result, when the robot deviates from the optimal direction, new calculations must be performed to find the best path toward the target.

In the next section a new method is proposed for avoiding collisions with moving obstacles while still pursuing the moving target.

4 Directive Circle: A New Tool for Obstacle Avoidance

In this section we present a solution to the obstacle avoidance problem in planning the motion of a circular wheeled mobile robot in presence of moving obstacles. The case of polygonal robots will be discussed in the next section.

First we introduce the concept of 'Collision Cone' as the set of relative velocities between the robot (R) and a colliding obstacle (O_i), which is developed in [10] and defined as:

$$CC_{R,O_i} = \left\{ v_{R,O_i} \middle| \theta_{R,O_i} \cap O_i \neq \emptyset \right\}. \tag{8}$$

In Fig. 6, for a circular robot R and the concave obstacle O_i moving at velocity $\vec{V}_{O_i}$ the collision cone CC_{R,O_i} is shown (in pink), which is bounded by the robot, the obstacle, and the two rays λ_r and λ_l tangent to the obstacle. In Fig. 6, $\vec{V}_{R,O_i} = \vec{V}_R - \vec{V}_{O_i}$ is the relative velocity of the robot and the obstacle, and θ_{R,O_i} is its direction. If $\vec{V}_{R,O_i}$ falls within the CC_{R,O_i} then a collision will be unavoidable. Note that in two dimensions the collision cone appears as a subset of a circular sector.

In [10] the collision cone is translated by $\vec{V}_{O_i}$ and its Minkowski sum with the obstacle's velocity forms a Velocity Obstacle (VO). Since for each robot-obstacle pair there is a unique collision cone, the Velocity Obstacles of all robot-obstacle pairs must be calculated.

Furthermore, a set of Reachable Avoidance Velocities (RAV) are also computed for taking into account the robot's dynamic constraints, after which the robot must select directions outside of all VOs and inside the RAV. In case of multiple obstacles, the robot gives priority to avoiding the obstacle which causes the most imminent collision, and keeps this behavior until all obstacles are avoided.

We should note that despite the mathematic correctness of the Velocity Obstacle method, computing all VOs and the RAV and performing set sums and differences at each time-interval is quite time-consuming, and given that for m obstacles the RAV set consists of at most $3m$ subsets (proved in [10]), the method is hardly applicable to real-time applications or problems with lots of obstacles.

In this section we introduce a new simple tool for online obstacle avoidance called *Directive Circle* (DC), which can be implemented very easily in real-time with much less calculations than the VO method and without suffering from high number of obstacles. In the DC method, first the Directive Circle for a robot and one or more obstacles is constructed, and then the best feasible direction is calculated.

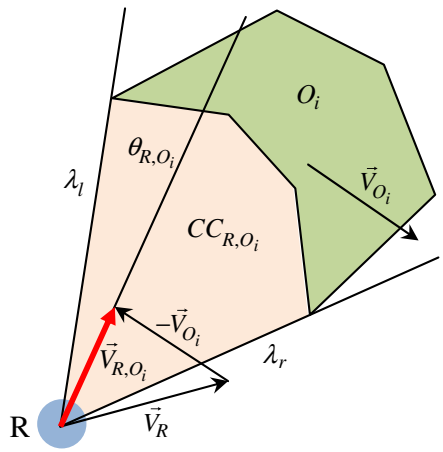

Fig. 6 The relative velocity $\vec{V}_{R,O_i}$ and the collision cone CC_{R,O_i}

4.1 Construction of the Directive Circle

The DC specifies all admissible and forbidden moving directions for a robot. Forbidden directions are those θ_{R,O_i} that lie inside the collision cone CC_{R,O_i} and therefore lead to collision with the obstacle O_i. The DC is constructed by shifting the robot's position along the obstacle's reverse direction (i.e. $-\vec{V}_{O_i}$) and then by drawing a Velocity Circle VC centered at the robot's origin with a radius of the robot's maximum velocity, $\vec{V}_R$, as shown in Fig. 7a.

Regarding the relative positions of the velocity circle VC and the collision cone CC_{R,O_i}, five possible cases may occur:

(C1) The velocity circle VC intersects both the λ_r and λ_l, each at one point:
In this case, suppose that the circle VC intersects the λ_l and λ_r at points A and B, respectively (Fig. 7a). If the relative velocity vector $\vec{V}_{R,O_i}$ lies in the collision cone CC_{R,O_i} (or in other words, $\vec{V}_{R,O_i}$ intersects the arc $\widehat{AB}$), then the robot will collide with the obstacle. Consequently, the Directive Circle is formed by signifying a forbidden zone within the circular sector APB, as illustrated in the upper circle of Fig. 7a.

(C2) The velocity circle VC intersects either λ_r or λ_l at two points:
This case is similar to the case C1 except for that both the points A and B lie on either λ_l or λ_r. The Directive Circle has the same form as in the case C1, i.e., the forbidden directions lie in the circular sector APB.

(C3) The velocity circle VC intersects both λ_r and λ_l, each at two points:
In this case, suppose that VC intersects the line λ_r at points A and B, and the line λ_l at points C and D (Fig. 7b). Now, if the relative velocity vector $\vec{V}_{R,O_i}$ intersects the arc $\widehat{AC}$ or $\widehat{AB}$, then the robot will collide with the obstacle. Consequently, the Directive Circle is formed by signifying two forbidden zones within the circular sectors APC and BPD, as illustrated in the upper circle of Fig. 7b.

(C4) The velocity circle VC does not intersect λ_r and λ_l and falls inside the CC_{R,O_i}:
In this case all possible directions are forbidden and the Directive Circle is totally blocked (Fig. 8a). This situation usually happens when the robot is in a cluttered space and/or obstacles have surrounded the robot tightly. Although a collision is seemingly inevitable, the robot should increase its speed (if possible) to enlarge the velocity circle VC until it intersects with either of the tangent lines λ_r or λ_l and move toward the direction of the intersection. If a speed increase is not possible, the robot must move along a direction perpendicular to either λ_r or λ_l (the one that is closer to the circle), and thus clear the obstacle's track as much as possible, in a hope for a change in the obstacle's speed or course of motion.
Another solution for unblocking some sectors of the DC is to decrease the robot's sensing range r_s, as a result of which only

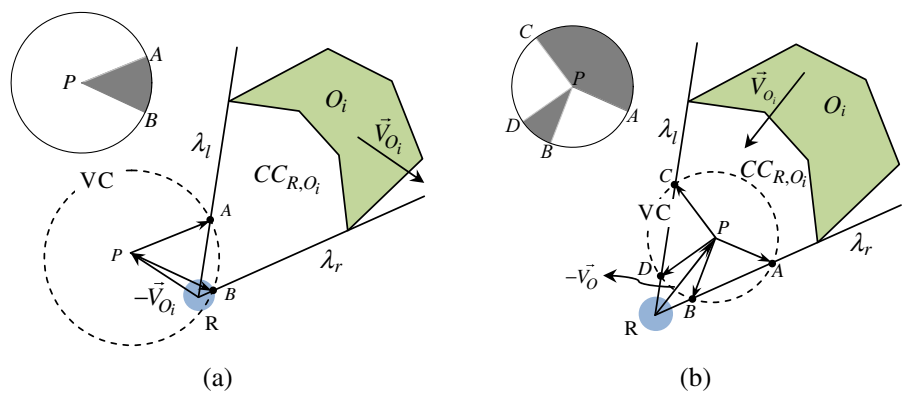

Fig. 7 **a** Case C1 with forbidden zone APB; **b** Case C3 with forbidden zones APC and BPD

