Available online at www.sciencedirect.com
ScienceDirect
Robotics and Autonomous Systems 56 (2008) 373–384
Robotics and Autonomous Systems
www.elsevier.com/locate/robot

Closed loop motion planning of cooperating mobile robots using graph connectivity

Guilherme Augusto Silva Pereira[a,*], Vijay Kumar[b], Mario Fernando Montenegro Campos[c]

[a] *Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais, Av. Antônio Carlos 6627, Belo Horizonte, MG 31270-010, Brazil*
[b] *GRASP Laboratory, University of Pennsylvania, 3330 Walnut Street, Philadelphia, PA 19104-6228, USA*
[c] *Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Av. Antônio Carlos 6627, Belo Horizonte, MG 31270-010, Brazil*

Received 22 August 2006; received in revised form 8 August 2007; accepted 8 August 2007
Available online 24 August 2007

Abstract

In this paper we address the problem of planning the motion of a team of cooperating mobile robots subject to constraints on relative configuration imposed by the nature of the task they are executing. We model constraints between robots using a graph where each edge is associated with the interaction between two robots and describes a constraint on relative configurations. We develop a decentralized motion control system that leads each robot to their individual goals while maintaining the constraints specified on the graph. We present experimental results with groups of holonomic and non-holonomic mobile robots.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Cooperative robots; Decentralized control; Mobile robot motion planning; Hamiltonian Graph

Cooperating mobile robots must be able to interact with each other using either explicit or implicit communication and frequently both. Explicit communication corresponds to a deliberate exchange of messages that is in general made through a wireless network. On the other hand implicit communication is derived through sensor observations that enable each robot to estimate the states and trajectories of its teammates. For example, each robot can observe relative state (position and orientation) of its neighbors (implicit communication), and through explicit communication exchange this information with the whole team in order to construct a complete configuration of the team.

A fundamental limitation related to these forms of interactions among the robots is the limited field of view of the physical sensors and the limited range of transmitters and receivers. When communication is essential for the completion of the specified task, the robots must move in order to maintain such constraints. In this paper, we address the problem of controlling the motion of a team of mobile robots subject to such constraints that we will call *formation constraints*. Formation constraints also arise in other situations. For example, in an object manipulation task, the robots must cooperate in order to keep the object contained in a subset of the configuration space. This requirement on cooperation can be translated into formation constraints that are functions of the object's position and orientation [1]. Additionally, robots must always avoid collisions with each other, which is also a motion constraint. Finally, mapping and tracking tasks may require constraints on relative positions and orientations to guarantee observability [2].

In this paper we are interested in the problem of controlling a team of robots performing a task that requires cooperation. More specifically, we are interested in tasks in which cooperation can be specified by constraints on relative configurations between pairs of robots in the team. We will model the team of robots as a set of independent agents that are decoupled except for these formation constraints. Each robot may be assigned its own motion plan toward its goal position, but it must conform to these constraints. The main goal of this paper is to develop a simple strategy for modifying the individual robot motion plans in runtime to maintain

* Corresponding author. Fax: +55 31 3499 4810.
E-mail addresses: gpereira@cpdee.ufmg.br (G.A.S. Pereira), kumar@grasp.cis.upenn.edu (V. Kumar), mario@dcc.ufmg.br (M.F.M. Campos).

0921-8890/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2007.08.003

the formation constraints. Thus, we require each robot to be able to reach its destination while satisfying the formation constraints. While our main focus in this paper is on sensing and communication constraints, the basic approach is applicable to other kinds of formation constraints, including those that arise in cooperative manipulation and mapping tasks. We will show the results with two different teams of robots that show the robots moving towards a goal under such constraints.

1. Previous work

The multi-robot motion planning problem has been addressed with centralized motion planners by a number of groups. The paths are constructed in the composite free configuration space $\mathcal{C}_{\text{free}} = \mathcal{C}^1_{\text{free}} \times \mathcal{C}^2_{\text{free}} \times \cdots \times \mathcal{C}^n_{\text{free}}$ [3]. This approach in general guarantees completeness but its complexity is exponential in the dimension of the composite configuration space [4]. Other groups have pursued decentralized approaches to motion planning. This generally involves two steps: (i) individual paths are planned independently for each robot; and (ii) the paths are merged or combined in such a way that collisions are avoided. Some authors call these approaches *coordinated path planning* [5–7].

Completely decentralized, behavior-based control algorithms have been analyzed by Arkin, Balch and their coworkers [8,9]. In this approach, several desired behaviors are prescribed for each agent and the final control is derived by composing these behaviors. For example, the behaviors can be summed using weights to emphasize the relative importance of each behavior. Typically, each agent has a main behavior that guides it to the goal and secondary behaviors that are used in order to avoid obstacles and other robots in the team. These behaviors are generally based on artificial potential fields [10].

Our approach uses potential field controllers based on navigation functions [11]. In a single robot navigating an obstacle field, the navigation function provides a Lyapunov function that guarantees the robot's convergence to the goal. The navigation function can be modified to accommodate unmodeled obstacles or dynamic constraints [12]. We will use a similar approach, but in a multi-robot setting, to solve the motion planning problem with formation constraints.

