Reactive Navigation of Multiple Moving Agents by Collaborative Resolution of Conflicts

K. Madhava Krishna
CSCE Department
University of Arkansas
Fayetteville, Arkansas 72701;
International Institute of Information Technology
Hyderabad, India
e-mail: mkrishna@uark.edu

Henry Hexmoor
CSCE Department
University of Arkansas
Fayetteville, Arkansas 72701
e-mail: hexmoor@uark.edu

Srinivas Chellappa
CS Department
Carnegie Mellon University
Pittsburgh, Pennsylvania
e-mail: schellap@andrew.cmu.edu

Received 2 January 2004; accepted 4 October 2004

In navigation that involves several moving agents or robots that are not in possession of each other's plans, a scheme for resolution of collision conflicts between them becomes mandatory. A resolution scheme is proposed in this paper specifically for the case where it is not feasible to have *a priori* the plans and locations of all other robots, robots can broadcast information between one another only within a specified communication distance, and a robot is restricted in its ability to react to collision conflicts that occur outside of a specified time interval called the reaction time interval. Collision conflicts are resolved through velocity control by a search operation in the robot's velocity space. The existence of a cooperative phase in conflict resolution is indicated by a failure of the search operation to find velocities in the individual velocity space of the respective robots involved in the conflict. A scheme for cooperative resolution of conflicts is modeled as a search in the joint velocity space of the robots involved in conflict when the search in the individual space yields a failure. The scheme for cooperative resolution may further involve modifying the states of robots not involved in any conflict. This phenomenon is characterized as the propagation phase where cooperation spreads to robots not directly involved in the conflict. Apart from presenting the methodology for the resolution of conflicts at various levels (individual, cooperative, and propagation), the paper also formally establishes the existence of the cooperative phase during real-time navigation of multiple mobile robots. The effect of varying robot parameters on the cooperative phase is presented and the increase in requirement for cooperation with the scaling up of the number of robots in a system is also illustrated. Simulation results involving several mobile robots are presented to indicate the efficacy of the proposed strategy.

1. INTRODUCTION

Robotic navigation for single-robot systems has been traditionally classified into planning and reactive based approaches. A scholarly exposition of various planning methodologies can be found in ref. 1. A similar exposition for dynamic environments is presented by Fujimora.[2] Multi-robot systems has become an active area of research since they facilitate improved efficiency, faster responses due to spread of computational burden, augmented capabilities, and discovery of emergent behaviors that arise from interaction between individual behaviors. Multiple mobile robot systems find applications in many areas such as material handling operations in difficult or hazardous terrains,[3] fault-tolerant systems,[4] covering and exploration of unmanned terrains,[5] and in cargo transportation.[6] Collaborative collision avoidance (CCA) between robots arises in many such multi-robot applications where robots need to crisscross each other's path in rapid succession or come together to a common location in large numbers. Whether it is a case of navigation of robots in a rescue and relief operation after an earthquake or while searching the various parts of a building or in the case of a fully automated shop floor or airports where there are only robots going about performing various chores, CCA becomes unavoidable.

The paper presents a novel three-tiered collaborative resolution strategy based on velocity control. The strategy is implemented in a distributed fashion across all robots in the system. In the first tier a robot attempts to resolve its conflict with another robot without anticipating a similar response from its counterpart. In other words each robot tries to resolve its conflict under the impression the other robot is careless or dumb. We call this as individual resolution where a conflict is said to be resolved if either of the two robots involved in the conflict resolves it independently. At the second tier robots involved in the conflict modify their velocities in a synchronized fashion such that the conflict is resolved. The second tier is resorted only when individual resolution fails. The second tier is also denoted as the cooperative phase of resolution where each robot makes an action by anticipating the action of the other. When the cooperative phase fails the third tier of resolution is adopted where cooperation spreads to robots that are not a part of the conflict and their help is sought in resolving the conflicts between the robots actually involved. The term cooperative is not a misnomer for it helps in achieving the following capabilities:

1. Avoid collision conflicts in a manner that conflicting agents do not come too near while avoiding one and another where and whenever possible. Thus agents take action in a fashion that benefits one another apart from avoiding collisions.
2. Provides a means of avoiding conflicts in situations where a single agent is unable to resolve the conflict individually.
3. Serves as a pointer to areas in the possible space of solutions where a search for solution is likely to be most fruitful.

Mathematically the individual resolution is characterized as a search in the velocity space of the respective robots involved in the conflict. When the search fails to yield a solution a search is resorted in the joint space of the robots involved in the conflict, which characterizes the cooperative phase. A formal depiction of the existence of the cooperative phase during navigation of a system of mobile robots is also presented and the effects of parametric variations on this phase are portrayed.

The rest of the paper is organized as follows. Section 2 places the work in the context of related works found in the literature and presents a brief literature review. Section 3 formulates the problem and the premises based on which the problem is formulated. Section 4 mathematically characterizes the three phases or tiers of resolution briefly mentioned above. Section 5 presents the algorithm while Section 6 validates the efficacy of the algorithm through simulation results. Section 7 discusses the limitations of the current approach and its future scope and ramifications and Section 8 winds up with concluding remarks.

2. PLACING IN CONTEXT OF OTHER WORK

In a manner similar to single-robot systems that are classified as planning or deliberative and reactive approaches, multi-robot navigation algorithms are traditionally classified as centralized[7–10] or decentralized.[11–13] In centralized approaches a single processor computes the plans for all the robots and the robots are controlled from a unified command. In the decentralized approach each robot computes its own plan and coordination between robots occurs when conflicts are detected while plans are exchanged and broadcast. The tradeoffs between the two approaches are well documented in the literature. In the case of a centralized approach that computes all possible conflicts over 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. Also, the requirement that all the world knowledge be localized at a single place often turns out to be not practical. Complete recalculation of paths is required even if one of the robot's plans is altered or environment changes. However, centralized approaches can guarantee completeness and optimality of the method. 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. It is more tolerant to changes in the environment or alterations in objectives of the agents. However, they are intrinsically incapable of satisfying optimality and completeness criterion.

A number of recent approaches try to provide algorithms that combine the advantages of both the approaches. Li and Chou[14] present a grouping strategy based on the hierarchical sphere tree that groups robots dynamically. Though the approach is purely a centralized one, the grouping strategy reduces planning time greatly for a large number of robots. The method is especially suitable in cases where the robots are crowded at their starting and goal configurations. Guo and Parker[15] present a distributed algorithm that provides for optimality. The algorithm is distributed in that each robot computes its own plan and the computations for optimal collision-free motion in the form of the modified velocity profile is done on each robot. A performance index based on the velocity profile is also computed for each robot and is broadcast to all other robots along with the velocity profile. All the robots adopt the profile corresponding to the minimum performance index as the optimal profile. Since each robot ends up calculating the velocity profiles for every other robot along their entire trajectories, the computational feasibility of the proposed method when the number of robot increases is in question. The complexity of the search space is also exponential in the number of robots in the system. In ref. 16 a methodology that is centralized within a network and distributed across networks is proposed. Networks get formed when robots are within a distance where communication is possible between them. A plan merging protocol (PMP) is presented in ref. 17 as a solution for the deadlock problem that occurs in distributed approaches.

The approach presented in this paper does not require the exchange or broadcast of complete plans as is the case with the typical decentralized approaches,[11,12,15] nor does it rely on assigning priorities to robots such as in refs. 8 and 12. The approach is based on changing velocities of the robots involved in a conflict in a synchronized fashion that is termed as cooperative resolution.

The present work is novel and different from others as the resolution of collision conflicts is attempted at three levels, namely the individual, cooperative, and propagation levels. Functionally cooperation is a methodology for pinning down velocities in the joint solution space of velocities of the robots involved in conflict when there exists no further solution in the individual solution spaces of those robots. When joint actions in the cooperative phase are not sufficient for conflict resolution, assistance of other robots that are in a conflict-free state at that instant is sought by the robots in conflict by propagating descriptions of the conflicts to them. When such free robots are also unable to resolve the conflict collision is deemed inevitable. The concept of propagating conflict resolution requests to robots not directly involved in a conflict is not found mentioned in robotics literature. Such kind of transmission of requests to robots, though not invoked frequently, is, however, helpful in resolving a class of conflicts which otherwise would not be possible as our simulation results reveal.

The method presented here is more akin to a real-time reactive setting where each robot is unaware of the complete plans of the other robots and the model of the environment. The work closest to the present is a scheme for cooperative collision avoidance by Fujimora's group[18] and a distributed fuzzy logic approach as reported in ref. 19. Their work is based on devising collision avoidance for two robots based on orientation and velocity control and extending this strategy for the multi-robot case based on the usual technique of priority-based averaging (PBA). However, we have proved in an earlier effort of ours[20] that such PBA techniques fail when individual actions that get weighted and averaged in the PBA are conflicting in nature. The work of Lumelsky[21] is of relation here in that it does not entail broadcast of plans to all other robots. It describes an extension of one of the *Bug* algorithms to a multi-robotic setting. There is not much mention of cooperation or collaborative efforts between the robots except in the limited sense of "reasonable behavior" that enables shrinking the size of the collision front of a robot that is sensed by another one.

Also of relevance here is the method of Guo and Parker[15] briefly discussed earlier. Their method involves first computing the complete plans for individual robots using a D* algorithm. Collisions are checked in the coordination diagram, which is the diagram obtained by decomposing the paths and velocities of each robot in the workspace (similar to Kant and Zucker[22]) and combining all the mappings to the *N*-dimensional space called the *coordination diagram* (CD). Collision areas in the CD are mapped as static obstacles and the D* algorithm is used sequentially on robots based on decreasing priority and collision-free paths recomputed. The method is similar to ours in that the eventual algorithm churns out velocity profiles for all robots in the workspace that are collision-free. However, since the entire path from the current location until the destination is evaluated for collision checks and computation of velocity profiles, the algorithm is prone to become intensive as the number of robots increase. Consequently, the scope of the algorithm to replan trajectories during real-time is severely limited with authors reporting 2 min of velocity planning on a *SPARC60* workstation. Hence all trajectories need to be planned before execution. On the contrary, since the current algorithm considers collisions only within a given T samples into the future, it is capable of generating collision-free trajectories for several robots in real-time with trajectories recomputed every time a space-time collision is detected during execution.

As mentioned before, conflict resolution through cooperation and propagation is pertinent to many applications that entail a number of robots that criss-cross each other in quick succession or in situations where robots find themselves coming together to get across an intersection from various directions which would otherwise result in a logjam.[23] In many such situations it is not reasonable to expect that the information about all such robots be maintained and their actions controlled from a central command nor does exchange or broadcast of their entire plans to one another at the time of eye contact appear intuitive.

3. PROBLEM FORMULATION AND PREMISES

The following premises have been made for algorithm development and simulations:

(a) Each robot Ri is assigned a start and goal locations and it has access to its current state and its current and aspiring velocities. The current state of Ri is represented as $\psi_i=\{vc_i, vn_i, \theta_i\}$ where vc, vn represent its current and aspiring velocities and θ its current motion direction.
(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 the same for all robots. Hence a robot Ri can predict when another robot Rj would attain its aspiring velocity vn from its current velocity vc.