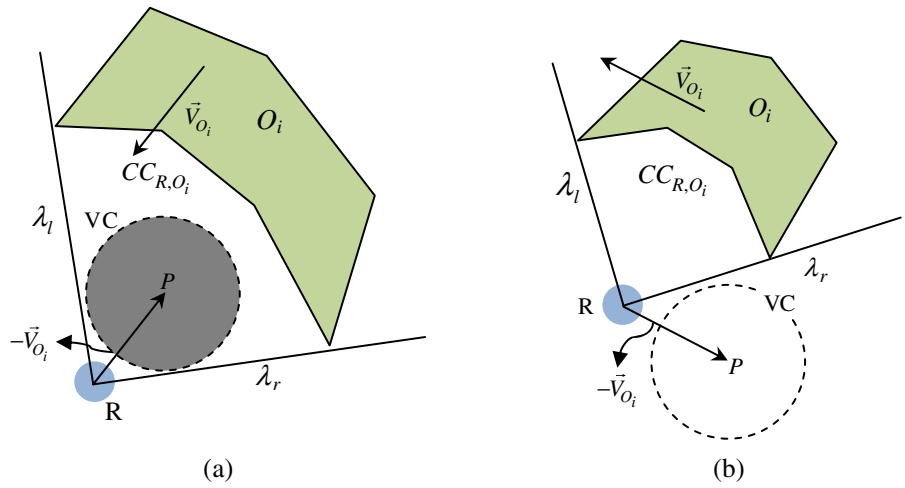

Fig. 8 **a** Case C4 with no admissible direction; **b** Case C5 with no forbidden zone

very close obstacles will be considered in forming the DC, and thus some admissible zones may appear by disregarding further obstacles. This will enable the robot to escape from narrow passages temporarily, after which the sensing range should be reverted to its original value.

(C5) The velocity circle VC does not intersect λ_r and λ_l and falls outside the CC_{R,O_i}:
In this case the Directive Circle has no forbidden zone (Fig. 8b) and the robot can freely move along the calculated optimal direction to the target.

When there are more than one obstacle in the workspace, the Directive Circle must be computed for all moving or static obstacles. Let F_{DC_i} and A_{DC_i} respectively stand for the forbidden and admissible zone(s) of the Directive Circle of obstacle O_i. Then the Overall Directive Circle (ODC) for all obstacles denotes all admissible and forbidden directions, and its forbidden zone F_{ODC} is the union of all F_{DC_i}, and its admissible zone A_{ODC} is the complement of F_{ODC}:

$$F_{ODC} = \bigcup_{i=1}^{m} F_{DC_i}, \quad A_{ODC} = S^1 \backslash F_{ODC}. \tag{9}$$

For instance, the ODC of two obstacles O_1 and O_2 is illustrated in Fig. 9 by superimposing two partial DCs.

For expressing the Overall Directive Circle in an exact and canonical form we propose a 'signature', which is a string of ordered pairs such that in each pair the first element indicates the starting angle of a forbidden zone and the second element shows the angular span of the zone, counted

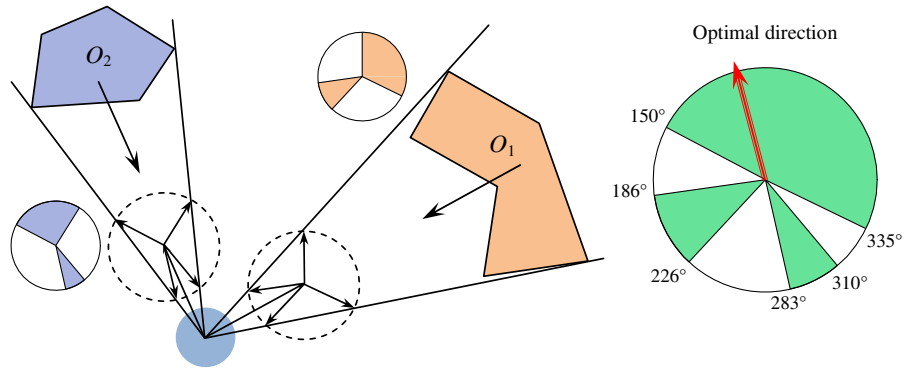

Fig. 9 The Overall Directive Circle (*right*) is formed by superimposing the Directive Circles of all obstacles. In this example the optimal direction lies in the forbidden zone

counterclockwise. Meanwhile, the pairs are sorted in ascending order of their first elements. For example, the signature of the ODC shown in Fig. 9 is [(186, 40), (283, 27), (335, 175)].

4.2 Calculating the Best Feasible Direction

After constructing the Overall DC, the robot can select any direction within admissible zones to avoid collisions. However, since at the same time it is pursuing a moving target, it should select a direction that is both collision-free and pointing toward the target.

Assuming that θ_{opt} is the robot's optimal direction for intercepting the target as early as possible (as calculated in Section 3), there are two possibilities regarding the relative positions of the θ_{opt} and admissible zones of the DC:

(1) The optimal direction falls within the admissible zone; that is, $\theta_{\text{opt}} \in A_{ODC}$: Then the robot can freely move along the $\vec{V}_R$ and avoid any collision.

(2) The optimal direction falls within the forbidden zone; that is, $\theta_{\text{opt}} \in F_{ODC}$: Then the robot must abandon the optimal direction and seek for the best feasible free direction. For this purpose, the robot should consider its next direction θ_{t+1} based on three criteria:

   (i) The best next direction must be admissible;
   (ii) The best next direction should have minimum deviation from the optimal direction θ_{opt} (to maintain optimality);
   (iii) The best next direction should have minimum deviation from the current direction θ_t (to maintain trajectory smoothness).

Therefore, a weighted aggregation of these criteria can provide a sound decision, as formulated in Eq. 10:

$$\theta_{t+1} = \arg\min_i \left(w_1 \cdot \left|\theta_{opt} - \theta_i\right| + w_2 \cdot \left|\theta_t - \theta_i\right|\right);$$
$$\forall \theta_i \in A_{ODC};\ w_1 + w_2 = 1,\ w_1,\ w_2 > 0. \quad (10)$$

Through extensive experiments and simulations the best value for w_1 was specified in the range of [0.7, 0.9].

4.3 Translational Collision Checking

Up to now, all calculations for determining the collision cone, Directive Circle, and the Best Feasible Direction have been based on the center of a circular robot. However, in order to effectively avoid collisions with obstacles, we need to check the motion safety for the whole robot.

The motion of a circular robot will be safe if all its points are guaranteed to avoid an obstacle, which can be realized by making sure that the relative velocity of each point on the robot and the obstacle lies outside the collision cone of that point and the obstacle. However, there is no need to check this condition for every point on the robot, and considering only the robot's two most extremal points will suffice. For example, in Fig. 10 the velocity vectors of the robot R and the obstacle O_i are $\vec{V}_{O_i}$ and $\vec{V}_R$, respectively. Then, the robot's two extremal points are A and B, which are endpoints of the diameter perpendicular to the robot's velocity vector. The next step is to calculate the relative velocities and collision cones of the extremal points and the obstacle, namely, $\vec{V}_{A,O_i}$, $\vec{V}_{B,O_i}$, CC_{A,O_i} (the polygon ACDEF) and CC_{B,O_i} (the polygon BCDEF). If both $\vec{V}_{A,O_i}$ and $\vec{V}_{B,O_i}$ lie outside of their respective collision cones, then the robot will not collide with the obstacle. In case of multiple obstacles, the translational collision checking procedure must be performed for each obstacle.