For multi-robot systems, potential fields were used to deploy robots in known environments [13]. The idea of having artificial potential fields in order to have each robot repelling the others was also used in [14]. These approaches are not directly applicable to our problem because they cannot be easily applied to maintain formation constraints. Further, they do not use the Lyapunov function properties of the potential functions in any meaningful way. In this paper, we present proofs for our methodology which show that besides going to their independent goals the robots also satisfy the formation constraints in the problem.

There has been a great deal of recent interest in the problem of formation control [15,16]. In problems where a rigid geometric arrangement of robots is desirable, it is appropriate to model the set of robots in formation as a rigid structure, sometimes called a *virtual structure* [17,18]. A desired motion is then assigned to the virtual structure, which is decomposed into trajectories that each member of the formation will follow. Virtual structures are, in general, centralized and deliberative methods but some approaches use reactive *leader-following* in order to maintain formation. In such a framework, each robot has a designated leader, which may be other robots in the group or a virtual robot that represents a pre-computed trajectory supplied by a high level planner. Thus, each robot is a follower that tries to maintain a specified relative configuration (a fixed separation and bearing for example) with respect to its leader(s) [15,19,20]. In contrast, in this paper we are interested in formation constraints. Instead of controlling the formation to maintain a rigid shape, we will want the shape to stay within bounds specified by a constraint graph that describes the cooperative task. But, it is worth pointing out that the approach proposed here can be used to drive a set of robots to a desired shape while maintaining formation constraints [21].

2. Problem definition

The basic multi-robot motion problem is to find a motion plan for all the robots in a group such that each robot reaches its goal while avoiding collisions with other robots and with the obstacles in the environment. We will extend this problem by defining the *coordinated motion planning problem*, where besides avoiding collisions the robots need to cooperate and maintain formation constraints in order to reach their goals.

Definition 1 (*Coordinated Motion Planning Problem*). Consider a world, $\mathcal{W}$, occupied by a set, $\mathcal{R}$, of n robots. The ith robot R_i can be represented by a configuration q_i in the configuration space $\mathcal{C}$. Let $\mathcal{F}_i \subseteq \mathcal{C}$ denote the free configuration space for R_i. Additionally, let $\mathcal{C}_{R_i}(\mathcal{R} \setminus R_i, t) \subseteq \mathcal{F}_i$ denote R_i's valid configuration space imposed by its *formation constraints*. The coordinated motion planning problem is to steer each robot, R_i, $1 \leq i \leq n$, from an initial configuration q_i^{init} at time $t = t_0$ to the goal configuration $q_i^{\text{goal}} \in \mathcal{F}_i$ at some time $t = t_f > t_0$ such that $q_i \in \mathcal{C}_{R_i}(\mathcal{R} \setminus R_i, t)\ \forall t \in (t_0, t_f]$.

Formation constraints are constraints on individual robots induced by the other robots in the team. Thus, $\mathcal{C}_{R_i}(\mathcal{R} \setminus R_i, t)$ depends on the robots' characteristics, their configurations, and also on the nature of the task. Notice that our problem statement differs from the previous definition of multi-robot motion planning problem in the sense that, besides inter-robot collisions, we are adding other kinds of constraints that include, for example, sensor field-of-view constraints and communication range constraints. On the other hand, our definition of multi-robot motion planning is not too different from the definition of robot motion planning for single robots [22]. While the traditional definition considers the problem of moving the robot in a limited free space, where the constraints are induced by the (non-controlled) obstacles in the environment, here part of the valid (free) configuration space for the robots is controlled by the position of the other robots. However, instead of developing a single algorithm for coordinating the motion of all the robots, we develop a decentralized algorithm which allows each robot to choose its

motion based on the available free space and the formation constraints.

3. Approach

In this paper we will solve the cooperative motion planning problem defined in Section 2 for a planar world $\mathcal{W} = \mathbb{R}^2$ and will focus our examples and experiments on sensing and/or communication constraints. However, it can be shown that the solution can be extended for other constraints induced by cooperation.

We consider a two-level motion planner where the superior level is able to specify a deliberative plan [8] in terms of previously computed navigation functions for each robot and desired *neighborhood relationships*. The navigation functions are discussed in Section 3.3. The neighborhood relationships are pairwise formation constraints that are formalized in Section 3.2. The second level of the planner is the level we are concerned with. We address the real-time modification of the pre-planned functions to accommodate the formation constraints. Before we proceed further, we will make some assumptions.

Assumption 1. All robots are identical in terms of geometry, and in terms of capabilities and constraints related to sensing, communication, control, and mobility, and all constraints are symmetric.

Assumption 2. The robots are point robots: $q_i = (x_i, y_i)$.

Assumption 3. The robots are holonomic. For the ith robot, the dynamical model is then given by: $\dot{q}_i = u_i$.

We will need these three assumptions to derive the basic results. But, as discussed in Section 3.6, we relax all three assumptions in our experiments (Section 4). A fourth assumption that we will need for an important proof is an assumption on the individual robot plans. This is introduced in Section 3.3.

