Automation, Control and Intelligent Systems
2013; 1(2) : 16-23
Published online April 2,2013 (http://www.sciencepublishinggroup.com/j/acis)
doi: 10.11648/j.acis.20130102.11

Dynamic obstacle avoidance in multi-robot motion planning using prediction principle in real environment

Suparna Roy[1], Dhrubojyoti Banerjee[1], Chiranjib Guha Majumder[1], Amit Konar[1], R. Janarthanan[2]

[1]ETCE Department, Jadavpur University, Kolkata-700032
[2]Jaya College of Engineering, Chennai, Tamil Nadu

Email address:

suparna.ry03@gmail.com (S. Roy), dhrubo_jyoti_banerjee@yahoo.co.in (D. Banerjee), cguhamajumder@yahoo.com (C. G. Majumder), konaramit@yahoo.co.in (A. Konar), srmjana_73@yahoo.com (R. Janarthanan)

To cite this article:

Suparna Roy, Dhrubojyoti Banerjee, Chiranjib Guha Majumder, Amit Konar, R. Janarthanan. Dynamic Obstacle Avoidance in Multi-Robot Motion Planning Using Prediction Principle in Real Environment, *Automation, Control and Intelligent Systems.* Vol. 1, No. 2, 2013, pp. 16-23. doi: 10.11648/j.acis.20130102.11

Abstract: This paper provides a new approach to the multi-robot path planning problem predicting the position of a dynamic obstacle which undergoes linear motion in the given workspace changing its direction at regular intervals of time. The prediction is done in order to avoid collision of the robots with the dynamic obstacle. First the work is done in simulation environment then the entire work has been implemented on Khepera II mobile robot. The performance of the above mentioned approach has been found to be satisfactory compared to the classical non-predictive approaches of dynamic obstacle avoidance.

Keywords: Linear Prediction, Particle Swarm Optimization, Multi-Robot Motion Planning

1. Introduction

Mobility is an important aspect of modern robots. Several approaches to mobility management of a mobile robot have been studied over the last four decades. Some of these popular methods for path planning include Voronoi Diagram, A* Heuristic Algorithm, Neural Networks, Fuzzy algorithms. Since the beginning of this decade, researchers are taking keen interest to study the scope of optimization algorithms in mobility management of robots. The justification of using optimization techniques arises particularly when the motion planning of a number of robots are considered together in the same workspace. This paper attempts to overcome one fundamental problem in multi-agent robotics.

One interesting problem in multi-agent robotics is Multi-robot Motion planning[1][2], where the robots have to determine their trajectory of motion from predefined starting point to fixed goal point without hitting any obstacles in the environment. Most of the multi-robot motion planning algorithms developed so far only considered static obstacles. The multitude of the Multi-robot Motion Planning problem grows significantly, when one or more dynamic obstacles are present in the scenario. This paper addresses one such problem of Multi-robot Motion Planning taking into consideration the predictive motion of the dynamic obstacle.

The prediction logic employed here assumes a linear motion of the dynamic obstacle within a short span of time. When the speed of the dynamic obstacle is relatively slower than that of the robots, we may presume that the linear motion of the obstacle is maintained within two successive sampling instances of the obstacle by the robots. This consideration provides a new opportunity to formulate the multi-robot motion planning as an optimization problem.

The formulation of the problem is concerned with designing an objective function considering two important issues. The first issue refers to determining the minimum distance between each robot and its respective goal without hitting any static obstacle during the trajectory planning by the robot team. The second issue deals with maximization of the distance between a dynamic obstacle and its nearest robot. These two objectives are put together to construct a single objective optimization function, which here has been optimized by the well known Particle Swarm Optimization (PSO) algorithm.

After predicting the location of the dynamic obstacle, the robot takes the decision of the next position accordingly, the advantage being improved efficiency. The distributed approach to solve the path planning problem has been un-

dertaken in this paper. Here we consider n-iterative algorithms for n robots, and the i^{th} algorithm determines the next position for the i^{th} robot, satisfying the necessary constraints.

The rest of the paper is organized as follows. Section II considers a formulation of the problem presuming static and dynamic obstacles separately, and then combines them together to construct a general objective function for the problem. The prediction approach has been incorporated here. The principle of PSO is outlined in section III. Section IV provides the algorithm to solve both the static and dynamic obstacle avoidance problem by using PSO. Computer simulation over simulated platforms is given in section V. Online experimental details with snapshots as well as the overview of the Khepera Robot used in the experiment are discussed in section VI. Performance analysis with respect to two standard metrics is undertaken in section VII. Inferences drawn from the paper are given in section VIII with a tinge of references at the end.

