Prediction-Based Path Planning with Obstacle Avoidance in Dynamic Target Environment

Zahraa Y. Ibrahim
Electrical Engineering Department
University of Basrah
Basrah, Iraq

Abdulmuttalib T. Rashid
Electrical Engineering Department
University of Basrah
Basrah, Iraq

Ali F. Marhoon
Electrical Engineering Department
University of Basrah
Basrah, Iraq

***Abstract*: In this paper, a new algorithm for mobile robot navigation and polygonal obstacles avoidance in dynamic target environment is introduced. In the dynamic target path planning the agent (robot) trying to reach a moving target in minimum path cost. The introduced algorithm which called Prediction-based path planning with obstacle avoidance in dynamic target environment planning a path to a moving target by predicting the next target location, then computing a path from the robot current location to the predicted target location representing each visible obstacle by the smallest circle that enclosing the polygon obstacle, then determine the visible tangents between the robot and the circular obstacle that intersect its shortest path and compute the shortest path. Three target movement scenarios were suggested and tested in different environment conditions. The results show that the target was reached in all scenarios and under all environment conditions with good path cost.**

Keywords: **Path planning, polygonal obstacles avoidance, randomized incremental, prediction technique.**

I. INTRODUCTION

A large number of path planning applications require planning a path to a moving target as in video games, virtual simulations, and robotics. Since the target changes its location, the agent must modify its path according to the new target location. The execution operation speed is very important because the path must be updated in real time. The quality of the updated path also important because it affects the cost of reaching the target.

Path planning algorithms classified into two types: on-line and off-line path planning. Off-line path planning was not suitable for the large dynamic environment and moving target problem handling because it consumes time since the complete path from start to the target is computed at the start position [1]. Off-line path planning algorithms converted to incremental algorithms in order to be more efficient [2]-[3]. Incremental algorithms suitable for a partially known environment but not solve moving target problem. On-line path planning suitable for a partially known environment with a static target, since the robot uses its sensors to know about the environment but not all they can be solve moving target problem [4]-[5].

Moving target search algorithm (MTS) is suitable for moving target problem handling. MTS compute a matrix of heuristic values. Each value represents the function *h(x,y)* for a pair of states x and y. the matrix initialized by the heuristic values computed by the static evaluation function. During the search operation the heuristic values updated by checking the differ values and sorting them, so the accuracy was improved [6]. Although the complete matrix may be very large, it is a sparse matrix, however, the original MTS is not a very efficient algorithm, and hence two types of updates were proposed called the Commitment to Goal (MTS-c) and the Deliberation (MTS-d). In the Commitment to Goal, the robot needs to know the target position at some point before the final known target position was reached. The idea is to ignore some of the targets positions, while the focus on heuristic table filling. Deliberation the robot updates the heuristic table without moving, so it is performing an off-line path planning [7].

Type of graphs similar to visibility graph is used for increasing the MTS efficiency called sub-goal graph. The sub-goal graph constructed by defining the obstacles corners as sub-goals then connects those sub-goals that are necessary for shortest path finding and can be reached from each other by edges. The sub-goal graph used for shortest path finding [8]-[11]. Although the sub-graph provides a short path from start to the target it requires memory space larger than other algorithms.

In this paper, a new moving target search algorithm called Prediction based path planning with obstacle avoidance in dynamic target environment was introduced. The introduced algorithm suitable for a target moving slower than the robot, and assume that the robot knows about the target position all the time. The proposed algorithm starts by predicting the next target location using the current and previous target locations. After predicting the next target location, the robot path is the straight line from its current location to the predicted target location. The robot represents each obstacle in its sensing range by its smallest enclosing circle. If any collision was detected, then the robot computes the visible tangents from the robot lo-

cation to the circular obstacle; now the robot path is the shortest tangent path. The process was repeated until the target reached.

II. Low-level path planning

In high-level path planning algorithm, the paths are represented by straight and curved lines. In low-level path planning the digital differential analyzer algorithm (DDA) is used to aim mobile robot to follow these paths. This section explains the methods used to investigate the straight and curved paths using the digital differential analyzer algorithm. This algorithm is an incremental method and the coordinate axis of each point is depend on the coordinate axis of the computed point in last steps. The DDA algorithm using the raster characteristics for eliminating the number of multiplications, therefore, it is faster than the direct line equation in determining the line pixels [12].

A.DDA line drawing

The procedure for line drawing in DDA algorithm depending on the line slope (m). At first, consider a line with positive slope with value equal to or less than one as shown in Fig.1 then the next point online is computed from the following equations:

$$x_{k+1} = x_k + \Delta x_{k+1} \quad (1)$$

$$. y_{k+1} = y_k + \Delta x_{k+1} * m \quad (2)$$

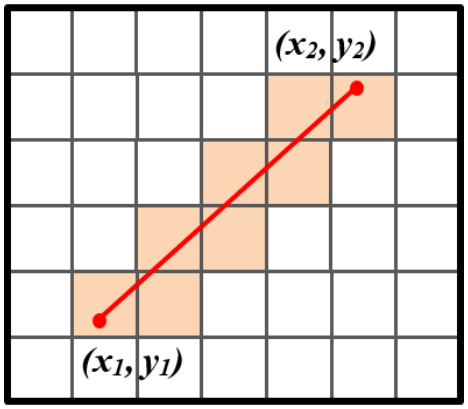

Fig.1 DDA line Drawing for the line has slope m<1.

For the positive line slope greater than one as shown in Fig.2 the next line point is computed from the following equations:

$$y_{k+1} = y_k + \Delta y_{k+1} \quad (3)$$

$$x_{k+1} = x_k + (\Delta y_{k+1}/m) \quad (4)$$

where Subscript k takes integer values starting from 1, for the first point, and increases by 1 until the endpoint is reached and Δx_{k+1} and Δy_{k+1} are the number of pixels in x-axis and y-axis direction respectively depend on the robot speed.

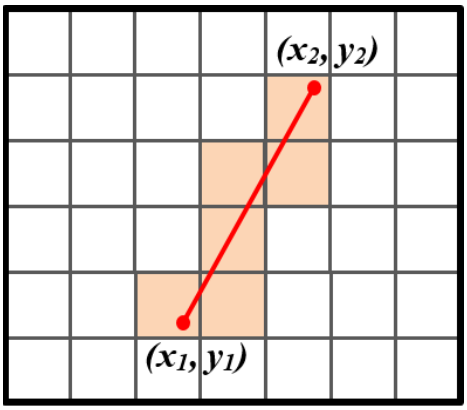

Fig.2.DDA line Drawing for line has slope m>1

B.DDA arc drawing

The DDA arc drawing concept is used in many algorithms that have described the rotation of the vector in the x-y plane. These algorithms are differing from each other in the trigonometric transformations. According to Fig.3 the rotation matrix that is used for vector rotation can be described as follows:

$$x_{n+1} = x_n \cos\Delta\theta + y_n \sin\Delta\theta \quad (5)$$

$$y_{n+1} = y_n \cos\Delta\theta - x_n \sin\Delta\theta \quad (6)$$

Substituting $\Delta\theta = \varepsilon = 2^{-m}$, where m *is* an integer $(m \geq 3)$,then equations (5) and (6) become as follows:

$$x_{n+1} = x_n \cos\varepsilon + y_n \sin\varepsilon \quad (7)$$

$$y_{n+1} = y_n \cos\varepsilon - x_n \sin\varepsilon \quad (8)$$

Equations (7) and (8) above can be described by rotation matrix A:

$$A = \begin{bmatrix} \cos\varepsilon & -\sin\varepsilon \\ \sin\varepsilon & \cos\varepsilon \end{bmatrix} \quad (9)$$

DDA algorithms eliminate the expensive computation of trigonometric functions by using simpler expressions. The rotation matrix determinant is one. The Matrices of DDA algo-

rithms have determinants approximately one, so the more accurate algorithm is the algorithm which has determinant closer to one [13].

