Collision Avoidance for Multiple Robots till Next Waypoints through Collision Free Polygons

Satish Pedduri* K. Madhava Krishna*
* Robotics Research Center, International Institute of Information Technology, Hyderabad, India

pedduri@research.iiit.ac.in mkrishna@iiit.ac.in

Abstract— **Kinematically consistent paths for multiple mobile robots that are collision free for the next T steps are generated by forming collision free polygons (CFP). The polygons can be generated in a distributive fashion for each robot. All paths inside CFP that are kinematically feasible for a certain robot are collision free with all other robots in the same collision cluster for the next T time samples and with any other robot in the workspace in general. The construction of CFP is through two log-linear complex operations. The first one involves computing the intersection of the path container polygon (PCP) of a robot with PCP of other robots in the cluster. The second involves computing the convex hull of intersecting points arising from the intersection of PCPs. Once CFP is computed for a robot it chooses a point within the CFP (its next waypoint) that minimizes a cost function and moves towards the point. The process is repeated till the goal is reached. The main advantage of this method is that the best and worst-case times for finding a T-step ahead collision free path is always log-linear. In many randomized search methods, finding the paths till the next waypoint results in a worst case complexity exponential in number of robots. The efficacy of the current method is well portrayed through simulations. Comparative results show that the following method finds a collision free path to next way point much quicker.**

1. INTRODUCTION

Multi robotic systems have been an active area of research, where multiple robots perform a task in a cooperative or individual fashion. While performing multi robotic tasks, it is often desirable that the system is collision free. Collisions can happen with the co-robots or with the static and dynamic obstacles, and these collisions also called as conflicts can be hazardous for the robots. In order to overcome these conflicts, we devise an algorithm, which can be used in various applications. The main advantage and novelty of this method is that the best and worst-case times for finding collision free path till next waypoint is always log-linear. Any point on this path cannot be accessed by any other robots for the next T Steps. In many methods that involve a randomized search for collision free paths the worst-case complexity is exponential in the number of robots. The efficacy of this method and its advantages are well portrayed through simulations. Comparisons with other heuristics that are used in randomized methods show that the following method finds a collision free path to the next waypoint much quicker than those heuristics.

The pivot of this algorithm is the computation of the *Collision Free Polygons* (CFP) for a robot. The CFP involves computing the intersection of *Path Container Polygon* (PCP) of a robot with all other robots. The PCP, a convex polygon can be considered as a container of a number of kinematically feasible paths for the next T instants. The vertices of the PCP (CP for short henceforth) are the current position of the robot, the coordinates that are reached with maximum linear acceleration and maximum angular deceleration (if the current angular acceleration is zero then that is maintained), the coordinates obtained by maximum linear and angular acceleration and those obtained by maximum linear deceleration and angular acceleration. The intersection of CPs of robots in collision gives rise to a number of intersection points on the boundary of the PCP as well as inside it for a robot. The convex hull of these points is considered. Then all points that are outside the hull but inside the CP are guaranteed to be collision free for next T samples. The CFP is the boundary of all such collision free points. The whole operation can be performed in two log-linear operations.

Points within CFPs of all the robots that belong to a collision cluster are chosen such that a cost function is minimized. The search for a point within the CFP can be made arbitrarily fast by choosing arbitrarily a small number of points to minimize the cost function. In fact choosing one point would suffice since any point in a CFP and path to it is collision free for next T steps. The robots belonging to a collision cluster are obtained through the collision dependency graph. Robot A that predicts a collision with another robot B within a certain stipulated time is said to have a collision dependency with that robot and is shown in the graph by a bi-directional link between the two. The algorithm currently operates in a semi-centralized fashion in that it is centralized with respect to the robots in a cluster while decentralized across clusters. However, a completely

decentralized implementation can also be achieved with extra bandwidth, allowing for more exchange of messages between the robots.

