Reactive Path Planning in a Dynamic Environment

Fethi Belkhouche, *Member, IEEE*

***Abstract*—This paper deals with the problem of path planning in a dynamic environment, where the workspace is cluttered with unpredictably moving objects. The concept of the virtual plane is introduced and used to create reactive kinematic-based navigation laws. A virtual plane is an invertible transformation equivalent to the workspace, which is constructed by using a local observer. This results in important simplifications of the collision detection process. Based on the virtual plane, it is possible to determine the intervals of the linear velocity and the paths that lead to collisions with moving obstacles and then derive a dynamic window for the velocity and the orientation to navigate the robot safely. The speed of the robot and the orientation angle are controlled independently using simple collision cones and collision windows constructed from the virtual plane. The robot's path is controlled using kinematic-based navigation laws that depend on navigation parameters. These parameters are tuned in real time to adjust the path of the robot. Simulation is used to illustrate collision detection and path planning.**

***Index Terms*—Autonomous agents, motion control, path planning for multiple mobile robot systems.**

I. Introduction

THE ABILITY of wheeled mobile robots to navigate safely and avoid moving obstacles is necessary for many real-world applications. Path planning in dynamic environments is still among the most difficult and important problems in mobile robotics. The literature that deals with this problem is rapidly growing.

The work of Erdmann and Lozano-Perez [1], Reif and Sharir [2], Kant and Zucker [3], and Fujimura and Samet [4] are among the earliest attempts to solve the problem of path planning in dynamic environments. Various classical approaches designed originally for static environments are extended to dynamic environments. For example, deterministic and probabilistic roadmap methods are used in [5]–[8]. A dynamic potential field method is suggested in [9] and [10]. A dynamic window approach is used in [11]. These methods perform well in static environments. However, this does not automatically imply good performance in dynamic environments. This is due to the major differences in collision detection and the necessity to control the speed of the robot as well. Additionally, global approaches, such as roadmap and potential field methods, have limited performance when obstacles are allowed to move in the workspace.

Other heuristic and probabilistic methods have been developed. The path-velocity decomposition is among the most popular heuristic methods used to navigate in dynamic environments. In this method, the motion-planning problem is addressed in two steps: 1) A geometric path is derived from a static environment, and 2) the linear velocity along this path is calculated to avoid moving obstacles. The main drawback of this method is when an object stops on the planned path. Another approach that is restricted to the velocity control based on predictive control is found in [12]. The method suffers from the same drawbacks as the path-velocity method. A deviation to adjacent paths is suggested when an obstacle blocks the robot's path [13].

There have been a few attempts to apply kinematic methods to solve the problem of collision detection [14], [15]. An approach for navigation and collision detection based on the kinematic equations is introduced in [14]. This approach uses the notion of collision cones (CCs). The collision conditions are established based on the range and the line-of-sight rates. The notion of indirect (weak) collision course is introduced and compared with the direct collision course in [15] and [16] by using the line-of-sight rate equations.

The space–time formulation is also used [4], [17]. In this formulation, the time dimension is included in the space configuration, which may result in additional algorithmic complexity. Algorithmic complexity can be a serious issue for navigation in dynamic environments. This problem is investigated by many authors [2].

The concept of velocity obstacles is introduced in [18]. This concept takes the velocity of the moving obstacles into account, which results in a shift of the CC. Collision avoidance is achieved by selecting the robot's velocities outside the velocity obstacles. The method is restricted to obstacles with linear motion. The nonlinear velocity obstacle approach [19] is an extended version of the velocity obstacle developed to cope with obstacles moving along arbitrary trajectories. The nonlinear velocity obstacle can be generated as a time integral of the colliding velocities. A reciprocal velocity obstacle is used for cooperative navigation in [20].

Probabilistic approaches are also used for path planning in dynamic environments [6], [21], [22]. These methods take uncertainties into account. A probabilistic roadmap method is combined with the notion of state–time in [6], where the

Manuscript received September 1, 2008; revised February 10, 2009. First published June 2, 2009; current version published July 31, 2009. This paper was recommended for publication by Associate Editor F. Thomas and Editor J.-P. Laumond upon evaluation of the reviewers' comments.

The author is with the Systems Engineering Program, Texas A&M International University, Laredo, TX 78041 USA (e-mail: fbelkhouche@tamiu.edu).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org, provided by the author. This material includes avi animation files that are shown to illustrate the collision-avoidance process (avi files can be opened using Windows Media Player). The scenario consists of three mobile objects moving in the coverage area of the mobile robot. The initial positions are $(-24, -24)$, $(-20, -10)$, $(-30, 10)$ for Di, and $(0, 0)$ for the robot. The speed is 2 m/s. The names of the avi files are given as follows. A.1: real_plane_speed_control_1.avi, A.2: virtual_plane_robot_transformed_speed_control_1.avi, B.1: real_plane_speed_control_2.avi, B.2: virtual_plane_robot_transformed_speed_control_2.avi, C.1: real_plane_orientation_control.avi, C.2: virtual_plane_robot_transformed_orientation_control.avi, C.3: virtual_plane_obstacles_transformed_orientation_control.avi. The names indicate the scenario. There is a correspondence in the letters, e.g., A.2 is the virtual plane corresponding to A.1. The environment required is Microsoft Windows operating system, and the run instructions include double click on the file. The size is between 210 and 345 KB. Contact fbelkhouche@tamiu.edu for further question about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2009.2022441

kinodynamic constraints are taken into account. A probabilistic approach for the velocity obstacles is suggested in [21].

Sensor-based path-planning methods are widely used to navigate in dynamic environments [23]–[27]. The robot calculates and estimates the motion of the moving obstacles based on its sensory system. Motion detection and probabilistic motion modeling are combined with smooth navigation functions in [23]. A detailed study about perceiving the presence of dynamic objects in the robot's neighborhood is discussed in [27]. Ultrasonic sensors are used with a CCD camera to predict the obstacle position in [25]. However, navigation in dynamic environments also requires the knowledge of the obstacle's velocity. A velocity predictor for future environment configuration by fusing multisensors is suggested in [28]. Two dynamic systems are combined together in [29]: The active robot system and the environment system. This allows the creation of a dynamic poisition/force control.

The problem of multirobot path planning is an important related problem. This requires a dynamic path-planning algorithm and coordination between robots. This problem is widely studied from different points of views, such as navigation, cooperation, and communication (see [30]–[33] for example). The use of coordination graphs for multiagent path planning is discussed in [34].

Path optimality is discussed in [35]–[38]. In path planning, optimality has a local aspect. This is a difficult problem, even in static environments. This led many authors to simplify the problem by making assumptions and restrictions on the path of the obstacles, such as fixed direction and constant speed.

In this paper, we suggest to solve the problem of path planning in dynamic environments by reducing arbitrary moving objects to stationary objects. The solution combines linear navigation laws with the notion of the virtual plane and velocity, as well as orientation windows. The virtual plane is a representation equivalent to the workspace, which is obtained by a transformation of observers. The method is reactive in the sense that only obstacles within a given radius are considered, and the path can be changed on the fly when a collision course is detected. The advantages of the suggested method can be summarized as follows.

1) While the problem of path planning in dynamic environments is difficult and complex, the suggested method is quite simple and highly effective.
2) The method is derived directly from the relative equations of motion of the robot and the dynamic obstacles.
3) The notion of the virtual plane allows finding the appropriate windows for the speed and orientation to move the robot in a collision-free path. The problem of navigation in a dynamic environment is reduced to a simple navigation problem in a static environment. Most classical methods used to navigate in static environments can be combined with the notion of virtual plane to navigate a wheeled mobile robot in a dynamic environment.
4) The method is reactive since the kinematic-based navigation laws used to navigate are parameter dependent. This property allows a change of the path in real time by tuning the parameters.