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 sensors. Current and future velocity estimates obtained through such techniques have been used to achieve desired results such as in ref. 24, where the authors report a successful real-time algorithm for fast navigation. The circularity assumption has been commonly used in literature[15,16,19] essentially due to the fact that many indoor robots are circular. In any case circularity does not have any bearing on the three-tiered resolution strategy presented in this paper. Its only effects are in the manner in which collision checks are to be performed. Assumption (c) is also common to multi-robotic systems with successful implementations.[15,18] Assumption (d) 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.

3.1. Formalizing a Collision Conflict (CC)

Since robots are not point objects, a collision is not merely a space-time collision, i.e., two or more robots

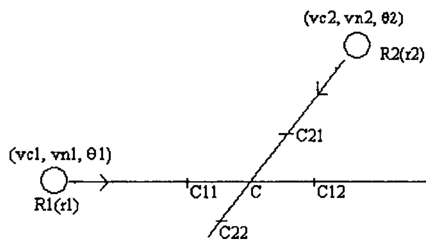


Figure 1. Two robots $R1$ and $R2$ with radii $r1$ and $r2$ along with their current states are shown. When $R1$ is shrunk to a point and $R2$ grown by radius of R1, C21 and C22 are centers of $R2$ where the path traced by $R1$ becomes tangential to $R2$.

reaching the same point at same time. Rather a CC is one that is spread over an interval of time and hence robots are prohibited from moving over a range of velocities for avoiding it. The CC is formalized here for the simple case of two robots moving at constant velocities.

Shown in Figure 1 are two robots R1 and R2 of radii $r1$ and $r2$, whose states are $\psi_1 = (vc_1, vn_1, \theta_1)$ and $\psi_2 = (vc_2, vn_2, \theta_2)$, respectively, where vc_1, vc_2 are the current velocities while vn_1, vn_2 are the aspiring velocities for R1 and R2, respectively. Point C in the figure represents the intersection of the future paths traced by their centers. For the purpose of collision detection, one of the robots R1 is shrunk to a point and the other R2 is grown by the radius of the shrunken robot.

The points of interest are the centers C21 and C22 of R2 where the path traced by the point robot R1 becomes tangential to R2. At all points between C21 and C22 R2 can have a potential collision with R1. C21 and C22 are at distances $(r1 + r2)\cos ec(|\theta_1 - \theta_2|)$ on either side of C. The time taken by R2 to reach C21 and C22 given its current state (vc_2, vn_2, θ_2) is denoted by t_{21} and t_{22}. Similar computations are made for R1 with respect to R2 by making R2 a point and growing R1 by $r2$. Locations C11 and C12 and the time taken by R1 to reach them, t_{11} and t_{12}, are thus computed. A collision conflict or CC is said to be averted between R1 and R2 if and only if $[t_{11}, t_{12}] \cap [t_{21}, t_{22}] \in \Phi$. The locations C11, C12, C21, and C22 are marked in Figure 1.

A direct collision conflict (DC) between robots R1 and R2 is said to occur if R1 occupies a space between C11 and C12 when the center of R2 lies between C21 and C22 at some time t.

A robot is concerned only about its time nearest CC with another robot within a given reaction time t_r that is same for all robots. A uniform t_r across all robots facilitates commutativity in collision relations, i.e., if R1 has a CC with R2 in t_r so does R2 with respect to R1.

4. THREE PHASES OF RESOLUTION

Conflict resolution between robots is attempted in three phases or three tiers as mentioned in Section 1. The modality of choosing the velocity determines which of the three phases is being adopted for resolution.

Consider the set S_T, representing the set of all possible solutions that resolve conflicts among the robots involved. Each member of the set s_i is an ordered tuple of velocities represented as $s_i = \{v_{1i}, v_{2i}, \ldots, v_{Ni}\}$, for each of the N robots involved in the conflict. The set is infinite and also not countable and hence the subscript i in s_i is used only for notational convenience. The robots avoid the conflict by attaining these velocities in the stipulated time tuplet $\{t_{1i}, t_{2i}, \ldots, t_{Ni}\}$ corresponding to the velocity tuplet. For ease of representation we denote each $s_i \in S_T$ by the velocity tuplet alone under the implicit assumption that these velocities are attained in the stipulated times as specified by the corresponding time tuplet.

The *cooperative space* is represented by the set $S_C \subseteq S_T$, i.e., the cooperative space is a subset of the total solution space and where every robot involved in the conflict is required to modify its current velocity to avoid the conflict. In other words, robots modify the velocity in such a manner that each of the robots involved has a part to play in resolving the conflict. Or, if any of the robots had not modified its velocity, it would have resulted in one or more collisions among the set of robots involved in the conflict.

The *cooperative phase* in navigation is defined by the condition $S_C = S_T$, where each robot has no other choice but to cooperate in order to resolve conflicts. In individual resolution robots choose velocities in the space of $S_I = S_T - S_C$, where the entailment for every robot to cooperate does not exist. When $S_I = \Phi$, the null set, we say the navigation has entered the cooperative phase.

Figure 2(a) shows evolution of trajectories of two robots, marked R1 and R2, moving orthogonal to one another. The arrows show the location of the two robots at time $t = 0$ sample. The robots move with identical speed of $v_{R1} = v_{R2} = 2.5$ units. The states of the two robots are represented as $\psi_1 = (v_{R1}, v_{R1}, 0)$ and $\psi_2 = (v_{R2}, v_{R2}, -90)$. The equality in the current and

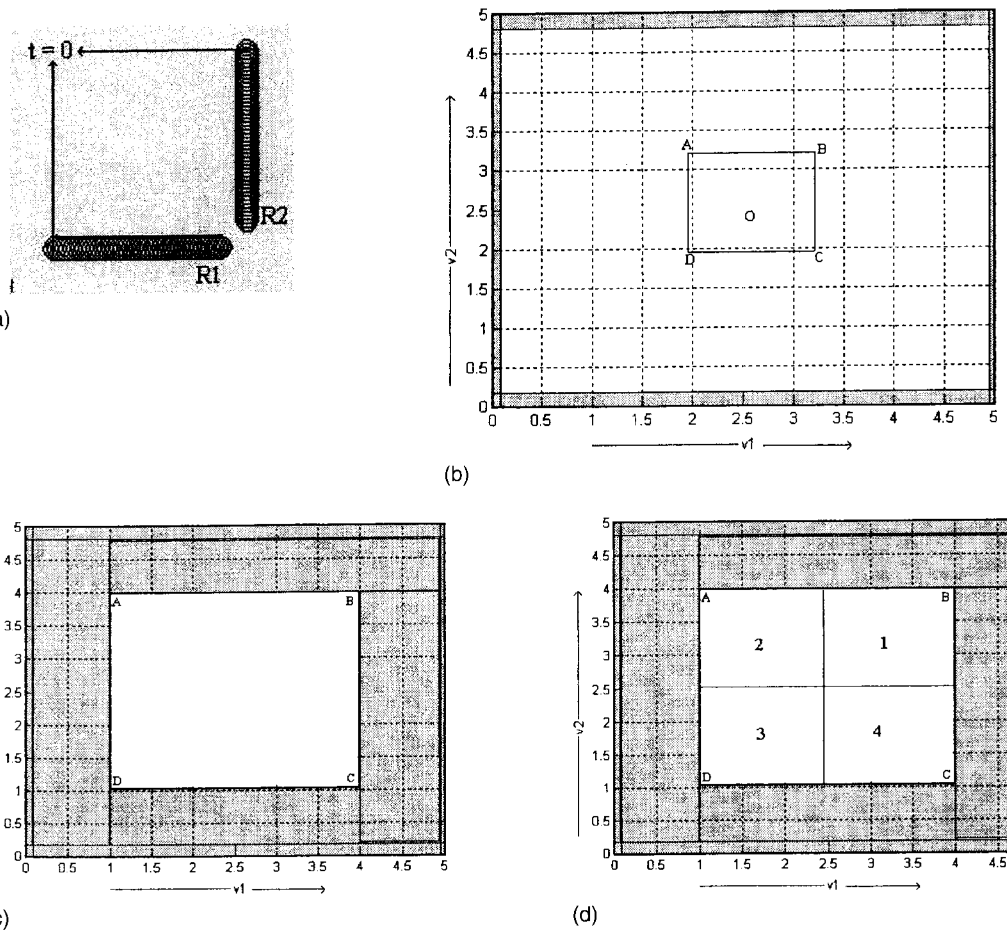


Figure 2. (a) Two robots approach each other along orthogonal directions. The locations of the robot at time $t=0$ is shown marked by arrows. (b) The range of possible velocities for either of the robots R1 and R2 shown along the x and y axis. The inner rectangle represents the area where the robots need to cooperatively find a solution for the pair of velocities $\{v_{R1}=2.5, v_{R2}=2.5\}$. (c) At $t=25$ the conflict area occupies the entire possible space of velocities. (d) Search is limited to quadrants 2 and 4 where robot actions are complementary.

aspiring velocities merely indicates that the robot moves with uniform velocity and is not a loss of generality from the case when the aspiring velocity differs from the current. The subsequent discussion holds equally for the case when the current and aspiring velocities differ. Corresponding to this location of the robots at the beginning of their trajectories, Figure 2(b) depicts the total space of velocities bounded within the outer rectangle (shown thick) whose length and breadth are 5 units, respectively. In other words, each robot can have velocities in the interval [0,5] units. The abcissa represents the range for one of the robots (R1) and the ordinate the range for the other (R2). The center of the figure marked as O indicates the location corresponding to their respective velocities of 2.5 units each. The strips of shaded region represent those velocities not reachable from O due to the limits of acceleration and deceleration. The inner rectangle, marked ABCD, represents the region of velocities where a possible solution can be found **if and only if** both robots alter their velocities. For $v_{R1}=2.5$ corresponding to R1's velocity on the abcissa, R2 must possess a velocity, which lies either above or below the segments AB and CD of the rectangle when projected onto the ordinate. Similarly for $v_{R2}=2.5$ on the ordinate, robot R1 must possess a velocity either to the right or left of the segments BC and AD when projected onto the abcissa to avert collision. We denote the velocities that make R1 reach the velocities at D and C from O as v_{11} and v_{12}, respectively, while the velocities that make R2 reach A and D from O are denoted by v_{21} and v_{22}, respectively. With reference to Figure 1, v_{11} and v_{12} correspond to velocities that enable R1 to reach C11 and C12 in the time R2 reaches C22 and C21, respectively, without changing its current aspiring velocity from v_{R2}.

4.1. Characterizing the Individual Phase

A pair of robots R1 and R2, which have a DC between them are said to be in individual phase of navigation if the conflict is resolved by either of the following two means:

(i) R1 controls its velocity to v_{12} such that it is able to get past C12 before R2 reaches C21 with its aspiring velocity as v_{R2} or R1 controls its velocity to v_{11} such that it does not reach C11 before R2 reaches C22 without changing its aspiring velocity from v_{R2}.