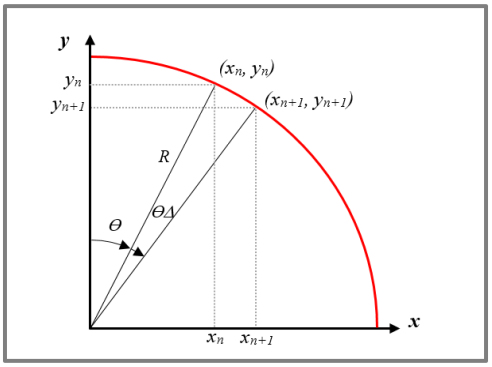

Fig.3 The rotation of vector R in the x-y plane.

One of DDA algorithms called fast exact DDA has accuracy higher than the accuracy of other known algorithms. It is a simple algorithm (it uses only elementary shift, addition, and subtraction), therefore this method can also be used in numerical control, planning mechanisms, and so forth [14]. This algorithm assumes the following initial conditions:

$$x_o = R \tag{10}$$

$$y_o = 0$$

And

$$x_1 = R * (1 - \varepsilon^2)^{0.5} \tag{11}$$

$$y_1 = R*\varepsilon$$

Where $R = 2^m$

Then x_{n+2} and y_{n+2} shown in Fig.4 are computed as follows

$$x_{n+2} = x_n - 2\,\varepsilon\, y_{n+1} \tag{12}$$

$$y_{n+2} = y_n + 2\,\varepsilon\, x_{n+1} \tag{13}$$

drawing circle starting from any point on its circumference with robot speed equal to the previous speed require some changes in the previous conditions as shown in Fig.5.

$$\theta_n = \cos^{-1}\left(\frac{x_n - c_x}{R}\right) \tag{14}$$

$$x_{n+1} = x_n + \Delta x_n \tag{15}$$

$$\theta_{n+1} = \cos^{-1}\left(\frac{x_{n+1} - c_x}{R}\right) \tag{16}$$

$$y_{n+1} = cy + R * \sin\theta_{n+1} \tag{17}$$

$$\varepsilon = |\theta_{n+1} - \theta_n| \tag{18}$$

Where Δx_n a number of pixels in x-axis direction is depends on the robot speed. After that the point (x_{n+2}, y_{n+2}) is computed from equations (19), and (20) below:

$$x_{n+2} = x_n + 2\,\varepsilon\,(y_{n+1} - cy) \tag{19}$$

$$y_{n+2} = y_n - 2\,\varepsilon\,(x_{n+1} - cx) \tag{20}$$

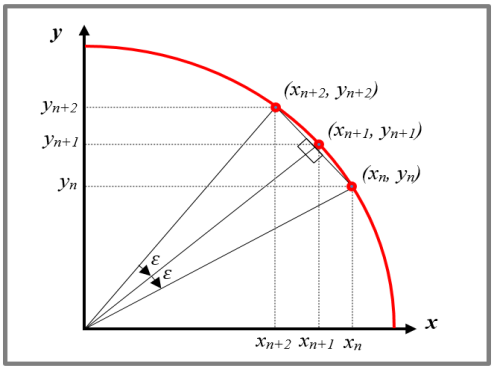

Fig.4 The computation of three subsequent points along the circle.

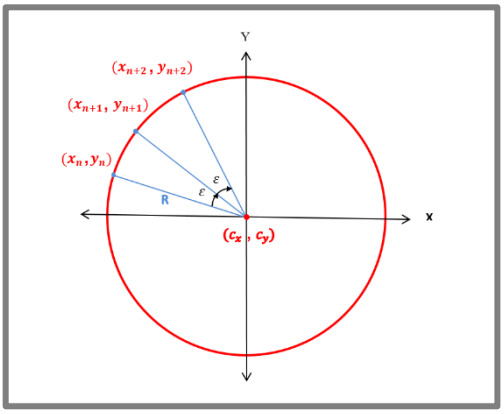

Fig.5 Drawing a circle from any given point.

III. Prediction-Based Path Planning with Obstacle Avoidance in Dynamic Target Environment Algorithm

Path planning is the process of finding a good path from the robot start position to the target position using path planning algorithm. The path planning algorithm must be capable of updating the target position during execution in order to be suitable for solving the moving target problem. In this paper, we assume that the current and previous positions of the target are

known and by using a prediction technique we can estimate the target next position. The robot movement is in a straight line toward the target and when it detects an obstacle in its sensing range then it computes the virtual smallest circle enclosing the obstacle using randomized incremental algorithm [15]. After that, if any circular obstacle intersects the robot path it computes two movement paths represented by the two tangents line to the virtual circle that collides its path. The shortest one is chosen to be the movement path to the robot. The algorithm is summarized by the following steps:

Step 1: determine the initial location, radius, direction, and velocity of a robot $(P_s(x_s^1, y_s^1), r_s, \theta_s, v_s)$ and the initial target location $(P_g(x_g^1, y_g^1))$.

Step 2: determine the initial distance (initial_dis) between the robot and the target initial positions.

$$initial\_dis = ((x_g^1 - x_s^1)^2 + (y_g^1 - y_s^1)^2)^{0.5} \quad (21)$$

Step3: the robot uses the current and previous actual target positions for predicting the next target position as follows:

Let (x_g^2, y_g^2) and (x_g^1, y_g^1) be the current and previous actual target positions respectively, and (x_g^3, y_g^3) the next predicted position as shown in Fig.6 then (x_g^3, y_g^3) is computed as follows:

$$x_g^3 = x_g^2 + (x_g^2 - x_g^1) \quad (22)$$

$$y_g^3 = y_g^2 + (y_g^2 - y_g^1) \quad (23)$$

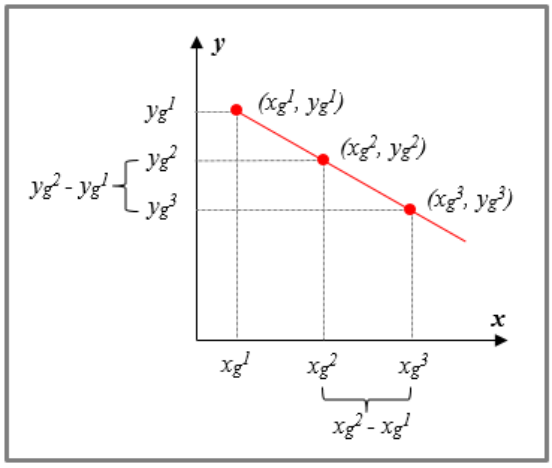

Fig.6 next target position prediction.

At the start position, the target current actual position assumed equal to the previous actual target position.

Step 4: Check if there are any obstacle lays within the sensing range of the robot. If there is any, then the robot increases its sensing radius by the value of the diameter of the largest obstacle in the environment in order to make sure that the robot detects all obstacles vertices.

Step 5: Represent each sensing polygon obstacle by virtual circle using randomized incremental algorithm [15] which can be summarized by the following points:

1. Arrange the polygon obstacle vertices in a set randomly. Let a number of obstacle vertices V equal to n ,then $V = <v_1, v_2, \ldots\ldots, v_n>$.

2. Let c_1 be the circle that contains v_1 and v_2.

3. Starting from $k=3$ to $k=n$ there are two cases:

Case 1: The vertex v_k inside or on the circle c_{k-1} boundary then $c_k = c_{k-1}$

Case 2: The vertex v_k outside the circle c_{k-1} boundary then the circle c_k must satisfy two conditions:

- The vertex v_k on its boundary.

- The vertices v_1 to v_{k-1} inside the circle c_k or maximum two of them on its boundary.

After computing the virtual circle, a safe obstacle is created by increasing the radius of each circular obstacle by the radius of the robot.

