Prediction-based Dynamic Obstacle Avoidance in Multi-Robot Motion Planning

[1] Chiranjib Guha Majumder, [2]Suparna Roy, [3]Dhrubojyoti Banerjee,
ETCE Department, Jadavpur University, Kolkata-700032

[1] cguhamajumder@yahoo.com, [2]suparna.ry03@gmail.com [3]dhrubo_jyoti_banerjee@yahoo.co.in

***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. The performance of the above mentioned approach has been found to be satisfactory compared to the non-prediction based approach of dynamic obstacle avoidance.**

***Keywords*- Linear prediction, Particle swarm optimization, Motion planning**

I. INTRODUCTION

It has been a significant progress in the field of multi-agent robotics since the last ten years of the twentieth century. The multi-robot system outperforms the single robot system in some given objectives.

In this paper, we address one interesting problem in multi-agent robotics, where the robots in a given workspace have to plan the optimal trajectory (path) from a predefined starting position to the destination without colliding with neighbouring robots and avoiding both static and dynamic obstacles, if any, in their vicinity. The new approach of this paper is – the prediction of the location of the dynamic obstacle undergoing linear motion in the given world map. 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 undertaken 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 multi agent motion planning problem has been realized by Particle Swarm Optimization (PSO)[1][2] The *fitness function* of the PSO, has two main components:

1) The objective function describing the selection of next position on an optimal trajectory.

2*)* The constraint representing collision avoidance with the fellow robots and both static and dynamic obstacles.

The paper is divided into six major sections. Section I presents a brief introduction about our work. Section II provides a formulation of the cost function for the distributed systems predicting the location of the dynamic obstacle. In section III, we briefly discuss the PSO algorithm. Section IV presents the algorithm for the distributed systems, predicting the next position of the dynamic obstacle. Computer simulation and experimental results are presented in section V which is followed by a conclusion in section VI.

II. 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 two dynamic obstacles. A set of principles listed below is first developed to formalize the path-planning problem by a uniform treatment.

A. Pre-assumptions

1) Current position of each robot is known with respect to given reference in the cartesian coordinate system.

2) The robots have a fixed set of actions for motions. A robot can select one action at a given time.

3) Obstacles are detected by their colour which is known to the robots.

4) *The robots can come to know of the next position of the dynamic obstacle by using a prediction approach.*

B. Principles

The following principles have been used in the present context, satisfying the above mentioned pre-assumptions.

A robot attempts to align itself towards the goal in a calculated optimal path calling PSO.

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 of the dynamic obstacle is done so that the prediction-based approach is satisfied.

On detecting a static obstacle in the environment, the robot moves from its current position to a next obstacle free position following a minimal path obtained by PSO algorithm.

If it predicts 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.

If no obstacle is found in the *figure of safety*, the i^{th} robot calculates the next optimal position to the goal and moves towards the goal, calling 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).Vi be the current velocity of the i^{th} robot.

Let (x_{ig}, y_{ig}) be the goal position of the i^{th} robot.

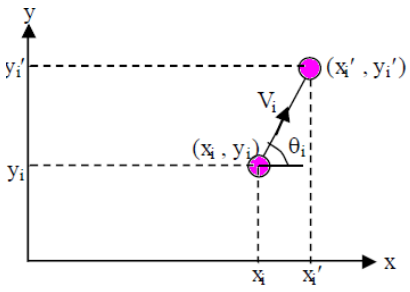

Fig 1. Current and next position of the i^{th} robot.

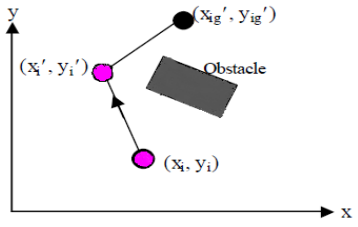

Fig 2. Selection of (x_i', y_i') from (x_i, y_i) to avoid collision with an obstacle.

It is evident from Fig.1 that

$$x_i' = x_i + v_i \cos\theta_i \Delta t \quad \text{.....(1)}$$

$$y_i' = y_i + v_i \sin\theta_i \Delta t \quad \text{.....(2)}$$

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,

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

.....(3)

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

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

Where (x_p, y_p) is predicted next position of the dynamic obstacle. Now $x_p = x_i' + u\Delta t$ and $y_p = y_i' + u\Delta t$.

