A negotiation model for cooperation among robots

João Sequeira, M. Isabel Ribeiro
Instituto Superior Técnico / Instituto de Sistemas e Robótica
IST - Torre Norte, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal
{jseq,mir}@isr.ist.utl.pt

Keywords: Cooperative mobile robots, Behavioral systems.

Abstract

This paper describes an architecture for the control of teams of robots. The architecture is developed from basis notions on the motion of single robots and includes the definition of a state space and basis operators that enable state transitions. Teams of robots can be also controlled by an architecture similar to the one used for single robots, with the adequate extension of the state. A negotiation procedure among the teammates is developed from the basis properties of the team state. The paper presents simulation results for a team of 3 unicycle robots flocking along a sequence of locations in the workspace.

1 Introduction - Robot control architectures

Since its appearance in the mid 80's, behavior-based control has been established as a relevant approach to purposive (*i.e.*, task achieving) robot control. With the recent expansion of robotics to the multiple agent systems a whole new set of applications emerges. The relevant ones include deep land mining, land mine sweeping, pesticide spraying in agriculture (using for instance free flying robots), surveillance of large areas, handling of large/heavy parts in a warehouse, coordinated removal of debris in catastrophe scenarios and small flying robots in space applications. This paper considers the behavior-based control of a team of unicycle robots, operating in a plane, and executing a set of tasks requiring flocking behavior.

The execution of most of the robot tasks requires the tracking of a reference path. Therefore, for a task to be properly executed by a robot, it is necessary that the task specifications be mapped into a feasible path along which the robot will travel. The mapping $task\ specs \mapsto path$ represents the functional control architecture of the robot. Functional hierarchies, [Saridis, 1996], and subsumption architectures, [Brooks, 1986], are two major examples of such mapping. The architecture considered in this paper differs from the previous ones in the support framework.

Path generation under position and/or velocity constraints (*i.e.*, constraints on the reachability of a point in the configuration space[1]) is closely related with the controllability problem which is known to be in the class of undecidable problems, [Laumond, 1993]. In addition, not every kinematically feasible path may be adequate to the execution of the task. Complex paths, with large number of maneuvers, often require small tracking errors and hence accurate sensors and control strategies. The complexity of the path planning problem has been part of the motivation for the study of reactive architectures (subsumption flavoured), in particular under dynamic environments where the path update cycle tends to require a fast dynamics from the architecture. However, in a wide class of applications (such as the aforementioneds one involving robot teams) the path tracking requisite can be relaxed down to constrain the paths of the robots to stay in a specified region of the workspace, rather than following a precise path.

The proposed behavioral approach to the control architecture design is characterized by the possibility to specify (directly or indirectly) a pattern for the trajectories to be followed by the robot or by each robot of a team of robots, [Sequeira, 1999]. This amounts to say that the paths leading to the execution of a task by each robot, rather than being precisely defined, are those contained in a connected region of the C-space. In addition, with the inclusion of a process of negotiation, among the team members, of state changes, the dynamics of the proposed architecture is extended to a team of robots. This extension, which is the novelty of this paper, has the

[1] C-space for short.

appealing feature that group behaviors and individual behaviors can be *a priori* specified in terms of team states and single robot states, based on the type of mission the team should perform.

The paper is organized as follows. Section 2 describes the control architecture for a single robot. The extension of the architecture to a team of robots is presented in Section 3. Section 4 presents an application example on flocking behavior and Section 5 points out the most relevant aspects of this work and discusses further developments.

2 A control architecture

The behavioral approach followed in this paper is supported on the formulation developed in [Sequeira, 1999]. This approach separates the problems of single robot and multi-robot control, and hence of cooperative task execution. This section addresses the problem of a single robot control.

In normal operation, a robot may follow several different paths without compromising the success of the task execution and hence any two of such paths are equivalent. This equivalence relation between paths requires that the whole C-space be considered. For example, any two paths reaching the same final configuration but visiting regions of the C-space too far apart are, in this sense, equivalent. Often, it is possible to consider aggregates of paths spanning only limited regions of the C-space. Using the previous reasoning any two aggregates spanning the same region of the C-space are equivalent. In a sense, equivalent path aggregates represent a behavior.