The suggested method belongs to a family of methods that use the relative velocity (similar to the velocity obstacles). These methods take the velocity of the obstacles into account. They are reactive, and thus, they present important advantages compared to global path-planning methods, such as the ability to change the path on the fly. Their simplicity give them an important advantage over the space–time formulation methods, where the search for a solution may be time consuming. Within the family of methods that use the relative velocity, the suggested method has the particularity of using the notion of the virtual plane, which represents a powerful and simple transformation integrable with various path-planning algorithms. While the velocity obstacles method uses a heuristic search to obtain the path solution, the suggested method achieves simple deviation using parameter-dependent linear navigation laws. The introduction of the notion of the virtual plane can benefit various applications in intelligent transportation and air-traffic management.

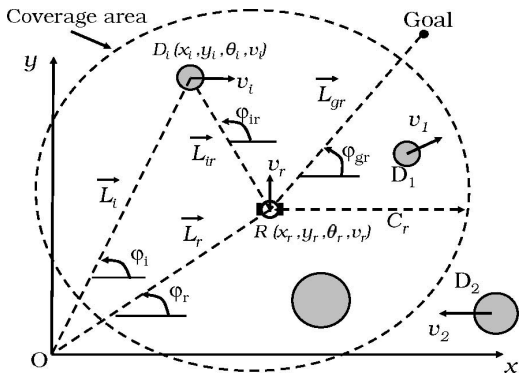

Fig. 1. Geometry of the navigation problem. Illustration of the kinematic and geometric variables.

The rest of the paper is organized as follows. In the next section, we introduce the problem. In Section III, we discuss the kinematic equations and the geometry of the path-planning problem. The concept of virtual plane is introduced and discussed in Section IV. The path-planning problem is discussed in Section V. Simulation is shown in Section VI.

II. Position of the Problem

Our objective is to develop algorithms for collision detection and control laws for the speed and the orientation to navigate the robot safely in a dynamic environment. Fig. 1 illustrates the navigation problem. The following definitions/assumptions are made.

1) The world is attached to a global fixed reference frame of coordinates $\{W\}$. Its origin is point O. It is possible to attach local reference frames to every moving object in the working space. We are particularly interested in the robot and the objects moving in its coverage area (CA).
2) The robot under consideration is a wheeled mobile robot with a circular shape denoted by R. Its radius is d_r, and its reference point situated at the center is $O_r(t)$. This point has coordinates (x_r, y_r) in $\{W\}$. The initial position of the robot is given by $R(t_0) = (x_{r0}, y_{r0})$.
3) The robot's final goal is a point G with coordinates (x_g, y_g) in $\{W\}$. The coordinates of this point are known to the robot $\forall t$.

4) The ith moving object is denoted by D_i, which is a circular object with radius d_i that moves along an arbitrary trajectory. No assumption is made on the motion of D_i. The coordinates of the reference point of D_i are (x_i, y_i). The initial position of D_i is $D_i(t_0) = (x_{i0}, y_{i0})$.
5) The robot has an omnidirectional sensory system that allows it to measure in real time the position, the linear velocity, and the orientation angle of the objects moving within a given range that defines the CA. The CA of the robot's sensory system is characterized by a radius C_r, which is centered at the robot's reference point. Only objects in the CA are considered.

Multiagent path planning and collaborative navigation are related problems that can be solved using the suggested method. They represent a natural extension of this study; however, they are beyond the scope of this paper.

III. Kinematic Equations and Geometry

In this section, we recall important geometric and kinematic variables, and we derive the relative kinematic equations. Our approach uses the relative motion in polar coordinates. Polar coordinates have been used by many authors for controlling mobile robots [39]–[41]. For more details concerning the derivation of the equations, see [41]. The robot under consideration is a wheeled mobile robot that moves according to the following equations:

$$\begin{aligned} \dot{x}_r &= v_r \cos\theta_r \\ \dot{y}_r &= v_r \sin\theta_r \\ \dot{v}_r &= a_r \\ \dot{\theta}_r &= \omega_r \end{aligned} \tag{1}$$

where θ_r is the robot's orientation angle, v_r and ω_r are the linear and the angular velocities, respectively, and a_r is the robot's linear acceleration. The control inputs in our approach are (θ_r, v_r) instead of (ω_r, a_r). Clearly, this is possible since $\omega_r = \dot{\theta}_r$ and $a_r = \dot{v}_r$. The variables v_r and ω_r are characterized by their maximum values $v_r^{\max}$ and $\omega_r^{\max}$, respectively. Let $a_a^{\max}$ be the maximum acceleration exertable by the motors and $a_d^{\max}$ be the maximum breakage deceleration. The state of the robot is characterized by $R : S_r = (x_r, y_r, \theta_r, v_r)$. The kinematic equations of D_i are given by

$$\begin{aligned} \dot{x}_i &= v_i \cos\theta_i \\ \dot{y}_i &= v_i \sin\theta_i \\ \dot{\theta}_i &= \omega_i. \end{aligned} \tag{2}$$

D_i has a linear velocity v_i and an orientation angle θ_i, and ω_i is its angular velocity. Its state is characterized by $D_i : S_{Di} = (x_i, y_i, \theta_i, v_i)$. Without loss of generality, the robot is reduced to its reference point while D_i is augmented in the size by the robot's radius. In the rest of the paper, R refers to the robot reduced to its reference point, and D_i refers to the augmented object. The velocity vectors for the robot and D_i are given by $\vec{v}_r$ and $\vec{v}_i$, respectively. With reference to Fig. 1, we define the following geometric quantities.

1) Quantities related to D_i.
   a) Line of sight of D_i ($\vec{L}_i$) is the imaginary straight line that starts from O and is directed toward the reference point of D_i.
   b) The line of sight angle of D_i (φ_i) is the angle made by $\vec{L}_i$.
   c) The range between O and D_i is denoted by l_i.
2) Quantities related to the robot.
   a) Line of sight of the robot ($\vec{L}_r$) is the imaginary straight line that starts from O and is directed toward the reference point of R.
   b) The line-of-sight angle of R (φ_r) is the angle made by $\vec{L}_r$.
   c) The range between O and R is denoted by l_r.
3) Relative quantities $R - D_i$.
   a) Relative line of sight $R - D_i$ ($\vec{L}_{ir}$) is the imaginary straight line that starts from the reference point of the robot and is directed toward the reference point of D_i.
   b) The line-of-sight angle $R - D_i$ (φ_{ir}) is the angle made by the line of sight $\vec{L}_{ir}$ and is given by

$$\tan\varphi_{ir} = \frac{y_i - y_r}{x_i - x_r}. \tag{3}$$

   c) The Euclidean distance between the robot and D_i (l_{ir}) is given by

$$l_{ir} = \sqrt{(y_i - y_r)^2 + (x_i - x_r)^2}. \tag{4}$$

4) Relative quantities $R - G$.
   a) Line of sight $R - G$ ($\vec{L}_{gr}$) is the imaginary straight line that starts from the reference point of the robot and is directed toward G.
   b) The line-of- sight angle $R - G$ (φ_{gr}) is the angle made by $\vec{L}_{gr}$, and is given by

$$\tan\varphi_{gr} = \frac{y_g - y_r}{x_g - x_r}. \tag{5}$$

Clearly, the relationship between the polar and Cartesian coordinate systems is given by

$$x = l\cos\varphi, \qquad y = l\sin\varphi \tag{6}$$

where l is the radial variable, and φ is the angular variable. By considering the kinematic models and the change of variables, we obtain the following equivalent kinematic equations:

$$\begin{aligned} \dot{l}_r &= v_r \cos(\theta_r - \varphi_r) \\ l_r\dot{\varphi}_r &= v_r \sin(\theta_r - \varphi_r) \end{aligned} \tag{7}$$

and

$$\begin{aligned} \dot{l}_i &= v_i \cos(\theta_i - \varphi_i) \\ l_i\dot{\varphi}_i &= v_i \sin(\theta_i - \varphi_i) \end{aligned} \tag{8}$$

for the robot and D_i, respectively. We consider the velocity of the robot with respect to D_i, i.e., $\vec{v}_{ri} = \vec{v}_r - \vec{v}_i$, and the velocity of D_i with respect to the robot, i.e., $\vec{v}_{ir} = \vec{v}_i - \vec{v}_r$.