Multi-robotic navigation algorithms are traditionally classified as centralized or decentralized approaches. In the centralized planners [1, 2], the configuration spaces of the individual robots are combined into one composite configuration space, which is then searched for, to obtain a path for the whole composite system. In case of centralized approaches that compute all possible conflicts over the entire trajectories, the number of collision checks to be performed and the planning time tend to increase exponentially as the number of robots in the system increases. Complete recalculation of paths is required even if one of the robot's plan is altered or the environment changes. However, centralized planners can guarantee completeness and optimality of the method, at least theoretically.

Decentralized approaches, on the other hand, are less computationally intensive as the computational burden is distributed across the agents and, in principle, the computational complexity of the system can be made independent of the number of agents in it, at-least to the point of computing the first individual plans. It is more tolerant to changes in the environment or alterations in the objectives of the agents. Conflicts are identified when the plans or commands are exchanged and some kind of coordination mechanism is employed to avoid the conflicts. However, they are intrinsically incapable of satisfying optimality and the completeness criterion. Prominent among the decentralized approaches are the decoupled planners [3], [4], [5]. The decoupled planners first compute separate paths for the individual robots and then resolve possible conflicts of the generated paths by a hill climbing search [3] or by plan merging [4] or through dividing the overall coordination into smaller sub problems [5]. However many of these planners are not viable in presence of non integrable constraints, moving obstacles and fast replanning. There is a vast volume of literature that addresses the problem of motion planning in presence non integrable constraints by incorporating the kinematics of the robot [6, 7] and their various extensions that address the problem of fast replanning [8] and moving obstacles [9] and multiple robots [10]. They do so by making use of the existing roadmap of curves constructed while computing the initial plan.

In the present method there is no initial plan for the robot since there is no map available and is closer to reactive collision avoidance approaches in this regard. It is crucially different from others in that the focus has been on computing a collision free path till the next waypoint that has the same log-linear complexity irrespective of the nature of the environment. The present approach can be dovetailed with the methods cited above since finding a path to the goal state through a sequence of paths till the next waypoint is a recurring theme in all. Heuristics are generally resorted to reduce the search in finding the path for multiple robots in collision [10]. However it is not guaranteed that these heuristics are efficient across the board for various situations *unlike the current approach where a search for a collision free path is* ***equivalent*** *to the computation of CFP*. Being log-linear it easily scales up gracefully over number of robots. It can also be dovetailed with approaches that do not incorporate a known map such as the Dynamic Window [11] to a multi robotic setting, where the search for a collision free path is for the next time sample only.

II PROBLEM FORMULATION

Given a set of robots $R = \{R_1, R_2, \ldots, R_n\}$, each assigned a start and goal configuration, the objective is to navigate the robot such that they reach the goal configuration avoiding all collisions. While collisions could occur with stationary and moving objects, in this paper we focus primarily on how the robots could avoid collisions that occur amongst them in a cooperative fashion. For this purpose the following premises have been made:

a. Each robot *Ri* is assigned a start and goal location and it has access to its current state and its current and aspiring velocities. The current state of *Ri* is represented as $\psi_i = \{vc_i, vn_i, \omega c_i, \omega n_i\}$ where vc, vn represent its current and aspiring velocities and $\omega c, \omega n$ its current and aspiring angular velocities.
b. Robots are capable of broadcasting their current states to each other. They do so only to those robots that are within a particular range of communication.
c. Robots accelerate and decelerate at constant rates that are same for all. Hence a robot *Ri* can predict, when another robot *Rj* would attain its aspiring velocity vn from its current velocity vc if it does not change its direction.

Assumption 'a' is a standard assumption made in a typical mobile robot system. In general sensor based localization algorithms are used to correct discrepancies between the expected estimate and the actual readings reported by sensor. Current and future velocity estimates obtained through such techniques have been used to achieve desired results such as in [11], where the authors report a successful real-time algorithm for fast navigation. Assumption 'b' is also common to multi robotic systems with successful implementations [10]. Assumption 'c' is done essentially to facilitate certain simplicity in the approach to reduce the amount of data transfer between robots. The algorithm's performance would not be affected if the rates of acceleration and deceleration vary between robots provided they are communicated to one another.