The behavioral control problem presented in this paper, rather than acting on a single path, takes place on a set of equivalence classes each of which is defined on a set of aggregates of paths. Each of these equivalence classes spans a region in the C-space containing a set of paths each producing similar effects in terms of the robot motion. This suggests the following definition for robot state.

Definition 1 (Robot state) *Let Q be the robot's C-space, and $a : Q \rightsquigarrow Q$ a parameterized set-valued map, hereafter called* action*, that generates paths for the robot to follow. A state is defined by the pair*

$$a(q) \equiv (q, a).$$

When the robot enters in a neighborhood of q, named the action initial configuration, the action a generates a path entirely contained inside a region $B_a(q) \subset Q$ (see Figure 1). The set $B_a(q)$ is named the state bounding region and defines an equivalence class in a space of aggregates of paths.

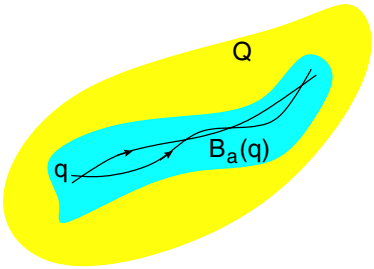

Figure 1: State representation.

Robotic tasks are often defined in terms of the robot motion in its C-space, *e.g.*, move to a specific configuration. Even functionally defined tasks can be translated into a set of configurations to be reached by the robot. For example, in tasks such as "clean a glass window" and "pick up the soda can on top of the table", sensing devices must detect the boundary of the window and the location of the table and of the soda can which are used to define points in the C-space that the robot must reach. Using Definition 1, the execution of a task can also be defined in terms of a state to be reached by the robot, *e.g.*, in the task "move in straight line from a given configuration" the action corresponds to the straight line motion whereas the configuration where the motion starts corresponds to the action initial configuration.

The control in the state space of Definition 1 is possible with two basis operators (see [Sequeira, 1999] for details).

1. **Composition of two states**, represented by the symbol $\circ$. This binary operator performs the transition between states. Given any two states, whose associated bounding regions overlap by a minimum amount of space, this operator returns a new state whose bounding region is the union of the bounding regions of the two argument states (see Figure 2). Furthermore, the operator provides the link path between the paths generated by the two states.

2. **Expansion of a state**, represented by the symbol $\boxtimes$. Additional new states can be created through the expansion of the aforementioned equivalence classes to accomodate other relevant paths not yet accounted for

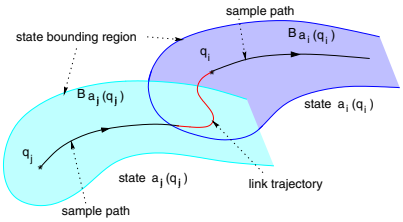

Figure 2: Composition of two states, $a_i(q_i) \circ a_j(q_j)$.

by the current state. This is a binary operator that expands the bounding region of the second state by an amount defined by the bounding region of the first state (see Figure 3).

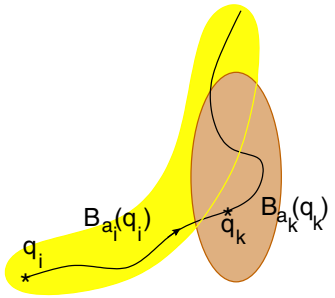

Figure 3: Expansion of a state, $a_k(q_k) \boxtimes a_i(q_i)$.

The decision on state changes is taken by a supervisor controller. This controller chooses, at given time instants, among a finite set of *a priori* defined actions, $a_1, \ldots, a_n$, which one is to be executed. The composition and expansion operators are used to get the next state. Assume that the robot is currently in state $a_i(q_i)$, that state $a_j(q_j)$ was chosen by the supervisor as the next state to be merged with the current one and that the resulting state is to be expanded by $a_k(q_k)$. The resulting dynamics of the behavioral architecture is of the form:

$$a(q_j) = (a_i(q_i) \circ a_j(q_j)) \boxtimes a_k(q_k) \qquad (1)$$

The choice of $a_i(q_i)$ and of $a_k(q_k)$ depends on the task and on the environment where the robot is operating. The state resulting from equation (1), $a(q_j)$, has two dual interpretations regarding the space occupied by the regions bounding each state.