The relative velocities are resolved in the polar coordinate system as follows:

$$\vec{v}_{ri} = \dot{l}_{ri}\vec{u}_{l_{ri}} + l_{ri}\dot{\varphi}_{ri}\vec{u}_{\varphi_{ri}} \tag{9}$$

$$\vec{v}_{ir} = \dot{l}_{ir}\vec{u}_{l_{ir}} + l_{ir}\dot{\varphi}_{ir}\vec{u}_{\varphi_{ir}} \tag{10}$$

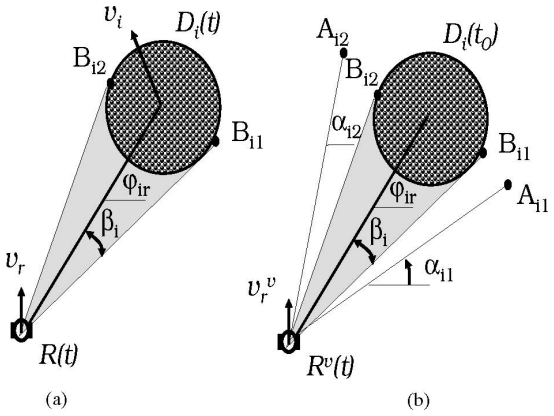

Fig. 2. CCs. (a) CC in the real plane between $R(t)$ and $D_i(t)$. (b) CC in the virtual plane between $R^v(t)$ and $D_i(t_0)$.

where $\vec{u}_{l_{ri}}, \vec{u}_{l_{ir}}$ and $\vec{u}_{\varphi_{ri}}, \vec{u}_{\varphi_{ir}}$ are the unit vectors along and across the line of sight, respectively. Clearly, $\vec{v}_{ri} = -\vec{v}_{ir}$, $l_{ri} = l_{ir}$, $\varphi_{ri} = \varphi_{ir} + \pi$, $\vec{u}_{l_{ri}} = -\vec{u}_{l_{ir}}$, and $\vec{u}_{\varphi_{ri}} = -\vec{u}_{\varphi_{ir}}$.

It is possible to prove that the tangential and normal components of the relative velocity are given by

$$\begin{aligned} \dot{l}_{ir} &= v_i \cos(\theta_i - \varphi_{ir}) - v_r \cos(\theta_r - \varphi_{ir}) \\ l_{ir}\dot{\varphi}_{ir} &= v_i \sin(\theta_i - \varphi_{ir}) - v_r \sin(\theta_r - \varphi_{ir}). \end{aligned} \tag{11}$$

The velocity components in (11) are valid for both $\vec{v}_{ri}$ and $\vec{v}_{ir}$ (i.e., $\dot{l}_{ir} = \dot{l}_{ri}$ and $l_{ir}\dot{\varphi}_{ir} = l_{ri}\dot{\varphi}_{ri}$). The first equation in (11) gives the evolution of the range. A negative sign of $\dot{l}_{ir}$ indicates that the robot is approaching from D_i. A zero rate for the range implies constant distance between R and D_i. The second equation in (11) gives the rate of turn of D_i as seen by R. A zero rate for the line-of-sight angle means that the motion of D_i as seen by the robot is a straight line. This system presents a nice and simple model that allows real-time representation of the relative motion between R and D_i.

By considering the relative motion between R and D_i, the states of the robot and the moving object are characterized by the relative range and the line-of-sight angle, i.e., $R : S_r = (\varphi_{ir}, r_{ir}, \theta_r, v_r)$, and $D_i : S_{Di} = (\varphi_{ir}, r_{ir}, \theta_i, v_i)$, respectively. Under this formulation, there exist two common variables (φ_{ir}, r_{ir}) between the state of R and the state of D_i.

The CC is a natural and simple approach that allows to avoid collision with a stationary obstacle. Usually, a CC is constructed by using sensor beams. The sensory system returns the upper and lower tangent limit points B_{i1} and B_{i2} in D_i, as shown in Fig. 2(a). The angle between the lines of sight made by these points is $2\beta_i$. The CC with D_i at a given time is characterized by

$$\text{CC}_i = [\varphi_{ir} - \beta_i, \varphi_{ir} + \beta_i]. \tag{12}$$

When D_i is allowed to move, the simple CC is not an indicator of collision. Therefore, it cannot be used for collision detection with moving objects. The solution suggested in this paper is to create an equivalent scenario where D_i is transformed into a stationary object. This is discussed in the next section.

IV. Transformation of Observer: The Virtual Plane

When navigating, the robot continuously checks for collisions with other moving objects. It is possible to use the range in the kinematic model given in (11) for collision detection. However, a decreasing range does not necessarily imply collision. In the model given in (11), both R and D_i are moving objects. Here, we suggest a model that transforms a moving object of interest into a stationary object. This is achieved by considering a local observer. Thus, the collision course between R and D_i (both moving) is reduced to a collision course between $D_i\,(t_0) : S^0_{Di} = (x_{i0}, y_{i0}, n/d, 0)$ and $R^v : S^v_r = (x^v_r, y^v_r, \theta^v_r, v^v_r)$, where R^v is a virtual moving robot that replaces R, and $D_i\,(t_0)$ is the initial position of D_i. The linear velocity and orientation angle of the virtual robot are given by v^v_r and θ^v_r, respectively. The coordinates of its reference point are given by (x^v_r, y^v_r). The path of R^v is different from the path of R. This formulation allows to construct another CC between R^v and $D_i\,(t_0)$. We call it "the CC in the virtual plane (CCVP)." Note that φ_{ir} and l_{ir} are invariant under the virtual plane transformation. Since the size of D_i does not change, β_i is also invariant under the transformation. This implies that the width and the direction of the CC in the real plane and the virtual plane are the same. The CCVP is shown in Fig. 2(b).

The components of the relative velocity between R^v and $D_i\,(t_0)$ along and across $\vec{L}_{ir}$ are given by

$$\begin{aligned} \dot{l}_{ir} &= -v^v_r \cos(\theta^v_r - \varphi_{ir}) \\ l_{ir}\dot{\varphi}_{ir} &= -v^v_r \sin(\theta^v_r - \varphi_{ir}). \end{aligned} \tag{13}$$

The linear velocity and orientation angle of R^v can be written as follows:

$$\begin{aligned} v^v_r &= \sqrt{(\dot{x}_i - \dot{x}_r)^2 + (\dot{y}_i - \dot{y}_r)^2} \\ \tan\theta^v_r &= \frac{\dot{y}_i - \dot{y}_r}{\dot{x}_i - \dot{x}_r}. \end{aligned} \tag{14}$$

The kinematic models given in (11) and (13) are equivalent. This is stated in the following result.

Proposition 1: The kinematic models given in (11) and (13) represent the same scenario.

Proof: Systems (11) and (13) represent the components of relative velocities along and across $\vec{L}_{ir}$. However, they are derived in different ways as follows.

1) System (11) is obtained by using vector subtraction and then resolution into components along and across $\vec{L}_{ir}$.
2) System (13) is obtained by using resolution into components along and across $\vec{L}_{ir}$ and then vector subtraction. ■

It is worth noting that the virtual plane as defined here is not unique; it is possible to define other forms for the virtual plane that can be used in specific applications.

The orientation angle and the linear velocity of the virtual robot are functions of the orientation angles of the robot and the obstacle (θ_r, θ_i), as well as their linear velocities (v_r, v_i), i.e.,

$$\begin{aligned} v^v_r &= f(\theta_r, \theta_i, v_r, v_i) \\ \theta^v_r &= g(\theta_r, \theta_i, v_r, v_i). \end{aligned} \tag{15}$$

Therefore, any sudden change in the motion of the moving object or the robot is captured in the virtual plane.