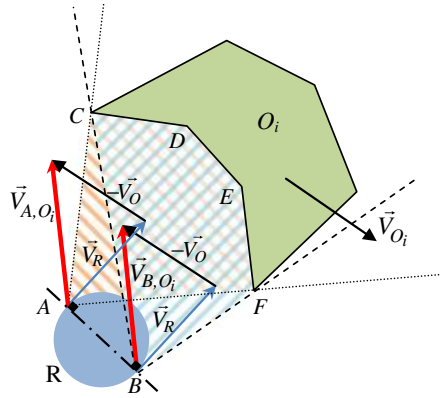

Fig. 10 Translational collision checking for the extremal points of a circular robot: here the point B will collide with the obstacle because its relative velocity $\vec{V}_{B,O_i}$ lies inside the collision cone BCDEF

It is noted that in order to save computational time, the collision cones of the robot's extremal points are calculated based on the obstacle points sensed from the origin point, rather than scanning the environment from each point separately. In fact, for relatively small robots, we assume that the scenes viewed from the two extremal points are equivalent, and thus ignore the slight differences in vision (i.e., the stereoscopic effect).

5 Motion Planning of Polygonal Robots

In Section 4 we proposed a locally optimal method for motion planning of circular robots in dynamic environments based on the Directive Circle. In this section, the proposed method is generalized to the case of polygonal robots. It is assumed that at each time-step the polygonal robot can rotate at most $\theta_{\max}$ degrees around its origin, which can be either inside or outside of the robot.

For motion planning of a polygonal robot the following procedure is performed:

(1) Similar to the circular robot case and based on the robot's origin point, the Overall Directive Circle is constructed by superimposing the DCs of all obstacles, and admissible and forbidden zones are determined.
(2) The Best Feasible Direction is calculated as in Section 4.2.
(3) The translational collision checking for all vertices of the polygon is done according to the Section 4.3. Actually, instead of the two extremal points in the circular robot case, here the navigability of all vertices must be examined. As mentioned earlier, the obstacle shape sensed from the robot's origin is used for all extremal vertices to save computational time. However, when some vertices are far from the robot's origin, or the workspace is cluttered with narrow passages, the visibility scan must be performed from all vertices to capture a more accurate image of the surrounding obstacles.
(4) In order to increase the ability of the polygonal robot to navigate through narrow passages and avoid collisions effectively, the robot must move with its minimal width perpendicular to the Best Feasible Direction. The minimal width of a polygon is the minimum value among the distances between all edge-edge and vertex-edge antipodal pairs of the polygon, and can be computed in $O(n)$ for a polygon with n vertices through the 'rotating calipers' method proposed by [32]. Thus we have assumed that the robot's axle is placed parallel to the minimal width, with the Origin point being at its middle.
For instance, in Fig. 11, the minimal width of the polygon is equal to CH, and the robot's swept area (i.e., the trace behind it) forms a rectangular 'strip'.
(5) After calculating the amount of rotation in the current time step, it must be ensured that the robot will not collide with any obstacle as a result of its rotation. This is done through rotational collision checking procedures explained in Section 5.1. If the rotation is collision-free, then the robot's local trajectory is planned using the exponential stabilizing control scheme described in Section 2.1.1.
(6) Steps 1 to 5 are repeated for each new position of the robot until it comes close to the target and initiates the target-catching procedure as described in Section 5.2.

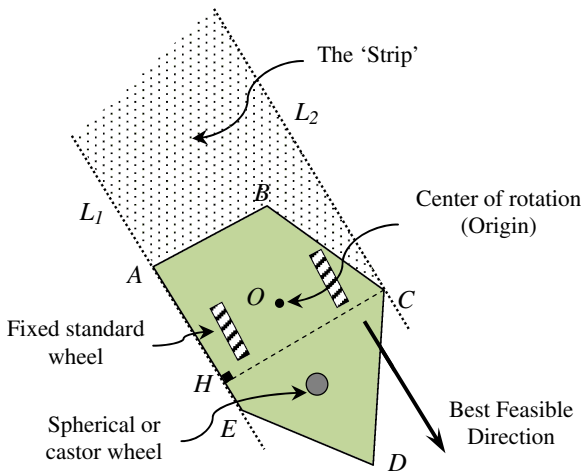

Fig. 11 The two parallel supporting lines L_1 and L_2 confine the polygonal robot at its minimal width

5.1 Rotational Collision Checking

As mentioned in Section 2.1, it is assumed that the robot uses a range sensor to detect the environment. This range sensor can be modeled as a raw distance function ρ: $\Re^2 \times S^1 \rightarrow \Re$. Assume that the robot's origin is on $X_R = (x_R, y_R) \in \Re^2$ with rays radially emanating from it. For $\theta \in S^1$, the value $\rho\ (X_R, \theta)$ denotes the distance to the closest obstacle along the ray from X_R at angle θ relative to the positive x-axis. Mathematically,

$$\begin{aligned} \rho\left(X_R, \theta\right) &= \min\left\{d\left(X_R, X_O^\theta\right)\right\}; \\ X_O^\theta &= X_R + \lambda \cdot [\cos\theta, \sin\theta]^T \in \bigcup_i O_i\ ; \\ \lambda &\in [0,\ r_s]\,, \end{aligned} \tag{11}$$

in which $d\left(X_R, X_O^\theta\right)$ is the Euclidean distance between the robot's origin and any obstacle intersecting the ray θ.

Also, we specify the minimum clearance of the robot to the closest obstacle as $Dis(X_R) = \min_{\forall\theta}\{\rho\ (X_R, \theta)\}$.

By denoting the Euclidean distance of vertex v_i of the robot from its origin as D_i and its direction as γ_{v_i}, the circumscribing circle (CS) of a polygonal robot is defined as the circle centered at the origin with a radius of $\max(D_i)$; that is, passing through the vertex farthest from the origin, as illustrated in Fig. 12.

The collision test is as follows: If for $\forall\ \theta \in S^1$: $\rho\ (X_R, \theta) > \max(D_i)$ (i.e., if the nearest obstacle lies outside of the circumscribing circle, as shown in Fig. 12), then there will be no collision between the robot and obstacle O_i.

Otherwise, two types of collisions may occur:

Type A Collision: An edge of the robot (E_i^R) intersects with a vertex of the obstacle $(v_n^{O_j})$, as shown in Fig. 13a.

Type B Collision: A vertex of the robot (v_i^R) intersects with an edge of the obstacle $(E_n^{O_j})$, as shown in Fig. 13b.

Since initially it is not known which of the above Collision Types will occur, through the following procedures we will calculate the maximum possible clockwise and counterclockwise rotations (τ_{cw} and τ_{ccw}, respectively) of the robot for each Type, and then integrate them to determine the amount of safe rotation at the current time-step.

Procedure A: In this procedure, the maximum clockwise and counterclockwise safe rotations of the robot, until the Type A collision occurs, are determined.

For each ray inside the CS we define circle C^θ centered at X_R and with a radius of $\rho\ (X_R, \theta)$.

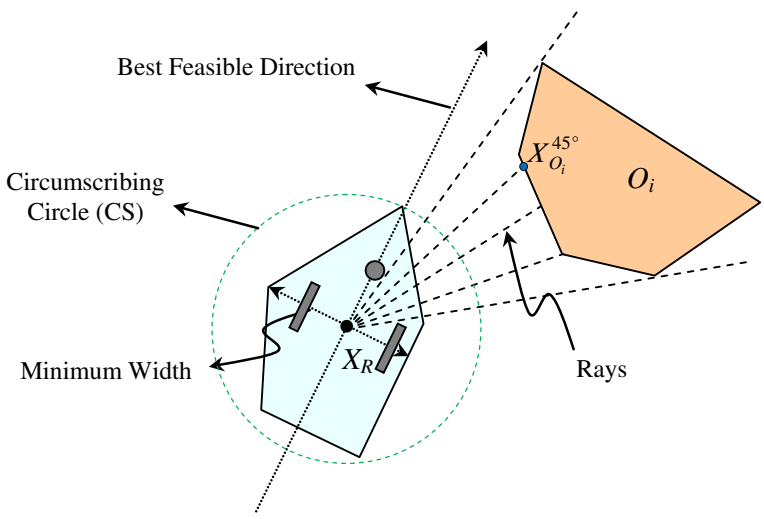

Fig. 12 The robot can rotate freely since the obstacle does not intersect its Circumscribing Circle