(ii) R2 controls its velocity to v_{22} such that it is able to get past C22 before R1 reaches C11 with its current aspiring velocity as v_{R1} or R2 controls its velocity to v_{21} such that it does not reach C21 before R1 reaches C12 without changing its aspiring velocity from v_{R1}.

In both cases it would suffice that only one of the two robots controls or modifies its aspiring velocity. This indeed is the crux of the individual phase where at least one of the two robots is able to individually avoid the conflict without requiring the other to take action. Thus the range of velocities that permit individual resolution of conflict by R1 is given by: $v \in [0,v_{11}]\cup[v_{12},v_{1M}]$, where v_{1M} represents the maximum permissible velocity for R1, which is 5 units in Figure 2(b). They are given by $v_{11}=vc_1+a_{-m}t_{22} \pm \sqrt{(vc_1+a_{-m}t_{22})^2+(vc_1^2+2a_{-m}s)}$. Here s denotes the distance from R1's current location to C11, a_{-m} is the maximum possible deceleration, and t_{22} is the time taken by R2 to reach C22 given its current state ψ_2. In the same vein the velocity that causes R1 to be ahead of C12 when R2 reaches C21 under maximum acceleration, a_m, is given by

$$v_{12}=vc_1+a_m t_{21}\pm \sqrt{(vc_1+a_m t_{21})^2+(vc_1^2+2a_m s')},$$

where s', the distance from R1's current location to C12, can also be written as $s'=s+(r1+r2)\cos ec(|\theta_1-\theta_2|)$ and t_{21} is the time taken by R2 to reach C21 given its current state ψ_2. In a similar fashion velocities v_{21} and v_{22} are computed. Thus some of the possible sets of solutions from the set S_T are enumerated as

$$s_1=\{v_{11},v_{R2}\},\quad s_2=\{v_{12},v_{R2}\},\quad s_3=\{v_{R1},v_{21}\},$$

$$s_4=\{v_{R1},v_{22}\},\quad s_5=\{v_{11},v_{22}\},\quad s_6=\{v_{21},v_{12}\}.$$

From the above list the first four solutions involve change in velocities of only one of the robots while the last two solutions involve change in velocities of both the robots. The last two solutions are examples of collaboration even in the individual phase as robots involve in a combined effort to avoid conflict even though they are not entailed to do so. The collaboration in the individual phase achieves the first capability mentioned in Section 1 of avoiding conflicts in a manner that conflicting agents do not come too near while avoiding one and another. Amongst the last two solutions (s_5,s_6), that one is selected which involves minimal change from the current state of the respective robots. The last two solutions indicate that collaboration involves complementary decision making since, as one of the robots accelerates from its current velocity, the other decelerates.

Henceforth, for any robot the lower velocity is denoted as v_1 and the higher velocity by v_2 with the robot index dropped for notational simplicity. In other words, the lower and upper velocities are denoted as v_1 and v_2 instead of v_{21} and v_{22} for R2 or instead of v_{l1}, v_{l2} for RI.

It is to be noted that the phrase that a robot change or modify its velocity is more precisely stated as the robot control or modify its *aspiring* velocity.

4.2. The Cooperative Phase

The area enclosed within the rectangle ABCD of Figure 2(b) is termed as conflict area for the pair of velocities $\{v_{R1},v_{R2}\}$ for time $t=0$ and denoted as $CA(v_{R1},v_{R2},t=0)$. Let $V_{r1}=[v_{l1},v_{h1}]$ represent the range of velocities for which there is a collision for robot R1 when R2 possesses a velocity v_{R2}. Similarly let $V_{r2}=[v_{l2},v_{h2}]$ represent the range of velocities for which there is a collision for robot R2 when robot R1 possesses a velocity v_{R1}. We define the conflict area for the velocity pair $\{v_{R1},v_{R2}\}$ for a given time t as $CA(v_{R1},v_{R2},t)=\{v_{R1},v_{R2}|v_{R1}\in V_{r1},v_{R2}\in V_{r2}\}$. The velocities v_{l1}, v_{h1} for R1 and v_{l2}, v_{h2} for R2 are arbitrarily close to their respective upper and lower control velocities v_1, v_2 that are used for resolving conflicts. In other words $|v_{l1}-v_1|<\varepsilon$ for R1, $|v_{l2}-v_1|<\varepsilon$ for R2 and similarly $|v_{h1}-v_2|<\varepsilon$, $|v_{h2}-v_2|<\varepsilon$ where ε is any arbitrarily low value. With progress in time, if control actions to avoid conflicts were not resorted, the conflict area expands to occupy

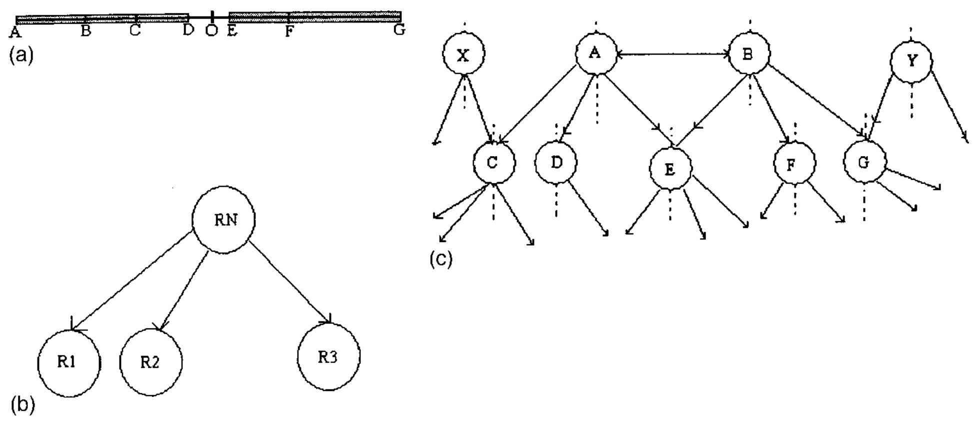


Figure 3. (a) The velocity axis of the robot whose current velocity is at O. Shaded portions represent inaccessible velocities or regions of conflict in the velocity space with other robots. (b) RN propagates requests to R1 and R2 on the left due to conflicts with lower velocities and on the right to R3 due to higher velocity. (c) Propagation can result in a generalized multiple tree or forest structure whose links represent the flow of conflicts between robots.

the entire space of possible velocities. This is shown in Figure 2(c) where the conflict area fills up the entire velocity space. Any combination of velocities outside the rectangle ABCD now falls inside the shaded border strips, which are not accessible from O due to the limits imposed by acceleration and deceleration. Hence individual resolution of conflicts by any one of the robots is ruled out since the upper and lower velocities v_1 and v_2 for both R1 and R2 now lie inside the shaded area.

Since the upper and lower velocities are situated well inside the shaded area, the velocity pairs corresponding to the vertices ABCD of the conflict area is unknown. Hence a cooperative search ensues for finding the pair of velocities that would resolve the conflict. Cooperation between robots averts an exhaustive search and restricts it two quadrants 2 and 4 [Figure 2(d)] of the conflict area where robot actions are complementary and yield best results for conflict resolution. Since a search is nonetheless time intensive, the rules (i) and (ii) mentioned earlier where robots resort to maximum acceleration and deceleration in a complementary fashion offer the boundary value solutions. A failure of the solutions at the bounds implies a failure anywhere inside and a pointer to resort to conflict propagation as the last resort.

A pair of robots R1 and R2 is said to be in cooperative phase of navigation if and only if they are able to resolve the collision conflict between the two through either of the following rules:

(i) R1 is able to get past C12 under maximum acceleration before R2 can get to C21 under maximum deceleration.

(ii) R2 is able to get past C22 under maximum acceleration before R1 can get to C11 under maximum deceleration.

The difference between the above rules and those mentioned in Section 4.1 is that in Section 4.1 R1 finds a control velocity that avoids conflict with R2 under the premise that R2 would not alter its aspiring velocity. Similarly R2 finds a control velocity under the impression R1 is dumb. However, in the cooperative phase R1 anticipates a modification in the aspiring velocity of R2 such as in rule 1 where R2 modifies its state (and hence its aspiring velocity) such that it reaches C12 under maximum deceleration. Under this anticipation of change in R2's control action R1 tries to attain the corresponding control velocity that would avoid conflict.

4.3. The Propagation Phase

Figure 3(a) shows the velocity axis for a robot RN. RN's current velocity is shown as O in the figure. The portions of the velocity axis shown shaded are those portions of the velocity forbidden from the current state of RN either because they are not reachable or they conflict with other robots. For example, portions AB and FG on the axis are not reachable while portions BC, CD, and EF conflict with robots R1, R2, and R3, respectively. At O, RN enters into a new conflict with a robot RM. Individual resolution of RN's conflict with RM results in conflict with R1 on the lower side and enters a forbidden region on the upper side. Similarly RM's individual resolution leads to conflict with other robots or results in access of forbidden regions. When RN cooperates with RM to resolve the conflict, it again results in conflict with R2 on the lower side and R3 on the upper side. In such a scenario RN propagates a cooperation request to R1, R2, and R3. The tree structure of Figure 3(b) depicts this propagation. All nodes on the left of RN are requests arising due to lower aspiring velocities while nodes on the right of RN are requests that arise due to higher aspiring velocities. This convention would be followed for all robots involved in the propagation phase. Thus robot RN's resolution of its DC (direct conflict) with RM results in indirect conflict (IDC) with robots R1, R2, and R3 and hence RN is considered to be in IDC with R1, R2, and R3. When R1 or R2 try to collaborate in conflict resolution of RN by changing their aspiring velocities it can lead to further conflict with other robots to whom requests are transmitted by R1 or R2 for collaboration. Thus propagation can be recursive and results in a multiple tree-like or forest data structure shown in Figure 3(c). A graph-like propagation is avoided since a robot-node that has already propagated a request to another node below does not entertain any new requests.

Thus any robot has the following functionalities with regard to propagating cooperation which are taken up for discussion below:

- Transmit requests
- Receive requests
- Reply to requests
- Receive replies

Transmitting requests: A robot RT transmits as a request to another robot RR a packet that contains the following information:

Source: The robot that originally sourced the request.

T-robot: The robot that is currently transmitting the request, which is itself (RT).

R-robot: The robot to which the request is transmitted (RR).

V-aspire: The velocity which the transmitting robot RT would aspire to have in order to avoid conflict which it has currently with some robot, R1, but which results in conflict with the robot to which the request is transmitted, RR.

t-collide: The minimum time to collision that RT currently has with R1.

Mode: If the aspiring velocity V-aspire is higher than RT's current velocity, then *mode* takes the tag *high* else it is assigned the tag *low*.

S-mode: If the S-mode has the tag *high*, then it indicates that RT and RR would be the right descendants of the source robot, else it indicates that they are left descendants.