A. Case of Multiple (n) Moving Objects: The Collision Window

It is natural to check collision between candidate pairs only. For n moving objects in the CA, the relative motion is described by n systems of kinematic equations. To each moving object D_i in the CA, we associate a virtual robot R^{iv}. The state of R^{iv} is given by $R^{iv} : S_r^{iv} = (\varphi_{ir}, r_{ir}, \theta_r^{iv}, v_r^{iv})$. The formulae for θ_r^{iv} and v_r^{iv} are the same as those for θ_r^v and v_r^v given in (14). All obstacles and their corresponding virtual robots are shown in the same virtual plane. It is important to have a global view by combining the CCs (each corresponding to a virtual robot) to construct the θ_r^{iv}-collision-window: θ_r^{iv}-window $= \bigcup_{i=1}^{n} \text{CCVP}_i$. The collision window is an angular representation that combines a set of CCs. It shows the free directions and the directions corresponding to the obstacles.

B. Moving Object as an Observer

Previously, we reduced the moving obstacle into a stationary obstacle. It is also possible to reduce the collision course between $R(t)$ and $D_i(t)$ to a collision course between a virtual moving object $D_i^v(t) : S_{Di}^v = (l_{ir}, \varphi_{ir}, \theta_i^v, v_i^v)$ and a stationary robot $R(t_0) : S_r^0 = (l_{ir}, \varphi_{ir}, n/d, 0)$. Here, $R(t_0)$ represents the initial position of the robot.

The components of the relative velocity along and across $\vec{L}_{ir}$ are given by

$$\begin{aligned} \dot{l}_{ir} &= -v_i^v \cos(\theta_i^v - \varphi_{ir}) \\ l_{ir}\dot{\varphi}_{ir} &= -v_i^v \sin(\theta_i^v - \varphi_{ir}). \end{aligned} \tag{16}$$

In the presence of n moving objects in the CA, the virtual robot is one stationary robot, i.e., $R(t_0)$. However, there exist n virtual objects $D_i^v, i = 1, \ldots, n$, which are shown in the same virtual plane. Note that the orientation angle and the linear velocity of each and every D_i^v depend on the orientation angle and the linear velocity of the real robot. Collision in the real plane can be avoided by controlling D_i^v to avoid $R(t_0)$.

C. Real Plane From the Virtual Plane

Collision detection is done in the virtual plane. However, the goal is to control the robot in the real plane. The orientation angle of the robot that allows to avoid collision is calculated by using θ_r^v. Clearly, we can write

$$\begin{aligned} \dot{x}_r &= \dot{x}_r^v + \dot{x}_i \\ \dot{y}_r &= \dot{y}_r^v + \dot{y}_i. \end{aligned} \tag{17}$$

This is the inverse transformation that maps the virtual plane into the real plane. It gives the velocity of the robot as a function of the velocities of the virtual robot and the moving object. The speed and orientation of the real robot can be deduced from the virtual robot and the moving object velocities as follows:

$$v_r = \sqrt{(\dot{x}_r^v + \dot{x}_i)^2 + (\dot{y}_r^v + \dot{y}_i)^2} \tag{18}$$

and

$$\tan\theta_r = \frac{\dot{y}_r^v + \dot{y}_i}{\dot{x}_r^v + \dot{x}_i}. \tag{19}$$

These equations will be used in the navigation process.

D. CCVP Transformed to Robot Speed and CCVP Transformed to Robot Orientation

The CCVP is characterized by a maximum angle $\theta_{r2}^v = \varphi_{ir} + \beta_i$, and a minimum angle $\theta_{r1}^v = \varphi_{ir} - \beta_i$. An orientation angle between these two values results in a collision. The CCVP transformed to robot speed (CCVPTV) is a transformation from the virtual robot's orientation space to the robot's linear velocity space. The CCVPTV is constructed by using (18), which gives v_r as a function of the velocities of the virtual robot and the moving object. The CCVPTV gives the interval of v_r corresponding to collision. The values of v_r corresponding to θ_{r1}^v and θ_{r2}^v are given by

$$\begin{aligned} v_{r1} &= \sqrt{(\dot{x}_i + v_r^v \cos\theta_{r1}^v)^2 + (\dot{y}_i + v_r^v \sin\theta_{r1}^v)^2} \\ v_{r2} &= \sqrt{(\dot{x}_i + v_r^v \cos\theta_{r2}^v)^2 + (\dot{y}_i + v_r^v \sin\theta_{r2}^v)^2}. \end{aligned} \tag{20}$$

In a similar way, it is possible to transform the virtual robot's orientation space to the robot's orientation space by using (19), which gives θ_r as a function of the velocities of the virtual robot and the moving object. This is called the CCVP transformed to robot orientation (CCVPTO). The CCVPTO represents the interval for the robot's orientation angle that results in a collision. The values of θ_r corresponding to θ_{r1}^v and θ_{r2}^v are given by

$$\begin{aligned} \tan\theta_{r1} &= \frac{\dot{y}_i + v_r^v \sin\theta_{r1}^v}{\dot{x}_i + v_r^v \cos\theta_{r1}^v} \\ \tan\theta_{r2} &= \frac{\dot{y}_i + v_r^v \sin\theta_{r2}^v}{\dot{x}_i + v_r^v \cos\theta_{r2}^v}. \end{aligned} \tag{21}$$

The CCVPTV and the CCVPTO are illustrated in Example 3 in Section VI and shown in Fig. 8.

V. Navigation Process

A. Linear Navigation Laws

Kinematic-based linear navigation laws are used to navigate the robot toward the final goal. These laws were introduced recently to generate smooth curved paths for robot path planning [42]. They depend on the navigation parameters, which are real numbers used to change the robot's path in real time. A linear navigation law is given by

$$\theta_r = M\gamma_{gr} + c_1 + c_0\, e^{-at} \tag{22}$$

where γ_{gr} is the line-of-sight angle of robot's final goal [as given by (5)]. The first term in (22) is a proportionality term. The variables c_1 and c_0 are deviation terms, where c_1 characterizes the final desired orientation angle of the robot, and c_0 characterizes the initial orientation of the robot. The exponential term $c_0\, e^{-at}$ is used for heading regulation. Term M is a navigation parameter with $M > 1$, and a is a given positive gain. Collision avoidance is achieved by tuning the value of M.

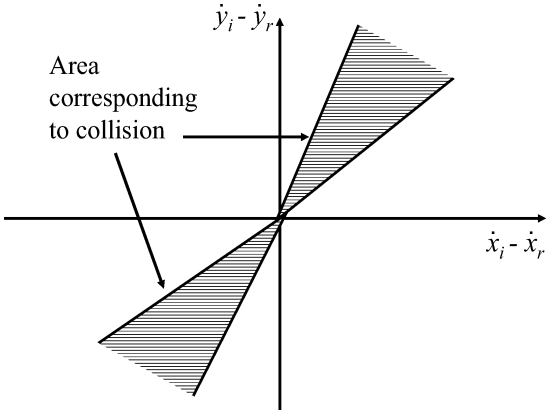

Fig. 3. CC in terms of the relative velocities.

B. Collision Course Characterization

An important property of the transformation resulting in the virtual plane is that l_{ir}, φ_{ij}, and β_i are invariant under the transformation. Thus, the CC in the virtual plane can be written as $\text{CCVP}_i = [\varphi_{ir} - \beta_i, \varphi_{ir} + \beta_i]$. Collision course in the virtual plane with $D_i(t_0)$ is characterized by

$$\theta_r^v \in \text{CCVP}_i. \tag{23}$$

In certain situations, (23) is not satisfied initially, but may become satisfied after a certain time; this is referred to as the indirect collision course, where $\theta_r^v \notin \text{CCVP}_i$, but $\theta_r \to \varphi_{ir}$. In the virtual plane, the characterization and detection of the direct collision course is straightforward. We are particularly interested in translating this to the real plane. This is stated as follows.