Step 6: The robot path is the direct path from the robot position to the target (target predicted next position). This path consists of curved part used for changing the robot direction toward the target and tangent line from that part to the target. The robot starting speed (v) which is represented by a number of pixels that the robot can move in the x-axis direction is the robot minimum speed (v_{min}). This speed can be changed according to the current distance between the robot and the target (dis_s_g).

$$dis\_s\_g = ((x_g^n - x_s^n)^2 + (y_g^n - y_s^n)^2)^{0.5} \quad (24)$$

There are four cases for computing the speed of robot depends on the relation between initial_dis and dis_s_g:

Case1: (dis_s_g > initial_dis) and ($v < v_{max}$)
In this case the current distance between the robot and the

target was largest the initial distance between them, that's mean the robot become far away from the target so that the robot speed must be increased by constant (s_1) in order to reach the target.

$$v = v + s_1 \quad (25)$$

Case 2: (*initial_dis* > = *dis_s_g* > *(2 * initial_dis / 3)*) and ($v < v_{max}$)

The robot is near to the target so that it is speed must be increased by constant value (s_2). It must be chosen less than (s_1).

$$v = v + s_2 \quad (26)$$

Case 3: (*(2 * initial_dis / 3)* > = *dis_s_g* > *(initial_dis / 3)*) and ($v < v_{max}$)

The robot become closest to the target so that its speed must be increased by constant value (s_3). It must be chosen less than (s_2).

$$v = v + s_3 \quad (27)$$

Case 4: (*dis_s_g* < *(initial_dis / 3)*) and ($v > v_{min}$)

In this case, the robot is closest to the target, therefore, the robot must decrease its speed to make sure that its speed equal to its minimum speed when reaching the target.

$$v = v - v * v_{stp} \quad (28)$$

Where v_{stp} is represented the number of pixels that must be subtracted from the robot speed for each step that the robot moves toward the target.

$$v_{stp} = (v - v_{min}) / dis\_s\_g \quad (29)$$

Step 7: Check if there is any collision may be occurring between the robot and the obstacles laying with the sensing range of that robot. If the distance between the robot and any obstacle less than the obstacle radius, then a collision may be occurring.

Step 8: If the collision was detected then the robot draws the inner and outer tangents to the virtual circle of the obstacle. This process is accomplished according to the following steps:

A. outer tangent lines

The two outer tangent lines represented by four endpoints $p_3(x_3,y_3)$,$p_4(x_4,y_4)$, $p_5(x_5,y_5)$,$p_6(x_6,y_6)$ as shown in Fig.7. The points $c_o(x_o, y_o)$ and $c_s(x_s, y_s)$ represent the center points of the virtual circle of obstacle and the circular trajectory of the robot and also, R_o and R_s are the radiuses of both circles respectively. The process of computing the four end points of the outer tangent lines start by drawing third circle centered at the virtual circle center and has radius R such that:

$$R = R_o - R_s \quad (30)$$

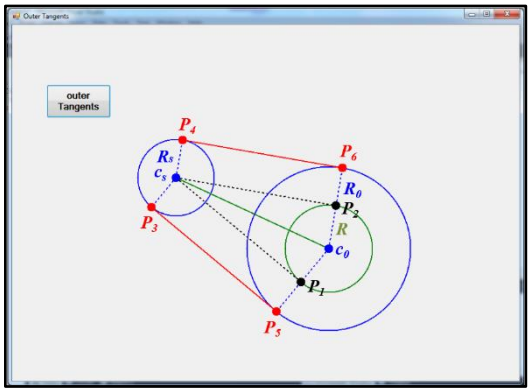

Fig.7 Drawing the outer tangents.

Then the third circle lines tangent points (p_1 and p_2) from the external point $c_s(x_s, y_s)$ computed. The two tangent points are the intersection points of the circle centered at c_0 and have radius equal to R with the circle centered at c_s and have radius equal to the distance(L) between c_s and p_2 as shown in Fig.8 .

$$x_1 = x_i + \frac{h(y_s - y_0)}{D} \quad (31)$$

$$y_1 = y_i - \frac{h(x_s - x_0)}{D} \quad (32)$$

Then p_5 can be computed from equations (22) and (23) be- low:

$$x_5 = x_1 + \frac{R_s(y_s - y_1)}{L} \quad (33)$$

$$y_5 = y_1 - \frac{R_s(x_s - x_1)}{L} \quad (34)$$

The same procedure repeated for computing p_2 and p_6.

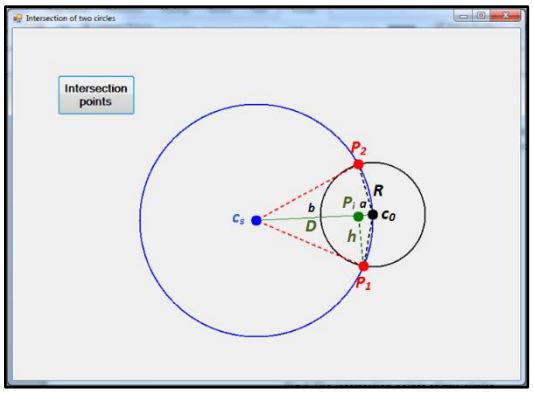

Fig.8 The intersection points of two circles.

B. Inner tangent lines

The inner tangent lines have four points $p_3(x_3,y_3)$,$p_4(x_4,y_4)$, $p_5(x_5,y_5)$, and $p_6(x_6,y_6)$ as shown in Fig.9 which computed using the same procedure used for outer tangents points but the difference is that the radius of the third circle is the sum of the radiuses of the other two circles.

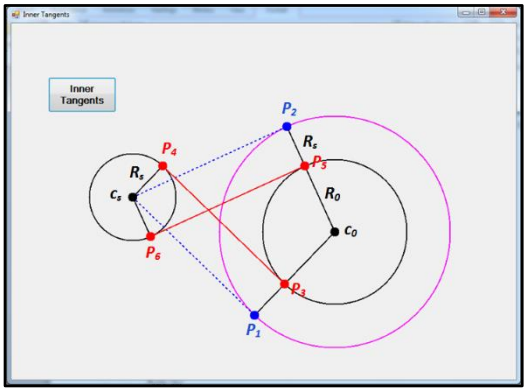

Fig.9 Draw the inner tangent lines.

According to Fig.9 the coordinates of p_5 computed from equations (35) and (36) below:

$$x_5 = x_2 - \frac{R_s(y_2 - y_s)}{L} \quad (35)$$

$$y_5 = y_2 - \frac{R_s(x_s - x_2)}{L} \quad (36)$$

Step 9: the robot computes the length of the two paths from the robot position to the target through the inner and outer tangents and follows the shortest one. During the robot motion toward the tangent point there are two cases for changing the speed of this robot according to the initial distance between the robot and the tangent point (dis_ir_t), and the current distance between the robot and the tangent point (dis_cr_t):

Case 1: dis_cr_t >= (dis_ir_t / 2)

In this case, the robot uses the same cases used in step6.

Case 2: dis_cr_t < (dis_ir_t / 2)

In this case the robot near the tangent point, therefore, the robot must decrease its speed to make sure that the robot speed equal to its minimum speed when the tangent point reached.

$$v = v - v * v_{stp} \quad (37)$$

where v_{stp} is the number of pixels that must be subtracted from the robot speed for each pixel that the robot moves toward the x-axis and computed from the following equation:

$$v_{stp} = (v - v_{min}) / dis\_ir\_t \quad (38)$$

Step10: When the robot reaches the tangent point, then it must update the value of the initial distance between the robot and the target (initial_dis) which become equal to the distance between the tangent point and the current target location.

Step11: Repeat the last steps until the robot reaches the target.

IV. Simulation Results

Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm is implemented using visual basic programming language. In our simulation differential drive mobile robot was assumed. The algorithm is tested in three target movement scenarios: straight line, circular arc, and sine wave scenarios. Each scenario tested in Different environments with a different number, size, and shapes of polygonal obstacles. The results were compared for three robot sensing radiuses.