III METHODOLOGY

We consider a differential drive robot that move along continuous curvature paths. We assume that for a small interval Δt the linear and angular velocities are constant. The path of a robot consists of piecewise circular arcs (linear segment are made of ∞ radius). From the current pose of the robot we get various curves corresponding to different values of final angular and linear velocities reached from the current state. These paths are generated such that there is only one change in the direction of acceleration, i.e. either the robot accelerates or decelerates but does not do both. In essence for certain T time steps into the future we get a set of reachable poses. Among these we select the pose reached through maximum linear and angular accelerations, along with the current position we form a polygon from these points that is convex and call it the PCP (figure 1). It can be shown that the set of reachable positions inside the PCP is much denser than those outside.

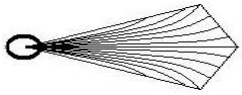

Fig. 1. The PCP and the Paths and set of reachable positions contained inside.

A. Dominantly Inaccessible Areas

We first find what we call as dominantly inaccessible areas within PCP of robot r_i with respect to all other robots r_j $j = 1 \to n;\ j \neq i$ with which it has a collision within next T samples. Let PCP of r_i constructed for the next t sample be CP_i^t we remove superscript t for ease of notation. Let $A(CP_i)$ be the area enclosed by CP_i. We extend the side vertices of CP_j be a factor ξ and form a new CP_j called $\overline{CP_j}$. The factor ξ further reduces the number of points accessible from current state of r_j that lie outside $A(CP_j) \forall j = 1 \to n$. We further grow all CP_j by radius of robot r_i and reduce r_i to a point. We denote the grown $\overline{CP_j}$ as $\widetilde{CP_j}$. We find the following area AFP_i such that any point in AFP_i is almost inaccessible from the current state of r_j $\forall j = 1 \to n;\ j \neq i$. We say "almost" in the sense that they are accessible only for extreme values of linear and angular acceleration or deceleration if at all. They can be made inaccessible by choosing an acceleration or deceleration slightly different from extreme value or by changing ξ.

$$AFP_i = \overline{\bigcup_{\substack{j=1 \\ j \neq i}}^{n} A(CP_i) \cap A(CP_j)} \cdots\cdots (1)$$

(Figure 2) shows AFP for robot A as the shaded region. $\widetilde{CP}$ for robots B and C are also shown.

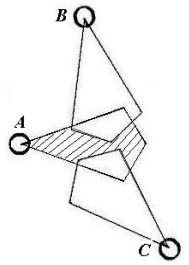

Fig. 2. The shaded region shows AFP for robot A. Any point in the region is inaccessible from B or C if values of acceleration are slightly different from the extreme values.

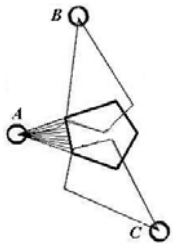

Fig. 3. CFP of the polygon A formed by convex hull.

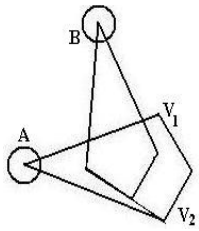

Fig. 4. Points chosen near V2 can pass the CP of B.

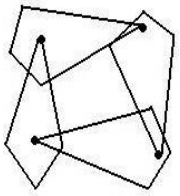

Fig. 5. Each robot's current state contain the container polygon of another robot.

B. Collision Free Paths

While any point in AFP for a robot is inaccessible from others it does not guarantee that the path reaching any such point from the current state is collision free. For this we compute the convex hull of the following points

- Point due to intersection of CP_i with CP_j and all points of CP_j inside CP_i.
- All point of CP_i except the current point corresponding to current pose of the robot.