Proposition 2: The direct collision course between R and D_i is characterized by

$$\frac{\dot{y}_i - \dot{y}_r}{\dot{x}_i - \dot{x}_r} \in [\tan(\varphi_{ir} - \beta_i), \tan(\varphi_{ir} + \beta_i)]. \tag{24}$$

Proof: Clearly, collision with $D_i(t_0)$ in the virtual plane is characterized by $\theta_r^v \in \text{CCVP}_i$. Using the equivalence between the real and the virtual planes, we obtain (24). ■

Fig. 3 shows a representation of the collision course in terms of the relative velocities. The shaded area corresponds to collision. The previous result states that in the case of a direct collision course, the line-of-sight angle varies but within a given interval centered at φ_i. Unlike the linear velocity of R, the linear velocity of R^v has no effect on collision and, thus, cannot be used for collision avoidance. However, it can be used to find or approximate the collision time. Changing the orientation angle of the virtual robot results also in a change in the linear velocity of R^v. Collision avoidance with $D_i(t_0)$ is achieved by controlling the orientation angle of R^v so that $\theta_r^v \notin \text{CCVP}_i$. The variable θ_r^v is controlled by v_r and/or θ_r. It is possible to write the expressions for v_r and θ_r as follows:

$$v_r = \frac{v_i(\tan\theta_r^v \cos\theta_i - \sin\theta_i)}{\tan\theta_r^v \cos\theta_r - \sin\theta_r} \tag{25}$$

$$\theta_r = \theta_r^v - \arcsin\left(\frac{v_i \sin(\theta_r^v - \theta_i)}{v_r}\right). \tag{26}$$

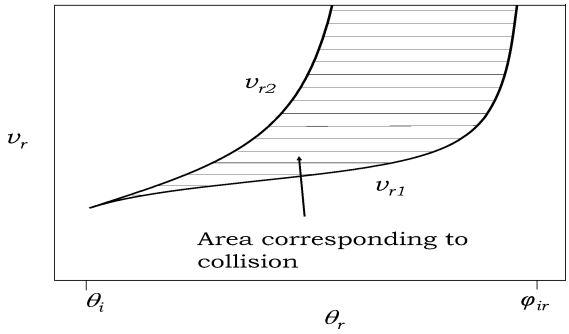

Fig. 4. Profile (θ_r, v_r) that results in a collision.

The combination between (23) and (25) or (23) and (26) allows the construction of some kind of CC for the robot's linear velocity and orientation angle. The CC is represented in the (θ_r, v_r) plane in Fig. 4.

C. Collision Avoidance With a Single Obstacle: Robot Transformed

The collision avoidance is accomplished by deviation in the virtual plane, which corresponds to the robot changing its speed or orientation angle in the real plane.

With reference to Fig. 2(b), A_{i1} and A_{i2} are the deviation points in the virtual plane. The line-of-sight angles from these points are given by α_{i1} and α_{i2}, respectively. For simplicity, the virtual robot applies a pursuit to deviate toward point A_{i1} or A_{i2}. Both right and left deviations are possible. Let t_1 be the time when the robot starts deviation.

1) Right detour: The robot applies a deviated pursuit to reach point A_{i1}, i.e.,

$$\theta_r^{v*}(t) = \alpha_{i1} + c_1 + c_0 \exp(-a(t - t_1)). \tag{27}$$

2) Left detour: The robot applies a deviated pursuit to reach point A_{i2}, i.e.,

$$\theta_r^{v*}(t) = \alpha_{i2} + c_1 + c_0 \exp(-a(t - t_1)). \tag{28}$$

Here, $\theta_r^{v*}(t)$ denotes the desired value for θ_r^v that allows avoidance the obstacle. The last term in (27) and (28) is a deviation term used to keep the smoothness of the path, a is a given positive gain, and c_0 and c_1 are deviations parameters calculated on the fly so that

$$\theta_r^{v*}(t_1) = \alpha_{i1,i2}(t_1) + c_1 + c_0. \tag{29}$$

The desired value θ_r^{v*} is controlled by using v_r or θ_r as follows.

1) Collision Avoidance by Changing the Speed: Knowing the desired value of θ_r^{v*}, the corresponding desired value for v_r is given by

$$v_r^* = \frac{v_i(\tan\theta_r^{v*} \cos\theta_i - \sin\theta_i)}{\tan\theta_r^{v*} \cos\theta_r - \sin\theta_r}. \tag{30}$$

2) Collision Avoidance by Changing the Orientation: Knowing the desired value of θ_r^{v*}, the corresponding desired value for θ_r is given by

$$\tan\theta_r^* = \frac{v_r^v \sin(\theta_r^{v*}) + v_i \sin(\theta_i)}{v_r^v \cos(\theta_r^{v*}) + v_i \cos(\theta_i)}. \tag{31}$$

The desired solutions v_r^* and θ_r^* need to be reachable, which is discussed in Section V-E.

D. Collision Avoidance With a Single Obstacle: Obstacle Transformed

Here, collision avoidance is accomplished by deviating the virtual obstacle (not the real obstacle), i.e., changing the orientation angle θ_i^v of D_i^v, from which we can write for the desired values of the robot linear velocity and orientation angle as follows:

$$v_r^* = \frac{v_i(\tan\theta_i^{v*}\cos\theta_i - \sin\theta_i)}{\tan\theta_i^{v*}\cos\theta_i - \sin\theta_i} \tag{32}$$

and

$$\tan\theta_r^* = \frac{-v_i^v\sin(\theta_i^{v*}) + v_i\sin(\theta_i)}{-v_i^v\cos(\theta_i^{v*}) + v_i\cos(\theta_i)}. \tag{33}$$

The virtual obstacle moves toward a point A_i near $R(t_0)$. The distance between A_i and $R(t_0)$ is greater than $d_r + d_i$. A comparison between the robot-transformed approach and the obstacle-transformed approach is shown in Example 2 in Section VI.

E. Collision Avoidance With Multiple (n) Obstacles: Robot Transformed

In an environment that is highly cluttered with moving obstacles, avoiding collision with one obstacle may result in collision with another obstacle. In this case, we have to consider all moving objects in the CA, especially those for which $\dot{l}_{ir} < 0$. The CCs corresponding to each D_i are combined into a single collision window in the real plane. It is possible to use the speed collision window (v-window) by mapping the solution to the v_r space or the orientation collision window (θ-window) by mapping the solution to the θ_r space.

1) Collision Avoidance Using the v-Window: To every object D_i in the CA, we associate a virtual robot R^{iv} with orientation angle θ_r^{iv}. Every $D_i(t_0)$ is characterized by a CC CCVP$_i$. The CCVPs are mapped to the v_r space by using the CCVPTV, from which the (v-window) is constructed, i.e.,

$$v\text{-window} = \bigcup_{i=1}^{n} \text{CCVPTV}_i. \tag{34}$$

2) Collision Avoidance Using the θ-Window: The CCVPs are mapped to the θ_r space using the CCVPTO, from which the (θ-window) is constructed, i.e.,

$$\theta\text{-window} = \bigcup_{i=1}^{n} \text{CCVPTO}_i. \tag{35}$$

The appropriate values for the robot's speed or orientation angle are then chosen in the free directions in the v-window or the θ-window, respectively. The v-window and the θ-window are illustrated in Example 3 in Section VI and shown in Fig. 9 at two different times.

Let v_r^* and θ_r^* be the desired values chosen from the v-window and the θ-window, respectively. The control laws used to track the desired values are given by

$$\begin{aligned} \dot{v}_r &= -K_v(v_r - v_r^*) \\ \dot{\theta}_r &= -K_\theta(\theta_r - \theta_r^*) \end{aligned} \tag{36}$$

where K_v and K_θ are given positive gains. Let t_1 be the time when the robot starts maneuvering to avoid collision and t_f be the estimated time to collision. The θ-window and the v-window give the admissible solutions for the robot's orientation angle and linear velocity. These solutions need to be reachable as well. The reachable solutions are characterized by

$$v_r^* \in [v_r(t_1) - a_d^{\max}t_f, v_r(t_1) + a_a^{\max}t_f] \tag{37}$$

and

$$\theta_r^* \in [\theta_r(t_1) - \omega_r^{\max}t_f, \theta_r(t_1) + \omega_r^{\max}t_f]. \tag{38}$$

Algorithm

for For all D_i in CA **do**
  Calculate $\dot{l}_{ri}$
  if All $\dot{l}_{ri} > 0$ **then**
    There is no collision risk, keep the same motion.
  else
    Construct the virtual plane.
    Test the collision in the virtual plane using the CCVP.
    if there is a collision risk **then**
      Create the CCVPTV(s) or the CCVPTO(s).
      Construct the θ-window or the v-window.
      Chose the appropriate values for θ_r or v_r.
      Execute a reachable solution.
    end if
  end if
end for

F. Collision Avoidance With Multiple Obstacles: Obstacles Transformed

In this case, we have one virtual robot $R(t_0)$ and various moving objects D_i^v. Collision is avoided by moving D_i^v to a single deviation point near $R(t_0)$. The distance to the deviation point must be greater than the distance $d_i + d_r$ from the center of $R(t_0)$. The orientation angle of the virtual obstacles is a function of θ_r^* and v_r^*; therefore, it is controlled according to θ_r^* or v_r^*. For every obstacle D_i in the robot's CA we associate a virtual obstacle D_i^v. An inverse CC allows to determine the intervals of θ_i^v that will result in a collision. The θ-window or v-window is constructed by combining these intervals and deducing the corresponding values of θ_r and v_r. Note again that controlling D_i^v does not imply controlling D_i. We have no control over the real moving objects.

