International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:5, 2007

Robot Motion Planning in Dynamic Environments with Moving Obstacles and Target

Ellips Masehian, and Yalda Katebi

***Abstract*—**This paper presents a new sensor-based online method for generating collision-free near-optimal paths for mobile robots pursuing a moving target amidst dynamic and static obstacles. At each iteration, first the set of all collision-free directions are calculated using velocity vectors of the robot relative to each obstacle and target, forming the Directive Circle (DC), which is a novel concept. Then, a direction close to the shortest path to the target is selected from feasible directions in DC. The DC prevents the robot from being trapped in deadlocks or local minima. It is assumed that the target's velocity is known, while the speeds of dynamic obstacles, as well as the locations of static obstacles, are to be calculated online. Extensive simulations and experimental results demonstrated the efficiency of the proposed method and its success in coping with complex environments and obstacles.

***Keywords*—**Dynamic Environment, Moving Target, Robot Motion Planning.

I. Introduction

MANY efforts have been conducted in robotics research for solving the fundamental problem of motion planning, which consists of generating a collision-free path between start and goal positions for a robot in a static and completely known environment, where there could be obstacles [1]. Mobile robot motion planning in dynamic environments has been studied extensively in [2]. On-line planning algorithms are needed for changing environments, but these typically suffer from a lack of generality in the knowledge of the space in which they are executed. Lots of studies exist on the motion planning for robotic systems using various approaches. There is strong evidence that a *complete* planner (i.e. one that finds a path whenever one exists and reports that no one exists otherwise) will take time exponential in the number of degrees of freedom (dof) of the robot, and that the algorithm belongs to a class of problem known as NP-Complete [3].

In recent years, a class of motion planning problems that has received more attention is the motion planning in dynamic environments with moving obstacles and moving targets. 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 [4].

A number of works exist on motion planning in dynamic environments with *moving obstacles*. These studies, in terms of the knowledge of the movement of obstacles, can be classified into two categories:

In the first category, movements of obstacles are completely unknown to the robot [5]. Our study belongs to this category. However, despite the lack of robot's knowledge of the environment, we try to optimize the robot's motion. Also, the safety of robot motion is an important concern.

In the second category, movements of obstacles are completely known. In recent studies 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 [6].

In addition to these categories, some works introduce the factor of uncertainty in obstacles' motion. In this realm a method was proposed in [7] to predict the motion of an obstacle and its uncertainty from the history of its movement. In another work, a probabilistic model of the uncertainty is used in order to select the motion that minimizes the expected time of reaching the destination [8].

An efficient neural network method was presented in [9] for real-time motion planning of a mobile robot, or a multi-joint robot manipulator, with safety considerations in a nonstationary environment. The optimal robot motion was planned through the dynamic neural activity landscape of the biologically-inspired neural network without any prior knowledge of the dynamic environment and without any learning procedures. A data-driven fuzzy approach was developed for solving the motion planning of a mobile robot in the presence of moving obstacle. The approach consists of devising a general method for the derivation of input-output data to construct a Fuzzy Logic Controller (FLC) off-line [10].

Some papers have developed motion planning with *moving target*. In [11] the authors consider the problem of planning the motions of a mobile robot equipped with a visual sensor, whose task is to track an unpredictable moving target in a workspace cluttered by obstacles. The planner decides in real time how the robot should move in order to keep the target within its field of view. It also proposed a framework by combining game theory and geometry to solve this multifold planning problem.

Several applications require persistent monitoring of a moving target by a controllable vision system. In applications that involve automated processes that need to be monitored, such as in an assembly workcell, parts or subassemblies might need to be verified for accuracy, or are determined to 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 target has introduced in a cluttered workspace [12]. In this work, the

Authors are with Faculty of Engineering, Tarbiat Modares University, Tehran, Iran.

robot and its target exist 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 [13].

In this paper, both the moving obstacles and moving target problem are addressed. It is assumed that the target's velocity is known while the speeds of dynamic obstacles, as well as the locations of static obstacles, are to be calculated online. The next section denotes the assumptions of the model. After describing the robot's sensing mechanism, and its technique for tracking the obstacles and the target, a new concept, the Directive Circle, is introduced. Next, the obstacle avoidance strategies are discussed. Finally, some simulation results are provided.