Global state - Every path may be considered as belonging to a single state, even if it results from a composition and/or expansion, as indicated by (1). In this sense, the bounding region of a single state may occupy an amount of space containing every possible path leading to the execution of a task.

Local state - Each bounding region occupies a limited amount of space. In this sense, the execution of a task will, in general, require the composition of a sequence of states in order that the paths leading to the task execution be contained in the bounding region defined by the corresponding global state.

By the local interpretation, each state can be regarded as the composition of a sequence of states each of which has a "small" bounding region and hence every bounding region has a covering[2] $B_a(q) = \cup_i C_{a_i}(q_i)$.

The action synthesis problem, and hence the computation of a state bounding region, depends on the specific task being executed. However, in reduced spatial horizons, the path planning/generation tends to be less constrained than in large horizons, e.g., often the obstacles far from the current robot position are not relevant for the current motion. From an implementation perspective, the coverings for the bounding regions support iterative approaches to global states building.

If the state on the left side of (1) is interpreted as a local state, then the execution of a task requires that the supervisor controller selects an adequate sequence of actions a_l and the corresponding initial configurations q_l. The region bounding the resulting global state is thus covered by the bounding regions of the local states $a_l(q_l)$ in the sequence, *i.e.*, $B_a(q) = \cup_l B_{a_l}(q_l)$.

For the sake of simplicity, the action a_l can be thought as being driven by a vector field whose role is to express a preferential motion direction inside the bounding region $B_{a_l}(q_l)$. Again, for the sake of simplicity, $B_{a_l}(q_l)$ is defined as a fixed radius ball in Q. The generation of a path from a vector field, $v_l(q)$, must be monitored to ensure that Definition 1 is verified, i.e., that the generated path lies entirely inside $B_{a_l}(q_l)$. This corresponds to solving the control problem defined as: given the robot at configuratinon q, $v_l(q_l)$, and the corresponding $B_{a_l}(q_l)$, compute a motion direction $v_l(q)$ such that any path approaches the path defined by $v_l(q_l)$ (the preferential motion direction). This control problem can be expressed

[2] This is a practical interpretation of the fact that Q is a compact set.

by

$$T_{K_l}(q) = \{v_l(q) \in Q : \lim_{h\to 0} \inf \frac{d_{K_l}(q+hv_l(q))}{h} = 0\}$$
$$d_{K_l}(q) = \inf_{b\in K_l} \|q - b\| \qquad (2)$$
$$K_l \subset B_{a_l}(q_l)$$

where K_l is a subset of the bounding region containing q_l and a "preferential path" generated by $v_l(q_l)$ (T_{K_l} is called the contigent cone to K_l, [Aubin, 1991]).

The set K_l, if properly chosen, is a viability domain for the robot, meaning that $\dot{q} \subset T_{K_l}(q)$. From Nagumo's theorem, [Aubin, 1991], if K_l is a *viability domain* for the robot then its trajectories lie in K_l and hence in $B_{a_l}(q_l)$.

The set K_l is chosen in the interior of the region bounding the local state, $B_{a_l}(q_l)$. If robot is inside K_l the contingent cone verifies $T_{K_l}(q) =$ any vector$v_l(q) = Q$, meaning that any motion direction v_l is possible. Whenever the robot moves outside K_l, expression (2) means that there is a sequence of $h_n \to 0+$ and motion directions $v_{l_n}(q) \to v_l(q)$ such that the path generated by the action approaches K_l (see Figure 4).

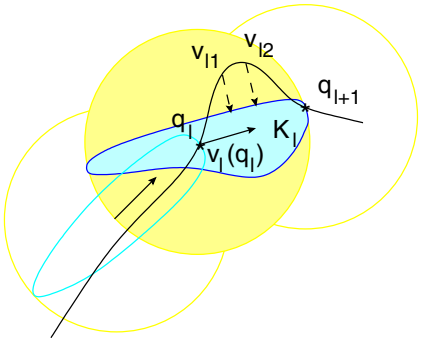

Figure 4: Viability-based action fundamentals.