RT transmits a request to RR only if RR is in a state of entertaining requests, else the request is not transmitted to RR. A robot RR accepts a request to collaborate to resolve RT's DC with another robot only if RR itself is not involved in a DC.

Receiving requests: A robot RR can receive single or multiple requests. A robot that receives requests from more than one robot to participate in its conflict, such as *C* receives requests from *A* and *X* in Figure 3(c) prioritizes the requests in order of time to collision of *A* and *C* with the robots with which *A* and *C* are in conflict. The requests are processed in the order of their priorities. If a request could be resolved, a success reply is propagated back to the robot that transmitted the request. A success reply indicates that RR intends to modify its aspiring velocity with respect to that request. Hence it cannot modify its velocity to the remaining requests it has received and hence propagates a failure back to the remaining robots that had requested RR. If a request is not solved, it is either propagated to another robot or a failure is transmitted back to the robot that transmitted. Unless all the requests had resulted in a failure being transmitted, RR does not entertain any new request for that sample. In other words, if RR has managed to solve at least one request or passed at least one to another robot, it does not accept any new request for that sample. A sample is one complete execution of the entire reactive loop or module across all robots.

Replying requests: A request is replied back as a success or failure to the robot that transmitted in the manner described in the previous paragraph.

Receiving replies: A robot RT that had transmitted requests to other robots receives a success or failure reply from the robots to which it had transmitted. If a success reply is received, RT sees whether the reply is from its left or right child. From the side on which the success was received a check is made if all other robots that had received the request from RT with respect to that particular aspiring velocity of RT have also replied a success. If all other children with respect to that v-aspire from that side (left or right accordingly) have propagated a success, then RT propagates a success to the parent whose request to RT has now succeeded. It removes links with all its remaining children since it has already achieved a success on one of its v-aspire, which would become its new aspiring velocity. To its remaining parents it propagates a failure reply. On the other hand, if RT receives a failure reply from its left or right child, it propagates a failure reply to RT's parent responsible for that request. Simultaneously it removes all other children on the particular side from which the failure reply was received with respect to that aspiring velocity.

This process of replying requests and receiving replies is recursed back till the original source or the root.

5. THE ALGORITHM

A robot can find itself involved in the following kinds of conflicts:

Mutual direct conflict (MDC): A pair of robots R1, R2 are said to be in MDC with one another if R1's first direct conflict (DC) in reaction time t_r is with R2 and R2's first DC in reaction time t_r is with R1.

Nonmutual Direct Conflict (NMDC): A robot R1 is said to be in NMDC with R2 when R1's first DC in t_r is with R2 while R2's first DC in t_r is, however, not with R1 but with some other robot R3.

Indirect conflict (IDC): A robot R1 is said to be in IDC with R2 if R1 has no MDC or NMDC at that instant and has received a request for resolving R2's conflict with some other robot R3.

The broad steps of the overall algorithm are delineated below. Each step of the algorithm itself is further discussed later.

For any robot RI do the following steps:

1. *If* RI has time nearest conflict within t_r with another robot RJ then do steps 1a to 1c.
    1a. $vn_I \leftarrow IresConf$(RI,RJ); Obtain the next aspiring velocity of RI, vn_I, through $IresConf$ module which attempts individual resolution of the conflict of RI.
    1b. *If* step 1a fails to resolve conflict *and* RI's conflict with RJ is of type *MDC then* obtain vn_I as $vn_I \leftarrow CResMDC$(RI,RJ); $CResMDC$ module attempts a cooperative resolution of the conflict between RI and RJ.
    1c. *If* step 1a fails *and* RI's conflict with RJ is of type *NMDC then* obtain vn_I as $vn_I \leftarrow CResNMDC$(RI,RJ); $CResNMDC$ module attempts a cooperative resolution of the non-mutual conflict between RI and RJ.
    1d. *If* either of the steps 1a or 1b or 1c leads to further conflicts with other robots propagate descriptions of the conflicts to those robots.
2. *If* RI has received a request from RK to solve RK's DC with some other RM *and* RI itself has no DC with any other robot *then* obtain vn_I as $vn_I \leftarrow ResIDC$(RI,RK); $ResIDC$ module attempts to resolve the indirect conflict between RI and RK.
3. Move RI on its current direction with its new aspiring velocity vn_I.
4. *If* RI has reached its target *then* halt navigation else repeat from step 1.

Every robot in the system maintains for itself a data structure termed *Vlist. Vlist* stores the areas in the velocity space of that robot that are either forbidden or conflicting given its current state. The velocity space of a robot is the set of all velocities that can be attained within its maximum permissible velocity as imposed by its motor ratings. Hence for a robot RI whose maximum permissible velocity is v_{MI}, the velocity space is the set of all velocities v_{RI} that belong to $[0, v_{MI}]$. This set is denoted by $V_S(RI)$. Thus the velocity space of the robot is also the velocity axis shown in Figure 3(a). *Vlist* is the data structure that maintains the end points of the shaded regions shown in Figure 3(a). A shaded region is considered forbidden if that region is not accessible for the robot from its current state. A shaded region is termed conflicting when aspiring for a velocity in that region by RI yields conflicts with another robot. The set of all conflicting regions is denoted as $V_C(RI)$ and the set of all forbidden regions by $V_F(RI)$. The set of all prohibited regions, $V_P(RI)$, is given by $V_P = V_C \cup V_F$, where the qualifier RI is omitted for ease of expression. The set $\bar{V}_P$ is the set of all collision-free velocities for R1, given its current state as ψ_I. Based on this data structure the various modules of the algorithm function as follows:

$IresConf$(RI,RJ) module: The module for individual resolution of RI's conflict with RJ computes the next aspiring velocity for RI, vn_I, that is conflict free if it exists.

Let t_{IC}, t_{JC} represent the time taken by RI to reach the point C shown in Figure 1, which is the intersection point of the future paths traced by the centers of RI and RJ. The following steps characterize this module:

1. Given ψ_I the lower and higher velocities for RI, v_1 and v_2 are computed.
2. *If* $v_1 \in V_C$ find the highest velocity lesser than v_1 that is collision free if it exists and denote that as the new lower velocity v_1.

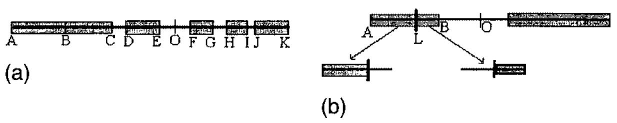


Figure 4. (a) An illustrative velocity axis. Shaded regions AB and JK are forbidden while other shaded regions are conflicting. (b) Illustrating indirect resolution of conflict in the propagation phase.

3. *If* $v_2 \in V_C$ find the lowest velocity higher than v_2 that is collision free if it exists and denote that as the new higher velocity v_2.
4. *If* $v_1 \notin V_P$ *and* $v_2 \notin V_P$ *do* steps 4a and 4b
   4a. *If* $t_{IC} < t_{JC}$ set $vn_I \leftarrow v_2$; If RI reaches C faster than RJ then assign the higher velocity as its next aspiring velocity.
   4b. *If* $t_{IC} > t_{JC}$ set $vn_I \leftarrow v_1$; If RI reaches C slower than RJ then assign the lower velocity as its next aspiring velocity.
5. If $v_1 \notin V_P$ *and* $v_2 \in V_P$ set $vn_I \leftarrow v_1$; Since only the lower velocity is collision free assign that as the next aspiring velocity.
6. If $v_1 \in V_P$ *and* $v_2 \in V_P$ set $vn_I \leftarrow v_1$; Since only the higher velocity is collision free assign that as the next aspiring velocity.

Similar computations are made for RJ. A check for noncomplementary actions is made to look out for both RI and RJ being assigned their respective higher or lower velocities, which could once again lead to a conflict. In such a case the old aspiring velocity is reassigned to the robot that needs to accelerate or decelerate by a larger amount than the other.

Steps 2 and 3 are explained through Figure 4, which shows an illustrative velocity axis for robot RI. Shaded regions AB and JK represent forbidden regions while other shaded regions are conflicting. The current velocity of the robot is marked as O on the axis. Shaded regions BC and DE on the lower side of O conflict with robots R1 and R2 while regions FG and HI conflict with R3 and R4 on the upper side. The current aspiring velocity of the robot, however, collides with a new robot RJ that has not been indicated on the velocity axis.

If the lower velocity v_1 belongs to the shaded region DE, step 2 of $IresConf$ module finds a conflict-free velocity that is just to the left of D and away from D by some threshold value. Similarly, if higher velocity v_2 falls in the shaded region FG, step 3 finds a conflict-free velocity that is just to the right of G.

$CresMDC$(RI,RJ) module: The module for cooperative resolution of RI's conflict with RJ computes the next aspiring velocity for RI, vn_I, that is conflict-free if it exists.

Computation for RI:

1. *If* $t_{IC} > t_{JC}$
   1a. If $v_1 \in V_C$ find v_{cfI}, the lowest free velocity for RI on the higher side of the shaded region where v_1 falls.
   1b. Find t_{I1}, time taken by RI to reach the point CI1 (same as point CI1 in Figure 1) assuming v_{cfI} as the next aspiring velocity.
   1c. If $v_1 \in V_F$ find t_{I1} as the time taken by RI to reach the point CI1 under maximum deceleration.
2. Corresponding to the condition $t_{IC} > t_{JC}$ RJ evaluates for itself steps 2a to 2c:
   2a. If $v_2 \in V_C$ find v_{cfJ}, the highest free velocity for RI on the lower side of the shaded region where v_2 falls.
   2b. Find t_{J2}, time taken by RJ to reach the point CJ2 (same as point C22 in Figure 1) assuming v_{cfJ} as the next aspiring velocity.
   2c. If $v_2 \in V_F$ find t_{J2} as the time taken by RJ to reach the point CJ2 under maximum deceleration.
3. RI and RJ exchange t_{I1} and t_{J2}.
4. If $t_{I1} > t_{J2}$ assign v_{cfJ} as the next aspiring velocity for RJ and v_{cfI} as the next aspiring velocity of RI.
5. If $t_{I1} < t_{J2}$ steps 1 to 4 are repeated with steps 1 and 2 swapped, i.e., RI tries to find a velocity on the higher side while RJ computes on the lower side that would satisfy the condition $t_{I2} < t_{JI}$.

Steps 1a and 2a need further discussion. Step 1a considers the lower velocity v_1 as a possible candidate for vn_I. Since v_1 is in conflict with some other robot and since it is already in the cooperative resolution module it implies search for a collision-free velocity further lower than v_1 has already failed in the individual resolution module. Hence a new velocity v_1 that is lower than the current velocity at O but is higher than the v_1, which is currently in conflict, is found. With respect to Figure 4(a), if v_1 falls in shaded region BC conflicting with R1, individual resolution would fail for there's no free velocity space on the left of BC. Hence step 1a finds a velocity that is just to the right of C that is conflict-free with R1 and other robots on the velocity axis but would still collide with RJ, the robot with which the conflict has been currently detected. Similarly step 2a does the same operation on the higher side for RJ. Steps 1b and 2b describe the cooperative resolution that occurs when velocities straight away fall in forbidden regions, which was also discussed in Section 4.2. Step 4 checks if v_1 and v_2 as computed in 1a and 2a (or 1b and 2b) would resolve the conflict between RI and RJ. This condition is given by $t_{I1} > t_{J2}$ or, in other words, the condition that RI reaches CI1 after RJ reaches CJ2. Step 5 checks for the case that the computed velocities do not resolve the conflict, in which case the process is repeated with RI searching on the higher side and RJ on the lower side.