3.1. Networks of robots

The physical locations of the robots coupled with the characteristics of the hardware and the requirements of the task dictate the *physical network* for the group of robots. This network determines neighborhood relationships among the robots and can be represented by a *cooperation graph*, G, where the robots themselves are the vertices and the relationship itself is represented by the edges. The cooperation graph is represented by a tuple $(\mathcal{R}, \mathcal{E}, \mathcal{G})$, where $\mathcal{R}$ is the set of robots, $\mathcal{E} \subseteq \mathcal{R} \times \mathcal{R}$ is the edge set representing the relationship among the robots, and $\mathcal{G}$ is the set of constraint functions that describe the conditions under which each edge is maintained. For each element of $\mathcal{E}$ there is at least one corresponding element in $\mathcal{G}$. When, for example, the task requires cooperation between robots R_i and R_j, the interaction between the robots is represented by the edge $e_{ij} = (R_i, R_j) \in \mathcal{E}$ and a set of m functions $\mathcal{G}_{ij} \subseteq \mathcal{G}$, where $\mathcal{G}_{ij} = \{g_1(q_i, q_j) \dots g_m(q_i, q_j)\}$.

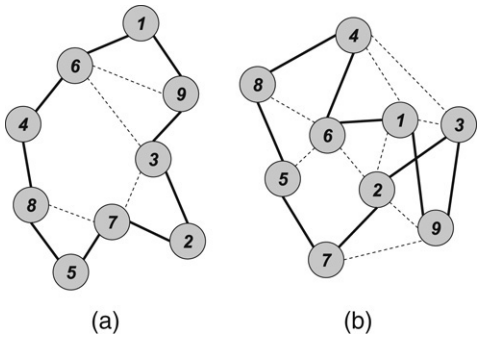

Fig. 1. *Hamiltonian Graph* as a common spanning subgraph of two different graphs. The continuous edges represent the *essential edges* that form the HG and dashed edges are *desirable edges*.

In this case we also say that R_j is a *neighbor* of R_i. Under the assumption that robots are identical and the constraints are symmetric, the graphs can be modeled as undirected graphs. In this case, $(R_i, R_j) \in \mathcal{E}$ is equivalent to $(R_j, R_i) \in \mathcal{E}$ and $\mathcal{G}_{ij} = \{g_1(q_i, q_j) \dots g_m(q_i, q_j)\}$ is identical to $\mathcal{G}_{ji} = \{g_1(q_j, q_i) \dots g_m(q_j, q_i)\}$. Also, the graph is simple — at most one edge connects any two nodes and there are no loops (edges of the form (R_i, R_i)).

For a given graph G we define the *valency* of a vertex as the number of neighbors of this vertex and *cycle* as a graph where every vertex has exactly valency two. Also, a *path* from R_i to R_j is a sequence of consecutive vertices starting with R_i and ending in R_j such that consecutive vertices are neighbors. We also say that a graph, G, is *connected* if there is a path between any two vertices of the graph. We define now a *Hamiltonian Path* (HP) as a path through a graph that includes the whole set of vertices of the graph. One can observe that this graph can have one or no cycles. When all vertices have valency two it has a cycle and it is called *Hamiltonian Cycle* (HC). It also can be noticed that for the same set of vertices, $\mathcal{R}$, several HPs and HCs are possible. In this paper we will call both graphs *Hamiltonian Graphs* (HG). Routing algorithms, for example, can take advantage of the existence of HGs in order to create a channel of communication between any two vertices of the graph, since the existence of a path is guaranteed. Let the *spanning subgraph* of a generic graph, X be the graph that has the same set of vertices of X. Then, by definition, each HG is a common spanning subgraph of a set of generic graphs with $\mathcal{R}$ as vertices. Fig. 1 shows a particular HC for a set of nine robots as a spanning subgraph of two different graphs.

We are interested in moving the robots in such a way that they can reach their goals and also maintain connectivity in the cooperation network. Thus, based on the previous definition of HG it is easy to verify that we can guarantee connectivity in a group of n robots if we constrain the robots movements in such a way that there is always at least one HG of n vertices.

In addition to maintaining connectivity, it may also be desirable to maintain additional edges in order to increase the system robustness to robot failures and to improve task performance. In sensing algorithms, for example, edges in the graph provide additional information for estimators and the additional edges can improve the quality of the estimates.

Thus, we may think of formation constraints including hard constraints, g, and secondary constraints, h. The edges corresponding to g are shown solid in Fig. 1, while those corresponding to h are shown dashed. Secondary constraints can be used to model constraints on the desired shape. In this way we can redefine $\mathcal{E}$ in G such that the graph has two types of edges: (i) *essential edges*, those which belong to the HG and (ii) *desirable edges*, those which do not belong to the HG. Accordingly, we redefine $\mathcal{G}$ as the set that includes, besides $\mathcal{G}_{ij}$, which correspond to essential edges, elements of the type $\mathcal{H}_{ij} = \{h_1(q_i, q_j) \dots h_m(q_i, q_j)\}$, corresponding to the desirable edges.

Based on these previous ideas we will develop decentralized controllers that (i) guarantee the existence of the essential edges for a given HG; (ii) try to create or maintain desirable edges; and (iii) lead the robots to their goals. We will establish guarantees on (i) and (iii), but (ii) is treated as a secondary objective. The determination of the desirable edges can be done in a centralized or decentralized fashion. We will address these problems in subsequent sections.