3 Cooperation through negotiation

In general, the cooperative execution of a task by a team of robots requires each single robot to control its motion and to send/receive state information to/from the rest of the team[3]. The information exchanged between robots is of two kinds: (i) state information, eventually evolving according (1), and (ii) event/symbolic information, signaling special conditions, e.g., the perception of a wall or a running error.

[3] Extreme cases where each individual robot executes its own task not exchanging information with the others are not considered. Nonetheless, the group behavior of such examples may appear as exhibiting cooperation.

The behavior of a team can be expressed using a group dynamics similar to the single robot dynamics defined by (1). The main idea is to retract the set of paths produced by the robots in the team to a single path, named the team-path and representing, for example, the center of mass of the team, e.g., the path of the center of mass of the team. This path can be thought as resulting from a team action $\mathcal{A}$ applied at initial team configuration $\mathcal{Q}$ (hence belonging to the team state $\mathcal{A}(\mathcal{Q})$). As for the team state bounding region, the trajectories of the individual robots span the region bounding the team paths, i.e., the paths generated by the team state. Therefore, the region bounding any team state contains the regions bounding the states of the individual robots.

Similarly to the single robot case, the duality between local and global team states requires that the global ones be achieved by the composition of a sequence of local team states. Therefore, the local team state defines a "small" region, $\mathcal{B}_{\mathcal{A}_l}(\mathcal{Q}_l)$, where each path of the individual robots is contained.

For the sake of simplicity, each of the $\mathcal{B}_{\mathcal{A}_l}(\mathcal{Q}_l)$ region, is defined as a ball in Q, centered at $\mathcal{Q}$ and with a fixed radius, such that, given the bounding region $B_{a_l^m}(q_l^m)$ of the m-th robot in the team,

$$\cup_{m=1}^{n} B_{a_l^m}(q_l^m) \subseteq \mathcal{B}_{\mathcal{A}_l}(\mathcal{Q}_l) \qquad (3)$$

At each local team state, $\mathcal{A}_l(\mathcal{Q}_l)$, the m-th robot has its own motion, generated from state $a_l^m(q_l^m)$. The corresponding path can not go outside $B_{a_l^m}(q_l^m)$ without permission from the team as this may lead to a violation of (3). This means that any change of state is negotiated among the team members. Figure 5 illustrates the negotiation principles considered in this paper. The regions bounding the local states are divided in two regions. Inside the inner negotiation boundary (NB) the robots are considered to be close to each other so they can move without querying the teammates. Between the inner and the outer NBs changes of state require permission from the teammates. Outside the outer NB it is assumed that no communication is possible and hence the robots operate isolated from the rest of team. This may lead to the formation of subteams, or coalitions. In Figure 5 robots R1 and R2 form a subteam while robots R2 and R3 form another subteam.

The negotiation process requires that each single robot had the knowledge on the team state and on its own state. This knowledge enables each supervisor to compute the motion direction of its

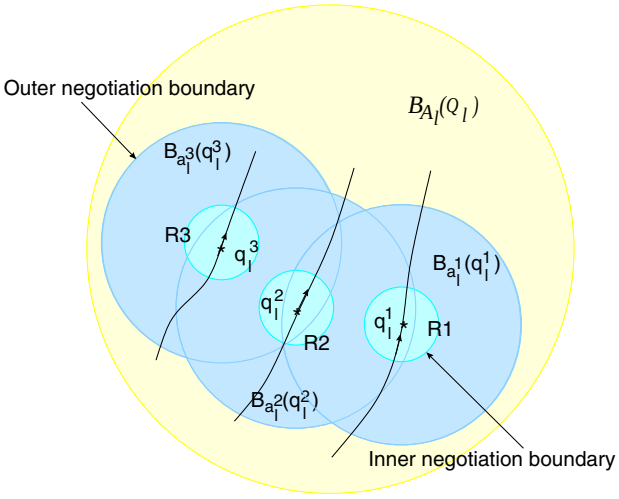

Figure 5: The team state and negotiation.

own subteam and then the appropriate motion direction for its on robot.