ResNMDC module: This module is very similar to the ResMDC module except with a delay where the robot RI that has a NMDC with RJ waits for RJ to resolve its MDC before modifying its aspiring velocity.

ResIDC(RR,RT,V-aspire(RT),Mode(RT),Source): ResIDC module involves the robot RR currently not in any kind of direct conflict with another robot to modify its current aspiring velocity in a manner such that the transmitting robot RT can have V-aspire as its next aspiring velocity. V-aspire enables RT to avoid its direct conflict with some other robot RK with which it currently experiences a conflict. Since V-aspire would in future conflict with RR, RT requests RR for cooperation. The ResIDC module is described graphically through Figure 4(b). Robot RT with current velocity at O has its lower aspiring velocity fall in the shaded region AB conflicting with robot RR. A thick vertical line at the location L shows the location of the lower velocity. Robot RT requires its aspiring velocity to be at L to avoid its current direct conflict with RK. Hence RR tries to modify its aspiring velocity such that the shaded velocity comes on the left or right of L as indicated by the two arrows in Figure 4(b), such that the velocity at L becomes collision-free. If RR succeeds in modifying its aspiring velocity such that L becomes collision-free, it transmits a success reply to RT. If these operations result in further conflicts with other robots, a request to cooperate is propagated to them. If RR's dynamic capabilities as imposed by its limits of acceleration fail to make the velocity at L collision-free, RR transmits a failure reply to RT.

The maintenance of the forest structure of Figure 3(c) and the role of every robot-node in creating new links, deleting existing links, and other associated operations is not discussed here for brevity.

6. SIMULATION RESULTS

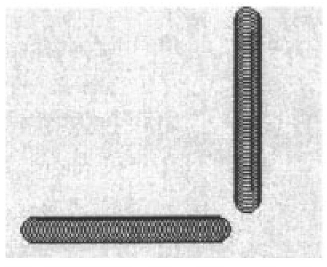
This section on simulation results is organized as follows. Initially the existence of the cooperative phase in a multi-robot navigation system is portrayed in Section 6.1 and the effects of parametric variations on the time span of the cooperative phase are presented. This section is the simulation counterpart of Section 4.2 where the existence of the cooperative phase was illustrated as the expansion of the conflict area to occupy the entire space of possible velocities. In Section 6.2 the inevitability of cooperative phase is discussed. Section 6.3 presents results of a multi-bodied system and illustrates the effects of scaling up of the number of robots on the requirement to cooperate and propagate.

6.1. Portraying the Existence through Simulation

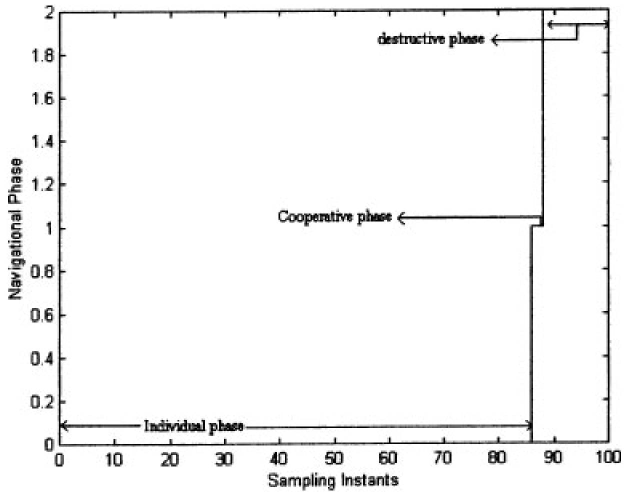
The existence of the cooperative phase in navigation and its time span of existence *vis-à-vis* the angular separation between robot heading angles, $(|\theta_1 - \theta_2|)$, for the two-bodied case is first presented. Robots are made to approach each other at various angular separations and the percentage of solution space available for choosing control velocities that could avoid collision is computed. The percentage availability of solution space for an individual robot is computed as $(L_{US}/L_T)\cdot 100$, where L_{US} is the length of the line that is not shaded on the velocity axis and L_T refers to the total length of the velocity axis.

However, the robots do not chose these velocities but continue to proceed until the solution space dries up completely, indicating the onset of cooperative phase. If the robots continue to navigate without entering into a cooperative scheme for collision avoidance, a stage arises where even cooperation would not prevent collision. This final phase is termed as the destructive phase, where the robots inevitably have to collide into each other.

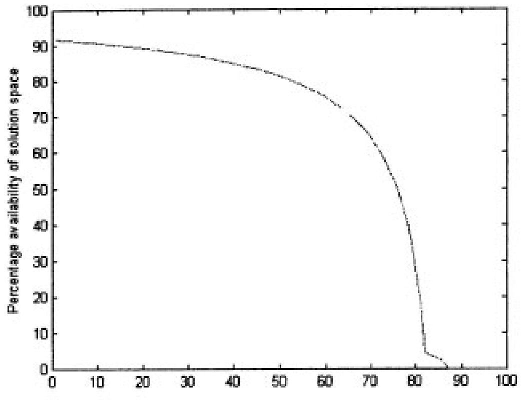
Figure 5(a) depicts a two-bodied case where the robots approach each other with an angular separation of 90 deg. Figure 5(b) illustrates a graph that takes discrete values on the y-axis versus sampling instants on the x-axis. Sampling instants denote the onset of a new reactive loop of the algorithm. The delays are appropriately introduced in the algorithm to make the time-length of every reactive cycle and hence every sample constant. For all the simulations portrayed in this section (6.1) the maximum velocity of either of the robots is 5 pixels per sample and the maximum acceleration for both the robots is 2 units. The discrete values on the ordinate (y-axis or vertical axis) of Figure 4(b) indicate the various phases of robot navigation. An ordinate value of 0 denotes the *individual phase* where the robot can avoid collision in-



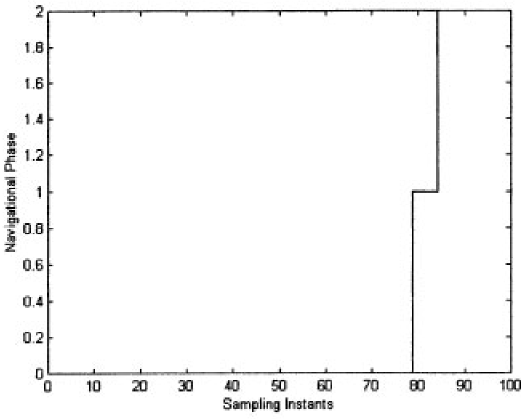
(a)



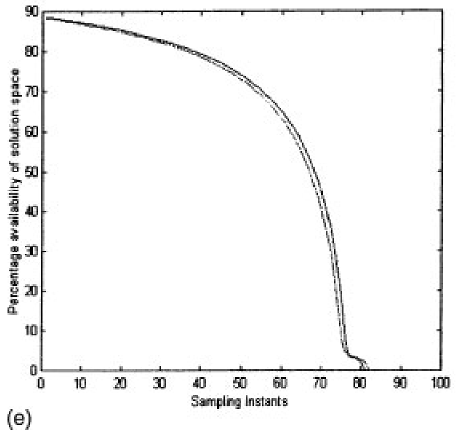
(b)



(c)



(d)



(e)

Figure 5. (a) Two robots approach each other with separation of 90 deg. (b) The various phases of navigation versus sampling instants for an angular separation of 90 deg between robot heading angles. (c) Percentage availability of solution space versus sampling instants. (d) Phases of navigation versus sampling instants for an angular separation of 45 deg between robots. (e) Percentage availability of the solution space does not overlap precisely in this case for the two robots and hence the demarcation between the two plots.

dividually without entering into cooperation. An ordinate value 1 signifies the *cooperative phase* of navigation where the solution space has dried up and the robots needs to cooperate for averting collision. Finally, value 2 on the ordinate implies the *destructive phase* where the robots inevitably need to collide or have already collided.

In Figure 5(b) the individual phase spans for 86 sampling instants from the start of navigation while the cooperative phase extends for only two instants after which the robots enter their destructive phase. Figure 5(c) depicts the percentage availability of solution space for choosing control velocities corresponding to the various navigational states of the robot in Figure 5(b). It is evident from Figure 5(c) that the range of options available in the solution space decreases with time and hits zero in the 86th sample where correspondingly in Figure 5(b) the robot enters the cooperative phase of navigation on that instant. Equivalently, the conflict area expands to occupy the entire space of possible velocities as depicted in 2c. Figures 5(d) and 5(e) depict the phases of navigation and the availability of solution space when robot pairs approach one another with an angular separation of 45 deg, while Figures 5(f) and 5(g) depict the same for a separation of 15 deg. These figures indicate that the cooperative phase onsets earlier as the angular separation decreases and correspondingly the range of options on the solution space reduces to zero faster. The span of the cooperative phase also increases with decrease in angular separation and in Figure 5(f) it becomes rather prominent. It is also worthwhile to note in Figures 5(e) and 5(g) the percentage availability of the solution space does not overlap precisely for the robot pair over sampling instants, hence the appearance of two distinct plots corresponding to the two robots. In Figure 5(e) the percentage availability of solution space hits zero for one of the robots ahead of the other. However, the system itself enters a cooperative phase only when the individual solution space exhausts for both the robots. The analysis indicates that the need to resort to cooperative phase for conflict resolution would increase when robots approach one another with reduced angles of separation. This is expected since the distance between C11 and C12 (C21 and C22) increases as the angular separation between the robots decreases. With increasing distances the conditions (i) and (ii) for individual resolution of conflicts mentioned in Section 4.1 becomes more difficult to meet. Equivalently, the percentage of individual solution space becomes less for the same reaction time for considering conflicts.

Table I. Robot parameters for which cooperation becomes mandatory for the two-bodied case.

Angular Separation (degrees)	Reaction Time (seconds)	Maximum Acceleration, Deceleration pixels/s^2	Maximum velocity (pixels/s)
90	5	0.15,–0.15	5
45	5	0.45,–0.45	3
15	12	2,–2	1

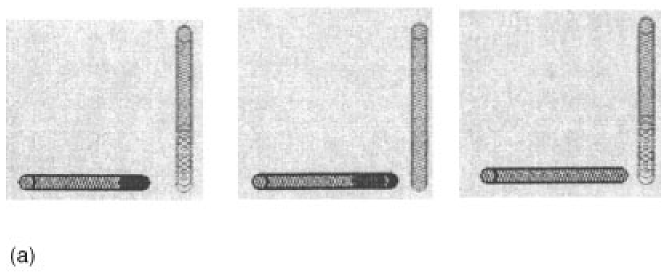
6.2. When Does Cooperation Become Inevitable?