2. Formulation of the Problem

Here we evaluate the next position of the robots from their current position in a given robot's world map with a set of static obstacles and one dynamic obstacle. A set of principles listed below is first developed to formalize the path-planning problem by a uniform treatment.

A. Pre-assumptions

Current position of each robot is known with respect to a given reference in the Cartesian-coordinate system. The robots have a fixed set of actions for motions. A robot can select one action at a given time. Obstacles are detected by their colour which is known to the robots. The robots can come to know of the next position of the dynamic obstacle by using a prediction approach.

B. Principles

i. A robot always attempts to align itself towards the goal position by calculating a optimal path [3] calling PSO.

ii. In each step the robot tries to predict the location of the dynamic obstacle moving in a linear path. The introduction of the linearity in the path for a small amount of time of the dynamic obstacle is done so that the prediction-based approach is satisfied.

iii. On detecting a static obstacle near it, the robot deviates from its current position to a next obstacle free position following a minimal path obtained by PSO algorithm.

iv. After predicting a dynamic obstacle, the robot has to move from its current position to a next obstacle free position following a minimal path obtained by the PSO algorithm.

Let (x_i, y_i) be the current position of the i^{th} robot at time t, $(x_{i'}, y_{i'})$ be the next position of the same robot at time (t+1). V_i be the current velocity of the i^{th} robot.

Let (x_{ig}, y_{ig}) be the goal position of the i^{th} robot. It is evident from Fig.1 that

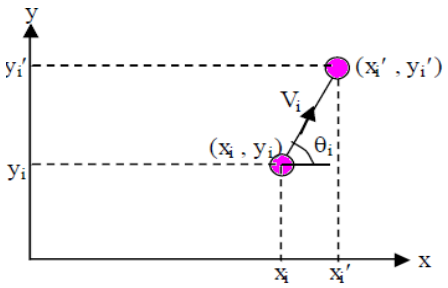

Figure 1. *Current and next position of the i^{th} robot.*

$$x_i^{'} = x_i + v_i \cos\theta_i \Delta t \tag{1}$$

$$y_i^{'} = y_i + v_i \sin\theta_i \Delta t \tag{2}$$

Where Δt =1,

C. Distributed Planning

The constraint for the i^{th} robot can be formulated as follows:

Let F be an objective function for the i^{th} robot that determines the length of the trajectory. For n number of robots, then

$$F = \sum_{i=1}^{n} \{v_i + \sqrt{\{(x_i + v_i \cos\theta_i - x_{ig})^2 + (y_i + v_i \sin\theta_i - y_{ig})^2\}}\} \tag{3}$$

The robot is predicting its next position avoiding the dynamic obstacle. The objective function incorporating the prediction principle is calculated as

$$F^{'} = F + \sum_{i=1}^{n} \sqrt{(x_p - x_{ig})^2 + (y_p - y_{ig})^2} \tag{4}$$

Where (x_p, y_p) is predicted next position of the dynamic obstacle. Now $x_p = x_{i'} + u\Delta t^{'}$ and $y_p = y_{i'} + v\Delta t^{'}$.

Where (x_i, y_i) is the current position of the dynamic obstacle.

$\Delta t^{'}$ is the sampling time where $\Delta t = arr[i] / arr[j]$, in arr[i] the total time needed for the movement of the dynamic obstacle is kept and the number of steps undertaken by the dynamic obstacle is stored in arr[j]. Here u stands for the velocity of the dynamic obstacle. The prediction made here is linear because extrapolation of the path of the dynamic obstacle is made considering the direction of motion constant over a given interval of time. The distance between robots at any point of time should not be less than a predefined threshold to avoid collision; this logic is used as a primary constraint to this problem.

In fig.3 D_i represents the current position of the dynamic obstacle [8] whereas D_p denotes the predicted position

of the same. R denotes the robot centroid whereas G denotes the goal centroid. The distance between the dynamic obstacle's predicted position and robot's current position is to be maximized to avoid any collision. Thus it becomes an objective function maximization problem. Let d_{ij} be the distance between i^{th} and j^{th} robots' current positions, $d_{i'j'}$ be the distance between i^{th} and j^{th} robots' next positions, then the constraint that the robot will not hit its kin is given by $d_{i'j'} - 2r \geq \in$, where r denotes the radius of the robots and $\in$ (>0) denotes a small threshold.