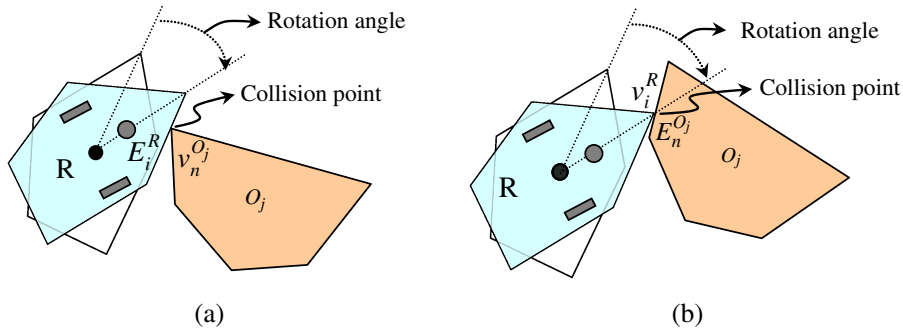

Fig. 13 **a** Type A collision, **b** Type B collision

Each circle C^θ will intersect the robot in $k \geq 1$ points. Assuming that the directions of the intersection points relative to the positive x-axis are called γ_k^θ (as shown in Fig. 14a), the clockwise and counterclockwise safe rotations are respectively denoted by τ_A^{cw} and τ_A^{ccw}, calculated as:

$$\tau_A^{cw} = \min\left(\left|\gamma_k^\theta - \theta\right|\right), \tau_A^{ccw} = \min\left(\left|\theta - \gamma_k^\theta + 360\right|\right), \quad \forall\, \theta \in S^1 :\ \rho\left(X_R, \theta\right) \leq \max\left(D_i\right). \tag{12}$$

Procedure B: In this procedure, the maximum clockwise and counterclockwise safe rotations of the robot, until the Type B collision occurs, are determined. For each vertex v_i of the robot lying outside of the robot's minimum clearance to the closest obstacle, we define circle C_{v_i} as a circle centered at X_R and passing through v_i. The direction of line connecting X_R and v_i is denoted as γ_{v_i}. Each circle C_{v_i} will intersect the sensed edges of the obstacle O_j in $k \geq 1$ points. Assuming that the directions of the intersection points relative to the positive x-axis are called $\eta_k^{v_i}$ (as shown in Fig. 14b), the clockwise and counterclockwise safe rotations are respectively denoted by τ_B^{cw} and τ_B^{ccw}, and calculated as:

$$\tau_B^{cw} = \min\left(\left|\eta_k^{v_i} - \gamma_{v_i}\right|\right), \tau_B^{ccw} = \min\left(\left|\gamma_{v_i} - \eta_k^{v_i} + 360\right|\right), \quad \forall\, v_i;\ D_i \geq Dis\left(X_R\right). \tag{13}$$

Now for the current time-step, the maximum safe clockwise and counterclockwise rotations of the robot (τ_{cw} and τ_{ccw}) around the robot's origin X_R are calculated by:

$$\tau_{CW} = \min\left(\theta_{\max}, \tau_A^{cw}, \tau_B^{cw}\right), \quad \tau_{CCW} = \min\left(\theta_{\max}, \tau_A^{ccw}, \tau_B^{ccw}\right). \tag{14}$$

5.2 Catching the Target

In most researches on dynamic targets the goal is achieved whenever the origin (or any) point of the robot touches the target or comes close to it at a specified clearance. However, sometimes the orientation of the robot at the moment of catching or nearing the target becomes important, such as in grasping parts moving on a conveyor belt at a certain direction or flocking of

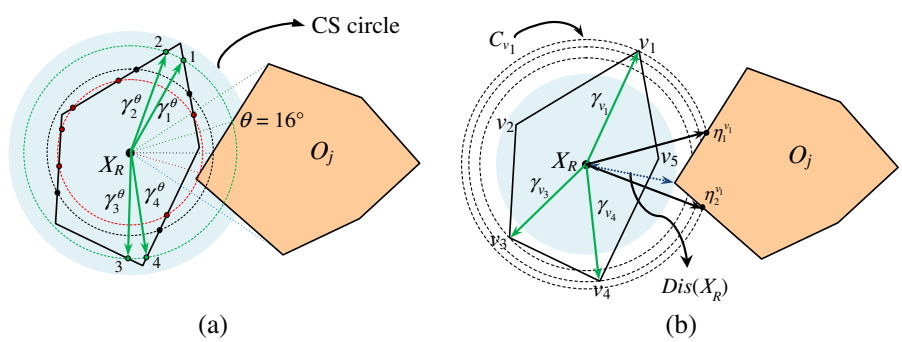

Fig. 14 **a** Procedure A is performed for all rays completely lying inside the circumscribing circle CS. **b** Procedure B is performed for all robot vertices lying outside of the robot's minimum clearance to the closest obstacle

aircrafts in a specific arrangement in airshows. By incorporating the exponential stabilizing control scheme into our planning method, the robot will be able to catch the target in the same orientation as that of the target. Meanwhile, for minimizing the required time for such an action, the robot must rotate and translate simultaneously toward the moving target. Therefore, supposing that the robot moves at its maximum speed $\vec{V}_R$ and the target will move at its current speed $\vec{V}_T(t)$, we can estimate the proper moment for shifting the motion policy to the aligning action based on the remaining time to reach the target.

Let us define $\tilde{T}_{\text{Trans}}(t)$ as the estimated time-steps (from the current time t) for reaching the target through translational moves, and $\tilde{T}_{\text{Rot}}(t)$ as the estimated time-steps (from the current time t) for aligning with the target through rotational moves, define as follows:

$$\tilde{T}_{\text{Trans}}(t) = \frac{\|X_T(t) - X_R(t)\|}{\vec{V}_R(t)},$$
$$\tilde{T}_{\text{Rot}}(t) = \frac{|\varphi_T(t) - \varphi_R(t)|}{\theta_{\max}}, \tag{15}$$

in which $X_T(t)$ and $X_R(t)$ are respectively the positions of the target and the robot at time t, and

