Cooperative Navigation Function based Navigation of Multiple Mobile Robots

Satish Pedduri* K. Madhava Krishna* Henry Hexmoor+
* Robotics Research Center, International Institute of Information Technology, Hyderabad, India
+Computer Science Dept, Southern Illinois University, Carbondale, IL, 62901, USA

pedduri@research.iiit.ac.in mkrishna@iiit.ac.in hexmoor@cs.siu.edu

***Abstract*—** *A cooperative methodology for collision avoidance of multiple wheeled robots, having respective goals to reach, is detailed in this paper. The paths executed by the robots are continuous not only in terms of positions reached by the robots but also in terms of their velocities. A navigation function is designed that, apart from taking into account goal reaching and avoidance behaviors, also implicitly captures cooperative behavior amongst robots by identifying spaces where collisions between robots tend to be minimized. A search in the joint space of linear and angular velocities of the robots results in selection of a linear and angular velocity tuple for each robot that minimizes the navigation function. Simulated results portray the efficacy of the methodology.*

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 (called as conflicts henceforth) can be hazardous for the robots. In order to overcome these conflicts, we device an algorithm, which can be used in various applications.

The pivot of this algorithm is a navigation function that, apart from modeling goal reaching and avoidance behavior, also captures the cooperative essence through a clearance behavior that essentially moves robots to spaces where the tendency to come across one another again in the future is reduced. A search in the joint space of linear and angular velocities of the robots in a collision cluster results in selection of a linear and angular velocity tuple for each robot in that cluster, minimizing the navigation function. The robots belonging to a collision cluster are obtained through the collision dependency graph. A 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 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].

The method presented here is different in that while being centralized with respect to the robots in a cluster it is decentralized across clusters. Moreover, complete plans are not computed. The locations of the robots for certain T time samples in future are exchanged for robots moving along arcs and for those moving with linear velocities along straight lines, it suffices to broadcast its current state. The

1-4244-0945-4/07/$25.00 ©2007 IEEE

collisions are avoided by searching in the velocity or the orientation space (the set of reachable orientations) of the robot. In that aspect, it resembles the extension of the Dynamic Window approach [6] to a multi robotic setting, however, with a difference. The difference is that in the dynamic window the acceleration command is applied only for the next time interval whereas in the present method the restriction is only in the direction of change in acceleration over a time interval $t < T$ for all the robots.

2. 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. All robots are circular and described by their radius
c. 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.
d. 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.

3 METHODOLOGY

Initially, the various components of the objective function, encoding the different behaviors are detailed.

3.1 Deviation Distance

Deviation Distance is computed using a pair of velocities (v, ω). The Expected position is computed through the velocity pair applied for certain time instants *t* from the current position (figure 1). The smaller angle between the heading line, and line connecting expected position and goal position is computed as θ as shown in figure 1. The distance from expected position towards goal position is *d*. Through the isosceles triangle we compute the *deviation distance* using the formula

Deviation Distance = $2d \sin(\theta/2)$. The deviation distance captures the target orienting behavior, as lesser the robot is deviated from its target lesser is the deviation distance. The reason for not merely using θ is to suppress the dominance of target orienting when the robot is already near its goal. In such a scenario deviation distance evaluates to a much lesser value than the mere deviation in angle preventing the robot from indulging in cumbersome orientation maneuver when near the goal.

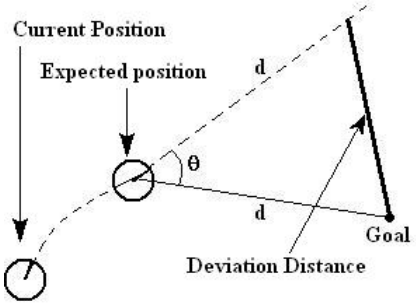

Figure 1: Computation of Deviation Distance for velocity pair (v, ω).

3.2 Proximity

Proximity is calculated the same way as *Deviation Distance*. The only difference being it is the distance between two robots after applying the velocity pairs (v_1, ω_1) and (v_2, ω_2) for certain time *t*. Proximity captures the essence of collision avoidance between robots as the proximity function is maximum when the robots are further away.