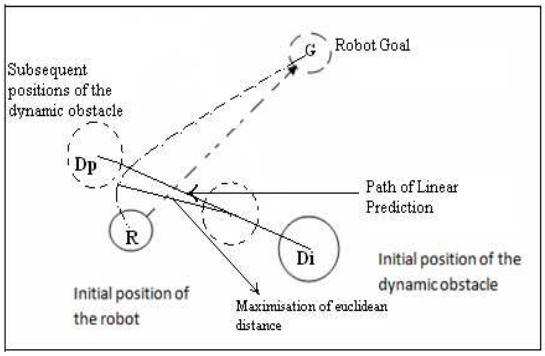

Figure 3. *Linear prediction of the path of the dynamic obstacle and the maximization of the Euclidean distance between robot current position and predicted next position of dynamic obstacle.*

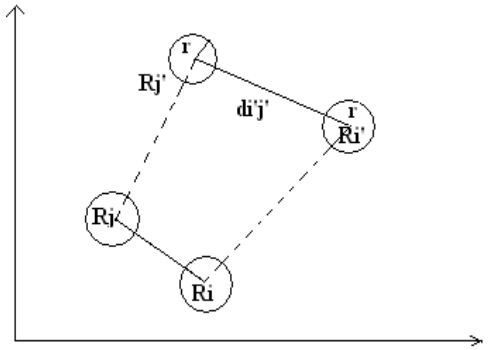

Figure 4. *Diagram showing the constraint such that no two robots can collide with each other*

The multi-robot path-planning as an optimization problem includes an objective function, concerning minimization of the Euclidean distance between the current positions of the robots with their respective goal positions, constrained by obstacles and other robots on the path. Thus, the constrained optimization problem in the present context for the i^{th} robot is given by,

$$F_i = \sum_{i=1}^{n} \{v_i + \sqrt{((x_i + v_i \cos\theta_i - x_{ig})^2 + (y_i + v_i \sin\theta_i - y_{ig})^2)}\} + f_{dp} \sum_{i'j'=1}^{n(n-1)/2} (\min_{\forall_i}(0,(d_{i'j'} - (2r+\varepsilon)))^2 + f_{st}/d_{i-obs} + f_{dy}/(n\times(R+r)) + 1/\sum_{i-1}^{n}\sqrt{(x_p - x_{ig})^2 + (y_p - y_{ig})^2} \quad (5)$$

where f_{dp} (>0) and f_{st} (>0) denote scale factors to the second and third terms in the right hand side of expression (5) d_{i-obs} represents the distance of the obstacle from the i-th robot, R is the radius of the dynamic obstacle.

1. Static obstacle

Consider the robot Ri is initially located at (x_i, y_i) .It needs to select point $(x_{i'}, y_{i'})$, i.e. next position of the robot, such that the line joining $\{(x_i, y_i), (x_{i'}, y_{i'})\}$ and $\{(x_{i'}, y_{i'})\ (x_g, y_g)\}$ do not touch the obstacle, as shown in Figure. 2. This is realized with PSO algorithm, such that it will always select a minimal path [9] to reach the respective destinations. To take case of static obstacles in the environment, we add one penalty function to the constrained objective function (5).Thus the present constraint optimization problem is transformed to –

$$F = \sum_{i=1}^{n} v_i + \sqrt{((x_i + v_i \cos\theta_i - x_{ig})^2 + (y_i + v_i \sin\theta_i - y_{ig})^2)} + f_{dp} \sum_{i'j'=1}^{n(n-1)/2} (\min(0,(d_{i'j'} - (2r+\in))))^2 + f_{st} \quad (6)$$

Where f_{st} = positive constant when a static obstacle is present on the planned local trajectory = 0, otherwise.

2. Dynamic obstacle

The main objective of the prediction principle is to anticipate the next position of the dynamic obstacle [4]. Thus the robot can take action accordingly and call PSO to move back to its optimal path.

Consider the robot Ri is initially located at (x_i, y_i) .It needs to select point $(x_{i'}, y_{i'})$, i.e. next position of the robot, such that the line joining $\{(x_i, y_i), (x_{i'}, y_{i'})\}$ and $\{(x_{i'}, y_{i'}), (x_g, y_g)\}$ do not touch the obstacle, as shown in fig. 2.

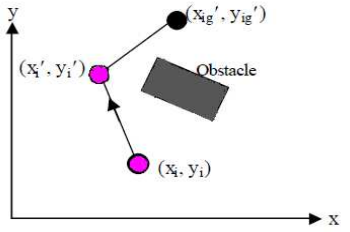

Figure 2. *Selection of* (x_i', y_i') *from* (x_i, y_i) *to avoid collision with an obstacle.*

To take case of dynamic obstacles in the environment, we add one maximization function. Thus the present constraint optimization problem is transformed to

$$F = \sum_{i=1}^{n} v_i + \sqrt{((x_i + v_i \cos\theta_i - x_{ig})^2 + (y_i + v_i \sin\theta_i - y_{ig})^2)} + 1/\sum_{i-1}^{n}\sqrt{(x_p - x_{ig})^2 + (y_p - y_{ig})^2} + f_{dy} \quad (7)$$

f_{dy} = a positive constant when a dynamic obstacle is predicted =0, otherwise.