4 A case study - Flocking robots

In this section the control architecture described in Section 2 is applied to a team of robots following the principles outlined in Section 3. A simulation of the specific case of a flock of robots is presented. The team is composed of identical robots of unicycle type. For the sake of simplicity ideal sensors are assumed. Each robot is equipped with *stop*, *go to task* and *avoid obstacles* actions. Team actions are *stop* and *go to task*. Both the linear and angular velocities of each robot are assumed limited. No environment disturbances are assumed and hence each robot has uniquely to avoid collisions with its teammates.

In the flocking behavior example considered in this paper, both single robot state bounding regions and the team state bounding region are set to discs (in the xy workspace).

The mission is to move the team through the tasks, maintaining a non rigid formation. Each task is defined as "reach the specified location and stop there for a while" and the task locations follow a rectangular pattern. A task is finished as soon as any of the robots reaches its location and the mission requires the execution in a clockwise sequence.

Figures 6 to 8 illustrate several aspects of the operation of the architecture. The locations of the tasks are marked with the symbol $\square$ in all the three figures. The robots start at configurations $(x, y, \theta) = \{(0,0,0);(2,0,0);(0,2,0)\}$.

Figure 6 shows the trajectories of the robots. Each robot position is represented by a symbol $\circ$ with its orientation represented by the dash $-$. The intense maneuvering shown, namely near Task 1, in the middle between Tasks 1 and 2, between Tasks 2 and 3 and near Task 4, is mainly a consequence of the collision avoidance actions. Nevertheless, the team is able to execute the mission. Between Task 2 and Task 3, the collision avoidance situations drive two of the robots too far away from the rest of the team. The negotiations at those points avoid the breakdown of the team.

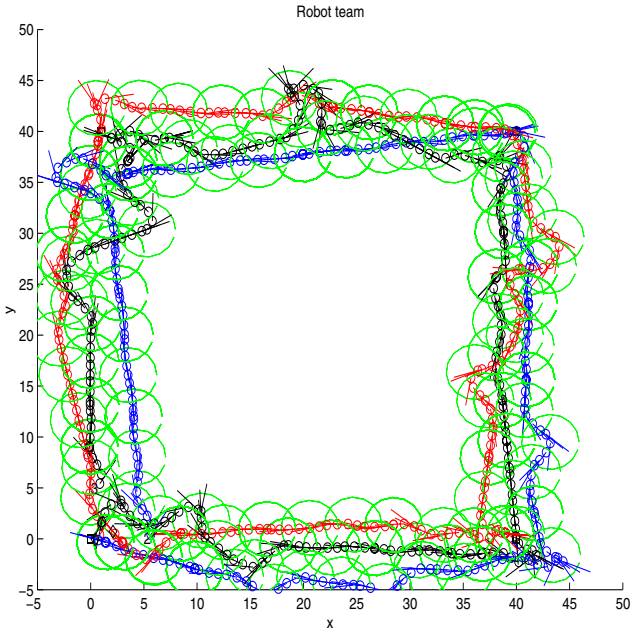

Figure 6: Individual trajectories

In Figures 6 and 7, the wide circles represent inner negotiation boundaries. For each of these inner NBs at least one negotiation occured (if a robot's request is denied then it stops and continues to issue the same request until it accepted by its teammates). Only negotiations to avoid an intentional team breakdown are shown. For example, when a robot is too far to interfere with the rest of the team, its supervisor is able to drive it towards the current task without any explicit negotiation. This means that the team state bounding regions were chosen wide enough so that whenever a robot lies out of contact with its teammates it is assumed that its supervisor is able to drive it towards the current task. The symbols $\triangle$ indicate intermediate goal locations computed by each robot's supervisor from the knowledge of the team state and its own state. These intermediate goal locations are used by the robots to compute their linear and angular velocities.