3.3 Forward Clearance

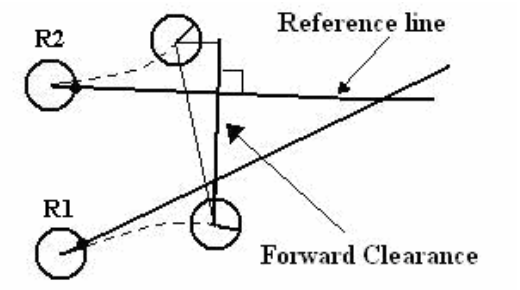

Figure 2: Computation of forward clearance for velocity pairs (v_1, w_1) and (v_2, w_2)

The forward clearance for robot R1 with respect to R2 is computed by dropping a perpendicular from R2's expected position onto the reference line drawn at R1's expected position. The reference line is drawn parallel to R1's current heading (figure 2). The forward clearance behavior makes sure that when the number of robots in a cluster increases a more judicious use of space between them is possible. For example based on proximity alone robots 1, 2 and 3 reach

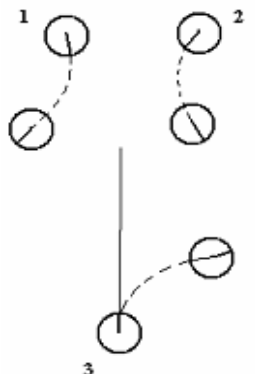

Figure 3a: Proximity based decision making the three robots to deviate from their current heading.

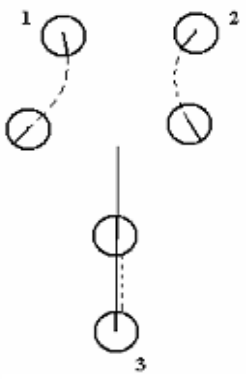

Figure 3b: After consideration of forward clearance the robots make right decision.

the expected positions as shown in figure 3a in dashed circles, where robots 2 and 3 come up on the same side of the reference line of 3 with respect to 2. However when forward clearance is also incorporated the situation changes to one shown in figure 3b where robot 3 does not deviate while 2 and 1 deviate on either side of 3. The role of forward clearance will be dealt more elaborately elsewhere. Intuitively the clearance behavior captures the cooperation between robots by assigning them such that robots make use of the space aesthetically avoiding clutter as well as sparseness. The proximity function alone does not facilitate this.

3.4 Objective Function Minimization

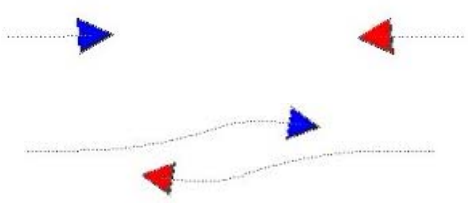

Figure 4a: Two robots moving head on with current angular velocities (0.01, 0.0). The robots heading to goals after avoiding collision.

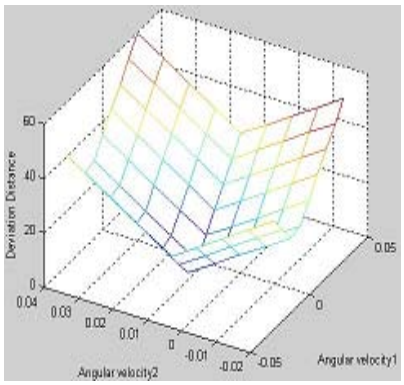

Figure 4b: Plot of deviation distance versus angular velocity 1, angular velocity 2.

After computing the Deviation distance, Proximity, and Forward clearances for all the combination of robots. The objective function is minimized for the current pairs of velocities. It takes the form $Obj = \alpha \sum_i d_i + \beta \sum_{ij} C_{ij} \Big/ P_{ij} + \gamma \sum_{ij} C_{ij} / F_{ij}$. Here *Obj* is the objective function that is evaluated, d_i is the deviation distance for robot *i* in its cluster, $C_{ij} = 1$ for every pair of robots *i,j* in a cluster that have a collision between them and 0 otherwise, P_{ij} is the proximity between any pair of colliding robots while F_{ij} is the forward clearance with respect to one of the robot's reference line and α, β, γ are constants. C_{ij} is 1 or 0 depending on whether it is in conflict or not. And α, β, γ being constants affecting the system behavior.

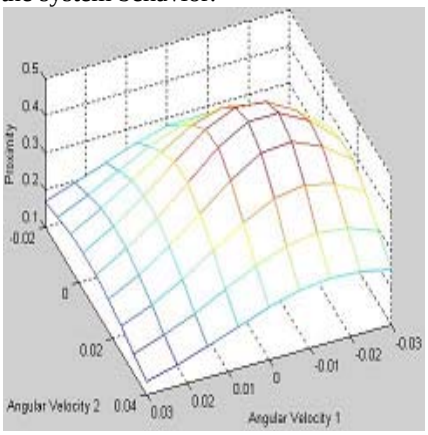

Figure 4c: Evaluation of proximity in angular velocity space.

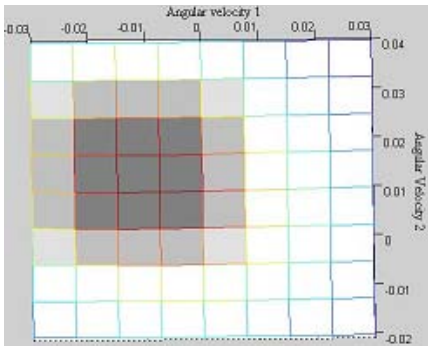

Figure 4d: Top view of 4c where the gray regions indicate the robots are too close.