3. The Particle Swarm Optimization

The PSO scheme has the following algorithmic parameters:

1) $V_{\max}$ or maximum velocity which restricts $\vec{V}_i(t)$ within the interval $[-V_{\max}, V_{\max}]$.

2) An inertial weight factor ω.

3) Two uniformly distributed random numbers φ1 and φ2 which respectively determine the influence of $\vec{p}(t)$ and $\vec{g}(t)$ on the velocity update formula.

4) Two constant multiplier terms C_1 and C_2 known as self confidence and swarm confidence respectively [5].

Initially the settings for $\vec{p}(t)$ and $\vec{g}(t)$ are $\vec{p}(0) = \vec{g}(0) = \vec{x}(0)$ for all particles. Once the particles are initialized, the iterative optimization process begins where the positions and velocities of all the particles are altered by the following recursive equations. The equations are presented for the d-th dimension of the position and velocity of the i-th particle.

$$\begin{aligned} V_{id}(t+1) &= \omega V_{id}(t) + C_1\varphi_1(P_{id}(t) - X_{id}(t)) \\ &+ C_2\varphi_2(g_d(t) - X_{id}(t)) \\ X_{id}(t+1) &= X_{id}(t) + V_{id}(t+1) \end{aligned} \quad (8)$$

The first term in the velocity updating formula represents the inertial velocity of the particle. The second term involving $\vec{P}(t)$ represents the personal experience of each particle and is referred to as 'cognitive part'. The last term of the same relation is interpreted as the 'social term' which represents how an individual particle is influenced by the other members of its society. Typically, this process is iterated until some acceptable solution has been found by the algorithm. Once the iterations are terminated, most of the particles are expected to converge to a small radius surrounding the global optima of the search space.

4. Solving the Constraint Optimization

Problem Using Pso

Pseudo Code:

Input: Initial position (x_i, y_i), goal position (x_{ig}, y_{ig}) and velocity v_i for n robots where 1<=i<=n and a threshold value €.

Output: Trajectory of motion P_i for each robot R_i from (x_i, y_i) to (x_{ig}, y_{ig})

Begin

Set for all robot i

$x_{icurr} \leftarrow x_i$; $y_{icurr} \leftarrow y_i$ //current position in both x & y coordinate of i^{th} robot//

For robot i=1 to N

Repeat

Check obstacle ( ) //at each (x_{icurr}, y_{icurr}) the robot rotates 360º with radius n*(R+r) [n >1] to search for obstacle//

IF static obstacle

Move away and call PSO // to find the next obstacle free optimal position //

Predict dynamic obstacle's next position

Compute sampling time // delta t that is time taken to undergo one step//

Update xp // next position of obstacle(dynamic) //

$$x_p = x_i' + u\Delta t \quad y_p = y_i' + u\Delta t .$$

Maximize the distance between robot(current->pos) and dynamic obstacle(predicted next->pos). // to avoid collision//

Call PSO

Move to -> (x_{inext}, y_{inext}) ;

// moves to next obstacle free position. //

$x_{icurr} \leftarrow x_{inext}$, $y_{icurr} \leftarrow y_{inext}$; //update the position//

Until $\|curr\_i - G_i\| \le \varepsilon$ // $curr\_i = (x_{curr\_i}, y_{curr\_i})$, Gi = (x_{ig}, y_{ig})//

End for;

End.

Procedure PSO

$(x_{icurr}, y_{icurr}, pos-vector)$

Begin

initialize 10 particles with random position and velocity;

For k < Maxiter do

Begin

1. update V_i and X_i i by (8);

2. determine local best position and global best position of

the particles;

End for;

Update:

$$x_{curr-i} \leftarrow x_{curr-i} + v_i \cos\theta_r$$

$$y_{curr-i} \leftarrow y_{curr-i} + v_i \sin\theta_r$$

Return;

End.

5. Experiments Using Computer Simulation

In this section, we provide the results of computer simulations of the proposed scheme of multi-robot motion planning avoiding both static and dynamic obstacles.

The multi-robot path-planning is realized in Turbo C environment on a Pentium processor. The number of robots (n) was varied from two to fourteen and the performance of the system was evaluated. The static obstacles are represented by Dark gray colour while the dynamic obstacles are represented by Orange colour in the world-map. The following figures show the screenshot of the program. The configuration of the world map with robots, dynamic obstacles and five static obstacles at different instant of time during the execution of the code is depicted in the following figures. The number of robots and the velocity of robots are changed to get different conditions and make sure in no cases the robots collide with any obstacle. The trajectories of the robot path avoiding both static and dynamic obstacles are clearly visible. The intermediate positions as well as the final position of the world map are shown where the robots are seen to reach their predefined goal safely.