The focus thus far has been on establishing the existence of a cooperative phase during navigation. A question may be asked while the existence of a cooperative phase during navigation is not denied, how essential is the need for it.

6.2.1. Requirement for Cooperation in Two-Bodied System

For the two-bodied system discussed in the last section cooperation could have been avoided if robots took preemptive actions before the onset of the cooperative phase. Table I illustrates under what set of parameters did an invocation of a cooperative scheme for collision avoidance became unavoidable. The table suggests for the case of 90 deg separation in robot heading directions cooperation becomes inevitable only when the robot's reaction time is considerably reduced to 5 s and when it possesses awful dynamic capabilities such as when it cannot accelerate faster or decelerate slower than 0.15 pixels/sample^2. However when the angular separation was 15 deg even default parameters entailed the cooperative phase. *Hence the requirement of a cooperative scheme in real-time navigation is not artificial even for a simple two-bodied system.*

Apart from its inevitability, cooperation also enables robots to avoid conflicts without coming too near to one another. The three snapshots in Figure 6(a) depict the closest distance of approach in a simulation of two robots. The leftmost snapshot is the case where both robots take complementary action with the left robot decelerating and the top accelerating. The next two snapshots are when only one of the robots acts to resolve the conflict. In the second the left decelerates while in the third the top accelerates. The distance between the two robots at the point of closest approach is maximal when robots take complemen-



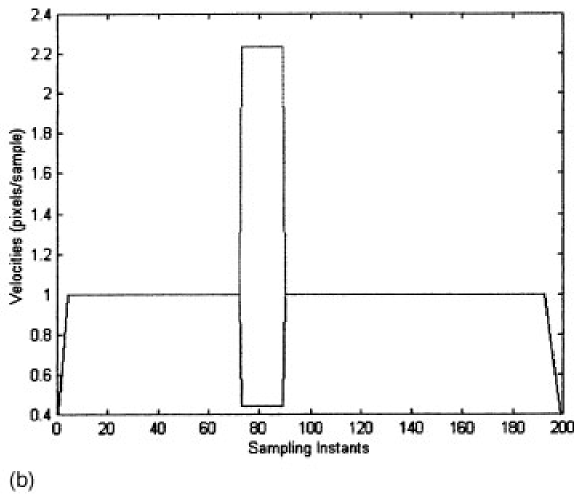


Figure 6. (a) The left-most snapshot depicts conflict resolution when both robots involved take appropriate actions while in the next two only one of the robots changes its aspiring velocity to avoid conflicts. (b) Velocity profiles for the two robots in the left-most snapshot of (a). The complementary nature of actions can be seen as one of the robots accelerates to a higher velocity and the other decelerates to a lower one.

tary actions. Figure 6(b) shows the velocity profile for the two robots for the case when cooperation is resorted through complementary actions. The peak and the trough in the profiles indicate that one accelerates and the other decelerates to resolve the conflict.

6.3. The Multi-Bodied Scenario

For all the simulations portrayed in this section the maximum velocity of either of the robots is 5 pixels/sample and the maximum acceleration for both the robots is 2 pixels/sample^2. The reaction time t_r is fixed at 12 samples. All robots are capable of communicating to one another within a range of 100 pixels.

Figure 7(a) shows an instant during the navigation of a system of five robots where robots 1 and 3 are unable to resolve their conflicts between them individually as well as cooperatively as cooperative solutions lead to indirect conflict with robot 4. Hence 1 and 3 propagate a request to resolve their conflict to 4, thereby embarking on the propagation phase as the last attempt to resolve their conflicts. Robot 4 accepts requests from 1 and 3 and is able to solve the request of 1 by modifying its current velocity such that 1 and 3 are able to avoid their mutual direct conflicts. This scenario is depicted in Figure 7(b) where 4 moves faster in such a way 1 and 3 are able to avoid their mutual direct conflict. Figure 7(c) shows the space-time evolution of trajectories for the robots of Figures 7(a) and 7(b). The x and y axes indicate the regions in the x-y plane occupied by a robot every time it samples the environment. Robot samples of the environment in time are shown along the z axis as sampling instants. The five solid lines of the figure correspond to the trajectories of the five robots. The figure shows that the robot trajectories do not overlap in space-time, confirming that all collision conflicts were resolved by the algorithm.

Figure 8 shows a sequence of snapshots during the navigation of a system of eight robots. The sequence is ordered left to right with the second row sequence following the first row. The traces of the robot are shown by thin lines rather than by the size of the robot. The rightmost snapshot in the first row and the leftmost snap shot in the second row are instances when propagation phase was effected for conflict resolution. The first and the last snapshots represent the initial and final configurations of the robots.

Figure 9 shows yet another sequence of six snapshots of a system of nine robots arranged in the same order as in Figure 8. The first and the last snapshots are the initial and final configurations of the nine robots. The robots are labeled 1 to 9 in the first and last figures. The traces of the robots are not shown for clarity. The initial and final configurations resemble a clocklike structure. In other words a robot placed at position 3 on a clock needs to get to a goal location which is near 9 and a robot placed near 9 initially has its goal configuration near 3. These examples depict simulations with increasing difficulty as the number of robots increases and all of them converge towards a common junction. Hence the trajectory of every robot intersects with every other robot and hence the number of collision conflicts of the total system is

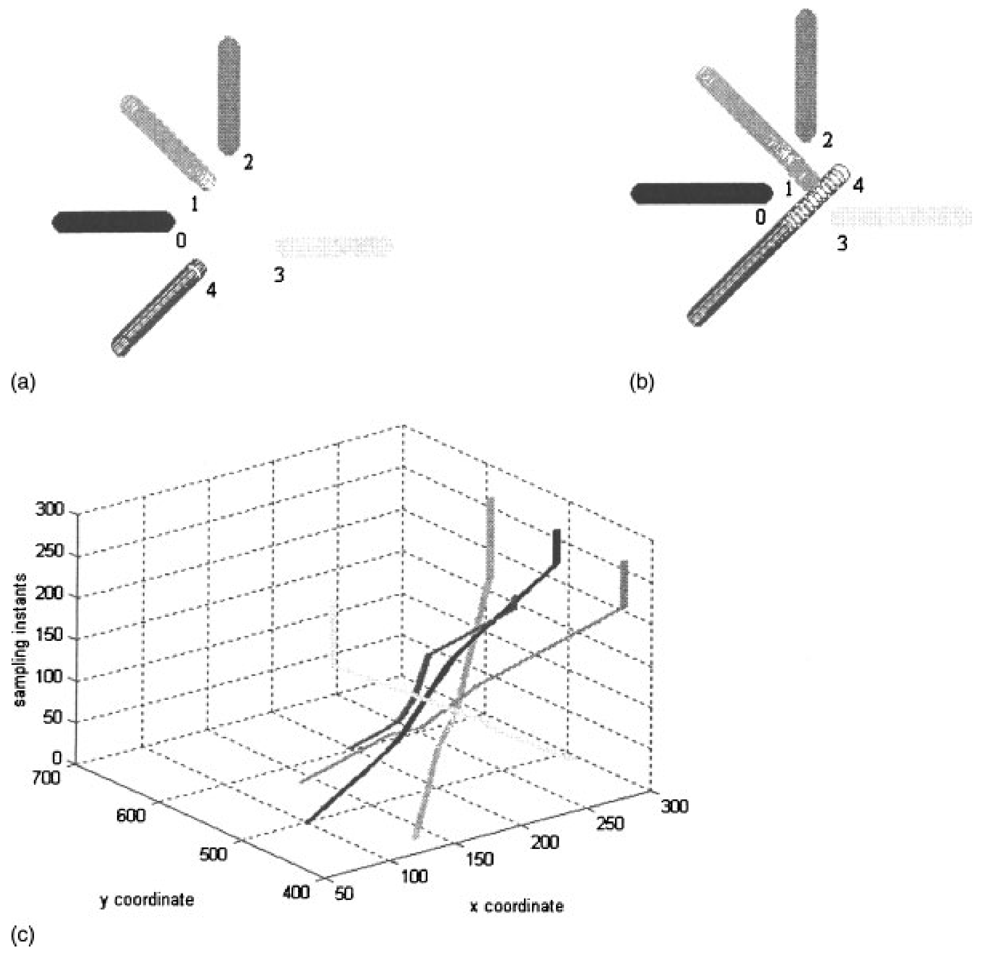


Figure 7. (a) A snapshot of a system of five robots. (b) Robots 1 and 3 propagate requests to resolve their conflicts to robot 4, which accepts the request and moves faster such that 1 and 3 are able to avoid their mutual direct conflict. (c) Space-time evolution of trajectories for the five-robot system.

high. It is also worth emphasizing that robots consider collision conflicts only within a reaction time of 12 samples by which time the robots have converged sufficiently close to one another.

The sequence of snapshots shown in Figure 10 highlight a more difficult example involving 11 robots at similar initial and final configurations as in Figure 9. When the number of robots was increased beyond 11 some of the conflicts could not be resolved and hence collisions were encountered between the robots. The second of these snapshots represents the instant when robots first begin to react to each other's presence by embarking on a strategy for resolving conflicts.

6.3.1. Effects of Scaling on the Requirement to Cooperate and Propagate

In Section 6.2 the inevitability of cooperation with respect to a two-bodied system was discussed. As the number of robots increases and if their navigation course leads to frequent crisscrossing of each other's path, the entailment to cooperate and propagate also increases. For instance, for the five-bodied system shown in the previous section there was one occurrence of propagation whereas in the seven-bodied system there were two such occurrences. Table II depicts the average number of such occurrences to cooperate and propagate over various simulation runs as the number of robots in the system is scaled up. The initial and final configurations in these runs were not similar to those in Figures 9 and 10. One such snapshot involving a simulation of 30 robots is shown in Figure 11. The results suggest that the necessity to cooperate and propagate in a multi-robotic system increases when the system scales up to a large number of robots. It is to be noted the number of attempts to