The simulation assumes that the minimum robot speed vmin=3 pixels per step that the robot can move toward the target, and the maximum robot speed v_{max}=15 pixels per step. The sensing range of the robot increased by the value of the diameter of the largest obstacle which equals to 110 pixels when a new obstacle is detected to make sure that all the obstacle vertices were seen. The produced algorithm tested for three target movement scenarios in different environments in terms of numbers, shapes, and dimensions of the obstacles. Each scenario is repeated for a number of obstacles ranged from 0 to 30 increasing by 5. We take about 10 distributed that arranged randomly and the means value of the distance between source and target has been calculated. For more comparisons each scenario is repeated for 100, 200 and 300 pixels represent the sensing radiuses of the robot. Snapshots of straight line target movement scenario with 15 obstacles have different shapes and dimension was shown in Fig.10-12 and the result indicated in Fig.13.The snapshots of the circular arc target movement scenario were shown in Fig.14-16 and the result indicated in Fig.17. The last scenario which represents sine wave scenario snapshots was shown in Fig.18-20 and the result indicated in Fig.21.

From all the simulation results shown in Fig.13, Fig.17 and Fig.21 we notice that by using the produced algorithm the robot reaches the target in all target movement scenarios and under any environment conditions with good path length, and the path length from the source position to the target was increased by increasing number of obstacles in the environment, and

also, the increasing of the robot sensing radius result from an increasing in the path length. The only explanation for this conclusion is that the robot predicates the target position for only one step and computes the safe path for that target, therefore, some obstacles may lay in the safe path but actually the robot moves away from these obstacles way before the robot become near them, therefore, it does not affect the robot path.

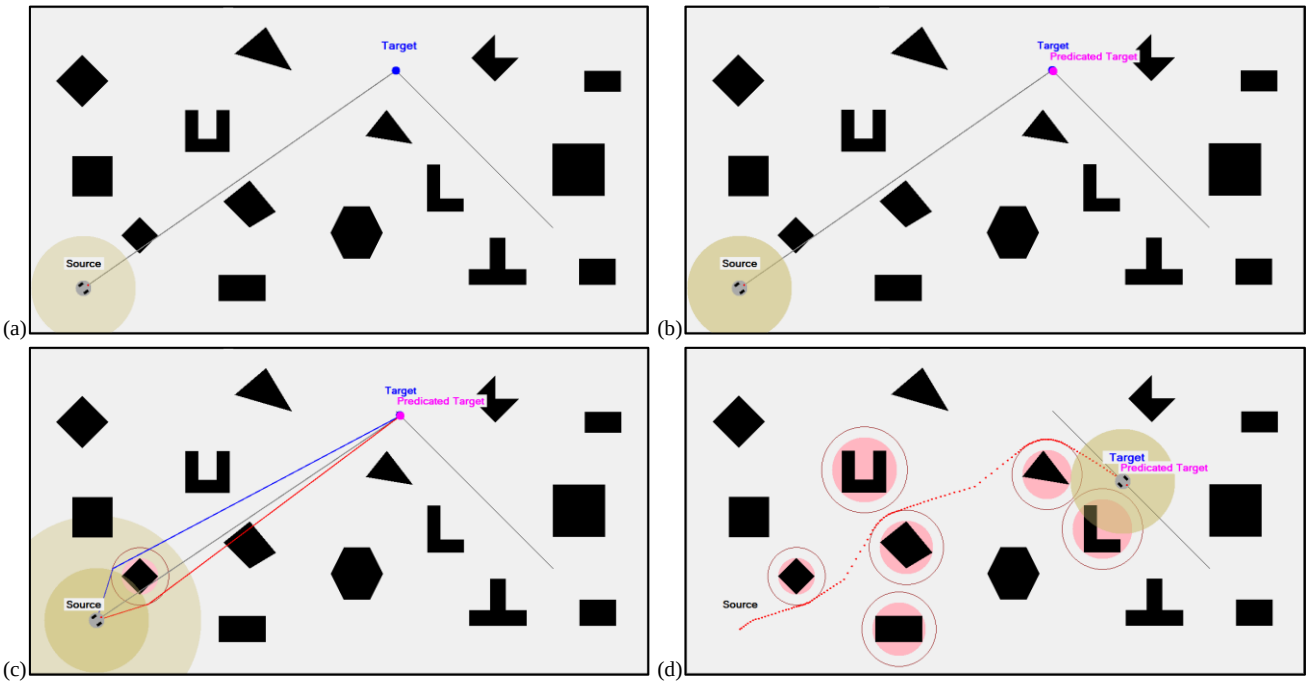

Fig.10 Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm with straight line target movement (robot sensing radius = 100 pixels): (a) path planning for the initial position of target; (b) path planning for the next position of target; (c) determine the shortest path to the next position of target; (d) the complete path of the robot.

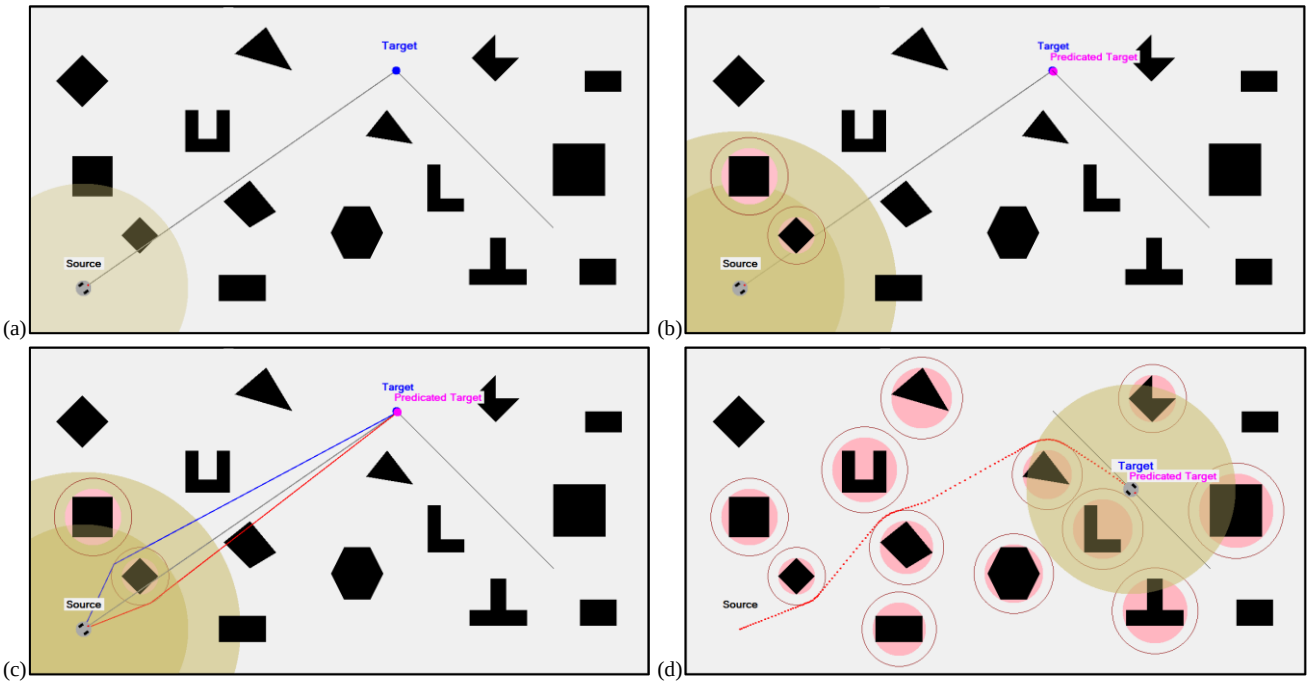

Fig.11 Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm with straight line target movement (robot sensing radius = 200 pixels): (a) path planning for the initial position of target; (b) path planning for the next position of target; (c) determine the shortest path to the next position of target; (d) the complete path of the robot.