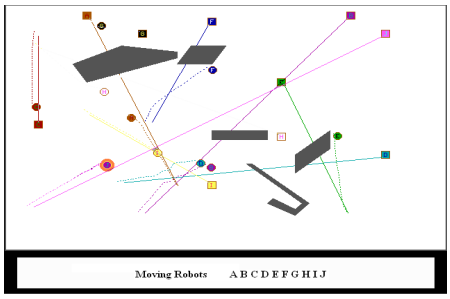

Figure 5. *Screenshot showing the Collision of the robot J with the orange coloured Dynamic obstacle in the world-map in classical non-prediction approach.*

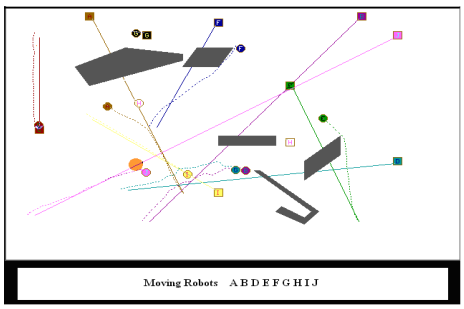

Figure 6. *Screenshot showing the path of Dynamic obstacle avoidance by the robot J in the world-map when 10 robots together are considered and prediction principle is used.*

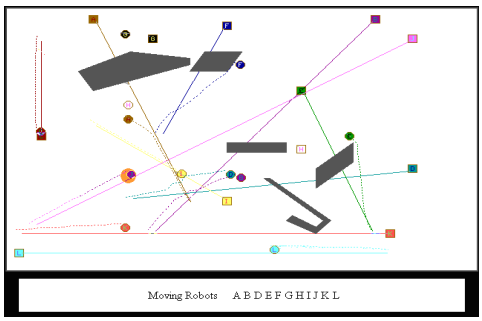

Figure 7. *Screenshot showing the Collision of the robot J with the orange coloured Dynamic obstacle in the world-map when 12 intelligent robots are considered at a time, in classical non-prediction based approach.*

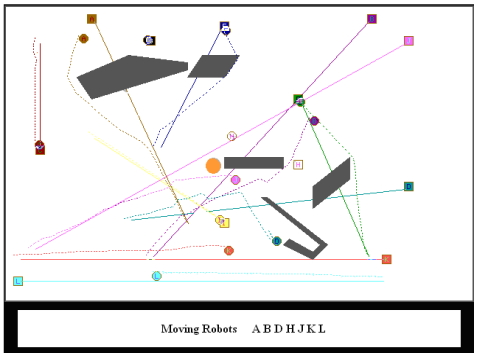

Figure 8. *Screenshot showing the path of Dynamic obstacle avoidance by the robot J and trajectories of eleven other robots in the world-map when 12 robots together are considered and prediction principle is undertaken.*

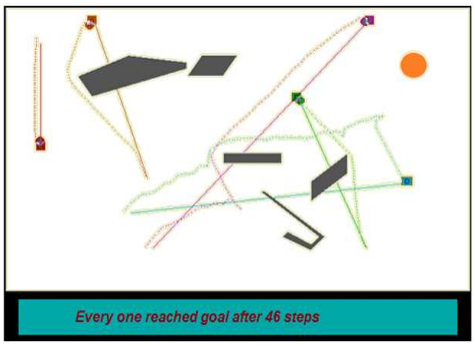

Figure 9. *Screenshot showing the complete path of the robots in the world map after the robots reached their goals and the dynamic obstacle has passed.*

6. Experiment Using Khepera-Ii Robot

Khepera II as shown in figure 10 is a miniature robot of diameter 7 cm and equipped with 8 in-built infrared proximity sensors, and 2 relatively accurate encoders for the two motors. The range sensors are positioned at fixed angles and have limited range detection capabilities[6][7]. The sensors are numbered from 0 to 7, with the leftmost sensor designated by 0, and the rightmost by 7. The sensor numbered 7 and 8 are on the back side of Khepera robot and not used in this experiment. The Khepera model we used is a table-top robot, connected to a workstation through a wired serial link.