For (figure 2) the convex hull is plotted in (figure3). Let us call the convex hull as $Hu(CP_j)$. Then the collision free polygon of r_i CFP_i is the polygon $CFP_i = CP_i \cap \overline{Hu(CP_j)}$. The shaded region in (figure 3) is the CFP for robot A. The characteristics of CFP is that any point in the CFP when reached from current pose of r_i, Cr_i with not more than one change in direction of angular velocity of r_{ji} will not intersect CP_j and is hence collision free from all paths emanating from $Cr_j;\ j = 1 \to n;\ j \neq i$.

C. Tough cases:

Case 1: When choosing the next waypoint of robot r_i within it's CFP at times the curve joining the current pose to the waypoint can cut CP_j of another robot. This happens if the point is chosen on the boundary of CFP, close to one of the vertices of the CP of r_i, usually when only one robot is intersecting r_i and the angle of approach is acute. This is shown in (figure 4).
However the probability of collision is negligible since

- The path even when it intersects the CP of other robot does it such that the length of the path inside the CP is very small.
- Since the other robot also chooses its next waypoint within its CFP the actual probability of paths colliding is zero.

Case 2: In this case we have a situation of each robot's current state contained in the container polygon of another robot. This is shown in (figure 5).
The situation can be overcome by continuing the robot along current motion direction for a few more samples. If the situation continues to persist the polygon size can be reduced for a lower time period T. This method of reducing polygon size can be applied for the previous case as well. It must be emphasized that the above situations are extremely rare and have been noticed not more than a few time in several runs.

D. Objective Function Minimization

We choose sample points for all robots within a cluster. These sample points for a robot is on the boundary that coincide with CFP of that robot and the convex hull. We take the point space of sample points of the robots in the cluster that minimizing the objective function

$$Obj = \alpha \sum_i p_i + \beta \sum_{jk} C_{ik} / P_{jk}$$

Here p_i is the perpendicular dropped on to the line joining the robots' current pose to the goal from the sample point. $C_{jk} = 1$ for any pair of robots r_j, r_k that are in collision in the cluster P_{jk} is the distance between r_j, r_k at the respective sample point considered. The first term in the objective function prevents large deviations from the line joining the current pose to the target, the second increases spacing between the robots.

It is to be noted again that the objective of evaluating Obj is only to obtain better shape over trajectories for the robot. The search space can be reduced as small as possible by choosing arbitrarily small number of sample points without affecting the ability of the method to chose a collision free trajectory for that is guided based on the theme discussed in the subsections A,B.

E. Dependency Graph and Collision Cluster

Robots that have a collision amongst them is shown by a bidirectional link in the dependency graph. For a snapshot shown in figure 6a, robot pairs (R1, R2), (R1, R3), (R4, R5), (R5, R6) detect collisions amongst them within a stipulated time. The dependency graph for such a situation is shown in figure 6b. All robots that are in the same connected component of the dependency graph belong to the same collision cluster. Hence for the graph of figure 6b there are two clusters once consisting of robots (R1,R2,R3) and the other (R4, R5, R6).

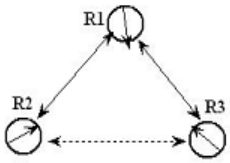

Fig. 6a. Robots R1, R2 and R1, R3 are in collision and included in one cluster.

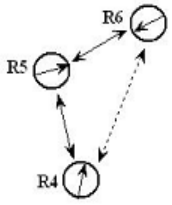

Fig. 6b. Robots R4, R5 and R5, R6 are in collision and included in another cluster.

Thus the algorithm for collision avoidance on the robot takes the form shown below

ALGORITHM: Collision Avoidance

1. Until all robots reach their goals, do steps 2 to 3
2. Find the dependency graph of robots yet to reach their goals and form clusters
3. For each cluster C_i do step 4
4. For all robots R_j in C_i do steps 5 to 7
5. Compute the CFP and obtain sample points.
6. Evaluate Obj and find the collision free point for each robot from the sample points obtained in 5 that minimizes Obj.
7. Traverse to the point obtained in 6 for each robot until one or all the robots reach to their obtained points while checking for the below conditions.
   i) If Robot from other cluster finds collision with the current cluster, go to step 1.
   ii) If goal is reached, go to step 1.