VI. Simulation

In this section, simulation results are presented. We show five different scenarios. The first scenario illustrates the collision-detection process with accelerating obstacles using the virtual

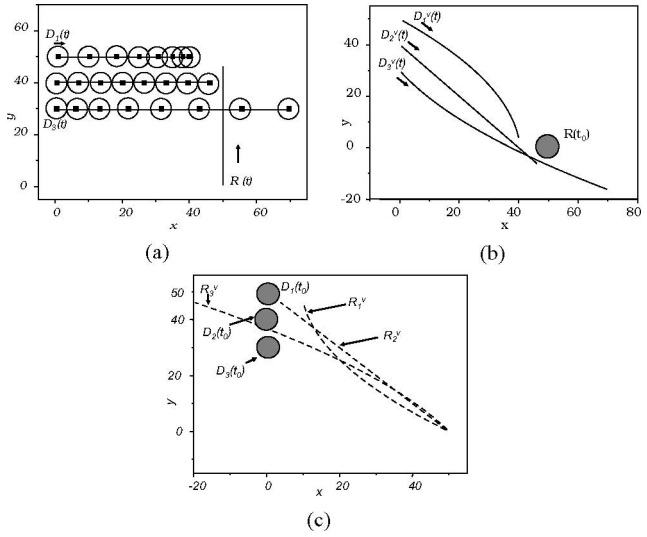

Fig. 5. Collision detection in the virtual plane. (a) Real plane R moving vertically, D_1, D_2, and D_3 moving horizontally, and D_1 and D_3 are accelerating. (b) Virtual plane with objects transformed. (c) Virtual plane with robot transformed.

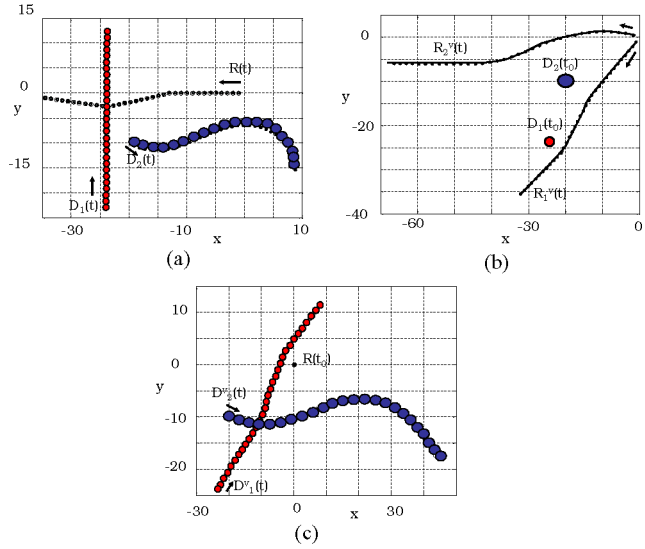

Fig. 6. Comparison between collision avoidance using the robot-transformed approach versus obstacle-transformed approach. (a) Real plane. (b) Virtual plane with robot transformed. (c) Virtual plane with obstacles transformed.

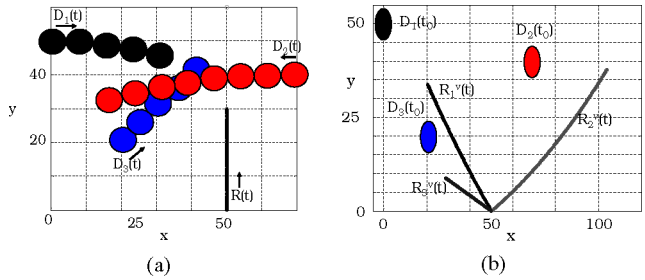

Fig. 7. (a) Robot and three moving objects in the real plane, $\theta_r = \pi/2, v_r = 2\,\mathrm{m/s}$. (b) Corresponding virtual plane.

planes. Note that collision detection with accelerating objects is a challenging problem. The second scenario shows a comparison between the obstacle-transformed and the robot-transformed approaches. The third scenario illustrates the CCVP and the transformation to the robot's velocity space or the robot's orientation space. The v-window and the θ-window are then constructed. In the fourth scenario, we show collision avoidance with moving obstacles using the virtual plane. The collision is avoided by changing the robot's path. The last scenario is similar, however, it illustrates both speed change and orientation change.

Example 1: This example is shown in Fig. 5, where some of the moving objects are accelerating (D_1 is decelerating and D_3 is accelerating). Our goal is to illustrate the collision-detection approach using the virtual planes that are shown in Fig. 5(b) (obstacles transformed) and (c) (robot transformed). While it is difficult to make a judgement on collision in the real plane, the virtual planes and their CCs give a good idea about the collision risk. From the figures, the equivalence between the real plane and the virtual plane is obvious.

Example 2: This example is shown in Fig. 6. It shows a comparison between the collision avoidance using the robot-transformed virtual plane [see Fig. 6(b)] and the obstacle-transformed virtual plane [see Fig. 6(c)]. The initial positions are $D_1(t_0) : (-24, -24)$, $D_2(t_0) : (-20, -10)$, and $R(t_0) : (0, 0)$. The real plane is shown in Fig. 6(a). The robot is in a collision course with $D_1(t)$, as it can be seen from the virtual planes, and thus it acts to avoid collision. In Fig. 6(b), R_1^v and R_2^v are created, and R_1^v changes path to avoid collision. In a similar way, in Fig. 6(c), $D_1^v(t)$ and $D_2^v(t)$ are created, and a collision with $R(t_0)$ is avoided by deviating $D_1^v(t)$.

Example 3: The following example shows the evolution of the orientation angles and the CCs in the virtual plane, and their transformation to construct the v-window and the θ-window. The motion in the real plane is shown in Fig. 7(a). The virtual plane (robot transformed) is shown in Fig. 7(b). The orientation angles of (R_1^v, R_2^v, R_3^v) as well as the corresponding CCs (CCVPs), are shown in Fig. 8(a), (c), and (e). These figures give good prediction of the possibility of collision. The CCs in the virtual plane are transformed to construct the CCVPTV and the CCVPTO, as shown in Fig. 8(b), (d), and (f), from which the v-window and the θ-window are constructed, respectively. From the CCVPs, there is no collision risk with D_2 or D_3 (since θ_r^{1v} and θ_r^{2v} are outside their corresponding CCVPs). However, the robot is in a collision course with D_1 in the time interval $0 \le t \le 7\,\mathrm{s}$. After $t = 7\,\mathrm{s}$, the collision risk disappears. This fact is captured in the CCVPTO and the CCVPTV, as shown in Fig. 8(b). Fig. 9 shows these windows at times $t = 5\,\mathrm{s}$ and $t = 10\,\mathrm{s}$. Any value of θ_r or v_r outside the shaded rectangles is safe for navigation.