II. Problem Assumptions

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

- *Robot*: The robot is circular and omnidirectional. It can move with a maximum speed of $\vec{V}_R$, which should be known at the beginning of planning. It is equipped with range sensors encircled around it.
- *Obstacles*: Each obstacle is presented with OB_i. Obstacles may be static or dynamic, with the speed of $\vec{V}_{Oi}$. Their speed must be set at the beginning, and be proportional to the robot's velocity. Obstacles can be of any concave or convex polygonal shape, and their instantaneous speed vector (velocity and direction) is unknown to the robot. We assume that the obstacles are identifiable by the robot, and move along arbitrary trajectories within a predefined rectangular border, without rotating about an internal axis. The border is considered a rigid body with reflective property in an Euclidean workspace. Upon colliding with the border or another obstacle, it bounces (reflects) back along its direction before the collision.
- *Target*: It is assumed that only one target exists in the problem, within the defined border. The velocity of target is shown with $\vec{V}_T$, and is proportional to the robot's velocity. $\vec{V}_T$ is known to the robot both at the beginning of motion planning, and later, if it undergoes a change.

III. Robot's Sensing

Since the velocity and position of obstacles is unknown for the robot, it must be equipped with detectors or range sensors to acquire necessary information. This is done by performing a visibility scan and detecting visible obstacle vertices.

Upon arriving at a new point in Configuration space, the robot first determines its distance to surrounding obstacles by means of its radial sensor readings, and stores the result in a *visibility matrix* which contains the angles of emanated sensor rays, the magnitude of each ray, and the coordinates of visible obstacle points. This matrix is then further processed to yield visible obstacles' convex vertices, which are included in a list of candidate moves.

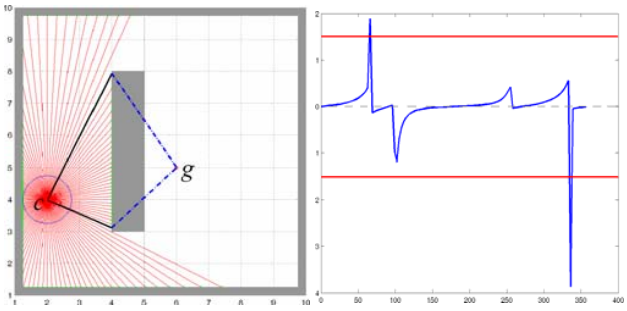
Assume that the mobile robot is located at the point c and has a circular shape with a radius of R and a number of S range-sensors situated equidistantly on its perimeter. Then each sensor projects a ray r_i (i = 1, …, S, mod S, counterclockwise) to find out its distance ρ_i from the nearest visible obstacle point $\mathbf{x}_i$ along the i-th direction, as shown in Fig. 1(a). Taking the metric $D(\mathbf{x}_i, \mathbf{x}_c)$ for the Euclidean distance of points $\mathbf{x}_i$ and $\mathbf{x}_c$, we have $\rho_i = D(\mathbf{x}_c, \mathbf{x}_i) - R$, where $\mathbf{x}_c$ is the coordinates of robot's current position in the Configuration space.

In order for the robot to avoid getting trapped in obstacles' concave regions and bypass any blocking obstacle, it should move toward the *tangent* rays of obstacles boundaries. A ray r_i is tangent to an obstacle iff in a neighborhood U of $\mathbf{x}_i$ the interior of the obstacle lies entirely on a single side of the line r_i. Otherwise, the robot's motion toward the middle of the obstacle will lead to collision. This strategy stipulates the robot to distinguish the obstacle's outermost vertices, or in a broader sense (if the obstacles are not polygons), the regions adjacent to tangent rays, as viewed from the robot's vantage point.

