2011 International Conference on Recent Trends in Information Systems

Multi-robot Motion Planning amidst Dynamic Obstacle

[1] Dhrubojyoti Banerjee, [1] Chiranjib Guha Majumder, [1] Suparna Roy, [1] Arpita Chakraborty, [1] Amit Konar, [2] R.Janarthanan
[1] ETCE Department, Jadavpur University, Kolkata-700032
[2] Jaya College of Engineering,Chennai

dhrubo_jyoti_banerjee@yahoo.co.in, cguhamajumder@yahoo.com, suparna.ry03@gmail.com, chakraborty_arpita2006@yahoo.com, konaramit@yahoo.co.in, srmjana_73@yahoo.com

***Abstract*-This paper provides a modern approach to multi-robot motion planning in a given world map amidst both static and dynamic obstacles. The distributed method for multi-robot motion planning has been realized with particle swarm optimization algorithm. The experimental results show that the variation in the path deviation from optimal trajectory of the mobile robots increases with the increase in the number of dynamic obstacles. But here we have presented an approach which minimizes the probability of collision of the robots with the obstacles.**

***Keywords*- Multi-robot motion planning, Fitness function, Figure of Safety**

I. INTRODUCTION

The last decade of the twentieth century added a new dimension to the field of multi-agent robotics. The multi-robot system is advantageous compared to the single robot system due to its enhanced performance with respect to some given objectives. The already existing works on multi-agent systems include Multi robot Box pushing, Multi-robot Path planning etc.

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 introduction of the dynamic obstacles in the given workspace. In this paper we employed a distributed approach to solve the path planning problem. 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] using distributed approach. 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 problem can be formulated by a centralized as well as a distributed approach. Here we have taken up the distributed approach for its proven faster convergence of the PSO and reduction of the planning time.

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. In section III, we briefly discuss the PSO algorithm. Section IV presents the algorithm for the distributed systems, avoiding collision with the static and the dynamic obstacles. 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 path-planning problem for each robot is executed in steps until all robots reach their respective (predefined) goal positions safely.

5) The robots may encounter more than a single obstacle (either static or dynamic) at a time.

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 searches for both static and dynamic obstacles inside a circular region around its vicinity, named as the *Figure of Safety*.

978-1-4577-0792-6/11/$26.00 ©2011 IEEE

If it detects a static obstacle in the environment, the robot has to move from its current position to a next obstacle free position following a minimal path obtained by PSO algorithm.

If it detects a dynamic obstacle within the figure of safety, 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.

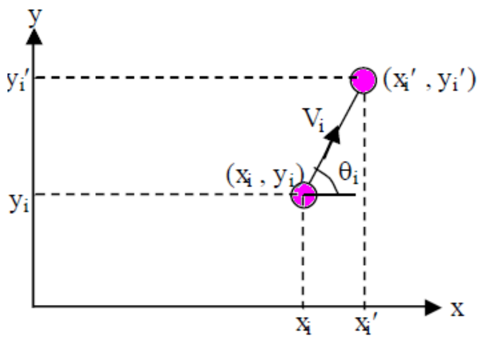


Fig 1. Current and next position of the i^{th} robot.

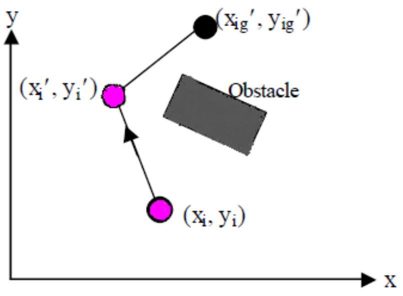


Fig 2. Selection of (x_i', y_i') from (x_i, y_i) to avoid collision with an obstacle.

It is evident from Fig.1 that

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

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

When Δ t =1, the above set of equation reduces to

$$x_i' = x_i + v_i \cos\theta_i \quad \text{.....................(3)}$$

$$y_i' = y_i + v_i \sin\theta_i \quad \text{...............(4)}$$

C. Distributed Planning

In distributed planning the total task is divided into n number of sub-tasks, where each subtask is realized by the PSO algorithm. In this section, we briefly outline the role of individual PSO algorithm, and then present how such distributed realization of PSOs can realize the overall system. 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,

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

.... (5)

Substituting x_i' and y_i' from expressions (3) and (4) in expression (5), we obtain

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

……………….(6)

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.