F. Complexity

The current algorithm complexity:

For the r robots, each container polygon for the robot has 4 points. Hence there are n = 4r points to be considered. So when done in a distributed way a robot r_i

finds intersection with remaining (r-1) polygons; it does so in $O(nLog(n)+c_1)$ time [12]. Indeed [12] presents an $O(n^{4/3})$ algorithm that for large n is quicker than $O(nLog(n))$ computation. If there are m intersection points of the r-1 polygons with the current one, the convex hull of those m points can be found in $O(mLog(m))$. Hence the overall complexity is $O(nLog(n)+mLog(m)+c)$. Here c is a constant compensating the time taken for growing the polygons and the constant c_1 from the intersection computations.

Complexity for choosing next waypoint by exhaustive search tempered with heuristics:

Among r robots each robot searches along m paths for the next waypoint or milestone. There are $O(m^r)$ searches for finding a milestone that is collision free with all others in worst case. Each path to such a mile stone is divided into k sub locations and collisions are checked at each of those k locations. Hence in the worst case the computation time is $O(km^r)$. If the first path integral to the new milestone is a collision free path, if that is the case for all the robots, for the k sub divisions of r robots' first paths the numbers of collision checks that were performed, is $O(kr)$. This is the best case scenario, where the first path searched for all the robots were collision free. While efficient heuristics can certainly make things better than $O(km^r)$ they can never guarantee $O(kr)$ except in very exceptional situations of robot configurations. However the time complexity of the present method is always the same.

IV. Simulation Results

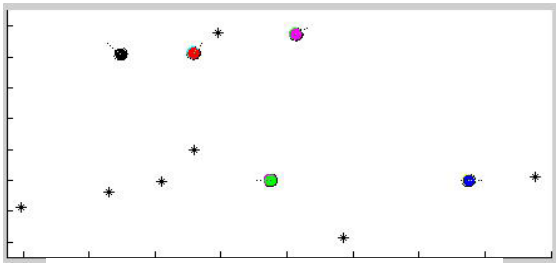

Fig. 7a. Initially five robots arriving in to the scenario.

Below are the simulations of 7 and 12 robots in different scenarios, where each robot having its goal location marked as asterisk. Figure 7a to 7d depicts the 7 robots case with each robot encountering collision with few other robots sequentially. Initially there are five robots in the environment, in which the two robots in top left have immediate collision and started avoiding each other. Figure 7b depicts a state where the robot at top left while avoiding the collision with a robot encountered previously, had detected collision with a robot coming towards it, and the total cluster of three started avoiding the collisions cumulatively. Figure 7d depicts the robots having reached their final goals through continuous curvature paths after avoiding each other.

Figures 8a-8c depicts 12 robots case with multiples of four robots colliding simultaneously, and reaching toward the respective goals. Figure 7a depicts the initial polygons computed immediately after finding the collisions. Figure 7b shows an intermediate stage during the collision avoidance maneuver. Figure 7c shows the robots reaching the goals and their respective paths of the robots. It must be noted that in both the scenarios the paths of the robots respect the kinodynamic constraints of the robot.

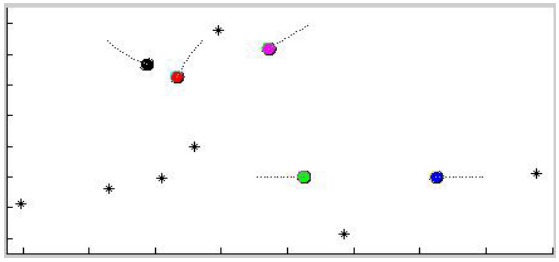

Fig. 7b. The 3 robots in top forming collision cluster.