For determining the tangent rays, a difference function is applied for successive adjacent rays to calculate the *ray difference* variables as

$$\hat{\rho}_i = \rho_{i+1} - \rho_i \,. \tag{1}$$

The magnitude-versus-angle plot of difference variables of the Fig. 1(a) is provided in Fig. 1(b). High peaks (both positive and negative) imply abrupt and large differences in successive ray magnitudes and so indicate the points where sweeping rays leave or meet a convex contour on obstacle boundary, and are detected by applying a notch filter to the plot. The result is determining obstacles' positions.



(a) (b)

Fig. 1 (a) A visibility scan of the environment by the robot located at c, and lines involved in calculating the cost of tangent rays. (b) The sensory data is processed to find obstacle vertices, represented by the two sharpest peaks

Now the velocities of obstacles should be detected. The robot records the positions of obstacles' edges in two successive iterations (time intervals) to determine each obstacle's speed.

Because of the dynamic nature of the problem, in which obstacles may mask each other or follow unknown directions, they are not necessarily identifiable by just their vertices. Therefore, a theoretical landmark is necessary to precisely track the obstacles. Since two visible edges of an obstacle will

definitely intersect, their intersection point can be computed as a landmark point. This point will help the robot in determining the obstacle's speed vector.

As shown in Fig. 2, suppose that e_1 and e_2 are the edges of obstacle O_i. The robot senses P_1 and P_2 on edge e_1, and P_3 and P_4 on edge e_2 at time t_1. At the next time interval (t_2) the points P'_1 and P'_2 are sensed on edge e'_1, and points P'_3 and P'_4 on edge e'_2. The slope of each edge can then be calculated based on these points.

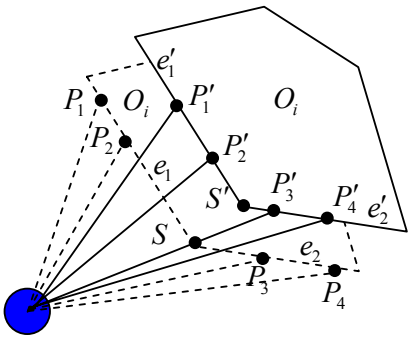


Fig. 2 Finding the velocity of obstacles

The algorithm looks for two equal slopes among visible edges at time interval [t_1, t_2]. In this case, S and S' are the landmark points. These points can be a real vertex of the obstacle, or a virtual one. The velocity vectors $\vec{V}_{x_O}$ and $\vec{V}_{y_O}$ of the obstacle then would be:

$$\vec{V}_{x_O} = x_{S'} - x_S$$
$$\vec{V}_{y_O} = y_{S'} - y_S \qquad (2)$$

When only one edge of the obstacle is visible at time t_1, and a landmark point cannot be found through the above procedure, the algorithm searches for an edge parallel to the visible edge at time t_2 and takes the endpoint of the edge as an approximate landmark for calculating $\vec{V}_{x_O}$ and $\vec{V}_{y_O}$. As a result, the obstacle's velocity will be estimated roughly, and as soon as two edges of the obstacle become visible, the velocity will be calculated exactly via an exact landmark.

IV. Target Following

In this section, we present the method of following the moving target. In [13] a novel method for the interception of moving targets in the presence of obstacles is proposed. The authors define a line-of-sight (LOS) as the relative position vector, $\vec{r}$, connecting the robot to the target, as shown in Fig. 3. The *Parallel Navigation* law states that the direction of LOS should remain constant relative to a non-rotating frame, while the robot approaches the target. Namely, the relative velocity, $\dot{r}$, between the robot and the target should remain parallel to the LOS (i.e. $\vec{r}$) at all times. If 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 relationships:

$$\vec{r} \times \dot{r} = 0, \qquad (3)$$
$$\vec{r} \cdot \dot{r} < 0. \qquad (4)$$

Equation (3) guarantees that the LOS and relative velocity remain parallel, while equation (4) ensures that the interceptor is not receding from the target.

The robot can move with a maximum speed of $\vec{V}_R$. However, because of the presence of obstacles, it is not always possible to directly follow the optimum path to the target.