Example 4: This example illustrates the collision avoidance process. The scenario shown in Fig. 10(a) has some similarity with the scenario of Example 1. As seen from the virtual plane shown in Fig. 10(b), there is a collision risk with D_3 in the time interval $0 \le t \le 5\,\mathrm{s}$. The robot changes its path by deviating to the left, as shown in Fig. 10(a), and its corresponding virtual plane (see Fig. 10c). The complete virtual plane is shown in Fig. 10(c).

Example 5: This scenario illustrates both speed change and orientation change. The robot moves toward its goal $(-10, 50)$

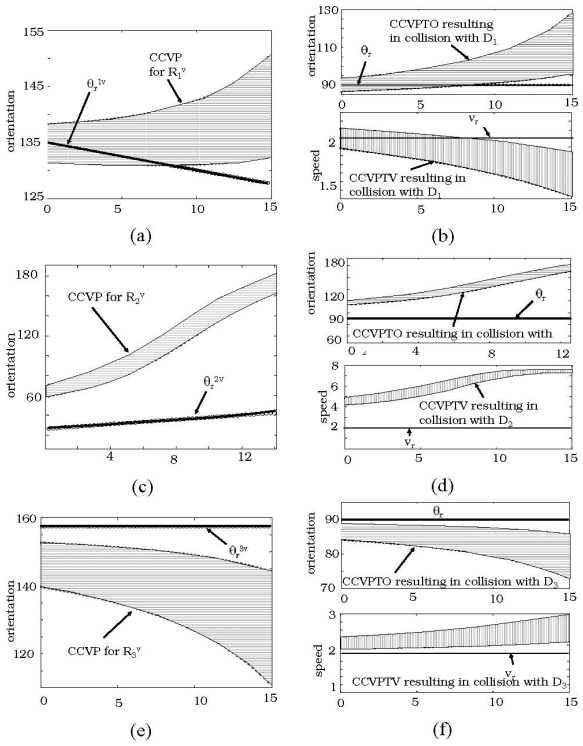

Fig. 8. (a), (c), and (e) Construction of the CCVPs. (b), (d), and (f) Their velocity and orientation transformations. The actual values of θ_r and v_r are shown in the figures.

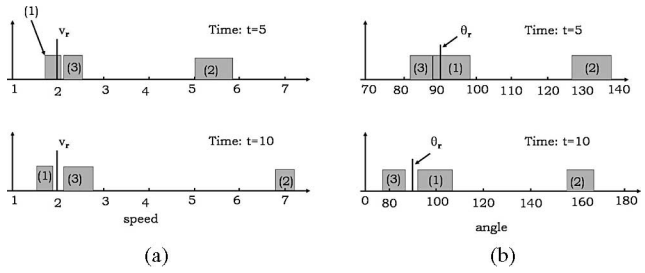

Fig. 9. v-window and the θ-window at times $t = 5$ s and $t = 10$ s. (1) Corresponds to D_1, (2) corresponds to D_2, and (3) corresponds to D_3.

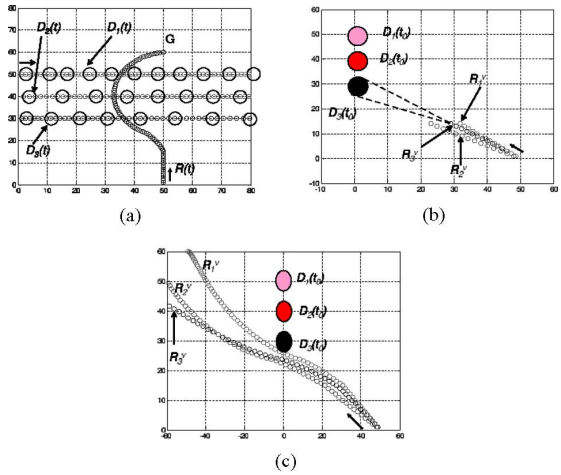

Fig. 10. Collision avoidance. (a) Real plane. (b) Virtual plane in $0 \leq t \leq 5$ s. (c) Complete virtual plane.

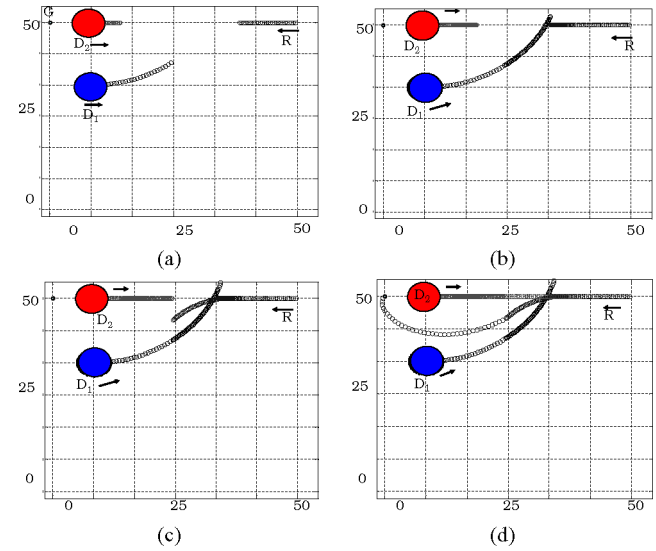

Fig. 11. Real plane. (a) $0 \leq t \leq 7$ s. Collision is detected with the moving objects. (b) $0 \leq t \leq 12.25$ s. Collision with D_1 is avoided by slowing down the robot. (c) $0 \leq t \leq 19$ s. Collision with D_2 is avoided by deviation. (d) $0 \leq t \leq 34$ s. Navigation is continued toward the goal.

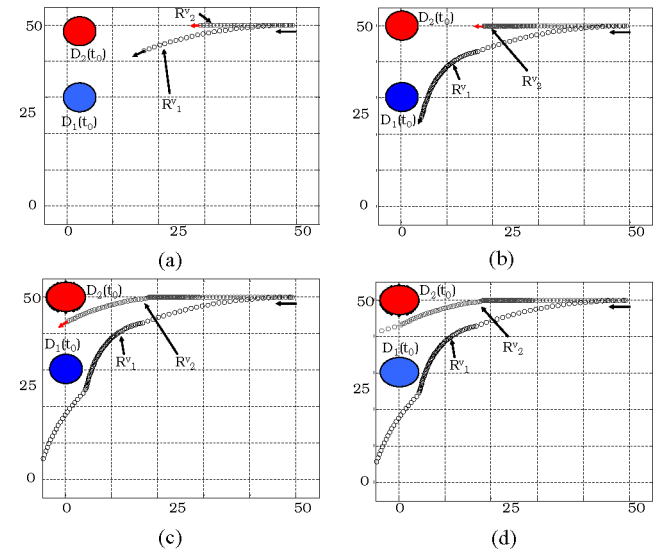

Fig. 12. Virtual plane. (a) $0 \leq t \leq 7$ s. Collision is detected with the moving objects. (b) $0 \leq t \leq 12.25$ s. Collision with D_1 is avoided by slowing down the robot. (c) $0 \leq t \leq 19$ s. Collision with D_2 is avoided by deviation. (d) $0 \leq t \leq 34$ s. Navigation is continued toward the goal.

from its initial position $(50, 50)$. The robot's speed is $v_r = 2$ m/s with an orientation $\theta_r = \pi$. Fig. 11 represents the scenario, while Fig. 12 shows the virtual plane in different time intervals: (a) $0 \leq t \leq 7$ s; (b) $0 \leq t \leq 12.25$ s; (c) $0 \leq t \leq 19$ s; (d) $0 \leq t \leq 34$ s. As seen from Fig. 12(a), the robot is in a collision course with both $D_1(t_0)$ and $D_2(t_0)$. However, it is more urgent to avoid collision with D_1, and the robot slows down to accomplish this. This corresponds to deviation from $D_1(t_0)$, as shown in Fig. 12(b). The robot deviates again to avoid collision with D_2, as shown in Fig. 11(c). Again, this corresponds to deviation from $D_2(t_0)$ in the virtual plane, as shown in Fig. 12(c).