Figure 4b shows the evaluation of the first term of *Obj* for the situation shown in figure 4a. The term is evaluated for For the sake of simplicity the search is performed by keeping the linear velocity constant and varying only the angular velocities of either of the robots. Figure 4b shows the function evaluates to a minimum for small values of aspiring angular velocities $(\omega n_1 = 0.0, \omega n_2 = 0.01)$ since the deviation distance is minimum for that case, the robots already are nearly oriented towards the goal. The current angular velocity pair is $(\omega c_1 = 0, \omega c_2 = 0.01)$.

Figure 4c and 4d shows the evaluation of the second term of *Obj* for the situation in 4a for two views of the mesh. Figure 4d shows in gray regions the areas of maximum evaluation. These correspond to aspiring velocity tuples having opposite signs indicative of the fact one of the robots turns clockwise and other counter clockwise leading to closer proximity between them. The area in white in figure 4d corresponds to regions of minimum evaluation of the second term in *Obj*. Figure 4e shows the evaluation of Forward clearance. Observe that there is a peak which indicates When the robots take clockwise and anti clockwise or vice versa then the forward clearance is maximum, which means the robots are close enough with respect to the reference line i.e. the horizontal line. Figures 4f shows the evaluation of the entire *Obj*. In this case a unique minima is obtained for $(\omega n_1 = 0.03, \omega n_2 = 0.04)$.

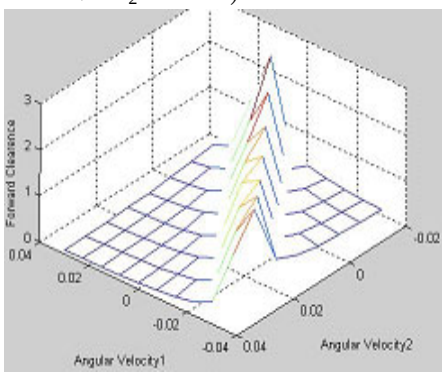

Figure 4e: Evaluation of forward clearance. Peak areas imply the robots are close with respect to reference line.

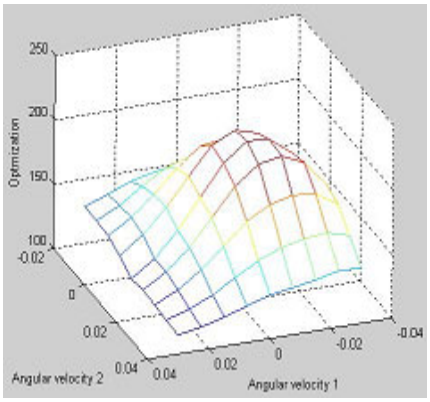

Figure 4f: Evaluation of objective function. The minimum is located at (0.03, 0.04).

However for cases where a unique minima does not exist the velocities are chose such that they are minimally deviant from the current velocities.

3.5 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 5a, 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 5b. 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 5b there are two clusters once consisting of robots (R1,R2,R3) and the other (R4, R5, R6).

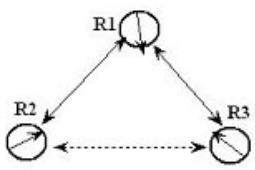

Figure 5a: Robots R1, R2 and R1, R3 are in collision and included in one cluster.

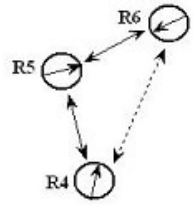

Figure 5b: Robots R4, R5 and R5, R6 are in collision and included in another cluster.

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

ALGORITHM: Collision Avoidance on Robot

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 *Rj* in c_i do steps 4 to 7
5. Evaluate *Obj* and find the aspiring velocity tuple $(vn_j, \omega n_j)$ for each robot *Rj* that minimizes *Obj*
6. Traverse along the trajectory described by $(vn_j, \omega n_j)$
7. If goal is reached, remove robot from the cluster.

4 SIMULATION RESULTS

Figure 6a: Initial phase of Robots heading

Figure 6b: Robots in 6a after a while avoiding collision

Figure 6c: Robots heading at 150 degree.

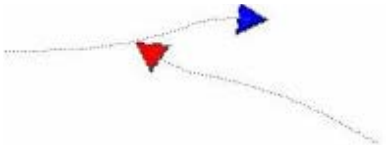

Figure 6d: Robots in 6c avoiding collision.

Figure 7a: Initial phase of three robots heading at different angles

Figure 7b: Robots in 7a avoiding collision.