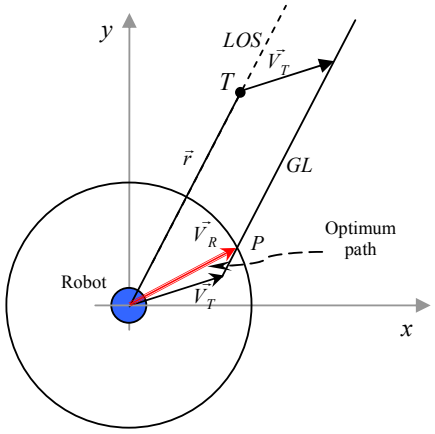


Fig. 3 Finding the optimum path toward the target

As long as the robot follows the optimum path, the optimum path is not required to be recalculated. However, upon deviating from the optimum path, the robot must perform new calculations to find the path to the target.

In Fig. 3 suppose that target T moves along $\vec{V}_T$. The origin of the frame of coordinates is located on the robot's center to show the instantaneous relative position of target. The velocity of target and any changes in direction of target's trajectory, and hence vectors $\vec{r}$ and $\vec{V}_T$ are known for robot. The endpoints of the velocity vectors present the position of the target and robot, after one time-interval has elapsed. The Guidance Line (GL) in Fig. 3 is a semi-line that is parallel to LOS. If the end-point of $\vec{V}_R$ falls on this semi-line, the direction of LOS would remain constant, and positional matching between the robot and the moving target is guaranteed.

The goal of the trajectory planner is to obtain a robot-velocity command for the next command instant, according to the parallel navigation law. First a circle with radius $\|\vec{V}_R\|$ is drawn around the robot. This circle intersects the semi-line GL at P. This is the end point of $\vec{V}_R$. It is evident that $\vec{V}_R$ must be at least equal to target's velocity in order to reach target.

Through this method we can find the optimal path. However, we consider the path's safety, which is also an important issue.

V. Obstacle Avoidance

In this section, we present the concept of the *Directive Circle* (*DC*) for single and multiple obstacles. This circle decides the acceptable and forbidden directions to move. At any instant, the robot recognizes the velocities of obstacles using the method explained in Section III.

The example in Fig. 4 illustrates the creation of the Directive Circle. Consider the robot R and obstacle OB, which is concave. The velocity of the obstacle is $\vec{V}_O$. We define the *Collision Cone*, $CC_{R,OB}$, as the set of relative velocities between the colliding R and OB [6]:

$$CC_{R,OB} = \left\{ v_{R,OB} \,\middle|\, \lambda_{R,OB} \cap \mathrm{OB} \neq \varphi \right\}$$

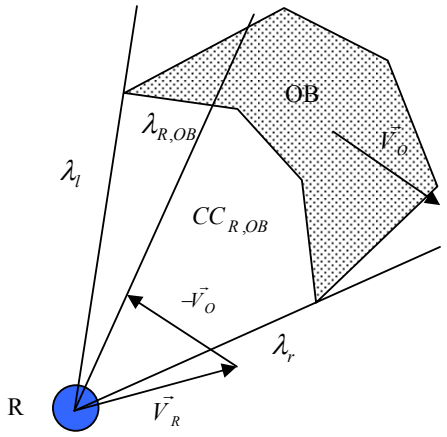


Fig. 4 The relative velocity $V_{R,OB}$ and the collision cone $CC_{R,OB}$

where $\vec{V}_{R,OB} = \vec{V}_R - \vec{V}_O$ is the relative velocity of *R* with respect to *OB*, and $\lambda_{R,OB}$ is the direction of $\vec{V}_{R,OB}$. λ_r and λ_l represent the two rays tangent to the obstacle, and are determined in visibility scan described in Section III. For every particular robot/obstacle pair, there is a unique Collision Cone.

In DC, we find the forbidden directions as sets of directions in $\lambda_{R,OB}$ lied on $CC_{R,OB}$. For forming the DC, the position of the robot along $-\vec{V}_O$ must be shifted. Then we draw a circle *C* around the robot with a radius of maximum velocity of robot.