3.2. Formation constraints

As mentioned before, with each edge $(R_i, R_j) \in \mathcal{E}$, we associate at least one configuration constraint for R_j induced by R_i as an inequality of the form $g_k(q_i, q_j) \leq 0$, which represents a formation constraint. The set $\mathcal{G}_{ij}$ of the m constraint functions due to edge e_{ij} is a representation of a region in the configuration space γ_{ij}, defined as:

$$\gamma_{ij} = \bigcap_{k=1}^{m} A_k, \tag{1}$$

where,

$$A_k = \{(q_i, q_j) \in \mathbb{R}^2 \times \mathbb{R}^2 \mid g_k(q_i, q_j) \leq 0\}, \quad 1 \leq k \leq m.$$

This formulation defines a configuration space for a robot R_i as a function of one of its neighbors, R_j, where formation constraints are satisfied. The intersection of the configuration spaces induced by all neighbors of R_i defines the region for q_i where all neighborhood relationships are maintained. It is represented as:

$$\Gamma_i = \bigcap_{k} \gamma_{ik}, \quad \forall (R_i, R_k) \in \mathcal{E}. \tag{2}$$

Analogous to the problem defined in Section 2, one can see that the region in the configuration space represented by Γ_i and defined by the constraints in $\mathcal{G}$, directly determines the valid configuration space for R_i as:

$$\mathcal{C}_{R_i}(\mathcal{R} \setminus R_i, t) = \mathcal{F}_i \cap \Gamma_i. \tag{3}$$

In this paper the formation constraints define three regions in the robot's configuration space (see Fig. 2). In the *safe region*, each constraint $g(q_i, q_k) < \delta$, where the small negative number δ can be thought of as a threshold. The region defined by $\delta \leq g(q_i, q_k) < 0$ is the *critical region* for the robot. The constant δ is designed in order to guarantee that the constraint

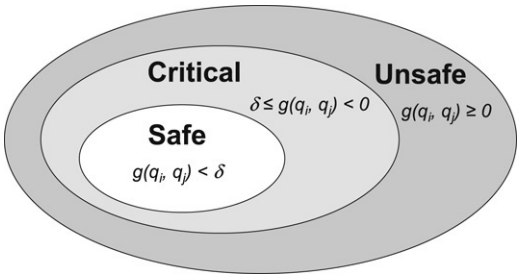

Fig. 2. The activation of the constraints define three regions in the robots' configuration space.

is still satisfied in the critical region and also to ensure that the robot does not leave this region. We say that a constraint is active when $g(q_i, q_k) \geq \delta$. If $g(q_i, q_k) \geq 0$ the robot is outside the valid configuration space that we call *unsafe region*. Depending on the nature of the constraints, the robots may not be able to return to the safe region of the configuration space. Our decentralized controllers are designed with the objective of keeping the robots in the safe configuration space.

3.3. Navigation functions

As discussed in the beginning of this section, a navigation function for solving the non-cooperative problem of steering each individual robot towards the goal while avoiding the static obstacles in the environment is assumed to be available from a higher level planner. Navigation functions are artificial potential fields without local minima [11]. Thus, for a navigation function, V_i, robot R_i input is given by $u_i = -k\nabla V_i$ where ∇V_i is the gradient of V_i. As pointed out in [12], this kind of navigation function can be thought of as a Lyapunov function for the system $\dot{q} = u(q)$, $u(q) = -\nabla V(q)$, because $V(q)$ is positive definite and the value of V is, by definition, always decreasing along system trajectories. Also, the navigation function can be modified in real-time if to its gradient is added a generic vector, given that the magnitude of this vector is smaller than $\|\nabla V(q)\|$, where $\|\cdot\|$ is the norm operator.

Proposition 1. *A navigation function $V(q)$ with goal in q^{goal} is a Lyapunov function for the holonomic robot R with dynamics $\dot{q} = u$ controlled by $u = -\nabla V(q) + U(q)$ if $\|U(q)\| < \|V(q)\|$.*

Proof. $V(q)$ is positive definite by the definition of navigation function. We need to show that $\dot{V}(q) \leq 0$.

$$\begin{aligned}\dot{V}(q) &= \nabla V(q) \cdot \dot{q} = \nabla V(q) \cdot u \\ &= \nabla V(q) \cdot [-\nabla V(q) + U(q)] \\ &= -\|\nabla V(q)\|^2 + \nabla V(q) \cdot U(q) \leq 0,\end{aligned}$$

since $\|\nabla V(q)\|^2 \geq \nabla V(q) \cdot U(q)$. Hence, $\dot{V}(q)$ always decreases along the robot's trajectory and vanishes at the goal position where $\|\nabla V(q)\| = 0$. Therefore $V(q)$ is a Lyapunov function for R. ■

The construction of navigation functions is beyond the scope of this paper. A methodology for constructing analytical

functions was proposed in [11]. A numerical method that allows for complex environments is presented in [23].

When robots are cooperating, there are formation constraints that force them to navigate near each other and their final goals are also reasonably close to each other. In this situation, the gradients of navigation functions for neighboring robots are similar. Based on this observation, our fourth assumption is:

Assumption 4. For any pair of cooperating robots, R_i and R_j, $\nabla V_i(q_i) \cong \nabla V_j(q_j)$, where V_x is the navigation function for R_x.

Obviously if the robots have goals that are not close to one another and the robots are close to their destinations, this assumption is not valid because their navigation functions will be very different. This fourth assumption is required for the proof of Proposition 3 in Section 3.4 and will be experimentally evaluated in Section 4.1.