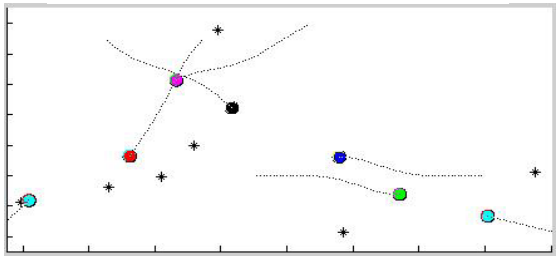

Fig. 7c. Intermediate state of robots in 7a.

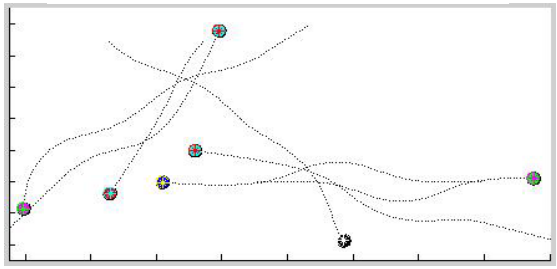

Fig. 7d. Robots in 7a with smooth curvature paths reached their respective goals.

A. Comparison with heuristics:

Table 1 depicts the time consumed by the presented algorithm and the different heuristics, for cluttered and scattered cases. We compared with three different heuristics the time taken to find a collision fee path to the next waypoint, on a Pentium-3 processor with 256MB memory, through MATLAB simulations. First

heuristic chooses the next way point based on the following order: extreme left with maximum accelerations, extreme right with maximum accelerations, extreme left with average velocities, extreme right with average velocities, immediate deceleration to either left or right. Similarly in the second heuristic we have the priority order: extreme left with maximum acceleration, extreme right with average velocities, and immediate deceleration. The third heuristic simply decelerates to either left or right. The first row shows the comparison for cluttered case, and the second row shows the results for robots which are far enough before detecting the collision.

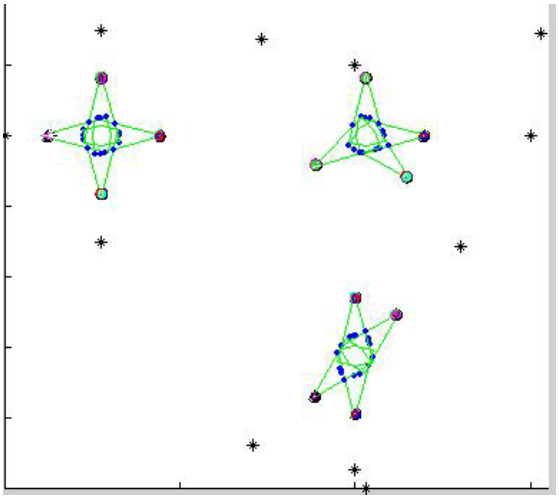

Fig. 8a. 12 Robots in cluttered situation forming four clusters, the initial polygons are shown in green, and sample points in blue.

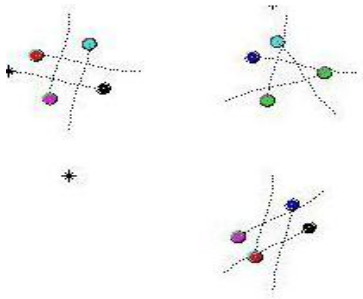

Fig. 8b. Robots in 8a at intermediate collision avoidance.

Scenario	Present algorithm	H1	H2	H3
cluttered	0.185	0.713	0.384	0.093
scattered	0.127	0.342	0.273	0.093

Tab. 1. Comparison between presented method and various heuristics in scattered and cluttered environments. All values shown in seconds.