Geometrically, one case from four cases may occur: 1) the circle intersects λ_r, 2) the circle intersects λ_l, 3) the circle intersects both λ_r and λ_l, and, 4) the circle intersects none of them. As shown in Fig. 5. In this sample, circle C has intersection with λ_l at point A, and with λ_l at point B. So, if the slope of $\lambda_{R,OB}$ falls between the slopes of *PA* and *PB* vectors, the robot will collide with the obstacle. The Directive Circle (dashed) is formed using the above calculation.

If the circle *C* doesn't have any intersection points with tangents λ_r and λ_l, DC does not have any forbidden zone, and robot can freely move along the calculated optimal direction to the target. If the whole circle *C* falls in $CC_{R,OB}$, then all possible directions are forbidden and the robot should stop. This condition usually happens when the robot is in a cluttered space, and/or obstacles have surrounded the robot tightly.

The DC must be built for all moving or static obstacles. Let F_i show the forbidden areas for each obstacle *i*. Then the total DC for all obstacles (i.e. the total forbidden and collision-free paths) is computed by superimposition of all partial DCs as below:

$$F = \bigcup_{i=1}^{m} F_i \,. \tag{5}$$

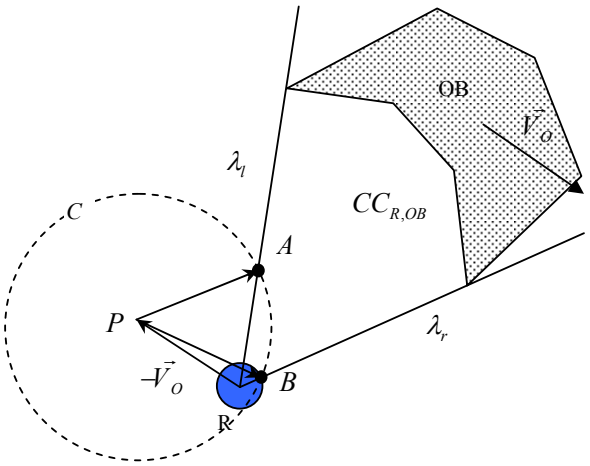


Fig. 5 Finding the forbidden zone and Directive Circle (DC)

A. Optimal Solution Search

By following the above algorithm, a set of collision-free paths are found for the center of robot. Our purpose is to find the *best* collision-free path. In Section IV the method of finding an optimum path toward target was discussed. Consider the DC of the problem shown in Fig. 6. By dividing the DC into N parts, there are N directions (r_i) to select. Some of these paths are located in forbidden areas and cannot be selected.

Acceptable directions are selected based on these conditions:

C1) The optimum path falls in acceptable area.

C2) The optimum path falls in forbidden area.

If condition 1 occurs, the algorithm chooses the optimum path calculated with the method in Section IV. But if condition 2 happens, the slopes of angle of optimum path is determined and shown with θ^*. Then, the slopes of all acceptable directions are measured and named θ_i. g_i represents the gap between θ^* and θ_i, as below:

$$g_i = |\, \theta_i - \theta^* |; \quad \forall\, \theta_i \in \Theta_f \,. \tag{6}$$

In order to find the optimum direction, the probability of each direction is computed by

$$P(r_i) = \frac{1}{\sqrt{2\pi}} e^{-x_i^2/2} \,, \tag{7}$$

in which $x_i = g_i \bmod \pi$, $0 \le x_i \le \pi$. So the direction with maximum probability is selected as below:

$$\vec{V} = \{ r_i \mid P(r_i) \ge P(r_j) \,,\ \forall\, i \neq j \} \tag{8}$$

In some conditions, there is an obstacle between robot and target, along the optimum path. In this situation, using

described algorithm will be helpful. Simulation of problem in next section shows that regard to optimum direction and target path, robot chooses the best solution that is short and safe. This property will be shown in the next section.