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. The constraints here have been modeled by three types of penalty, the first to avoid possible collision between any two mobile robots, the second to avoid collision of a mobile robot with a static obstacle whereas the third to avoid possible collision between the robots and the dynamic obstacles. Thus, the constrained optimization problem in the present context for the i^{th} robot is given by,

$$F_i = \sum_{i=1}^{n} \{v_i + \sqrt{((x_i + v_i\cos\theta_i - x_{ig})^2 + (y_i + v_i\sin\theta_i - y_{ig})^2)}\}$$
$$+ f_{dp} \sum_{i'j'=1}^{n(n-1)/2} (\min_{\forall_i}(0,(d_{i'j'} - (2r+\varepsilon)))^2 + f_{st} / d_{i-obs}$$
$$+ f_{dy} / (n \times (R + r)) \qquad \text{............(7)}$$

where f_{dp} (>0) and f_{st} (>0) denote scale factors to the second and third terms in the right hand side of expression (7) d_{i-obs} represents the distance of the obstacle from the *i*-th robot, R is the radius of the dynamic obstacle. In our experiments, we selected f_{st} =5000, f_{dp} =100 and f_{dy} =500 arbitrarily.

While moving in the workspace, where both static and dynamic obstacles are present, the robots first search in a circular area with radius (n*(R+ r)), [n>1] called the *figure of safety*, for obstacles where R and r are the radius of the Robot and the dynamic obstacle respectively.

1. Static 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. This is realized with PSO algorithm, that will always select a minimal path to reach the respective destinations. We now form an objective function F as has been deduced in equations (5), (6) and (7) which has to be minimized. 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 –

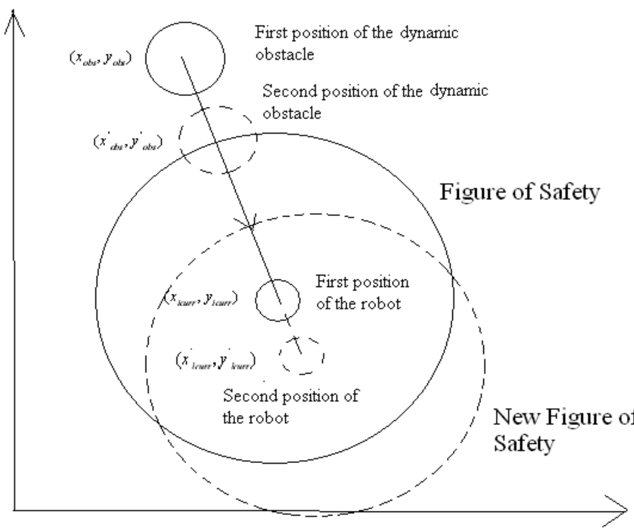
$$F = \sum_{i=1}^{n} v_i + v((x_i + v_i\cos\theta_i - x_{ig})^2 + (y_i + v_i\sin\theta_i - y_{ig})^2)$$
$$+ f_{dp} \sum_{i'j'=1}^{n(n-1)/2} (\min(0,(d_{i'j'} - 2r)))^2 + f_{st}$$
$$\text{………………..(8)}$$

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

2. Dynamic obstacle

We have introduced two dynamic obstacles in the workspace. The dynamic obstacle moves in a linear path but with varying direction. The direction of the moving obstacle is varied by changing the co-ordinates of the centre of the dynamic obstacle as a function of some random number. Mathematically it is represented by,

C(x,y) = f(p,q), where p,q=arbitrary random number. 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.