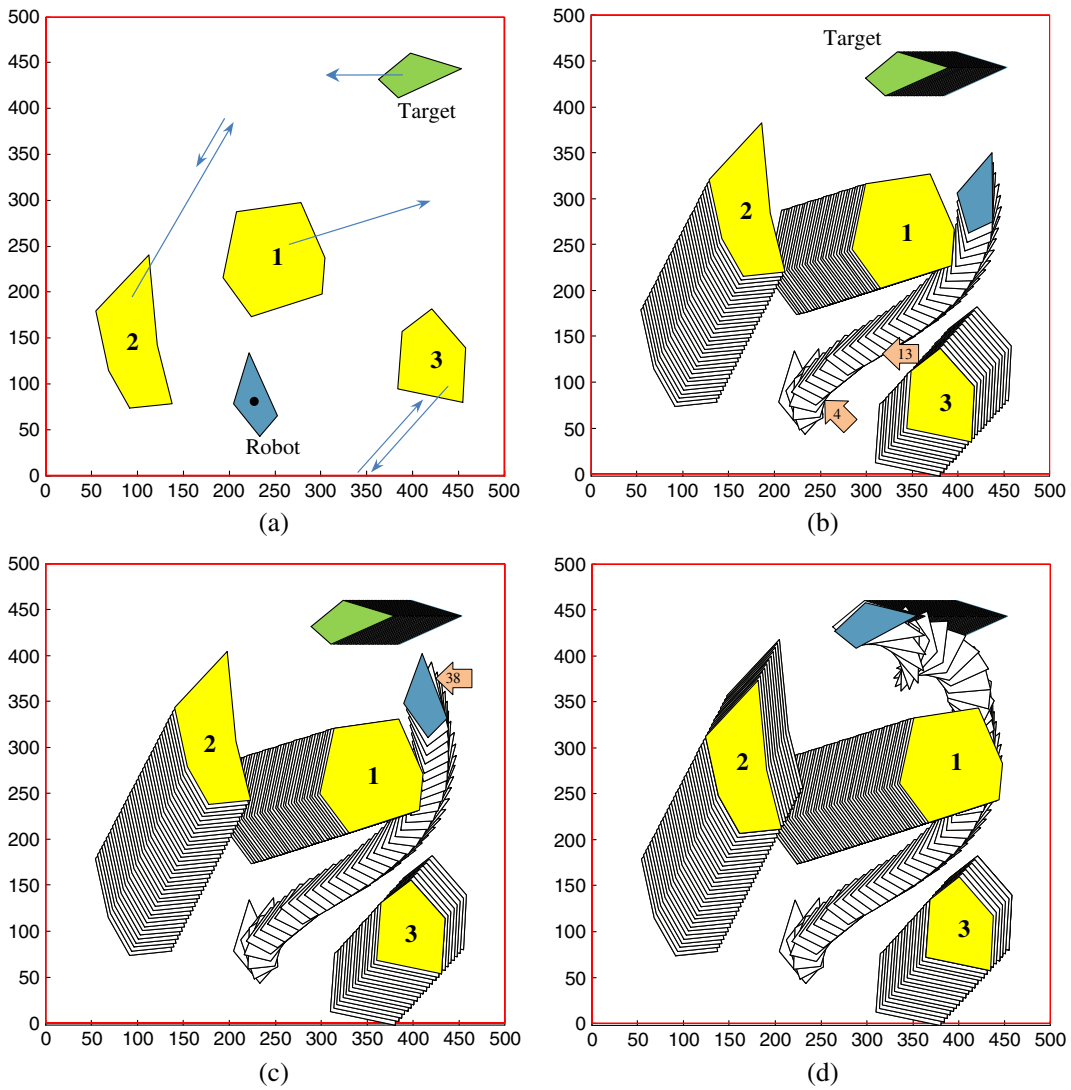

Fig. 15 A sample problem solved by the proposed method. Numbered arrows indicate the elapsed time-steps

$\varphi_T(t)$ and $\varphi_R(t)$ are the respectively the orientations of the target and the robot at time t. The aligning action starts when the following relation holds:

$$\tilde{T}_{\text{Trans}}(t) \leq (1 + \beta \cdot n_O(t)) \cdot \tilde{T}_{\text{Rot}}(t), \tag{16}$$

in which $n_O(t)$ is the number of obstacles sensed by the robot at time t, and β is a safety parameter empirically set to 1. Equation 16 means that the robot should start aligning itself with the target as late as possible, that is, when it estimates that the remaining time for reaching the target (plus a safety factor) will barely suffice for completing the alignment action and catching the target at the right orientation. In the meantime, if an obstacle intercepts the robot's path unexpectedly, it will stop the aligning action and shift to the obstacle avoidance mode by moving perpendicular to its minimal width. The aligning action is illustrated in a sample problem in Fig. 15. Since through the Eq. 15, the remaining time for translational and rotational moves are estimated based on the position and orientation of the target at time t, the robot can instantly respond to any variation in the target's velocity or course of motion and update its Best Feasible Direction.

6 Simulation and Comparisons

The presented method for online motion planning of circular and polygonal robots has been successfully programmed and implemented in various, from simple to cluttered environments, with single to numerous, and static to highly dynamic obstacles.

For instance, Fig. 15a illustrates the initial scene of a sample problem for a polygonal robot. As shown in Fig. 15b, the robot starts its motion by translating and rotating around its origin in order

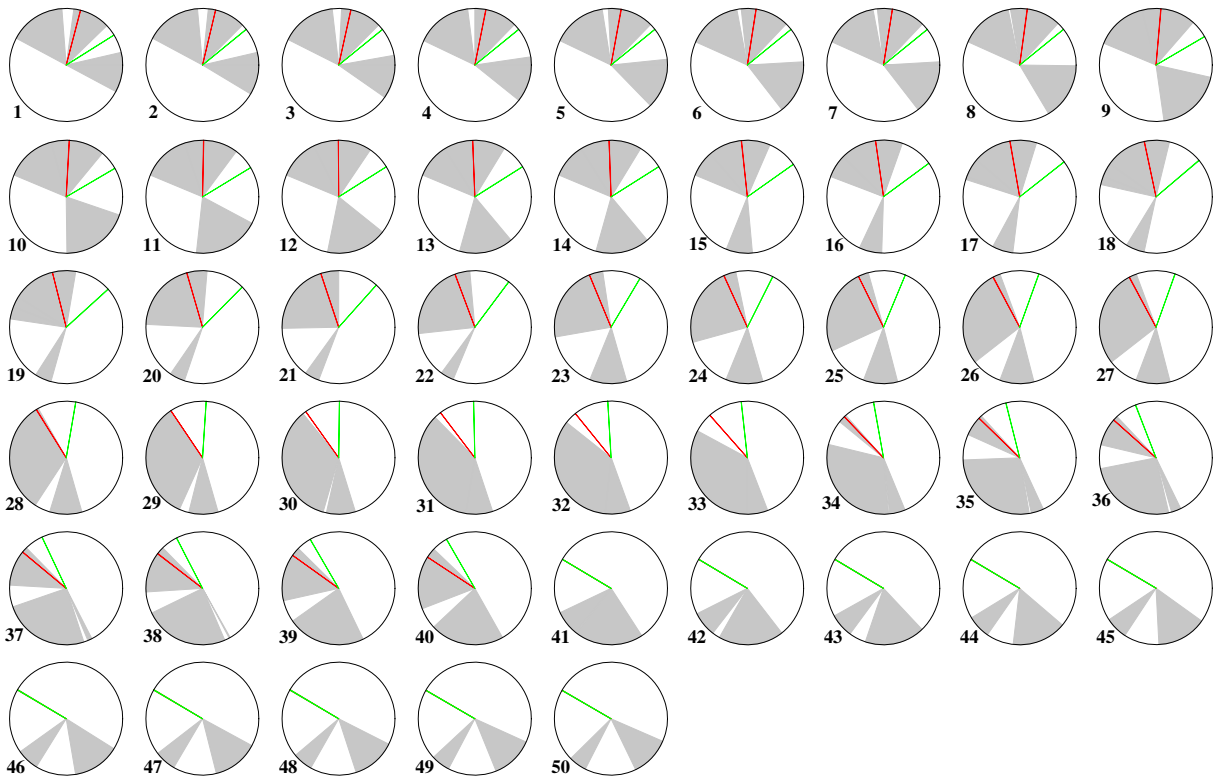

Fig. 16 Overall Directive Circles of the robot in consecutive time-steps. In each step, the grey sectors represent forbidden zones, the red beam shows the optimal direction, and the green beam points to the best feasible (collision-free) direction

to lay its minimal width perpendicular to the Best Feasible Direction, in step 4.

Then it performs translational moves along the Best Feasible Direction until step 13. Afterward, the robot tries to match its path with the optimal path by avoiding a Type A and B collision with the Obstacle 1, as explained in Section 5.1. So the direction of path changes gradually till step 41. In step 38, however, Eq. 16 holds true since $\tilde{T}_{\text{Trans}}(38) = 118.09/10 = 11.809$ and $\tilde{T}_{\text{Rot}}(38) = (111^\circ - 24^\circ)/20^\circ = 4.35$, and with $n_O(43) = 2$ and $\beta = 1$ we will have $11.809 \leq (1 + 2(1)) \times 4.35$, and therefore the robot must shift to the aligning action. From step 41 onwards (Fig. 15c), the optimal and the Best Feasible directions coincide and so the robot follows the target with its minimal width perpendicular to the optimal path. The aligning action completes in 12 time-steps and ultimately the robot catches the target in 50 time-steps (Fig. 15d).