3.4. Decentralized controllers

Our control system is decentralized and implemented using a set of three reactive controllers, one for each region of the configuration space shown in Fig. 2. The switching between these controllers is governed by activation of constraints that depend on the relative positioning of a robot with respect to its neighbors. Our control system allows each robot to have up to two assigned essential neighbors and several desirable ones. In other words it guarantees connection with up to two robots by maintaining at least one Hamiltonian Cycle and tries to maintain several other edges. We will assume for each robot the availability of a navigation function $V_i(q_i)$ with a unique minimum at q_i^{goal} which is presumably derived from a knowledge of the obstacles and the goal destination.

In this section we will call the two essential neighbors of R_i (those which belong to the HC) by R_a and R_b. We denote the constraints due to R_a, which have the form $g(q_a, q_i) \leq 0$, by g^a and those due to R_b, which have the form $g(q_i, q_b) \leq 0$ by g^b. In general, constraints induced by R_x will be denoted by g^x.

When a robot is in the *unsafe* configuration space it tries to move in order to satisfy the constraints induced by its essential neighbors without using the navigation function. In other words, the constraints themselves act as potential fields attracting the robots to each other and forcing them into a feasible configuration that satisfies all the constraints. The control input in this mode is:

$$u_i = -k_1 \left(\alpha \nabla g^a + \beta \nabla g^b \right), \tag{4}$$

where ∇g^x is a unit vector along the gradient of the constraint defined by:

$$\nabla g^x = \frac{\partial g^x / \partial q_i}{\| \partial g^x / \partial q_i \|}.$$

∇g^a is due to R_a and ∇g^b is due to robot R_b. The variables α and β assume a value 1 or 0 depending on whether the constraints are active or not, respectively and k_1 is a positive constant.

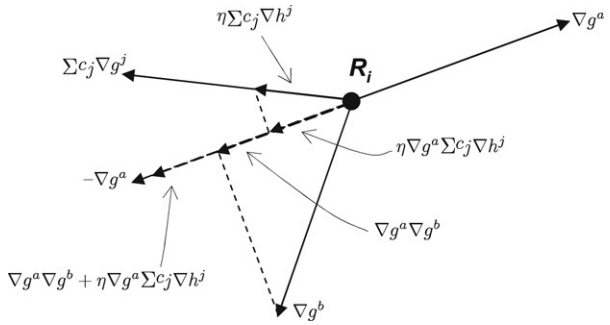

Fig. 3. η is chosen in order to avoid R_i moving in a direction contrary to ∇g^a and ∇g^b. Only the constraint relative to ∇g^a is illustrated for simplicity. Observe that a small η was chosen in such a way that $\nabla g^a \nabla g^b + \eta \nabla g^a \sum c_j \nabla h^j < -\nabla g^a$.

In the *safe* configuration space the robot has the following input:

$$u_i = -k_2 v_i, \tag{5}$$

where

$$v_i = \frac{\nabla V_i(q_i)}{\| \nabla V_i(q_i) \|},$$

is the normalized gradient vector of the navigation function $V_i(q_i)$.

In the *critical* configuration space a robot tries to navigate toward the goal while maintaining the formation constraints. The control input for this state is:

$$u_i = -k_1 \left(\alpha \nabla g^a + \beta \nabla g^b + \eta \sum_j c_j \nabla h^j \right) - k_2 v_i, \tag{6}$$

where $k_2 > 3k_1$ in order to guarantee convergence to the goal, as will be shown in the proof of Proposition 3. In this equation α, β, and c_j can each be 0, or 1. When $g < \delta$, the value 0 is assigned. When $g \geq \delta$, the value 1 is assigned. The summation $\sum c_j \nabla h^j$ is the part of the control action due to the desirable connections in the graph and corresponds to the summation of the active constraints induced by the desirable neighbors (those which do not belong to the HC). The non-negative gain η must be properly chosen in order to satisfy the following constraints:

$$\begin{aligned} -1 &< \alpha\beta \nabla g^a \cdot \nabla g^b + \eta\beta \nabla g^b \cdot \sum c_j \nabla h^j \leq 2 \\ -1 &< \alpha\beta \nabla g^a \cdot \nabla g^b + \eta\alpha \nabla g^a \cdot \sum c_j \nabla h^j \leq 2. \end{aligned} \tag{7}$$

As shown in Fig. 3, these two constraints for η are used in order to guarantee, by controlling the size of $\eta \sum c_j \nabla h^j$, that (i) the robots will never go in directions contrary to ∇g^a and ∇g^b, and (ii) the robots will never go in directions contrary to v_i. The value of η can then be chosen as the maximum non-negative value that satisfies the constraints. Observe that η is zero when ∇g^a and ∇g^b are anti-parallel vectors. In this case the robots give up on keeping the desirable edges and worry only about the necessary ones.

The three control laws (4)–(6) guarantee that the formation constraints are satisfied and, given Assumption 4, solve the n problems of individually leading the robots to their goals.