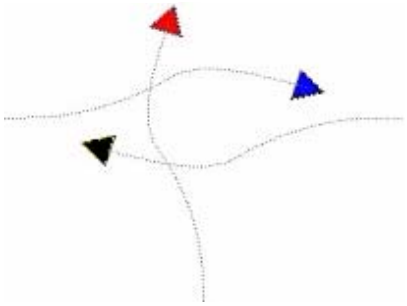

Figure 7c: Robots in 7a after avoiding collision reaching towards their goals.

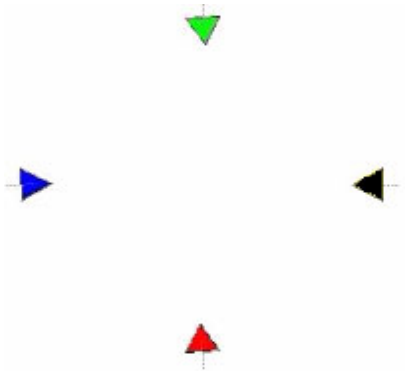

Figure 8a: Four robots in a cluster encountering collision.

Figures 6a and 6b show the trajectory executed by a pair of robots approaching each other at 90 degrees. Figure 6a shows the instant at which collision was first detected. Note that the evaluation of the objective function results in one of the robots not modifying its original trajectory at all. Similarly figures 6c and 6d show collision avoidance maneuver for robots that approach each other at 150 degrees. Figures 7a to 7c show various stages in collision avoidance between 3 robots while 8a to 8c show avoidance maneuver of 4 robots. These figures show the graceful change of state of robot from one velocity tuple to another respecting the kinematic constraints of the robots as well as the ability to avoid collisions simultaneously. Figure 9 shows the angular velocity profile for one of the four robots of figure 8 showing the continuous change in angular velocity whenever direction control is adapted.

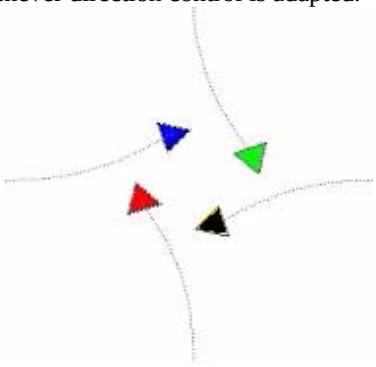

Figure 8b: Robots in 8a avoiding collision.

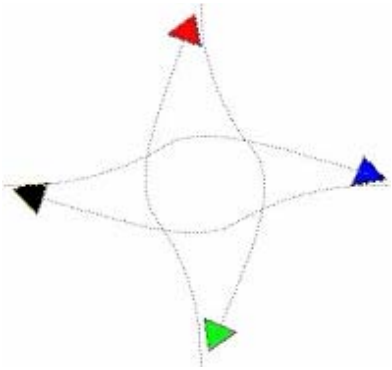

Figure 8c: Robots in 8a reaching their goals.

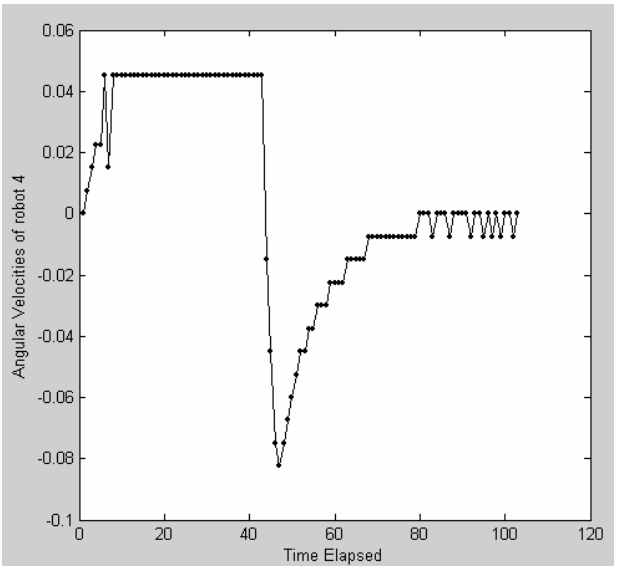

Figure 9: Angular velocity profile of a robot in figure 8a.

5 CONCLUSIONS AND SCOPE

A methodology for collision avoidance between multiple mobile robots, moving along straight lines or continuous curvature paths, is described in this paper. Simulation results vindicate that the said navigation function is able to come up with an aspiring linear and angular velocity for each robot, providing graceful paths avoiding collisions. Continuity in both linear and angular velocities are maintained, thereby preventing the need for stopping the robot whenever a directional change is entailed. Future scope of this effort is to investigate the complexity of the search procedure as well as to evolve efficient heuristics that result in further reduction of the search space.

ACKNOWLEDGEMENTS

This work is supported by CUBRC subcontract number 6665-2.

REFERENCES

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

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

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

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

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

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