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

Δt is the sampling time where $\Delta t = arr[i]/arr[j]$,in array[i] the total time taken for the movement of the dynamic obstacle is kept and we keep the number of steps undertaken by the dynamic obstacle in array[j].Here u stands for the velocity of the dynamic obstacle. Now the prediction made here is linear because extrapolation of the path of the dynamic obstacle is made with the direction of motion assumed constant over a given interval of time. The distance between robots at any point of time not being less than a predefined threshold (to avoid collision), we can use this as a primary constraint to this problem.

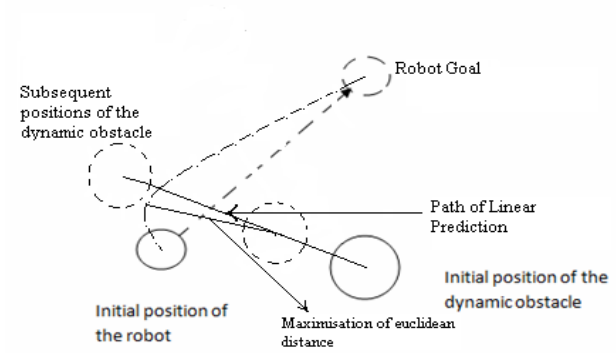

Fig.3 Linear prediction of the path of the dynamic obstacle and the maximization of the Euclidean distance between robot current position and predicted next position of dynamic obstacle

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 $d_{i'j'} - 2r \geq \varepsilon$, where r denotes the radius of the robots and ε (>0) denotes a small threshold.

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,

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

.....(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 R_i 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. This is realized with PSO algorithm, that will always select a minimal path to reach the respective destinations. To take case of static obstacles in the environment, we add one penalty function to the constrained objective function (7).Thus the present constraint optimization problem is transformed to –

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

$$+ f_{dp} \sum_{i'j'=1}^{n(n-1)/2} (\min(0,(d_{i'j'} - 2r)))^2 + f_{st} \text{ ……..(6)}$$

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

= 0, otherwise.

2. Dynamic obstacle

If randomly moving obstacle is detected within the *figure of safety* whose radius is (n*(R + r)), [n>1] then i^{th} robot deviates from its path to avoid the dynamic obstacle.
Consider the robot R_i is initially located at (xi, yi).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

To take case of dynamic obstacles in the environment, we add one penalty function to the constrained objective function (7).Thus the present constraint optimization problem is transformed to

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

$$+ f_{dp} \sum_{i'j'=1}^{n(n-1)/2} (\min(0,(d_{i'j'} - 2r)))^2 + f_{dy}$$

$$+ \sum_{i=1}^{n} \sqrt{(x_p - x_{ig})^2 + (y_p - y_{ig})^2} \quad \text{……(7)}$$

Where f_{dy} =positive constant when a dynamic obstacle enters the figure of safety

=0, otherwise

IV. PATH PLANNING AVOIDING STATIC AND DYNAMIC OBSTACLES

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 rotates360º 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

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

Maximize the distance between robot(current->pos) and dynamic obstacle(npredicted next->pos).
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 $\|curr\_i - G_i\| \le \varepsilon$ // $curr\_i = (x_{curr\_i}, y_{curr\_i})$, G_i = (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 (10);
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.

V. EXPERIMENTS AND 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[2][3] is realized in C on a Pentium processor. The number of robots (n), was varied from 2 to 14 and the performance of the system was evaluated. The configuration of the world map with 9 robots, two dynamic obstacles and 5 static obstacles at an instant of time during the execution of the code is depicted in the following figure.