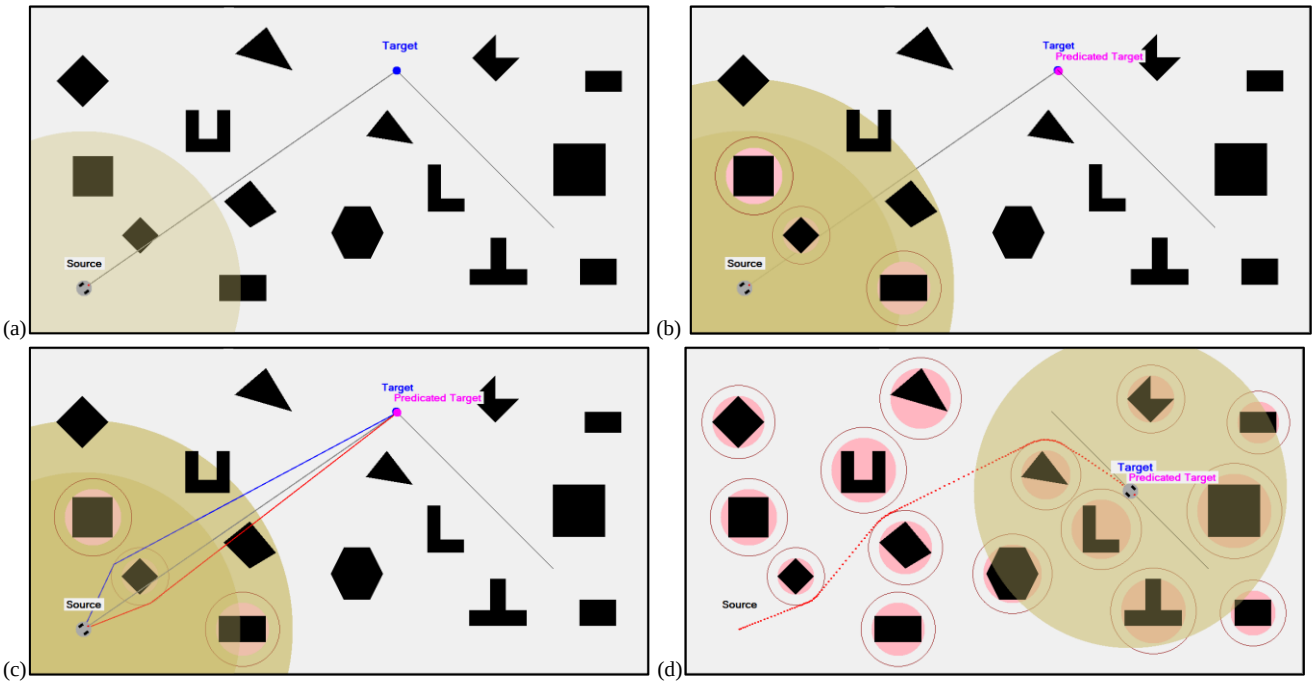

Fig.12 Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm with straight line target movement (robot sensing radius = 300 pixels): (a) path planning for the initial position of target; (b) path planning for the next position of target; (c) determine the shortest path to the next position of target; (d) the complete path of the robot.

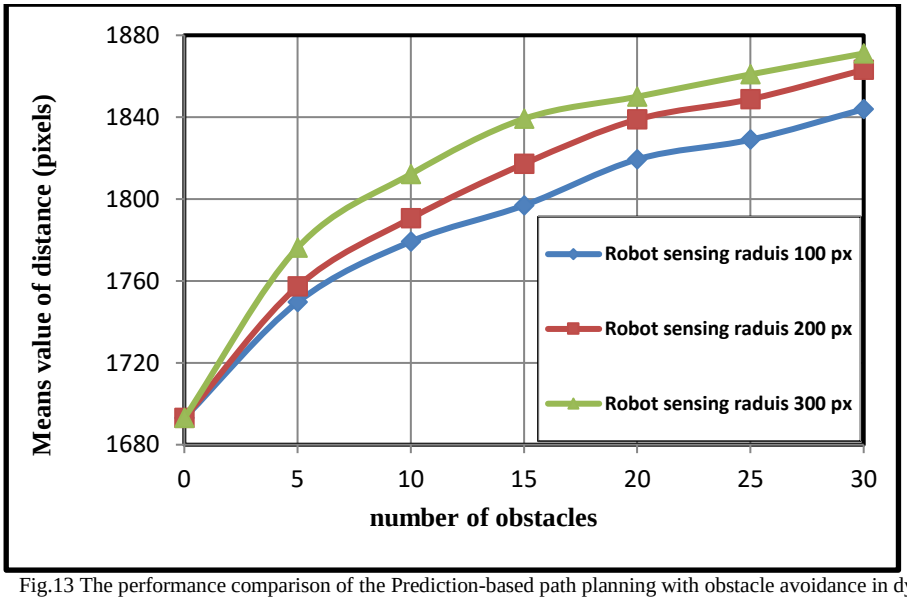

Fig.13 The performance comparison of the Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm for straight line target movement with a different number of obstacles and different sensing range.

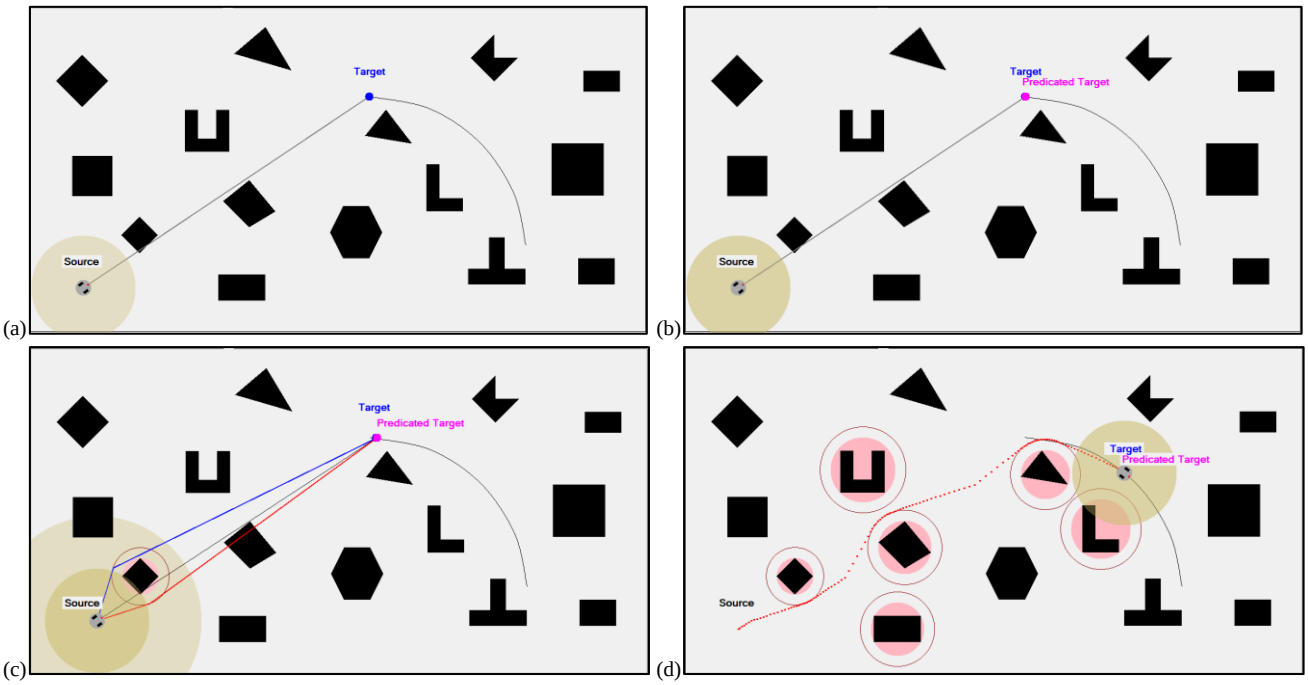

Fig.14 Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm with circular arc target movement (robot sensing radius = 100 pixels): (a) path planning for the initial position of target; (b) path planning for the next position of target; (c) determine the shortest path to the next position of target; (d) the complete path of the robot.