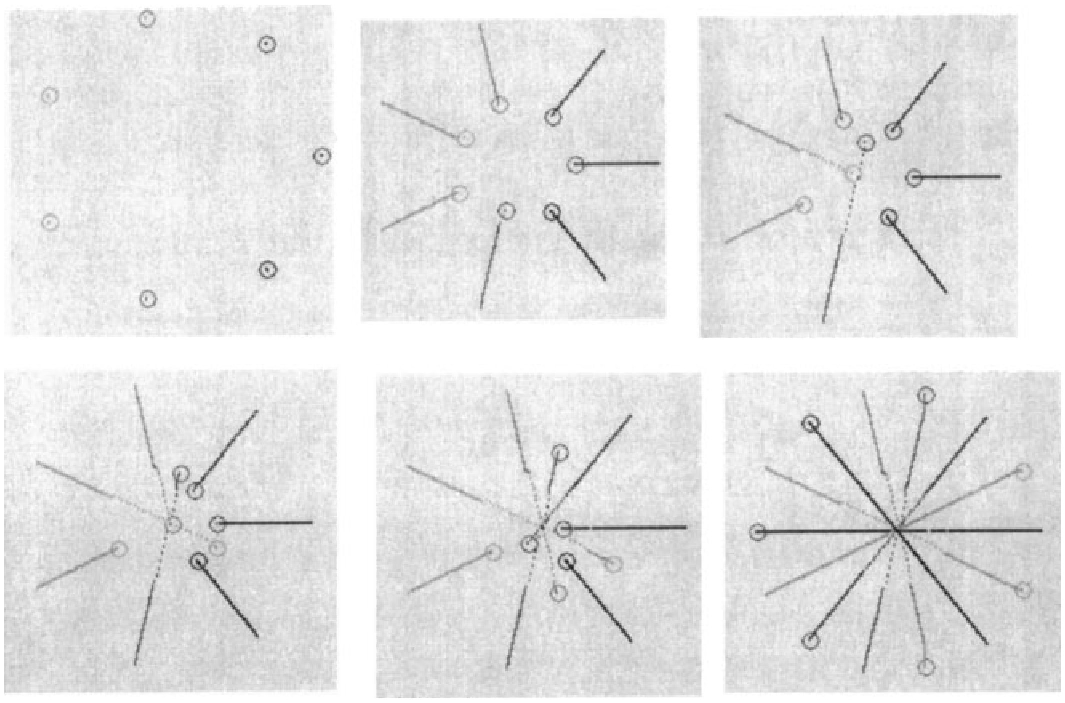


Figure 8. Sequence of snapshots arranged from left to right with the second row following the depicting navigation of a system of eight robots. The third and fourth snapshots depict instances where propagation of conflicts was resorted for conflict resolution.

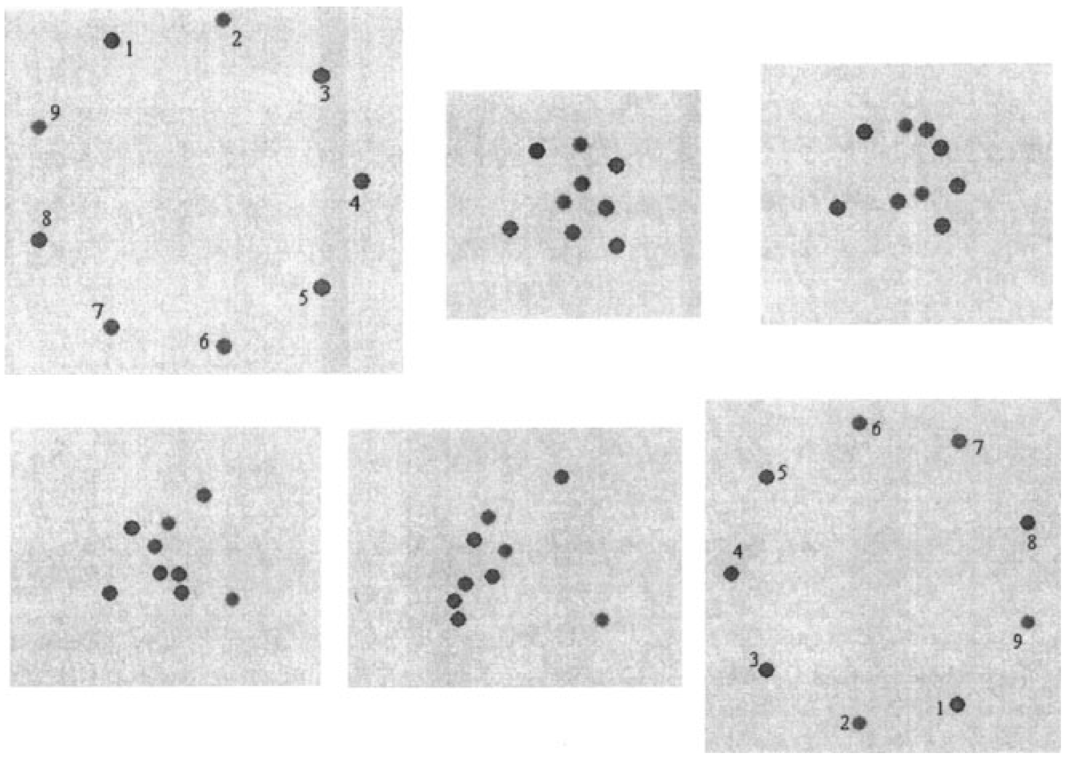


Figure 9. Sequence of snapshots during navigation of a system of nine robots.

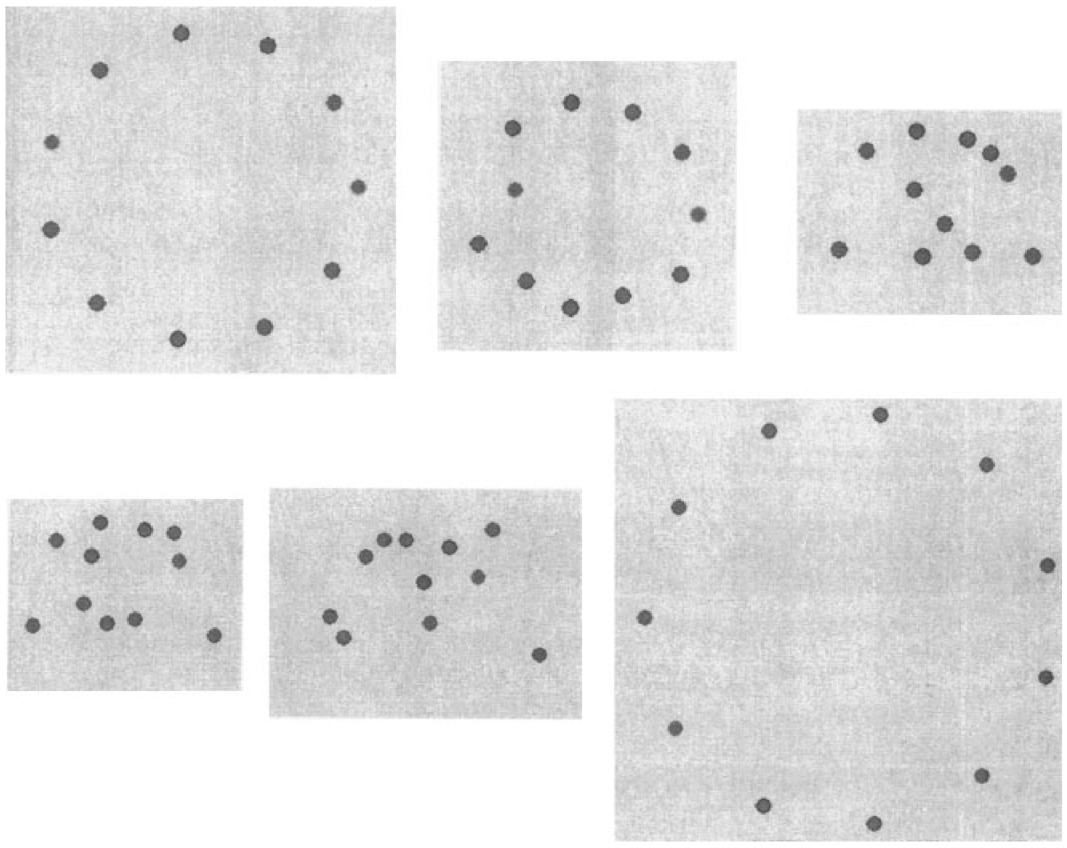


Figure 10. Sequence involving 11 robots. The second of the snapshots indicates the instance when robots begin to react to each other's presence.

cooperate signifies the number of times the robots had to enter the cooperative phase of navigation when individual resolution failed.

7. NAVIGATION EXPERIMENTS

This section shows two navigation experiments using ActiveMedia's Amigobots as the robots to verify the proposed resolution scheme. The Amigobot contains all basic components for robotics sensing and navigation in a real world environment such as battery power, motors and wheels, position encoders, and so-

Table II. The effect of scaling up on the need to cooperate and propagate conflicts.

No. of robots	No. of attempts to cooperate	No. of conflict propagations
10	2	2
15	4	3
20	8	4
30	12	5

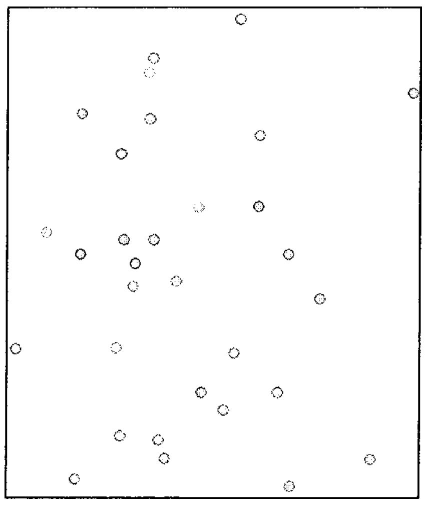


Figure 11. A snapshot during navigation of 30 robots.

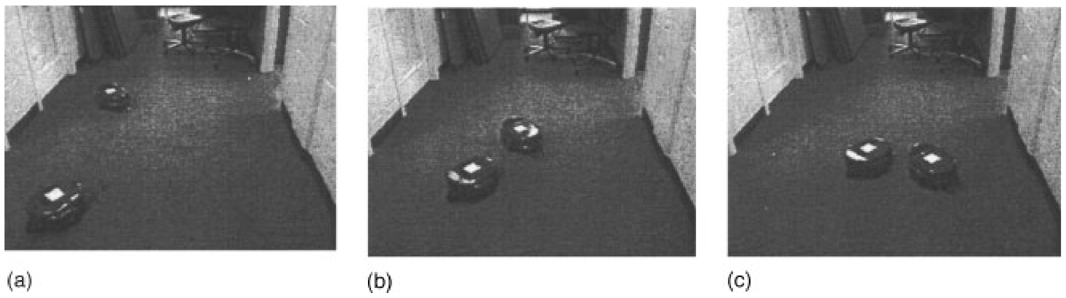


Figure 12. (a)–(c) Collision avoidance sequence between two Amigobots. The robot on the left in the second figure slows down while that on the right hastens to avoid one another.

nar transducers managed via an onboard Hitachi H8 microcontroller. The communication between the robot and the client computer happens through a wireless serial Ethernet. The control algorithm makes use of the classes and methods provided by ARIA (ActivMedia Robotics Interface for Application) Java Wrapper to interface with the methods that provide for low level control of the robot. Each robot to be controlled is invoked as a separate thread within the algorithm. Robots become aware of each other's states through shared objects between the threads.