The shapes of Overall Directive Circles during the runtime are plotted in Fig. 16. As the robot and obstacles change position, forbidden zones merge or split and block/unblock certain directions, and the optimal direction (shown in red) moves gradually and slowly counterclockwise. Finally, the optimal and Best Feasible directions coincide in step 41 onwards.

It is noted that the observed angular gaps between the Optimal and Best Feasible directions (red and green beams in Fig. 16) are due to the fact that the Overall Directive circle is calculated only for the robot's Origin (which is in the middle of the robot's axle), while in order to avoid collisions with obstacles the robot must check the safety of its boundary (as described in Section 5). Therefore, most of the time the robot deviates from the Optimal direction, even if the Optimal direction lies in Admissible zone.

The same problem was solved in offline mode to optimality and took 43 time-steps, which shows a small gap between our obtained results.

Some other simulations are done in order to illustrate the efficiency of this method. In Fig. 17a the robot follows its near optimum path and achieves to the target in 30 steps. In this problem the target does not move. In Fig. 17b obstacle no. 9 deviates the robot from near optimum path and finally the robot catches the target in 62 steps. This problem has clutter environment and the robot selects relatively smooth path toward the target unless near the obstacle 9.

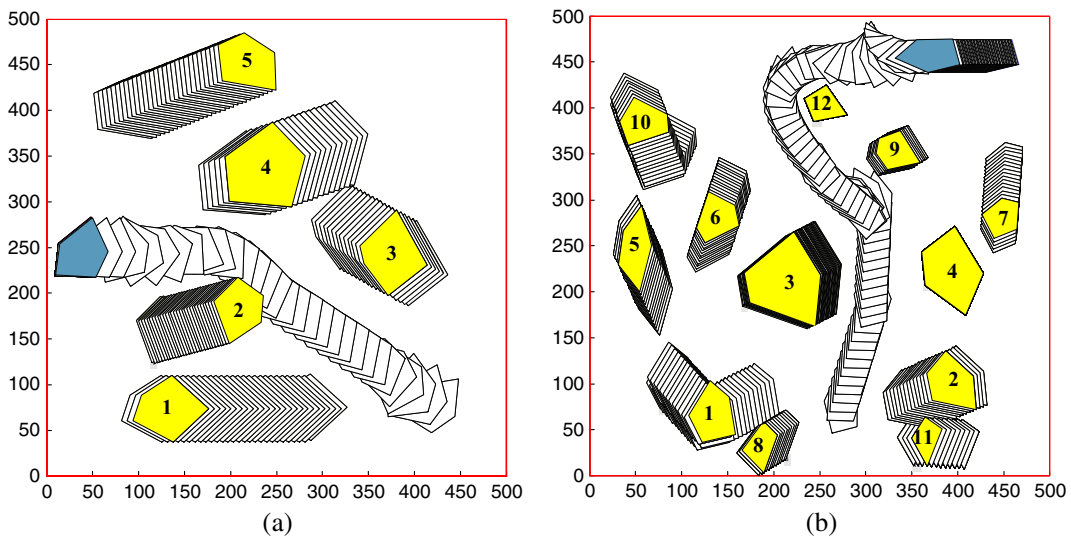

Fig. 17 **a** the robot follows near optimum path and the obstacles do not deviate the robot from near optimum path. **b** Obstacle No. 9 deviates the robot from its optimum path

6.1 Comparisons

In order to assess the efficiency of the developed Directive Circle-based method, its generated paths were compared with near-optimal paths obtained in offline mode, that is, by assuming the shapes, positions and velocities of all obstacles as known a priori. Near-optimal paths were generated by a generalized version of the Rapidly-Exploring Random Tree (RRT) method proposed in [33], which is widely implemented in motion planning problems as a powerful sampling-based method. The RRT incrementally generates random nodes in the free configuration space and gradually builds up a tree, and then searches it to find a start-to-goal path. For circular and Polygonal robots the configuration space of the RRT was 4D with $q(t) = (x_R(t), y_R(t), t)$, and for polygonal robots it was 4D with $q(t) = (x_R(t), y_R(t), \varphi_R(t), t)$.

For comparisons, 10 test problems with various environments and varied obstacle velocities were designed and solved by the two competing methods for 5 different scenarios, in which the obstacles' proportional speed to the robot was set to different values, selected from 0.15, 0.35, 0.55, 0.75 and 0.95. The robot was considered polygonal, since it is more challenging than the circular case.

The simulations were run on an Intel® Core 2 Duo 2.50 GHz CPU with 4 GB RAM. Since the RRT method is based on random samples, each run yields a different tree and hence a different path to the target. Therefore we ran the RRT planner 10 times for each problem and each scenario (adding up to a total of $10 \times 10 \times 5 = 500$ runs), and took the minimum of the 10 runs of each test as its near-optimal path in offline mode. On the other hand, since the Directive Circle method has no stochastic element, it produces the same result for each scenario/problem. The high repeatability and robustness of the method is a beneficial property in obtaining definite and exact results, thus obviating the need for multiple runs. The maximum allowable rotation at each step was set to $\theta_{\max} = 20°$ and the value of w_2 was set between [0.1, 0.3] in Eq. 10.

The computational results are presented in Table 1. For each scenario, the number of time-steps to reach the target in online (Directive Cir-

Table 1 Path length gaps (in percent) between the presented method (online) and near-optimal solutions obtained from the RRT method (offline)

Problem	Obstacle to robot speed ratio (scenarios)														
	15 %			35 %			55 %			75 %			95 %		
	Online	Offline	% Gap	Online	Offline	% Gap	Online	Offline	% Gap	Online	Offline	% Gap	Online	Offline	% Gap
1	42	40	5.0	43	40	7.5	65	55	18.1	64	55	16.3	55	51	7.8
2	43	41	4.9	58	54	7.4	59	51	15.6	49	47	4.2	74	68	8.8
3	46	42	9.5	81	72	12.5	56	49	14.2	52	48	8.3	49	44	11.4
4	50	45	11.1	71	62	14.5	62	54	14.8	52	47	10.6	84	68	23.5
5	50	48	4.2	63	60	5.0	59	53	11.3	59	48	22.9	60	50	20.0
6	47	43	9.3	55	45	22.2	55	46	19.5	62	46	34.8	57	48	18.7
7	46	46	0.0	41	41	0.0	42	42	0.0	44	42	4.8	57	48	18.7
8	46	45	2.2	57	48	18.8	50	50	0.0	44	44	0.0	45	44	2.3
9	42	42	0.0	60	51	17.6	51	47	8.5	46	44	4.6	43	41	4.9
10	48	48	0.0	46	46	0.0	46	46	0.0	46	46	0.0	50	48	4.2
Avg.	46	44	4.6	57	52	10.5	54	49	10.2	52	47	10.7	57	51	12.0

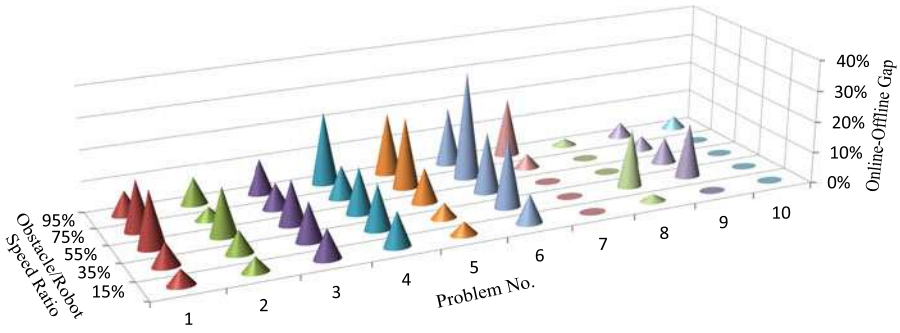

Fig. 18 A plot of the path length gaps between the Directive Circle and near-optimal solutions

cle) and offline (RRT) methods are reported, and the path length gaps in percent are also presented.