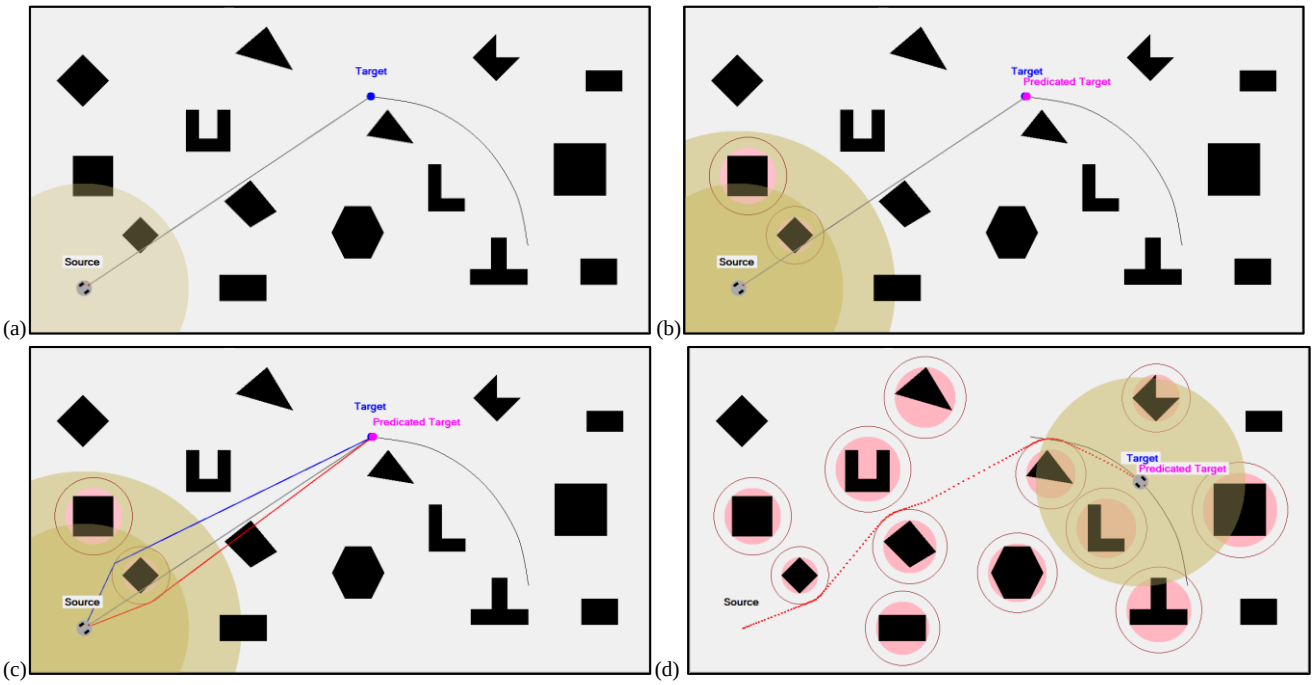

Fig.15 Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm with circular arc target movement (robot sensing radius = 200 pixels): (a) path planning for the initial position of target; (b) path planning for the next position of target; (c) determine the shortest path to the next position of target; (d) the complete path of the robot.

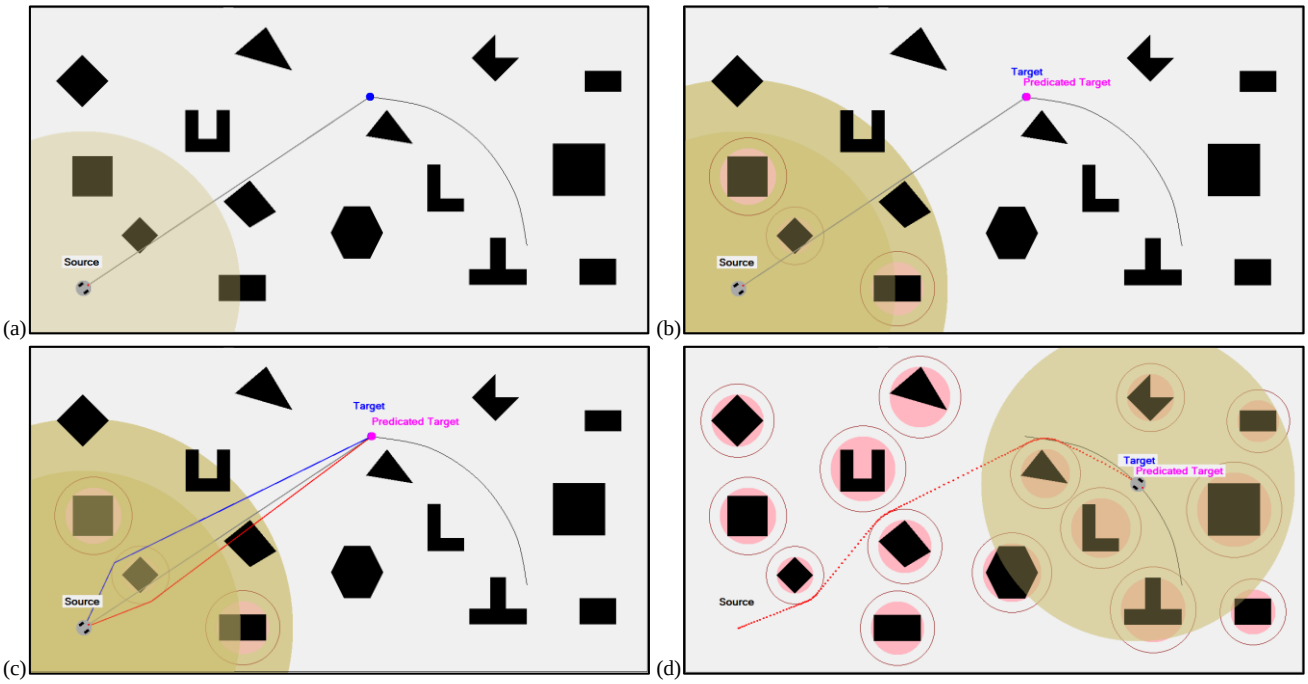

Fig.16 Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm with circular arc target movement (robot sensing radius = 300 pixels): (a) path planning for the initial position of target; (b) path planning for the next position of target; (c) determine the shortest path to the next position of target; (d) the complete path of the robot.

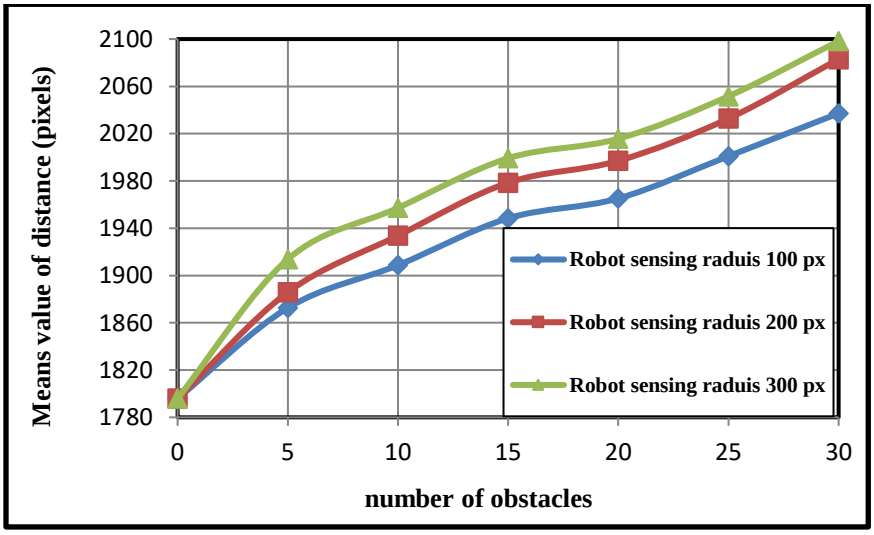

Fig.17 The performance comparison of Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm for circular arc target movement in an environment with a different number of obstacles and for different sensing range.