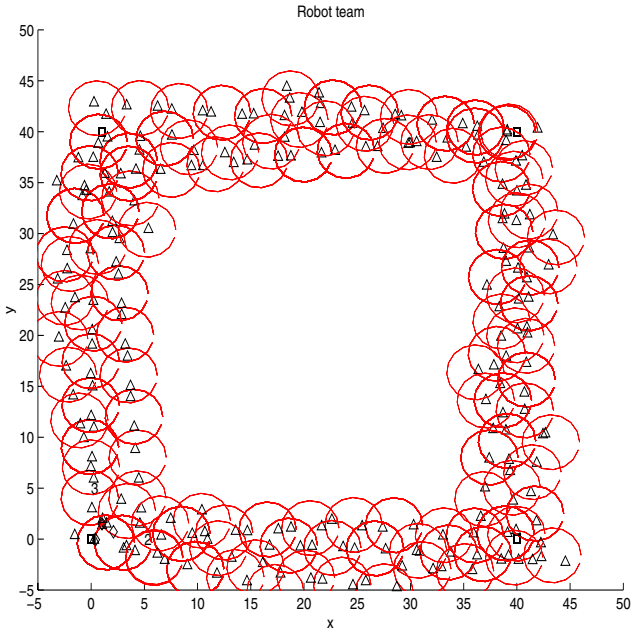

Figure 7: Inner boundaries for the explicit negotiations

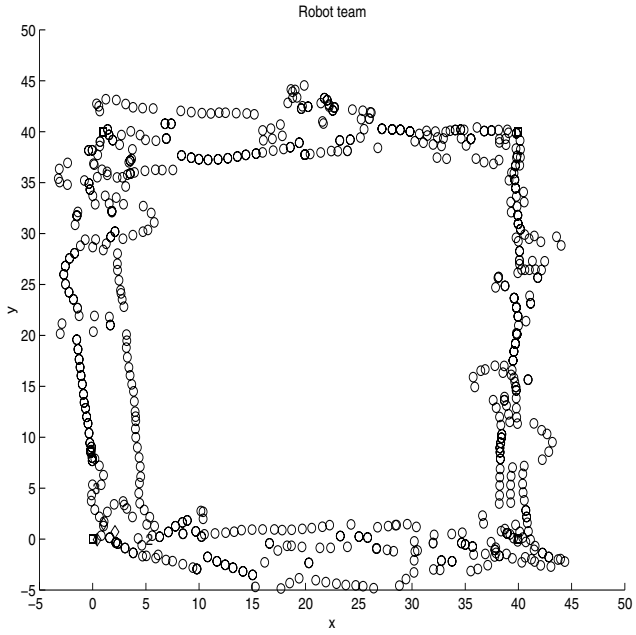

Figure 8: Center of mass of the subteams

Figure 7 illustrates the trajectory of the center of mass of the subteams. In the regions between the starting region and Task 1, between Task 1 and Task 2 and between Task 3 and Task 4, the formation of two subteams is clearly visible. Each of these subteams is composed of two robots, with one of them belonging, simultaneously, to the two of them. Between Task 2 and Task 3 a region with only one subteam is also visible. A correspondence with Figure 6 shows that the three robots were close to each other.

5 Conclusions

A complete architecture for the control of robot teams was presented. The approach considered separates the control architecture in two components:

1. a framework to model the control architecture of single robots;
2. a negotiation model to account for the interactions among the robots supported on the extension of the concepts developed for the single robot architecture.

The simulation results presented illustrate the main concepts of team cooperation introduced by the architecture.

Further work includes the analysis of team controllability under the framework described and its implications on the negotiation process.

Acknowledgements

This work was sponsored by the Portuguese research programme PRAXIS XXI, under the project COOPERA - 2/2.1/TPAR/2087/95.

References

[Aubin, 1991] Aubin, J. (1991). *Viability Theory*. Birkhäuser.

[Brooks, 1986] Brooks, R. (1986). A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, RA-2(1).

[Laumond, 1993] Laumond, J. P. (1993). Singularities and Topological Aspects in Nonholonomic Motion Planning. In Z. Li and J. Canny, editors, *Non-Holonomic Motion Planning*, pages 149–199. Kluwer Academic.

[Saridis, 1996] Saridis, G. N. (1996). On the Theory of Intelligent Machines: A Comprehensive Analysis. *International Journal of Intelligent Control and Systems*, 1(1):3–14.

[Sequeira, 1999] Sequeira, J. (1999). *Cooperation Among Robots: A Behavioural Approach Supported On Group Theory*. PhD thesis, Instituto Superior Técnico, Portugal. Departamento de Engenharia Electrotécnica e de Computadores.