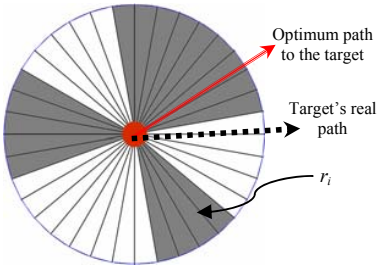


Fig. 6 Total Directive Circle for 3 disjoint obstacles

B. Safety Checking

The robot is circular with radius *r*. so we need to check the safety of the path in order to avoid collisions. We need to check the safety of the path when the robot is near to obstacles. It is possible to define a parameter for the safety level of the planning. An obstacle is said to be near to the robot when the minimum distance between the robot and obstacle is *K* times less than the obstacle's velocity. *K* is the safety parameter.

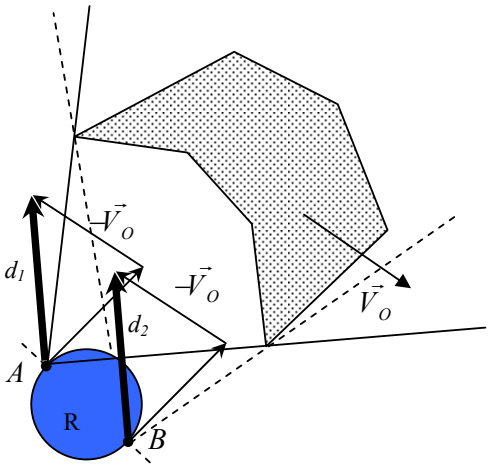


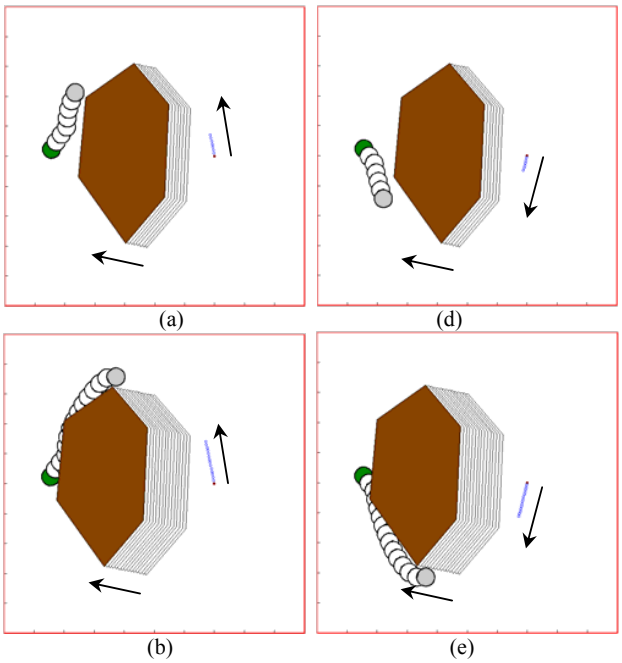
Fig. 7 Total Directive Circle for 3 disjoint obstacles

Aiming to verify the safety of the path, the robot's exterior points should be checked to make sure whether the robot contacts with obstacles or not. In Fig. 7, the exterior points are shown with *A* and *B*. Since the robot's shape is circular, these points can be determined easily. *A* is resulted from subtracting 90º from the gradient of chosen path, and *B* is obtained from adding 90º to the gradient of the chosen path. In order to ensure that these points get in touch with obstacle or not, the algorithm presented in Section V can be used (from [6]). As shown in Fig. 7, d_1 and d_2 are the sums of the robot's velocity vector and $-V_O$ at *A* and *B* respectively. d_1 falls out of $CC_{R,OB}$ in point *A* and so does not contact with the obstacle. However, d_2 falls in the $CC_{R,OB}$ of point *B*, so this direction is not acceptable for moving. This procedure ensures that the chosen direction is collision-free.