(x_{obs}, y_{obs})
First position of the dynamic obstacle
Second position of the dynamic obstacle
(x'_{obs}, y'_{obs})
Figure of Safety
(x_{icurr}, y_{icurr})
First position of the robot
(x'_{icurr}, y'_{icurr})
Second position of the robot
New Figure of Safety

Fig.3 Dynamic obstacle avoidance within the Figure of Safety

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

This is realized with the PSO algorithm, that will always select a minimal path to reach the respective destinations. We now form an objective function F as has been deduced in equations (5), (6) and (7) which has to be minimized.

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 –

$$F = \sum_{i=1}^{n} v_i + v((x_i + v_i\cos\theta_i - x_{ig})^2 + (y_i + v_i\sin\theta_i - y_{ig})^2)$$
$$+ f_{dp} \sum_{i'j'=1}^{n(n-1)/2} (\min(0,(d_{i'j'} - 2r)))^2 + f_{dy}$$
$$\text{…………………(9)}$$

Where f_{dy} =positive constant when a dynamic obstacle enters the figure of safety
=0, otherwise

III. 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 φ_1 and φ_2 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.

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.

$$V_{id}(t+1) = \omega V_{id}(t) + C_1\phi_1(P_{id}(t) - X_{id}(t)) + C_2\phi_2(g_d(t) - X_{id}(t))$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(10)$$

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.

IV. Path Planning Avoiding Static And Dynamic Obstacles

In this section we propose a solution to the constraint optimization problem introduced in section II avoiding both static and randomly moving obstacles. The proposed scheme presumes current position of n-robots and their speed, and determines next position of each robot by optimizing the given constraint objective function. An algorithm outlining the scheme is presented below:

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
rotates360$^o$ 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 //
IF dynamic obstacle
     Move away from the dynamic obstacle and Call PSO // to
find the next obstacle free optimal position //
   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.
Initially the experiment was conducted with one randomly moving obstacle approaching the positive x-direction, which was followed by two dynamic obstacles with the second one approaching the negative x-direction.

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.

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

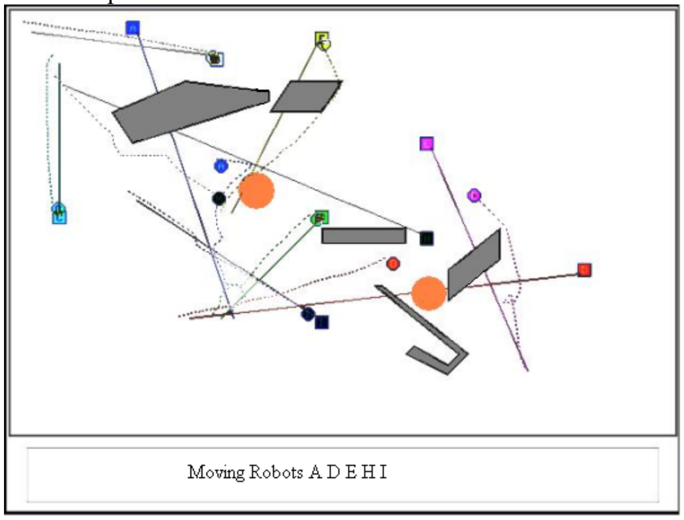


Fig.4 Screenshot showing the path of avoidance of the robots in the worldmap after dynamic obstacle has passed by.

Two 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, the uncovered distance for robot i is $\|G_i - C_i\|$, where $\|.\|$ denotes Euclidean norm. For n robots, uncovered target distance (UTD) is the sum of $\|G_i - C_i\|$ i.e. UTD = $\sum_{i=1}^{n} \|G_i - C_i\|$.

Average Path Deviation (APD)

Let the average time taken by the robots to reach their goal be t. Now, if the path traversed by the robot be $D_{traversed}$ and the actual path between the robot and the goal in the presence of only static obstacle be D_{static} ,the Average Path Deviation is given by APD= $D_{traversed}$ - D_{static} .

The plots of the results of the experiments conducted in simulation are given below. Fig.5 and Fig.6 show that, as the number of robots increases, the convergence of the curve is delayed. The nature of the third curve in Fig.6 is indicative of the fact that a dynamic obstacle has been encountered and clearly shows its random nature. In a word more number of robots signifies late convergence.