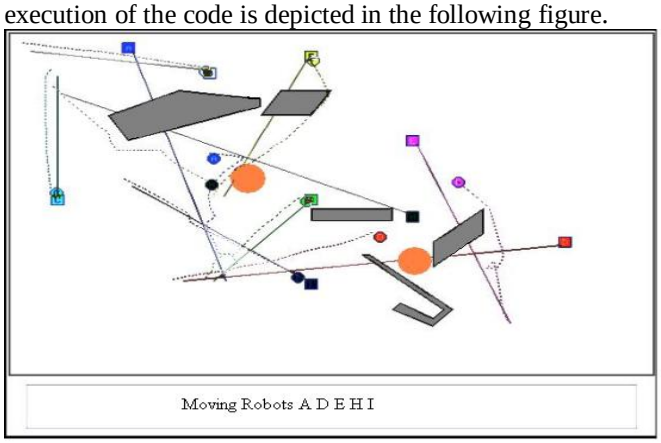

Fig.4 Screenshot showing the path of avoidance of the robots in the worldmap after dynamic obstacle has passed by.

The static obstacles are represented by Dark gray colour while the dynamic obstacles are represented by Orange colour in the world-map.

One metrics 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. For n robots, uncovered target distance (UTD) is the sum of $\|G_i - C_i\|$ i.e. UTD = $\sum_{i=1}^{n} \|G_i - C_i\|$.

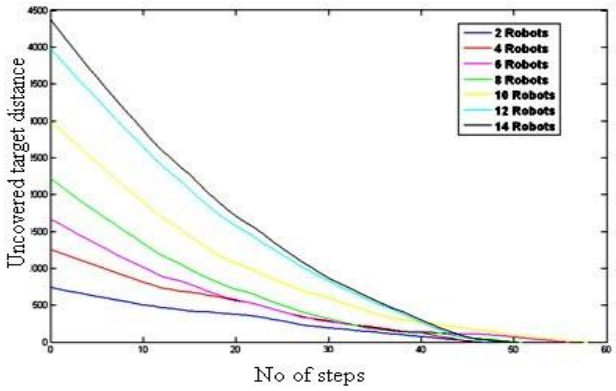

Fig.5 Plot of Uncovered Target Distance(UTD) vs Steps with number of robots as parameter with cocordinate (200,300)

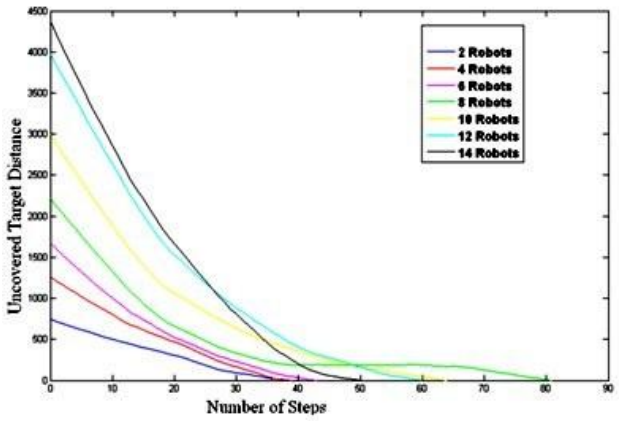

Fig.6 Plot of Uncovered Target Distance(UTD) vs Steps with number of robots as parameter with coordinate (50,300)

The above plots reflect that the number of steps increases with the number of robots. The initial position of the dynamic obstacle is different in fig.5 from fig.6. So the predicted position of the dynamic obstacle is also different in each case.As a result the trajectory of the robots are also different in each case.

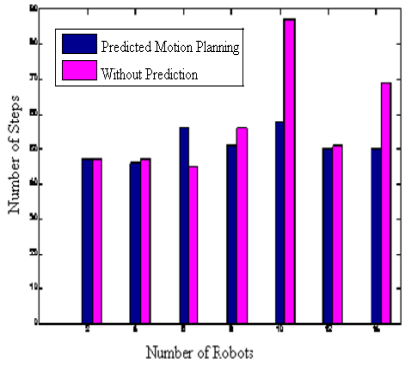

Fig.7 Bar chart showing performance evaluation of robots with and without prediction

The above plot clearly shows that the number of steps for robot in prediction based approach is much less than the approach without prediction.

VI. CONCLUSION

We have introduced the concept of the prediction of the location of the dynamic obstacle. The experimental results are in conformation with the fact that the prediction logic helps in minimizing the steps of the motion of the robots, which encounters the dynamic obstacle within its trajectory. The prediction logic help the robot to determine the location of the dynamic obstacle and plan its path accordingly minimizing the time. This is where our approach on multi-agent path planning amidst both static and dynamic obstacles outperforms 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