VI. Simulation and Discussion

This section presents simulations of the algorithm's performance. To substantiate the motion planner's capabilities, two types of problems are simulated: Example 1 shows the effectiveness of the Directive Circle (DC) and the method's optimality. Two cases are simulated and presented in Figs. 8(a), (b), (c), and Figs. 8(d), (e), (f). In both cases, initial configurations of the start and goal are the same. Also, the obstacle's position and velocity is the same in both cases. Therefore, the DC of both cases are equal, while the optimum paths are different and lying in forbidden areas. In Figs. 8(a), (b), (c) the target moves upward, and directions within [58°, 305°] are acceptable. The slope of the optimum direction is 16°. Directions are prioritized using (7), and the most probable direction is selected and checked for the safety of path. In Fig. 8(a) initial locations of the robot, target and obstacle are displayed. Up to the state shown in Fig. 8(b), condition C2 (in Section V.*A*) holds. Afterwards, condition C1 occurs and the optimum path falls in the acceptable area of DC.

Figs. 8(d), 8(e), and 8(f) show the second case in which the slope of target's movement is 255° and the slope of the optimum direction is 342°. Similar to the case 1, directions within [58°, 305°] are acceptable. Using equation (7), probabilities of selecting each direction are computed and the most likely direction is selected as the robot's moving direction, as in Fig. 8(d). The robot follows that strategy until the target is visible and the optimum path in the DC falls in the acceptable area. The robot's motion path, from start toward reaching the target, is shown in Fig. 8(e) and 8(f).



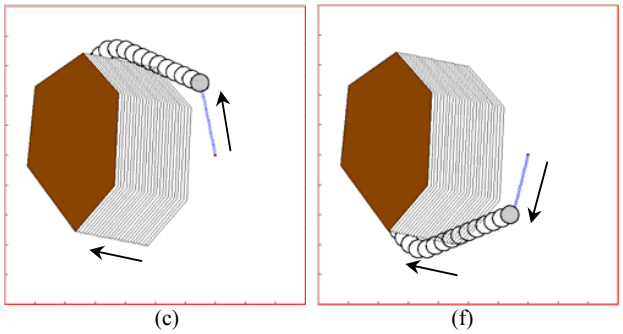


Fig. 8 The Directive Circle helps the robot in selecting the right direction in problem 1 (a, b, c), and problem 2 (d, e, f)

The presented planner can solve more complex problems with more static or dynamic obstacles in convex or concave forms. Fig. 9(a) demonstrates an example that the motion planning is to be done in the presence of one static and two dynamic obstacles. At starting moment the robot is close to a concave obstacle. By applying the obstacle avoidance technique, the robot moves away from the concave obstacle and toward the target. At the moment shown in Fig. 9(b), the moving convex obstacle collides with the boarder and bounces back. The robot senses this change in real-time, and realizes that following the previous direction would not be optimal. So, it changes its direction to the left. Fig. 9(c) displays the instance at which the concave obstacle bumped to the border. The overall trajectory of the robot is shown in Fig. 9(d).

These simulations demonstrate the efficiency of the method and its responsive performance. As shown in figures, the robot's trajectory is collision-free, safe, and near optimal. It can be implemented in any arbitrary environment.

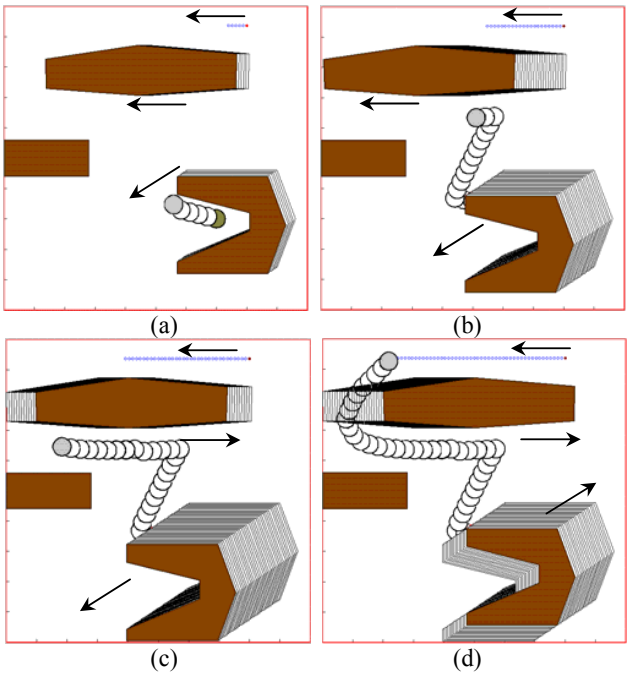


Fig. 9 The robot has real-time responses to unpredicted variations in environment