For our experiments the maximum translational speed was limited at 75 cm/s. Collisions were considered within a reaction time, t_r, of 10 s into the future. The first experiment is portrayed through Figures 12(a)–12(c). Figure 12(a) represents the instance when robots detect the collision within t_r. Figure 12(b) depicts the robot on the left slowing down while the robot on the right is speeding up to clear one another safely. Figure 12(c) is a few instants after the clearance. The second experiment involves four robots [Figures 13(a)–13(c)] requiring to get past a common intersection. Figure 13(a) is the instance when all four robots have detected collision with at least one of the other robots. Figures 13(b) and 13(c) show snapshots during the collision avoidance sequence among the four robots.

8. LIMITATIONS AND SCOPE

The main limitation of this approach is that complete collision-free resolution of conflicts cannot be guaranteed as is the case with typical reactive approaches that do not resort to planning as a strategy for avoiding conflicts. In fact, these limitations are also reported in approaches that take to reactive strategies for resolving conflicts.[10,18,19] For example, simulation results show the occurrence of conflicts even for the case of five robots.[18] In our approach the number of conflicts tends to increase as the number of robots in the system increases and they crowd or converge towards a common junction that needs to be crossed. In such cases the angular separation between the directions of approach of several robots becomes small and hence the possibility to collide increases. The situation is largely alleviated by the following measures:

- Increase the reaction time between robots whose angular separation is very low. By making reaction time as a function of angular separation we have found that the number of conflicts can be essentially reduced.
- For robots whose angular separation is less, an orientation control scheme in combination

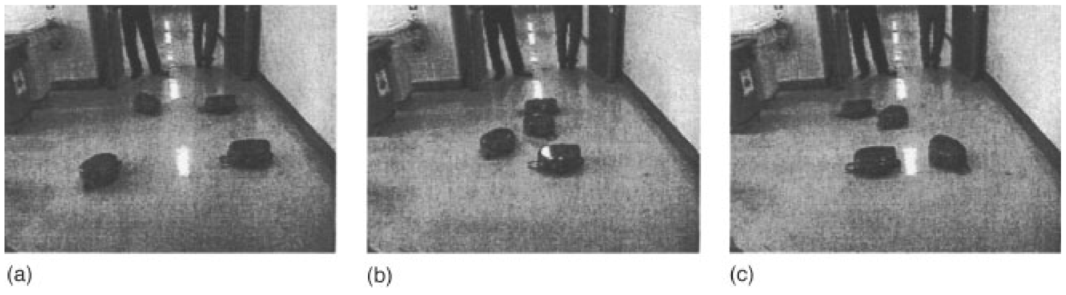


Figure 13. (a)–(c) Collision avoidance sequence between four Amigobots.

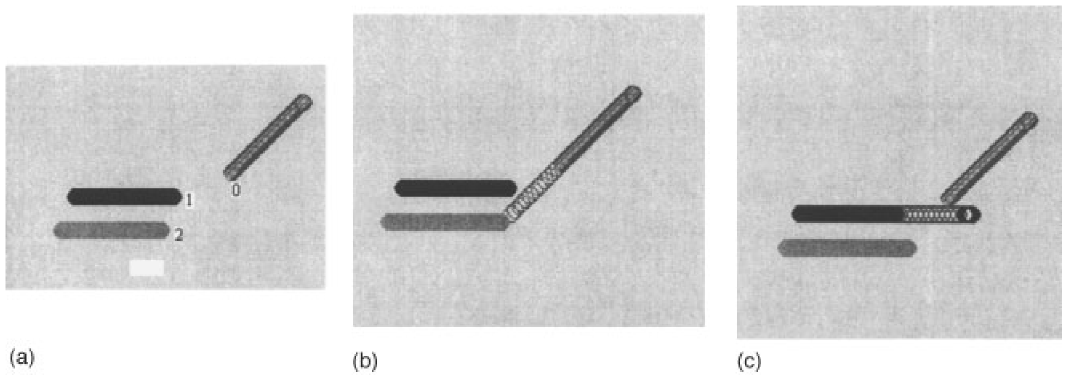


Figure 14. (a) Robots 1 and 0 detect conflict between each other. Robots 0 and 2 are not aware of each other. (b) Robot 0 accelerates and 1 decelerates, which results in collision between 0 and 2. (c) Collision between 0 and 2 averted through an act of benevolence of 1.

with velocity control can be employed for reducing collisions. However, orientation control can also increase instances of collision when the number of robots is large and they tend to crowd or converge towards narrow areas.

- A trivial solution is to bring to a halt all the robots involved in a conflict and through some priority scheme navigate them one after another.

8.1. Social Cues in Multi-robotic Systems

One of the main areas that has been identified for expanding the scope of this effort is the role of social cues and sociality in real-time navigation of several robots. For example, robots that are *benevolent* in transferring critical information to other robots at the apt instance can help avoid hidden collisions and alleviate the performance of the entire system. Benevolence among robots is particularly helpful when the communication range between robots is restricted to very small distances. In the example briefed below, robots communicate to one another only within a range of 50 pixels and the kinematic and dynamic capabilities of the robots are restricted.

Figure 14(a) depicts a situation where the robots numbered 0 and 1 are in collision. In the normal scheme of collision avoidance robot 1 accelerates and robot 2 decelerates, which, however, results in collision between robots 0 and 2 in the future as shown in Figure 14(b). At the time of decision making robots 0 and 2 are out of the field of vision of each other and robot 0 is not aware of the consequence of its decision made with respect to robot 1. However, robot 1 is in the knowledge of both robots 0 and 2 as they are within its range of vision. Robot 1 can anticipate the future collision between robot 0 and 2 and hence modify the decision-making strategy. It could suggest 0 to decelerate while itself accelerates; this would avoid the future collision between 0 and 2. We call this an act of benevolence where 1 comes forward and changes its own preferred mode of action (that of decelerating) to avert collision between 2 and 0. The act of benevolence avoids collision between 0 and 2 in Figure 14(c).

9. CONCLUSIONS

A novel distributed three-tiered approach for coordinated cooperative collision avoidance for a multirobot system from a reactive navigation standpoint has been presented and the simulation results confirm the efficacy of the proposed model. Robots resolve conflicts at three levels, namely, individual, cooperative, and propagation phases. The approach is particularly suitable for a large number of robots moving about in shop floors, factories, airports, and the like where a priori knowledge of the plans of all other robots in the system is not made available for every robot in lieu of computational complexity. Establishing the existence of a cooperative phase in navigation as well as ascertaining the entailment of cooperation in two robotic and multi-robotic systems involving several robots has also been a contribution of this effort. Cooperative phase needs to be invoked when individual resolution of collision conflicts does not yield a control action in the individual solution spaces of the robot. Cooperation can be considered as a search for control actions (here velocities) in the joint space of the system of robots involved in conflicts. The results reported also indicate that the need to cooperate and propagate conflicts increases as the system scales up to a large number of robots. Future areas of work include incorporating a cooperative orientation control scheme and the investigation of various social cues such as benevolence and deception in conflict resolution in a multi-robot system.

REFERENCES

1. J.C. Latombe, Robot motion planning, Kluwer Academic, Dordrecht, 1991.
2. K. Fujimura, Motion planning in dynamic environments, Computer Science Workbench, Springer-Verlag, 1991.
3. V. Genevose, R. Magni, and L. Odetti, Self-organizing behavior and swarm intelligence in a pack of mobile miniature robots in search of pollutants, Proc 1992, IEEE/RSJ Int Conf on Intelligent Robotics and Systems, Raleigh, NC, 1992, pp. 1575–1582.
4. L.E. Parker, ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation, IEEE Trans Rob Autom 14:(2) (1998), 220–240.
5. H. Choset, Coverage for robotics—a survey of recent results, Anna Mathe Artifi Intell 31 (2001), 113–126.
6. R. Alami, S. Fleury, M. Herbb, F. Ingrand, and F. Robert, Multi robot cooperation in the Martha project, IEEE Rob Autom Magazine 5(1), 1998.
7. J. Barraquand, B. Langlois, and J.C. Latombe, Numerical potential techniques for robot path planning, IEEE Trans Syst Man and Cybern 22:(2) (1992), 224–241.
8. M. Erdmann and T. Lozano-Perez, On multiple moving objects, Proc IEEE Int Conf on Robotics and Automation, 1986.
9. S.M. LaValle and S.A. Hutchinson, Optimal motion planning for multiple mobile robots having independent goals, IEEE Trans Rob Autom 14 (1998), 912–925.
10. C.W. Warren, Multiple path coordination using artificial potential fields, Proc IEEE Int Conf on Robotics and Automation, 1990, pp. 500–505.
11. R. Alami, F. Ingrand, and S. Qutub, A scheme for coordinating multi-robot planning activities and plans execution, Proc 13th European Conference on AI, 1998.
12. K. Azarm and G. Schmidt, Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation, Proc IEEE Int Conf on Robotics and Automation, 1997, pp. 3526–3533.
13. S. Carpin and E. Pagello, A distributed algorithm for multi-robot motion planning, Proc 4th European Conf on Advanced Mobile Robotics, 2001.
14. T. Li and H. Chou, Motion planning for a crowd of robots, Proc IEEE Int Conf on Robotics and Automation, 2003.
15. Y. Guo and L. Parker, A distributed and optimal motion planning approach for multiple mobile robots, Proc IEEE Int Conf on Robotics and Automation, 2002, pp. 2612–2619.
16. C.M. Clark, S.M. Rock, and J.C. Latombe, Motion planning for mobile robots using dynamic networks, Proc IEEE Int Conf on Robotics and Automation, 2003.
17. R. Alami, F. Robert, F. Ingrand, and S. Suzuki, Multi-robot cooperation through incremental plan merging, Proc IEEE Int Conf on Robotics and Automation, 1995, pp. 2573–2578.
18. A. Fujimoru, M. Teramoto, P.N. Nikiforuk, and M.M. Gupta, Cooperative collision avoidance between multiple mobile robots, J Robot Syst 17:(7) (2000), 347–363.
19. P. Srivastava, S. Satish, and P. Mitra, A distributed fuzzy logic based *n*-body collision avoidance system, Proc of the 4th Int Symposium on Intelligent Robotic Systems, Bangalore, January 1998, pp. 166–172.
20. K. Madhava Krishna and P.K. Kalra, Detection tracking and avoidance of multiple dynamic Objects, J Intell Rob Syst 33:(3) (2002), 371–408.
21. V.J. Lumelsky and K.R. Harinarayan, Decentralized motion planning for multiple mobile robots: The cocktail party model, Autonom Rob 4 (1998), 121–135.
22. K. Kant and Z.W. Zucker, Towards efficient trajectory planning: The Path Velocity Decomposition, Int J Rob Res 5:(3) (1986), 72–89.
23. K. Madhava Krishna, H. Hexmoor, and P. Subba Rao, Avoiding collision logjams through cooperation and conflict propagation, Proc IEEE KIMAS'03 (Int Conf on Knowledge Integrated Multi Agent Systems), Boston, MA, 1–3 October, 2003.
24. K. Madhava Krishna, R. Alami, and T. Simeon, Moving safely but not slowly: Reactively adapting paths to better time-lengths, Proc IEEE Intl Conf on Advanced Robotics, June 2003.