Proposition 2. *If the robots start in a feasible configuration, i.e. a configuration which satisfies all formation constraints, the switched control law represented by (4)–(6) guarantees that those constraints are satisfied during the robot's motion.*

Proof. We consider a generic constraint involving a generic pair of robots R_i and R_j, $g(q_i, q_j) \leq \delta$, and show that when the constraint is active, the control input makes $\dot{g}(q_i, q_j) \leq 0$. The time derivative of $g(q_i, q_j)$ is given by:

$$\dot{g}(q_i, q_j) = \frac{\partial g}{\partial q_i}\dot{q}_i + \frac{\partial g}{\partial q_j}\dot{q}_j. \tag{8}$$

For the ith robot, if $g(q_i, q_j)$ is active, then for the jth robot, $g(q_j, q_i)$ is also active (since we assume symmetry in Assumption 1). In the control law (6), $\nabla g^a = \nabla g^j$ for R_i and $\nabla g^b = \nabla g^i = -\nabla g^j$ for R_j. Let ∇g^α be the term associated with the constraint induced by the other necessary neighbor of R_i and ∇g^β be the term associated with the constraint induced by the other necessary neighbor of R_j. Substituting for $\dot{q}_i$ and $\dot{q}_j$ in (8) from (6), the time derivative of $g(q_i, q_j)$ is given by:

$$\begin{aligned}
\dot{g}_j &= \frac{\partial g}{\partial q_i}\cdot\left[-k_1\left(\alpha\nabla g^\alpha + \nabla g^j + \eta_i\sum c_k\nabla h^k\right) - k_2 v_i\right] \\
&\quad + \frac{\partial g}{\partial q_j}\cdot\left[-k_1\left(-\nabla g^j + \beta\nabla g^\beta\right.\right. \\
&\quad \left.\left. + \eta_j\sum c_l\nabla h^l\right) - k_2 v_j\right] \\
&= \left\|\frac{\partial g}{\partial q_i}\right\|\nabla g_j^k\cdot\left[-k_1\left(\alpha\nabla g^\alpha + \nabla g^j\right.\right. \\
&\quad \left.\left. + \eta_i\sum c_k\nabla h^k\right) - k_2\, v_i\right] \\
&\quad - \left\|\frac{\partial g}{\partial q_i}\right\|\nabla g_j^k\cdot\left[-k_1\left(-\nabla g^j + \beta\nabla g^\beta\right.\right. \\
&\quad \left.\left. + \ \eta_j\sum c_l\nabla h^l\right) - k_2 v_j\right] \\
&= \left\|\frac{\partial g}{\partial q_i}\right\|\left[-k_1\left(\nabla g^j\cdot\nabla g^j + \alpha\nabla g^j\cdot\nabla g^\alpha\right.\right. \\
&\quad \left. + \eta_i g^j\cdot\sum c_k\nabla h^k\right) - k_2\nabla g^j\cdot v_i \\
&\quad - k_1\left((-\nabla g^j)\cdot(-\nabla g^j) + \beta(-\nabla g^j)\cdot\nabla g^\beta\right. \\
&\quad \left.\left. + \eta_j(-\nabla g^j)\cdot\sum c_l\nabla h^l\right) - k_2(-\nabla g^j)\cdot v_j\right].
\end{aligned}$$

Under the assumption that $v_i = v_j$ we have:

$$\begin{aligned}
\dot{g} &= -\left\|\frac{\partial g}{\partial q_i}\right\|\left[k_1\left(1 + \alpha\nabla g^j\cdot\nabla g^\alpha + \eta_i\nabla g^j\cdot\sum c_k\nabla h^k\right)\right. \\
&\quad \left. + k_1\left(1 + \beta(-\nabla g^j)\cdot\nabla g^\beta + \eta_k(-\nabla g^j)\cdot\sum c_l\nabla h^l\right)\right] \\
&\leq 0,
\end{aligned}$$

since η_i and η_k were chosen in order to satisfy constraints (7).

If $v_i \neq v_j$ eventually, $\dot{g}(q_i, q_k) > 0$. In these cases at a certain point we can have the activation of the constraint $g(q_i, q_j) \leq \delta$ and consequently the control will switch to the mode relative to the unsafe space. The control law in this mode makes $\nabla v_i = \nabla v_j = 0$ and consequently, by the previous conclusion, the constraints are preserved. Therefore, given the initial conditions, $g(q_i, q_j) \leq 0$, for all $1 \leq i \leq n$ and R_j essential neighbor of R_i, and the fact that the derivatives $\dot{g}(q_i, q_j) \leq 0$ when this constraint is active, the proposition is proved. ■

Proposition 3. *If the robots' initial and goal positions are valid configurations and during the motion the gradient of the navigation function of two neighbor robots can be considered the same, the switched control law represented by (4)–(6) leads the robots to their goals.*

Proof. Observe by the proof of Proposition 2 that if the robots' initial configuration satisfy $g(q_i, q_j) \leq \delta$ and $v_i = v_j$, then they will never enter in the unsafe configuration space. In that way, only Eqs. (5) and (6) will be used as control inputs. Thus, notice by Proposition 1 that $V_i(q_i)$ is a common Lyapunov function for these control modes since constraints (7) guarantee η_i such that the term added to the normalized gradient of the navigation function is non-negative and smaller than or equal to 3, and $k_2 > 3k_1$. ■

One may question the assumption $v_i(q_i) = v_k(q_k)$ that underlies the validity of Proposition 3. However, it is a good assumption when the robots are far away from their respective destinations, as discussed in Section 3.3. Also, it is important to notice that convergence to the goal will only be guaranteed if the constraints are satisfied when the robots are at the goal position. More specifically:

$$\left(q_1^{\text{goal}}, q_2^{\text{goal}}, \ldots, q_n^{\text{goal}}\right) \in \mathcal{C}_{R_1}(t) \times \mathcal{C}_{R_2}(t) \times \cdots \times \mathcal{C}_{R_n}(t),$$

for $t \to \infty$. If this constraint is not satisfied, the robots would eventually need to violate the formation constraints in order to reach their goals. Since this is not allowed by our controllers, as shown in the proof of Proposition 2, the robots will be stuck in a local minimum introduced by the mode of the controller relative to their unsafe configuration spaces. If this behavior is not acceptable, a high level planner should act and change the cooperation graph, G, during the task execution.