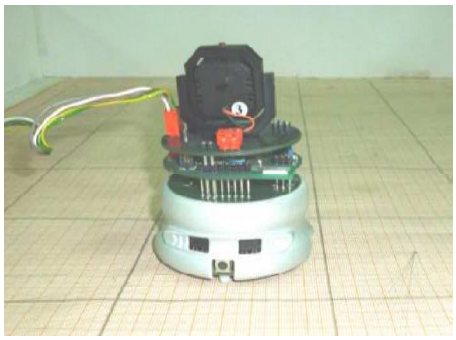

Figure 10. *The Khepera II Robot.*

Now during online execution of the robot path traversal, both the robots are placed in their initial positions. They

have to reach their goal avoiding all static and dynamic obstacles present in the world-map. The dynamic obstacles are pink in colour and move in the world map randomly. Sensors and camera are the main source of information. These data acquisition tools are already embedded in the robot.

The optimal path to the goal is decided by the PSO algorithm and static obstacles are detected by the sensory value. Here we have taken account of the first six sensors of the Khepera-II Mobile Robot. So the sensory data set is given by R={*R*0,*R*1,*R*2,*R*3,*R*4,*R*5}.The robots are programmed in such a way that whenever a static obstacle is in the robot's vicinity the reflected sensory value is more than the predefined threshold which denotes presence of obstacle. So the robots deviate from their path to avoid the obstacle and then again come back to its optimal path thus avoiding collision.

The dynamic obstacle as well as the kin is detected by its colour. The vision system generates information corresponding to the result of the processed image captured by the camera of the functional robot. After the online capture of images by the Khepera robot the set of images are sent to the PC for colour recognition. This is done so that the co-operating agent can recognize its kin as well as dynamic obstacle. The colour is expressed as an RGB triplet (r,g,b) each component of which can vary from 0 to a defined maximum value.

In this problem, the dynamic obstacle is pink in color and is identified through color image processing using Vector Distance approach. The images of.bmp format being of pink colour, the individual components of the RGB triplet will result in a value which is more than the predefined threshold for pink colour. Now, while searching for pink the program in the PC connected to the co-operating robot sets a tolerance or a threshold so that the images or colour close to pink can be recognized. The same procedure is applied for kin recognition which is done by identifying black colour of khepera robot. The only difference is that the images of.bmp format being of black colour, the individual components of the RGB triplet will result in zero value ideally.

That is the system works for the following situations in case of kin recognition:

$$\forall I \in P \cup P_{close\_to}$$

where I represents the set of images captured and B denotes pink set and P_{close_to} near to pink.

In practical world, the nearness factor is considered to remove the constraint of ideality of any particular colour.

The Euclidian distance between those two vectors is computed as follows:

$$D = \sqrt{(R-t)^2 + (G-t)^2 + (B-t)^2} \qquad (9)$$

where t is the tolerance limit considered for near to pink factor.

The pseudo code for dynamic obstacle recognition is given below.

Pseudo-code :
Begin
{
For each pixel
{
Compute the distance between that pixel and the reference color (D)(Pink and black here) //calculation of distance vector//
If D < Threshold //Thresholding operation//
Then Current pixel is accepted as pink or black //selection of pink and black colour
else
Current pixel is not accepted
}
End

The real time experiment of multi-robot motion planning was supposed to be done with varying number of robots from two to twelve as done in case of simulation, but here for the sake of simplicity and ease of experimental set-up only two robots are considered. The white colored objects are the static obstacles, the pink colored ball being the dynamic obstacle is moved randomly in the world map. Figure 11 to 14 shows the actual image of the complete world map during the time of path planning in real environment.

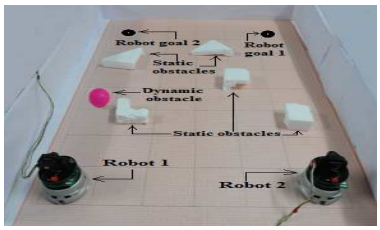

Figure 11. *The complete world-map showing the initial position of the Robots (R), the static obstacles and the dynamic obstacle, the goal (G) position of the respective robots and the boundary.*

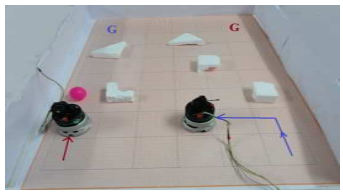

Figure 12. *The complete world-map showing the Robots (R) during online motion, here the robot-1 detects the dynamic obstacle in front of it as a result it will turn 90º to its right and continue its motion.*

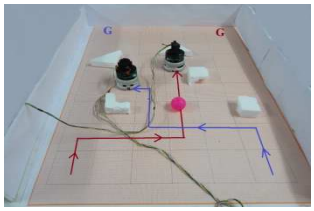

Figure 13. *The complete world-map showing the Robots (R) during online motion, its trajectory of motion of the robot-1 isshown is red line and that of of robot-2 is shown in blue line,the obstacles, the goal (G) and the boundary*