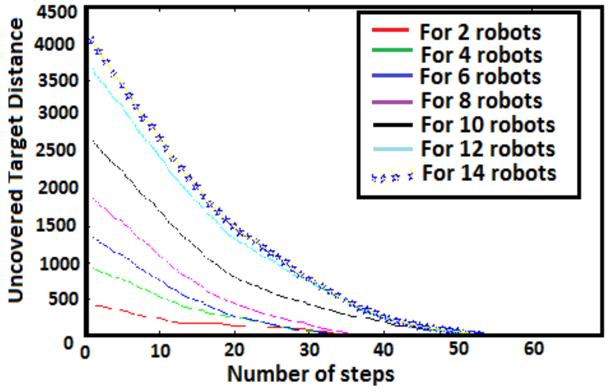


Fig.5 Plot of Uncovered Target Distance(UTD) vs Steps with number of robots as parameter with 1 dynamic obstacle.

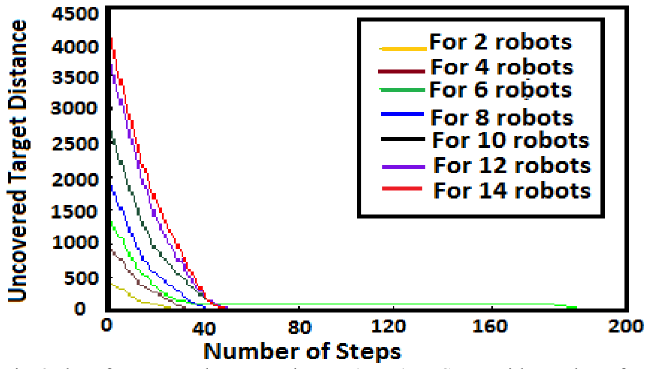


Fig.6 Plot of Uncovered Target Distance(UTD) vs Steps with number of robots as parameter using 2 dynamic obstacles.

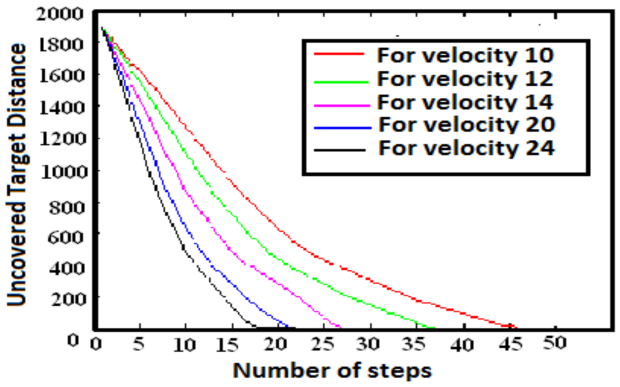


Fig.7 Plot of Uncovered Target Distance(UTD) vs Steps with robot velocity as parameter using 1 dynamic obstacle.

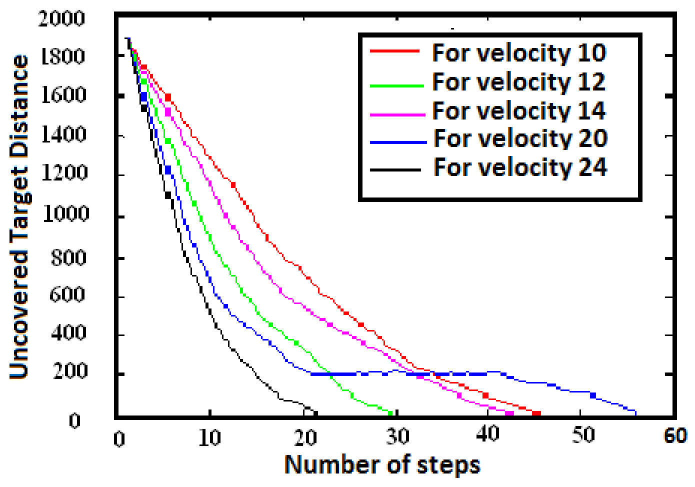


Fig.8 Plot of Uncovered Target Distance(UTD) vs Steps with robot velocity as parameter using two dynamic obstacles.