VII. Conclusion

A new method for planning the motions of a mobile robot in dynamic environments with moving and/or static obstacles and moving target is presented. By setting the target's velocity to zero, the problem turns into a classic moving obstacles problem and can be solved successfully by this planner. The motion planner's decisions are based on the robot's current position, and the velocities of the target and obstacles. Obstacles are not limited in shape, and can be concave or convex.

The main focus has been on two aspects: (a) time-optimal interception with a moving target; (b) obstacle avoidance. Key features of this approach are: (1) it finds a safe path; (2) it incorporates simple geometric estimations on possible deadlocks and local minima; (3) it is practical in a real-time environment with various obstacles. Another property of this method is the smoothness of the resulting trajectory.

This method can be extended to multi-robot applications. For future work, an interesting extension to the algorithm can be the problem of pursuing a set of moving targets by a set of robots; such each robot selects the nearest target.

Numerous simulations have verified the system to be efficient and robust in regards to interception of moving targets with various different interception parameters and situations. In order to estimate exact velocities of moving objects and find the optimum path, the robot's sensors must be sufficient in number and coverage.

References

[1] J. C. Latombe, *Robot Motion Planning*, Kluwer Academic Pub., Boston, MA, 1991.
[2] T. Tsubouchi and M. Rude, "Motion planning for mobile robots in a time-varying environment", *J. of Robotics and Mechatronics*, Vol. 8, No. 1, pp. 15-24, 1996.
[3] J. Canny, *The Complexity of Robot Motion Planning*, MIT Press, Cambridge, MA, 1988.
[4] J. Canny and J. Reif, "New lower bound techniques for robot motion planning", in *Proc. IEEE Symposium on the Foundations of Computer Science*, Los Angeles, CA, 1987.
[5] S. Ishikawa, "A method of indoor mobile robot navigation by using fuzzy control", in *Proc. IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, pp. 1013-1018, 1991.
[6] P. Fiorini and Z. Shiller, "Motion planning in dynamic environment using velocity obstacles", *International Journal of Robotics Research*, Vol. 17, No. 7, pp. 760-772, July 1998.
[7] A. Inoue, K. Inoue, and Y. Okawa, "On-line motion planning of autonomous mobile robots to avoid multiple moving obstacles based on prediction of their future trajectories", *J. of Robotics Society of Japan*, Vol. 15, No. 2, pp. 249-260, 1997.
[8] J. Minura, H. Uozumi, and Y. Shirai, "Mobile robot motion planning considering the motion uncertainty of moving obstacles", in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, Tokyo, pp. 692-698, 1999.
[9] S. X. Yang and M. Meng, "An efficient neural network method for real-time motion planning with safety consideration", *J. of Robotics and Autonomous Systems*, Vol. 32, pp. 115-128, 2000.
[10] M. Al-Khatib and J. J. Saade, "An efficient data-driven fuzzy approach to the motion planning problem of a mobile robot", *J. of Fuzzy Sets and Systems*, Vol. 134, pp. 65-82, 2003.
[11] P. Fabiani, H. H. Gonzalez-Banos, J. C. Latombe and D. Lin, "Tracking an unpredictable target among occluding obstacles under localization uncertainties", *J. of Robotics and Auton. Sys.*, Vol. 38, pp. 31-48, 2002.
[12] S. M. Lavalle, H. H. Gonzalez-Banos, C. Becker, and J. C. Latombe, "Motion Strategies for Maintaining Visibility of a Moving Target", in *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 731-736, 1997.
[13] F. Kunwar, F. Wong, R. Ben Mrad, B. Benhabib, "Guidance-based on-line robot motion planning for the interception of mobile targets in dynamic environments", *J. of Intelligent and Robotic Systems*, Vol. 47, Issue 4, pp. 341-360, 2006.