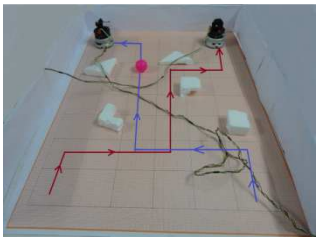

Figure 14. *The complete world-map showing the Robot(R) has reached its goal (G) successfully. The trajectory of the path is also shown.*

7. Performance Analysis

The performance of the prediction based system can be evaluated in many ways. Performance evaluation is a wide topic, and covers many techniques to measure the quality of the system. Different performance evaluation metrics covers different areas, which include how algorithms cope in different physical conditions in the scene, i.e. changing number of robots and obstacles, changing velocity of the robots, change in position of the obstacles etc. Two metrics average path deviation and average uncovered target distance have been considered for evaluating the performance of the system.

Average Uncovered Target Distance (AUTD)

Given a goal position G_i and the current position C_i of a robot on a 2-dimensional workspace, where G_i and C_i are 2-dimensional vectors, the uncovered distance for robot i is $\| G_i - C_i \|$, where $\| . \|$ denotes Euclidean norm. Initially when the robots start from its starting position the uncovered Target Distance is maximum and it decreases as the robots trends towards their respective goals. As soon as the robots reach their goals UTD becomes zero. For n robots, uncovered target distance (UTD) is the sum of $\|G_i - C_i\|$ i.e. UTD = $\sum_{i=1}^{n} \|G_i - C_i\|$.

Now, for k runs of the program, the average of UTDs is evaluated and it is called the average uncovered target distance (AUTD). For all experiments conducted in this study, k = 10 is considered.

Average Path Deviation (APD)

When no obstacles are present in the world map then the robots will follow the minimum path decided by the PSO algorithm. Whenever static and dynamic obstacles are introduced in the world map the robots deviate from their optimal path in order to avoid collision. Average path deviation is the difference between these two paths. For more number of robots the average path deviation will be the sum of individual APD for separate robots. The average path deviation should be as low as possible for better performance.

Let the average time taken by the robots to reach their goal in presence of no obstacle be t. Now, if the optimal path traversed by the robot be $D_{traversed}$ and the actual path between the robot and the goal in the presence of static and dynamic obstacles be $D_{static} + D_{dynamic}$, the Average Path Deviation is given by.

$$APD = | D_{traversed} - (D_{static} + D_{dynamic}) |$$

Figure.15 and figure.16 reflect that the uncovered target distance increases with the increasing number of robots. This happens due to the congestion in the path of the robots at the initiation of the iteration with more number of robots. The initial position of the dynamic obstacle is different in figure.15 from figure.16. So the predicted position of the dynamic obstacle is also different in each case. As a result the trajectories of the robots are also different in each case.

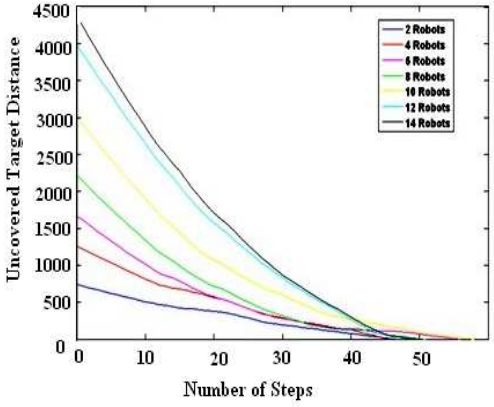

Figure 15. *Plot of Uncovered Target Distance(UTD) vs Steps with number of robots as parameter and initial co-ordinate of the dynamic obstacle (200,300).*

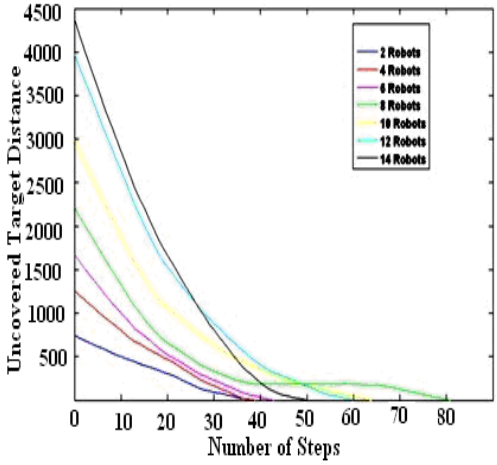

Figure 16. *Plot of Uncovered Target Distance (UTD) vs Steps with number of robots as parameter and initial co-ordinate of the dynamic obstacle (50,300).*

For a fixed uncovered target distance the number of steps undertaken by the robots decreases with increasing velocity of the robots, which is quite evident from the figure 17 and figure 18. This happens because robot with more speed reaches the goal fast.