VII. Conclusion

This paper deals with the problem of collision detection and path planning in dynamic environments. The method is based on the virtual plane from which an angle window or a speed window is constructed. The virtual plane is a transforation of observer that allows to map moving obstacles to stationary obstacles. This brings important simplifications to the problem of collision course and path planning in a dynamic environment. The notion of the virtual plane can be combined with various classical methods for path planning and navigation in dynamic environments.

References

[1] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1, pp. 477–521, 1987.

[2] J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," in *Proc. 26th Annu. Symp. Found. Comput. Sci.*, Jul. 1994, vol. 41, no. 4, pp. 764–790.

[3] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, 1986.

[4] K. Fujimura and H. Samet, "A hierarchical strategy for path planning among moving obstacles," *IEEE Trans. Robot. Autom.*, vol. 5, no. 1, pp. 61–69, Feb. 1989.

[5] K. Fujimura and H. Samet, "Roadmap-based motion planning in dynamic environments," *IEEE Trans. Robot.*, vol. 21, no. 5, pp. 885–897, Oct. 2005.

[6] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot. Res.*, vol. 21, no. 3, pp. 233–255, 2002.

[7] M. Kallmann and M. Mataric, "Motion planning in the presence of moving obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 2004, pp. 4399–4404.

[8] J. V. der Berg and M. Overmars, "Kinodynamic motion planning on roadmaps in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Diego, CA, Oct. 2007, pp. 4253–4258.

[9] S. Ge and Y. Cui, "Dynamic motion planning for mobile robots using potential field method," *Auton. Robots*, vol. 13, no. 3, pp. 207–233, 2002.

[10] K. Valavanis, T. Hebert, R. Kolluru, and N. Tsourveloudis, "Mobile robot navigation in 2-d dynamic environments using an electrostatic potential field," *IEEE Trans. Robot. Autom.*, vol. 30, no. 2, pp. 187–196, Mar. 2000.

[11] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, Roma, Italy, 2007, pp. 1986–1992.

[12] A. F. Foka and P. E. Trahanias, "Predictive control of robot velocity to avoid obstacles in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Las Vegas, NV, Oct. 2003, pp. 370–375.

[13] T. Fraichard and C. Laugier, "Dynamic trajectory planning, path velocity decomposition and adjacent paths," in *Proc. Int. Joint Conf. Artif. Intell.*, Chambery, France, Aug. 1993, pp. 1592–1597.

[14] A. Chakravarthy and D. Ghose, "Obstacle avoidance in dynamic environment: A collision cone approach," *IEEE Trans. Syst., Man Cybern. A, Syst., Humans*, vol. 28, no. 5, pp. 562–574, Sep. 1998.

[15] F. Belkhouche and B. Belkhouche, "Kinematics based characterization of the collision course," *Int. J. Robot. Autom.*, vol. 23, no. 2, pp. 127–136, 2008.

[16] F. Belkhouche, T. Jin, and C. Sung, "Plane collision course detection between moving objects by transformation of coordinates," in *Proc. IEEE Multi-Conf. Syst. Control*, San Antonio, TX, Sep. 2008, pp. 1013–1018.

[17] T. Fraichard, "Trajectory planning in dynamic workspace: A state-time space approach," *Adv. Robot.*, vol. 13, no. 1, pp. 75–94, 1999.

[18] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, pp. 760–772, 1998.

[19] F. Large, S. Sckhavat, Z. Shiller, and C. Laugier, "Using non-linear velocity obstacles to plan motions in a dynamic environment," in *Proc. Int. Conf. Control, Autom., Robot. Vis.*, Dec. 2002, pp. 734–739.

[20] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Pasadena, CA, May 2008, pp. 1928–1935.

[21] B. Kluge and E. Prasseler, "Recursive agent modeling with probabilistic velocity obstacles for mobile robot navigation among humans," in *Autonomous Navigation in Dynamic Environments* (Tracts in Advanced Robotics 14). Berlin, Germany: Springer, 2007, pp. 121–134.

[22] M. J. Matney, P. Anz-Meadorb, and J. L. Fosterc, "Covariance correlations in collision avoidance probability calculations," *Adv. Space Res.*, vol. 34, no. 5, pp. 1109–1114, 2004.

[23] R. Philippsen, B. Jensen and R. Siegwart, "Towards real-time sensor-based path planning in highly dynamic environments," in *Autonomous Navigation in Dynamic Environments* (Tracts in Advanced Robotics), vol. 35. Berlin, Germany: Springer, 2007, pp. 135–148.

[24] H. Koyasu, J. Miura, and Y. Shirai, "Mobile robot navigation in dynamic environments using omnidirectional stereo," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, Sep. 2003, pp. 893–898.

[25] K.-T. Song and C. Chang, "Reactive navigation in dynamic environment using a multisensor predictor," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 29, no. 6, pp. 912–925, Dec. 1999.

[26] R. Alami, T. Simeon, and K. M. Krishna, "On the influence of sensor capacities and environment dynamics onto collision-free motion plans," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Lausanne, Switzerland, Oct. 2002, pp. 2395–2400.

[27] K. Krishna and P. Karla, "When does a robot perceive a dynamic object?," *J. Robot. Syst.*, vol. 19, no. 2, pp. 73–90, 2002.

[28] C. Chang and K.-T. Song, "Environment prediction for a mobile robot in a dynamic environment," *IEEE Trans. Robot. Autom.*, vol. 13, no. 6, pp. 862–872, Dec. 1997.

[29] M. Vukobratovic, V. Potkonjak, and V. Matijevic, "Contribution to the study of dynamics and dynamic control of robots interacting with dynamic environment," *Robotica*, vol. 19, no. 2, pp. 149–161, 2001.

[30] S. Akella and J. Peng, "Time-scaled coordination of multiple manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 2004, pp. 3337–3344.

[31] C. Clark, T. Bretl, and S. Rock, "Applying kinodynamic randomized motion planning with a dynamic priority system to multi-robot space systems," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, Mar. 2002, pp. 3621–3631.

[32] K. Krishna, H. Hexmoor, and S. Chellappa, "Reactive navigation of multiple moving agents by collaborative resolution of conflicts," *J. Robot. Syst.*, vol. 22, no. 5, pp. 249–269, 2005.

[33] N. Bourbakis, "A traffic priority language for collision-free navigation of autonomous mobile robots in dynamic environments," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 27, no. 4, pp. 573–587, Aug. 1997.

[34] Y. Li, K. Gupta, and S. Payandeh, "Motion planning for multiple agents in virtual environments using coordination graphs," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, Apr. 2005, pp. 378–383.

[35] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 912–925, Dec. 1998.

[36] L. Yun-Hui and S. Arimoto, "Finding the shortest path of a disc among polygonal obstacles using a radius-independent graph," *IEEE Trans. Robot. Autom.*, vol. 11, no. 5, pp. 682–691, Oct. 1995.

[37] K. Fujimura and H. Samet, "Planning a time-minimal motion among moving obstacles," *Algorithmica*, vol. 10, no. 1, pp. 41–63, 1993.

[38] J. G. de Lamadrid and J. Zimmerman, "Avoidance of obstacles with unknown trajectories: Locally optimal paths and path complexity, part I," *Robotica*, vol. 11, pp. 299–308, 1993.

[39] Y. Kim and M. Minor, "Path manifold-based kinematic control of wheeled mobile robots considering physical constraints," *Int. J. Robot. Res. Arch.*, vol. 26, no. 9, pp. 955–975, 2007.

[40] E. Badreddin and M. Mansour, "Fuzzy-tuned state feedback control of a nonholonomic mobile robot," in *Proc. 12th Triennial World Congr. IFAC*, Sydney, Australia, Jul. 1993, pp. 769–772.

[41] F. Belkhouche and B. Belkhouche, "Modeling and controlling a robotic convoy using guidance laws strategies," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 4, pp. 813–825, Aug. 2005.

[42] F. Belkhouche, "Nonholonomic robots navigation using linear navigation functions," in *Proc. Amer. Control Conf.*, New York, Jul. 2007, pp. 5328–5332.

Fethi Belkhouche (S'02–M'05), photograph and biography not available at the time of publication.