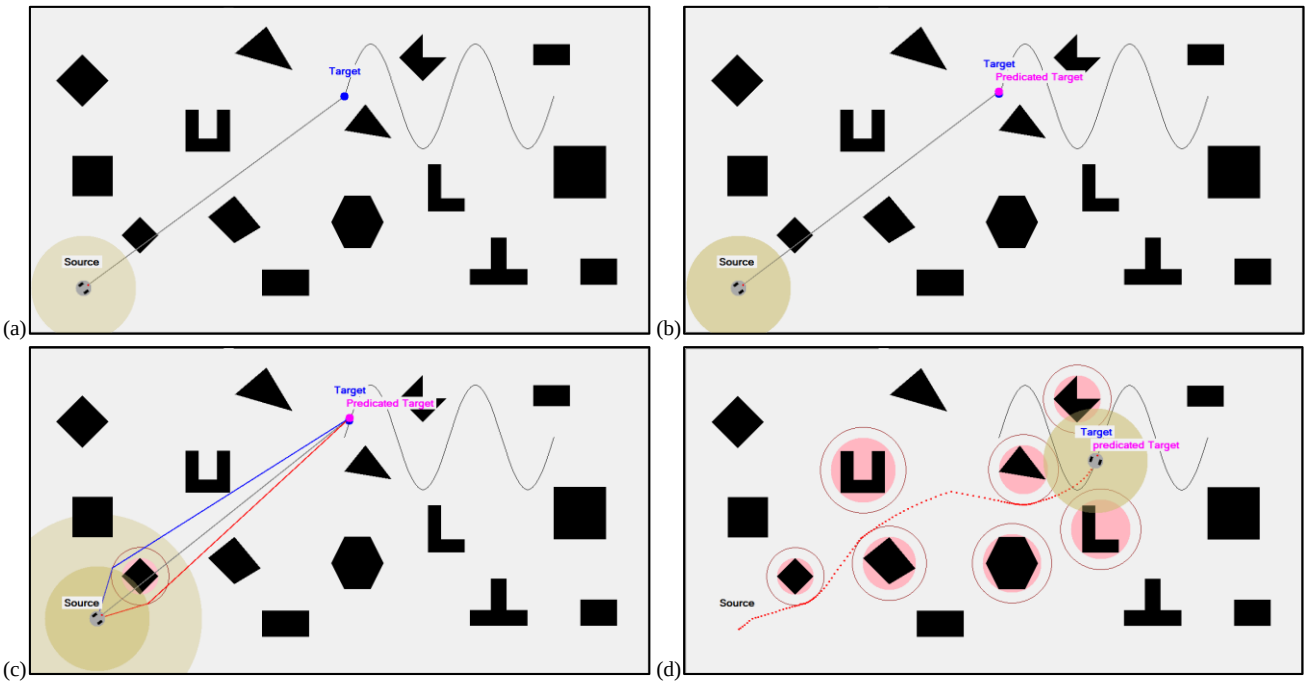

Fig.18 Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm with sine wave target movement (robot sensing radius = 100 pixels): (a) path planning for the initial position of target; (b) path planning for the next position of target; (c) determine the shortest path to the next position of target; (d) the complete path of the robot.

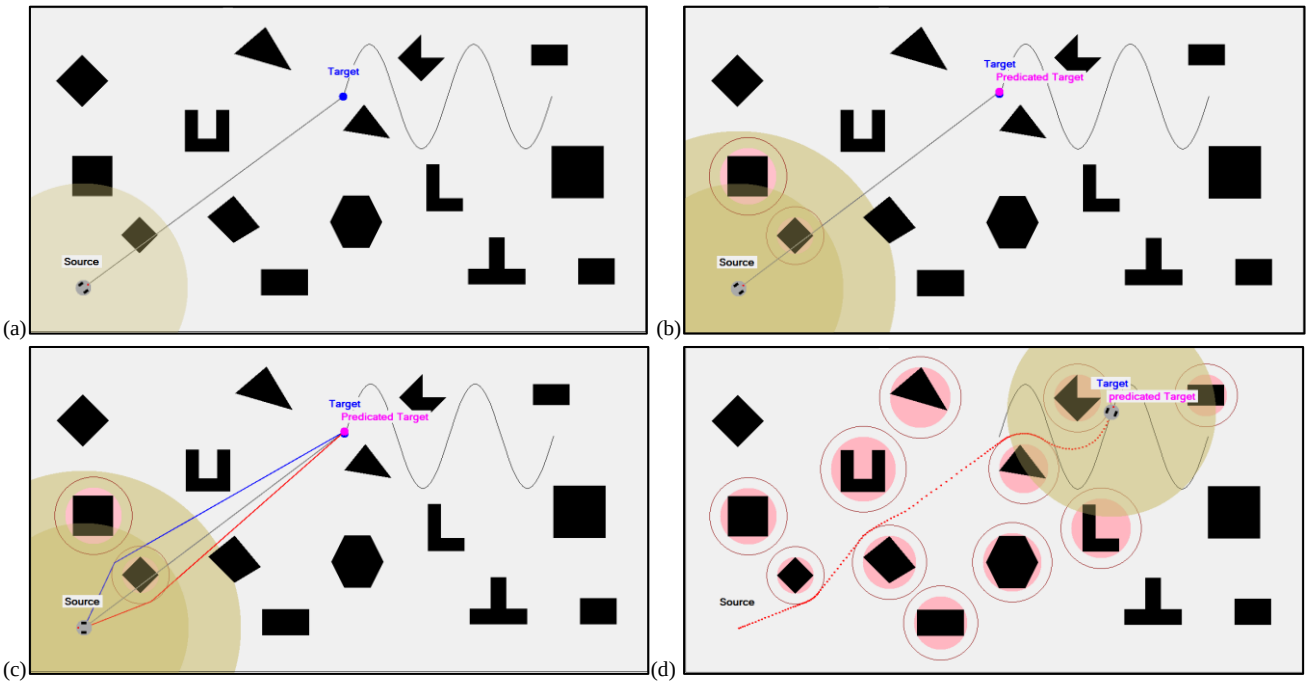

Fig.19 Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm with sine wave target movement (robot sensing radius = 200 pixels): (a) path planning for the initial position of target; (b) path planning for the next position of target; (c) determine the shortest path to the next position of target; (d) the complete path of the robot.