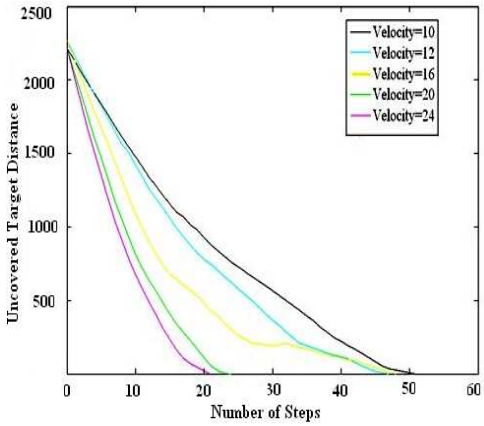

Figure 17. *Plot of Uncovered Target Distance(UTD) vs Steps with robot velocity as parameter and the initial co-ordinate of the dynamic obstacle (200,300).*

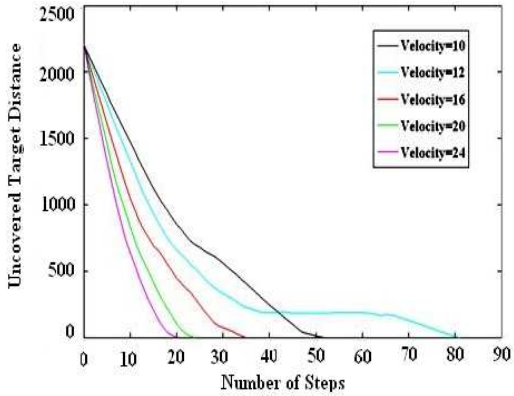

Figure 18. *Plot of Uncovered Target Distance vs Steps with robot velocity as parameter and the initial co-ordinate of the dynamic obstacle (50,300).*

The bar chart in figure.19 clearly shows that the number of steps covered by the robot to reach their goal in prediction based approach is much less than the approach without prediction.

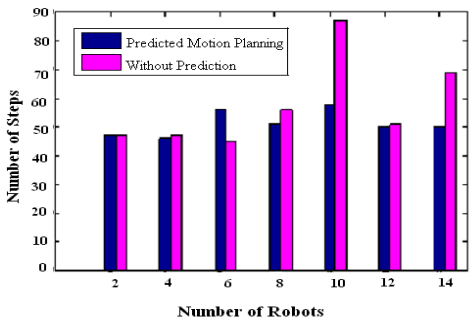

Figure 19. *Bar chart showing performance evaluation of robots with and without prediction.*

This proves the excellence of our work thereby increasing the efficiency of the system of dynamic obstacle avoidance thereby reducing the risk factor of collision.

8. Conclusion

We have introduced the concept of prediction to determine the next position of the dynamic obstacle. This prediction principle is also employed in online path planning using Khepera robot as well as in simulation apporach.In both the cases, the robots reach their goal without hitting any obstacle. The experimental results are in conformation with the fact that the prediction logic helps in minimizing the steps of motion of the robots, which encounter the dynamic obstacle within its trajectory compared to the non-prediction based approach. The prediction logic helps the robot to determine the location of the dynamic obstacle and plan its path accordingly minimizing the time of traversal. Thus both the number of steps as well as the execution time is minimized in the above problem. This is where our approach on multi-agent path planning amidst both static and dynamic obstacles supersedes the other works on multi-agent systems already existing in this domain.

References

[1] J. Kennedy, R. Eberhart, “Particle swarm optimization”, In Proceedings of IEEE International conference on Neural Networks. (1995) 1942-1948.

[2] Jayasree Chakroborty, Amit Konar, Aruna Chakroborty, “Multi-robot co-operation by Swarm and Evolutionary Algorithms”.

[3] M. Ryan, “Graph Decomposition for Efficient Multi-robot Path planning,”in Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 2003-2008, Jan. 2007.

[4] A.Fujimori and S. Tani, ‘A navigation of mobile robots with collision avoidance for moving obstacles’, in Proc. IEEE International Conference on Industrial Technology, Bangkok, Thailand, Dec. 2002, pp. 16.

[5] F. van den Bergh and A.P. Engelbrecht, “Cooperative learning in neural networks using particle swarm optimizers,” South African Computer Journal, 26:84-90, 2000.

[6] 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.

[7] 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.

[8] F. Kunwar, F. Wong, R. Ben Mrad, B. Benhabib, “Guidance-based online robot motion planning for the interception of mobile targets in dynamic environments”, Journal of Intelligent and Robotic Systems, Vol. 47, Issue 4, pp. 341-360, 2006.

[9] 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.