In Fig.9 more fluctuation in the Average Path Deviation (APD) is shown when two dynamic obstacles are encountered as against that in case of one dynamic obstacle.

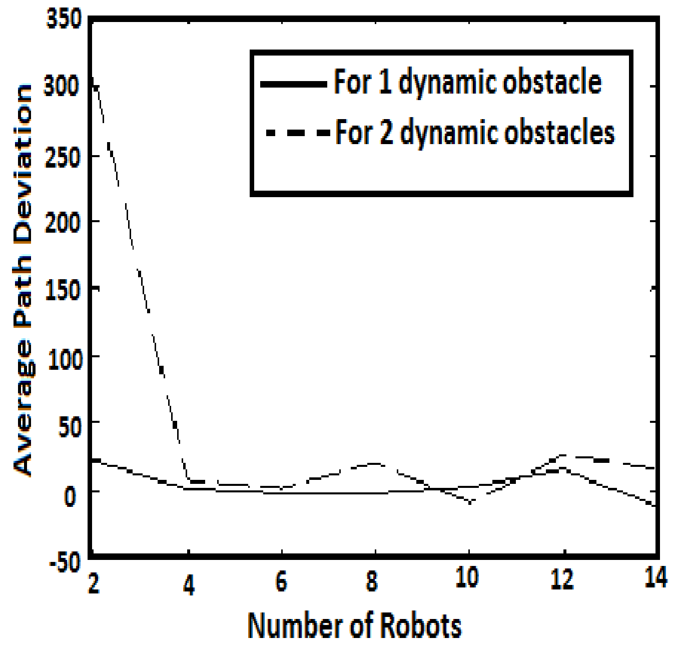


Fig.9 Performance evaluation using plot of Average path deviation (APD) vs Number of steps for 1 and 2 dynamic obstacles.

Finally Fig.10 shows the performance evaluation of the robots in case of static as well as dynamic obstacles.

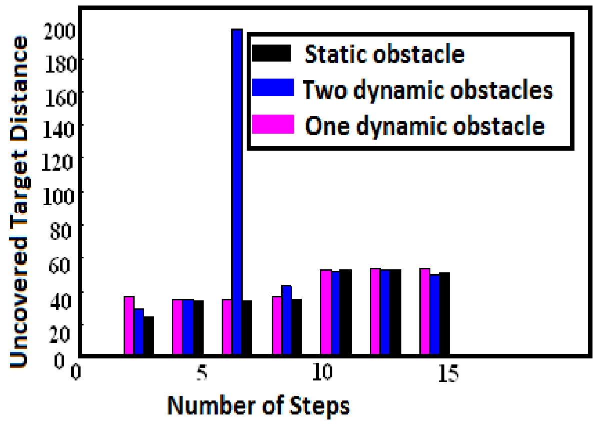


Fig.10 Bar chart showing performance evaluation of robots encountering static and dynamic obstacle.

The study shows that the number of steps required by the robots while encountering the dynamic obstacle is more than that of its static counterpart. But, in some cases, the numbers of steps covered by the robots, while encountering the static obstacles as well as the dynamic obstacles are equal. In those cases the robots have not encountered with the dynamic obstacle during their traversal.

The nature of the plot in the Fig.10 owes its origin completely to the random nature of the dynamic obstacle. The high peak in the Fig.10 indicates that the robot gets trapped in a localized region on encountering a dynamic obstacle repeatedly.

VI. Conclusion

We have introduced the concept of the *figure of safety* within which collision can never occur. The experimental results are in conformation with the fact that the variation in the path deviation from optimal trajectory of the mobile robots increases with the increase in the number of dynamic obstacles. The more the chances of the dynamic obstacle, getting closer to the robot, the more will be the tendency of the mobile robots getting trapped or localized in a particular region. But our approach aims to alleviate such problems by introducing the new concept of the *figure of safety*. 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

[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