3.5. Neighborhood assignment

The problem of neighborhood assignment is a difficult task to be executed in a decentralized fashion mainly because global knowledge of the robots' position is necessary. It consists basically in specifying the graph G, which is a representation of the cooperation network. Thus, it is very dependent on the task the robots are performing. In this paper we propose a heuristic that is based on two steps (i) selecting the Hamiltonian graph and (ii) selecting the desirable edges.

3.5.1. HG Selection

Selecting the HG is an operation that involves global knowledge of the group position. Therefore more resources, such as communication bandwidth, processing time, power, etc., need to be used. Because of this, we suggest the algorithm to be used only once, or when it is necessary during the execution of the task (e.g., one of the robots fails or a new robot is incorporated).

In the first step of the algorithm the robots discover the graph topology by "flooding" packages on the network and listening

to the answers. This is a traditional way to determine network topologies and may be substituted by more efficient ones [24]. Based on this topology, each robot determines n (or what they think it is) and the IDs of its teammates. One of the robots is then chosen to execute the computation, in case of centralized algorithms, or coordinate it, in case of parallel ones. Leader election algorithms such the one presented in [20] can be used in this processes.

Exact algorithms for finding a Hamiltonian Cycle are NP-complete. However, some heuristics and parallel algorithms can be used to improve the computation [25]. The only necessary condition for the algorithms is that there is at least one HC. In this case, if no solution is found in a given time, it is assumed that there is no HC in the graph and the robots need to move in order to create more connections.

3.5.2. Desirable edges selection

An easy and obvious way of determining which edges of the graph are desirable is to consider all existing edges as such. In this way the robots try to maintain only the current edges and consequently as soon as an edge is broken it is not considered desirable any more. As simple as the previous solution is considering that only the m closest robots are important. In this case, because the gradient of the desirable edge constraints are summed together in (6), the smaller m is, greater is the probability of maintaining the edges.

A more complex way of selecting the edges is based on routing and communication algorithms. In this case desirable edges are those which make the communication simpler and more efficient. In certain tasks, for example, it could be necessary to increase the bandwidth between two nodes, making a specific edge to be selected as desirable.

3.6. Extension to real robots

Our control laws were derived under Assumptions 1–3. However, most real robots fail at least one of these assumptions and, therefore, it is natural to ask whether our methodology can be used or not for other kinds of robots.

Assumption 1 cannot be entirely relaxed in applications where the formation constraints depend on the physical characteristics of the robots. It happens because our proofs need the constraints for both neighbor robots to be active at the same time and then, they need to be identical. However, a simple way to avoid the problem is to choose common constraints for each pair of robots. For example, consider a group of robots maintaining communication constraints. In this case, it is important that the robots' antennas have the same characteristics. If they do not, a very good approximation is using the constraints related to the weakest antenna in a pair of neighboring robots. For this specific situation, other differences such as mobility, size of the robots, etc. do not affect the algorithm.

Assumption 2 is easily relaxed if the obstacles are grown to the size of the robots during the navigation function construction.

Assumption 3 is more difficult to deal with. For non-holonomic robots, u_i, which is a two-dimensional vector, can be used as a set-point for controllers that take into account the non-holonomic constraints. Experimental results will show that the methodology works in several cases. A different approach is shown in [12] where dipolar potential fields are used to generate potential fields for non-holonomic systems. This is a promising direction for future research.

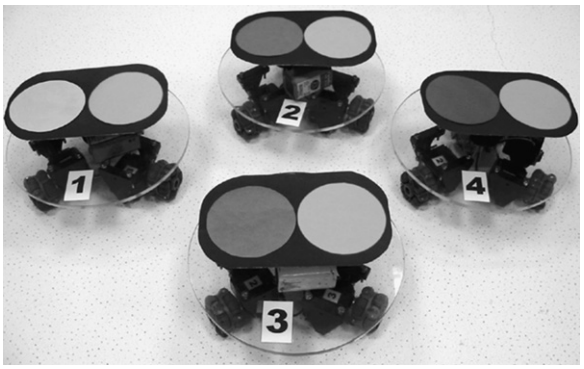

Fig. 4. The VERLab holonomic robots.

The next section will show experiments with small teams of both holonomic and non-holonomic robots in order to illustrate the proposed methodology.

4. Experiments

4.1. Flocking

The first group of experiments presented in this paper was performed with a team of holonomic robots. Each holonomic robot has three omnidirectional wheels mounted in a circular, 10 cm radius, plexiglass platform as shown in Fig. 4. They do not have any local sensors or communication devices and are remotely controlled by a computer that relies on an overhead camera to localize the robots in the environment.