It can be seen that in all the cases except for the third heuristic, the presented algorithm performed well. The performance gain of the current algorithm is more pronounced in a cluttered than a scattered environment. This is indeed expected since the search begins to grow exponentially in heuristic approaches when clutter is more. Note that the third heuristic is utterly trivial as it simply stops all the robots and hence has negligible utility. Also it is observed that these heuristics perform well for certain cases only, for the rest of the cases the computation time grows exponentially. The comparisons are reported in seconds. The time for each operation for all methods would have been much faster had they been done in C/C++ rather than MATLAB.

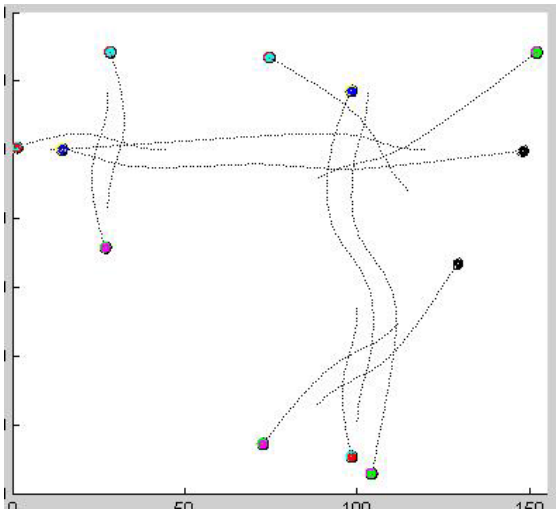

Fig. 8c. Robots in 8a reaching their goals, along the continuous smooth paths.

V.CONCLUSIONS

A method for collision avoidance among multiple robots through collision free polygons has been presented.. The essential contribution of this paper has been the computation of a collision free path till the next way point in a sequence of two log linear operations. Heuristic approaches that compute a collision free path to the next waypoint in randomized search methods invariably tend to become exponential in the number of robots in cluttered scenarios howsoever good the heuristics are designed. The method has shown to be useful for online collision avoidance and goal reaching maneuvers that respect the kinodynamic constraints of the robot. Extensive comparisons with carefully designed heuristics show the performance gain of this method over such heuristics.

REFERENCES

[1]J. Barraquand and J. C. Latombe., "A monte-carlo algorithm for path planning with many degrees of freedom". *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA),,* 1990.

[2]Svetska P. and Overmars M, "Coordinated motion planning for multiple car-like robots using probabilistic roadmaps" *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA),* 1995

[3]Bennewitz M., Burgard W. and Thrun S., "Finding and Optimizing Solvable Priority Schemes for Decoupled Path Planning Techniques for Teams of Mobile Robots", Proceedings *Robotics and Autonomous Systems*, 41 (2), pp 89-99, 2002

[4]F. Gravot and R. Alami, "An extension of the plan-merging paradigm for multi-robot coordination", *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA)*, 2001.

[5]S. Leroy, J. P. Laumond, and T. Simeon, "Multiple path coordination for mobile robots: A geometric algorithm", *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI),* 1999.

[6] O. Brock and L. Kavraki, "Decomposition-based motion planning: a framework for real-time motion planning in high-dimensional configuration spaces," *Proceedings of IEEE International Conference on Robotics and Automation, vol. 2,* 2001.

[7] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[8] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," *Proceedings of International Conference on Intelligent Robots and System (IROS),* October 2002

[9] M Zucker, J Kuffner and M Branicky, "Multipartite RRTs for Rapid Replanning in Dynamic Environments", *ICRA 2007 to appear.*

[10] C.M. Clark and Stephen M. Rock and J.C. Latombe , "Motion Planning for Multiple Mobile Robots using Dynamic Networks" *Proceedings of IEEE International Conference on Robotics and Automation,* ICRA 2003: 4222-4227.

[11] Fox D, Burgard W, and Thrun S., "The Dynamic Window Approach to Collision Avoidance", *IEEE Robotics & Automation Magazine*, 4(1). 1997

[12] P. Gupta, R. Janardan and M. Smid, "Efficient algorithms for counting and reporting pairwise intersections between convex polygons", Proceedings of Information Processing Letters, (69), pp. 7--13, 1999.