The amounts of gaps are also plotted in Fig. 18 for better visualization. As can be seen, when the environment changes slowly (i.e., in scenario 15 %), the online method produces paths quite close to the near-optimal paths, but by increasing the speed of obstacles the robot deviates from near-optimal paths. This happens because the moving directions of obstacles change rapidly and unexpectedly as they bump to each other and bounce back, creating a high level of entropy and uncertainty for the robot. Since the robot lacks any knowledge about the environment, such unexpected changes in the paths of nearby obstacles cause deviations from the optimum path.

A relatively large deviation from the near-optimal solution (34.8 %) can be seen in the problem 6 according to scenarion 75 %. This instance is illustrated in Fig. 19, in which the online and offline (near-optimal) paths are superimposed. The third dimenstion shows the time axis, and each layer represents a time-step. Here the path of obstacles O_1 and O_2 and O_3 at first steps of the problem causes deviation of the robot from near optimum path. The robot responds by changing its course of motion for safety reasons, resulting in a deviation of 34.8 % from the near-optimal path. The target is caught in 62 and 46 time-steps in online and offline modes, respectively. The RRT method produced 8791 nodes in the form of a cone-shaped tree because any expansion in the

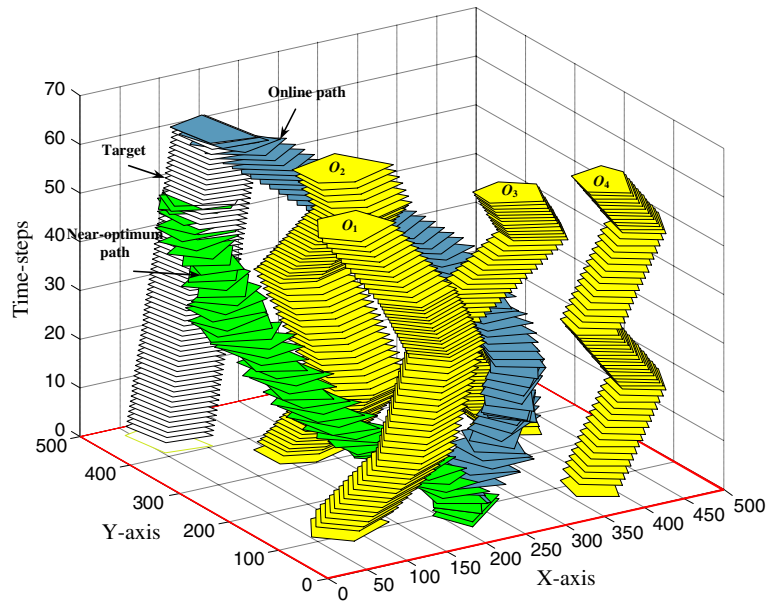

Fig. 19 Simulation of the problem 6 under scenario 75 %. The target is caught in 62 and 46 time-steps in online and offline modes, respectively

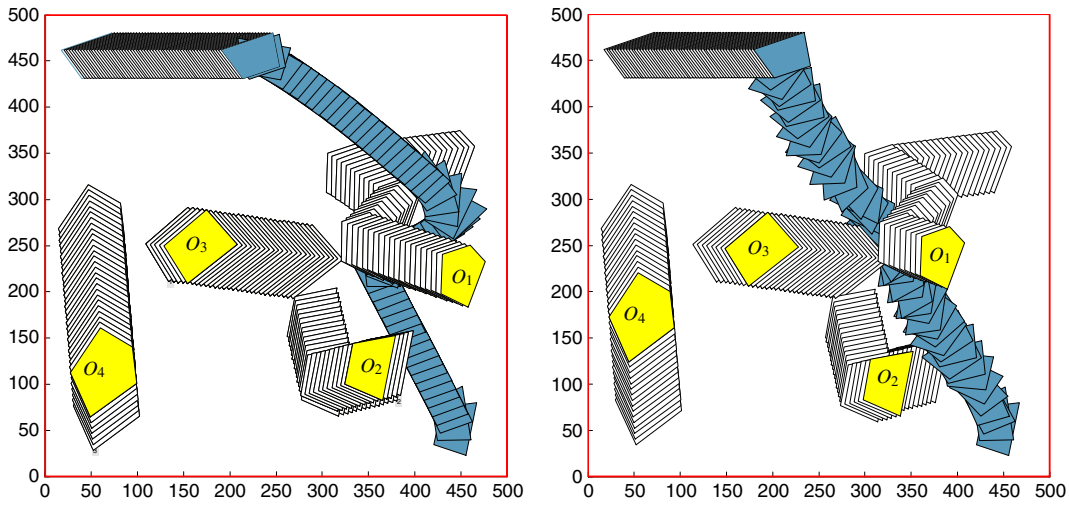

Fig. 20 **a** The robot's trajectory calculated by the Directive Circle method. **b** A near-optimal solution obtained by the RRT sampling-based approach. Note the irregularities in the robot's orientation in **b** due to the randomness of the method

course of time creates a node in one time-layer above the previous layer. The offline solution was constructed within several minutes.

Figure 20a and b show another example where the target is intercepted in 59 and 55 time-steps by online and offline methods, respectively. It should be noted that the near-optimal solution was obtained by taking the minimum path among 50 RRT runs. The 4-step gap occurs because of the unexpected change in the path of obstacle O_1, which is near to the robot and has a direct influence on its motion. Note that the obstacles can also change their motion path without colliding with other obstacles.

Table 2 The main steps of the presented method and their time complexities

Step		Time complexity
1	Visibility scan (S = the number of radial senses at each scan)	$S \times O(1)$
2	Calculating the optimal direction to the target	
	• Drawing the Guidance Line	$O(1)$
	• Drawing the velocity circle	$O(1)$
	• Finding their intersection	$O(1)$
3	Forming the Overall Directive Circle	
	• Calculating the Collision Cones for all m obstacles	$m \times O(1)$
	• Merging all Directive Circles into the Overall Directive Circle	$O(1)$
4	Calculating the best feasible direction	$O(1)$
5	Checking translational collision (n = the number of vertices of a polygonal robot; $n = 2$ in circular robots)	$n \times O(1)$
6	Checking rotational collision (for polygonal robots only)	
	• Calculating Type A collision	$O(1)$
	• Calculating Type B collision	$O(1)$
7	Aligning with the target (for polygonal robots only)	
	• Calculating $\tilde{T}_{\mathrm{Rot}}(t)$	$O(1)$
	• Calculating $\tilde{T}_{\mathrm{Trans}}(t)$	$O(1)$
Total		$O(1)$

The presented method is a fast local obstacle avoidance method which can be performed in real-time, with little memory requirements. Table 2 summarizes the main steps of the method together with the required time complexity of each step. Altogether, the method can be run in constant time at each time-step, and so is suitable for real-time implementations.

7 Conclusions

For any mobile robot operating in real-world conditions, motion planning in dynamic environments is a challenging topic. In this paper we presented a new sensor-based method for online motion planning of circular and polygonal differential-drive wheeled mobile robots which pursue a moving target and get obstructed by moving obstacles. Successive visibility scans enable the robot to detect the instantaneous velocities of the obstacles in real-time. Also, the robot must catch the target at the same orientation as that of the target.