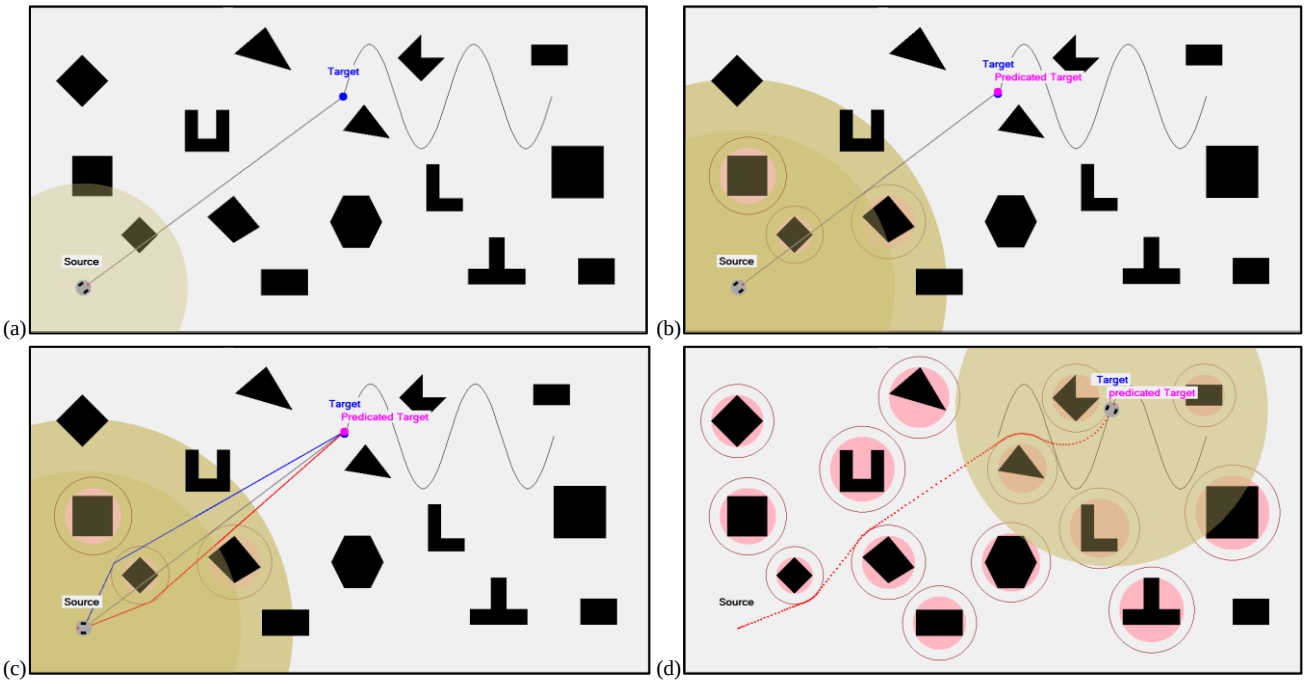

Fig.20 Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm with sine wave target movement (robot sensing radius = 300 pixels): (a) path planning for the initial position of target; (b) path planning for the next position of target; (c) determine the shortest path to the next position of target; (d) the complete path of the robot.

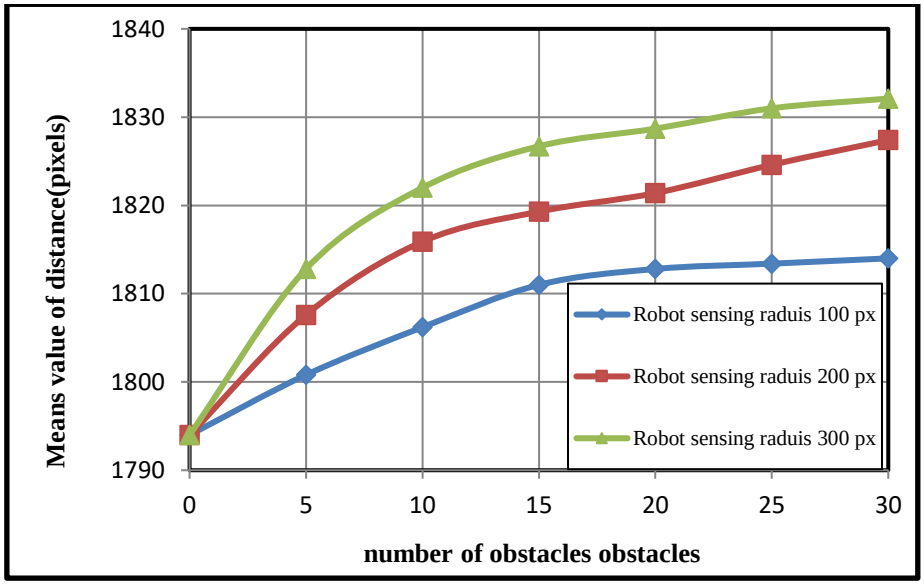

Fig.21 The performance comparison of the Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm for sine wave target movement in an environment with a different number of obstacles and for different sensing range.

V. Conclusion

This paper introduces path planning algorithm for mobile robot navigation in an environment with a dynamic target and polygonal obstacles. This algorithm which called Prediction-based path planning with obstacle avoidance in dynamic target environment applied in a straight line, circular arc, and sine wave target movement scenarios. The paper first part explains the low-level path planning which represented by the digital differential analyzer for line and arc drawing. We use digital differential analyzer algorithm because of its simplicity and integer calculations which reduce the required resources. The paper second part explain in details how the problem of path planning with dynamic target solved by prediction the next target location then searching for the shortest path from the current position to the predicted target. The explanation of the low-level and high-level of the introduced algorithm shows that it requires simple resources for implementation in the hardware system. The simulation results discuss different environment conditions such as obstacles number, shapes, and dimensions for each target movement scenario and the result compared to three different robot sensing radiuses. The main simulation result shows that by using the introduced algorithm the robot reaches the target in all target movement scenarios. Prediction-based path planning with obstacle avoidance in dynamic target environment algorithm has good performance using low robot sensing radius for different obstacles number, shapes, and dimensions.

VI. References

[1] J.F. Canny, J.M. Malik, D.D. Edwards"Artificial Intelligence A Modern Approach",1995.

[2] S. Koenig, and M. Likhachev," Fast Replanning for Navigation in Unknown Terrain", IEEE TRANSACTIONS ON ROBOTICS, VOL. 21, NO. 3, JUNE 2005.

[3] A.Stentz," The Focussed D* Algorithm for Real-Time Replanning", International Joint Conference on Artificial Intelligence, August 1995.

[4] A.T. Rashid, A. A. Ali, M. Frasca, and L. Fortuna," Path planning with obstacle avoidance based on shortest distance algorithm", Robotics and Autonomous Systems, Vol. 61, No. 12, p.p. 1440-1449, 2013.

[5] C.Undeger, and F.polat," Real-Time Edge Follow: A Real-Time Path Search Approach", IEEE Transactions on Systems Man and Cybernetics Part C, October 2007

[7] T.Ishida, and R.E. Korf,"Moving-Target Search: A real-Time Search for Changing Goals",IEEE TRANSACTION ON PATTERN ANALYSIS AND MACHINE INTALIGENT ,Vol.17,NO.16,JUNE 1995

[8] T.Uras, S.Koenig, and C.Hern´andez" Subgoal Graphs for Optimal Pathfinding in Eight-Neighbor Grids", Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, 2013

[9] D.Nussbaum, and A.Yörüçkü," Moving Target Search with Subgoal Graphs", Proceedings of the Eighth International Symposium on Combinatorial Search (SoCS-2015).

[10] C.Hern´andez,and J.A. Baier," Fast Subgoaling for Pathfinding via Real-Time Search", Association for the Advancement of Artificial Intelligence,2011.

[11] Y.Björnsson, Ramon Lawrence,and V. Bulitko," Case-Based Subgoaling in Real-Time Heuristic Search for Video Game Pathfinding", Journal of Artificial Intelligence Research,September 2010.

[12] D.Hearn, M.P.Baker,Computer Graphics C (1996).

[13] L.Moroz, J.L. CieVliNski, M.Stakhiv and V.Maksymovych, "A Comparison of Standard One-Step DDA Circular Interpolators with a New Cheap Two-Step Algorithm", Hindawi Publishing Corporation Modelling and Simulation in Engineering, 2014.

[14]] J.L. Cie´sli´nski and L.V.Moroz, "Fast exact digital differential analyzer for circle generation", 2013.

[15] E.Welzl, "Smallest enclosing disks (balls and ellipsoids)", APPEARED IN"New Results and New Trends in Computer Science 555, p.p. 359-370, 1991.