At each time-step of the algorithm, first the optimal direction to the target is calculated through simple geometric calculations, and then a translational collision check is performed to calculate all the directions that lead to collisions with nearby obstacles. All forbidden and admissible directions are integrated in the Directive Circle of that time-step, based on which the robot calculates its best feasible direction. If the robot is polygonal, a further rotational check is done for ensuring that the robot's edges or vertices do not collide with obstacles during its rotation about the origin point. The motion of the robot at each time-step is governed by a exponential stabilized control scheme, through which the robot regulates its translational and rotational velocities in order to smoothly reach its next configuration. At last, when the robot is near to the target and is expected to reach it in a few time-steps, the robot aligns itself with the target, considering a maximum of $\theta_{\max}$ degrees rotation at each time-step.

Computational results and comparisons with near-optimal solutions obtained from RRT-based offline path planning showed that the online algorithm guides the robot close to near-optimal paths quite well, and deviations occur only when the robot avoids narrow passages for maintaining a safety clearance, or encounters abrupt changes in the velocity and direction of nearby obstacles. Overall, in 10 problems each solved with five obstacle-to-robot speed scenarios, the average solution gap between our online and RRT offline methods was less that 10 %, which is quite satisfactory. Aside from simplicity and ease of implementation, another advantage of the method is that it requires no post processing procedure for path smoothing.

There are some issues that must be considered in future versions of the method: here the polygonal robot is convex, but this limitation can be removed by generalizing the rotational collision checking procedure. Besides, currently the robot does not estimate the obstacles' expected (future) trajectories, and responds to their motions based on their current velocity and position. But we can incorporate methods that try to model the behavior of the obstacles and act based on a probabilistic estimation of their position, such as [34]. So the algorithm can be improved further by adding a memory component to record the observed behaviors of obstacles and estimate their future positions. Lastly, due to lack of equipment, we were not able to implement the algorithm in practice, which is hoped to be realized in future.

References

1. Latombe, J.C.: Robot Motion Planning. Kluwer Academic Pub, Boston, MA (1991)
2. Canny, J.: The Complexity of Robot Motion Planning. MIT Press, Cambridge, MA (1988)
3. Canny, J., Reif, J.: New lower bound techniques for robot motion planning. In: Proceedings of IEEE Symposium on the Foundations of Computer Science, pp. 49–60. Los Angeles, CA (1987)
4. Tsubouchi, T., Rude, M.: Motion planning for mobile robots in a time-varying environment. J. Robotics Mechatronics. **8**(1), 15–24 (1996)
5. Du Toit, N.E., Burdick, J.W.: Robotic motion planning in dynamic, cluttered, uncertain environments. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 966–973 (2010)
6. Petty, S., Fraichard, T.: Safe motion planning in dynamic environments. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 2210–2215 (2005)

7. Belkhouche, F.: Reactive path planning in a dynamic environment. In: Proceedings of IEEE Transactions on Robotics, vol. 25, no. 4, pp. 902–911 (2009)
8. Liu, C.L., Tang, J., Yang, J.Y.: Path planning of mobile robot using new potential field method in dynamic environments. In: Seventh International Conference on Natural Computation (ICNC), vol. 2, pp. 1011–1014 (2011)
9. Ishikawa, S.: A method of indoor mobile robot navigation by using fuzzy control. In: Proceedings of IEEE/RSJ Int. Workshop on Intelligent Robots and Systems, vol. 2, pp. 1013–1018 (1991)
10. Fiorini, P., Shiller, Z.: Motion planning in dynamic environment using velocity obstacles. Int. J. Robot. Res. **17**(7), 760–772 (1998)
11. van den Berg, J.P., Overmars, M.H.: Roadmap-based motion planning in dynamic environments. In: Proceedings of IEEE Int. Conf. on intelligent Robots and Systems, vol. 2, pp. 1598–1605 (2004)
12. Inoue, K., Okawa, Y.: On-line motion planning of autonomous mobile robots to avoid multiple moving obstacles based on prediction of their future trajectories. J. Robotics Society Japan **15**(2), 249–260 (1997)
13. Minura, J., Uozumi, H., Shirai, Y.: Mobile robot motion planning considering the motion uncertainty of moving obstacles. In: Proceedings of IEEE Int. Conf. on Systems, Man, and Cybernetics, vol. 4, pp. 692–697 (1999)
14. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. In: IEEE Robotics and Automation Magazine, vol. 4, no. 1, pp. 23–33 (1997)
15. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion. MIT Press (2005)
16. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
17. Fulgenzi, C., Spalanzani, A., Laugier, C.: Dynamic obstacle avoidance in uncertain environment combining PVOS and occupancy grid. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 1610–1616 (2007)
18. Yang, S.X., Meng, M.: An efficient neural network method for real-time motion planning with safety consideration. J. Robotics and Autonomous Systems. **32**, 115–128 (2000)
19. Al-Khatib, M., Saade, J.J.: An efficient data-driven fuzzy approach to the motion planning problem of a mobile robot. J. Fuzzy Sets Systems **134**, 65–82 (2003)
20. Gal, O., Shiller, Z., Rimon, E.: Efficient and safe online motion planning in dynamic environments. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 88–93. Japan (2009)
21. Fabiani, P., Gonzalez-Banos, H.H., Latombe, J.C., Lin, D.: Tracking an unpredictable target among occluding obstacles under localization uncertainties. J. Robotics Auton. Sys. **38**, 31–48 (2002)
22. Chunyu, J., Qu, Z., Pollak, E., Falash, M.: A new reactive target-tracking control with obstacle avoidance in a dynamic environment. In: American Control Conference, pp. 3872–3877. St. Louis, USA (2009)
23. Ge, S.S., Cui, Y.J.: Dynamic motion planning for mobile robot using potential method. Auton. Robot. **13**, 207–222 (2002)
24. Huang, L.: Velocity planning for a mobile robot to track a moving target—a potential field approach. Robot. Auton. Syst. **57**, 55–63 (2009)
25. LaValle, S.M., Gonzalez-Banos, H.H., Becker, C., Latombe, J.C.: Motion strategies for maintaining visibility of a moving target. In: Proceedings of IEEE Int'l Conf. on Robotics and Automation, vol. 1, pp. 731–736 (1997)
26. Kunwar, F., Wong, F., Ben Mrad, R., Benhabib, B.: Guidance-based online robot motion planning for the interception of mobile targets in dynamic environments. J. Intell. Robot. Syst. **47**(4), 341–360 (2006)
27. Jaradat, K., Garibeh, M.A., Mohammad, H., Feilat, E.A.: Dynamic motion planning for autonomous mobile robot using fuzzy potential field. In: Proceedings of Int. Symposium on Mechatronics and its Applications (ISMA09), pp. 1–6 (2009)
28. Masehian, E., Katebi, Y.: Robot motion planning in dynamic environments with moving obstacles and target. International Journal of Aerospace and Mechanical Engineering **1**(1), 20–251 (2007)
29. Pourboghrat, F.: Exponential stabilization of nonholonomic mobile robots. Comput. Electr. Eng. **28**(5), 349–359 (2002)
30. Patrick, H.L., Selzer, S.M., Warren, M.E.: Guidance laws for short-range tactical missiles. J. Guid. Control Dynam. **4**(2), 98–108 (1981)
31. Speyer, T.J., Kim, K., Tahk, M.: Passive homing missile guidance law based on new target maneuver models. J. Guid. Control. Dynam. **1**(13), 803–812 (1990)
32. Shamos, M.I.: Computational geometry. Ph.D. dissertation, Yale University (1978)
33. LaValle, S.M.: Rapidly-Exploring Random Trees: A New Tool for Path Planning. Technical Report, TR 98-11, Computer Science Department, Iowa State University, USA (1998)
34. Chattopadhyay, I., Ray, A.: Optimal path-planning under finite memory obstacle dynamics based on probabilistic finite state automata models. In: American Control Conference, pp. 2403–2408. St. Louis, USA (2009)