We want to control the holonomic robots to perform a task known as *flocking* [26]. In this task a group of robots is suppose to move from an initial region of the free configuration space to another region. In order to move as a group the robots must not exceed a maximum distance from their neighbors and also avoid possible collisions with static obstacles and with other robots. These behaviors can be modeled as relative constraints to the robots' motion. Such constraints are very similar to real world communication and sensing constraints as will be shown in Section 4.2.

In the experiments presented in this section, three holonomic robots were used and the graph G was set to be a complete triangular graph (see Fig. 5) with two constraints per edge: (i) an avoidance constraint represented by a circle with 24 cm of diameter and (ii) a maximum distance constraint represented by a circle with 45 cm of diameter. Inside the first and outside the second circle, which are centered in each of its neighbors, each robot is in the unsafe region of its configuration space.

Fig. 6 shows trajectories of the three holonomic robots flocking from an initial position at time t_0, to a final goal at t_f in an environment with a single circular obstacle. Observe that in the initial configuration the constraints are not satisfied and

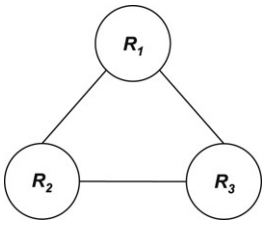

Fig. 5. Cooperation graph for the experiment in Fig. 6.

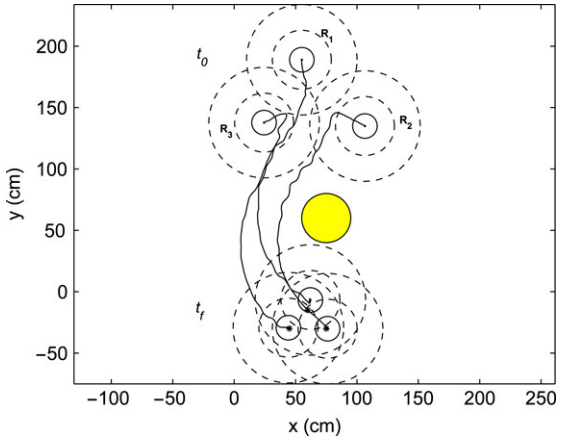

Fig. 6. Three holonomic circular robots flocking from initial configurations to a target region avoiding a single circular obstacle. The dashed lines represent both the collision and the maximum distance constraints.

the robots need to move towards each other. As proposed in our methodology, a navigation function was independently constructed for each robot. Because the navigation function defines paths from every free configuration to the desirable configuration, observe in Fig. 6 that at t_0, R_2 could move towards the goal taking either side of the obstacle. However, due to formation constraints the robots travel together to the goal area.

In order to evaluate the effect of possible differences in gradients of navigation functions for each robot (which is contrary to Assumption 4), and also the effect of different values of the threshold δ, we have executed several trials and obtained quantitative results. We have chosen four sets of values for δ. In the first set, the robots enter in the critical region of the configuration space when they are outside a circle with diameter of 37 cm or inside a circle with diameter of 33 cm, centered on their neighbors. In the second set of values these circles were set to have 40 cm and 30 cm of diameter respectively, and

Table 1
Percentage of time in each region of the configuration space for four different sets of values for δ: Set 1 — big δ, large critical regions; Sets 2 and 3 — intermediate values of δ; Set 4 — small δ, small critical regions

	Configuration space	R_1	R_2	R_3
Set 1	Safe (%)	21.16	19.71	16.08
	Critical (%)	77.24	79.80	83.05
	Unsafe (%)	1.60	0.49	0.87
Set 2	Safe (%)	63.77	67.66	60.94
	Critical (%)	33.76	31.78	36.81
	Unsafe (%)	2.47	0.56	2.25
Set 3	Safe (%)	78.10	81.08	77.62
	Critical (%)	20.30	18.39	21.08
	Unsafe (%)	1.60	0.53	1.30
Set 4	Safe (%)	76.81	77.17	78.17
	Critical (%)	10.22	10.56	10.91
	Unsafe (%)	12.98	12.27	10.92

in the third set they have diameters 42 cm and 27 cm. In the forth set, the critical configuration space was made very small with circles diameters set at 44 cm and 25 cm respectively. The differences among these four sets of values can be visualized in Fig. 7, where the regions of the configuration space were plotted to scale.

For each value of δ we have performed 20 runs in a single obstacle environment (Fig. 6). In all runs the robots were initialized in the same region of the workspace but not in the same exact configuration (due to positioning errors). After entering for the first time in the critical or safe regions of their respective configuration spaces (and therefore satisfying the constraints), the percentage of time each robot spent in each region was computed and is shown in Table 1. An average of these values for each experiment is plotted in Fig. 8. Despite the time the three robots stayed in their unsafe spaces, in all 80 runs (20 for each set) the robots successfully completed their tasks and reached their goals.

The first observation from Table 1 is that Assumption 4 is either a good approximation or it is actually not necessary when the robots have enough time to respond to an active constraint. Observe that, except for set 4 where the critical region is very small, the control laws relative to the critical region of the configuration space are sufficient to prevent the system from moving to its unsafe space. This conclusion comes from the small percentage of time that the robots stay in their unsafe regions of the configuration space.

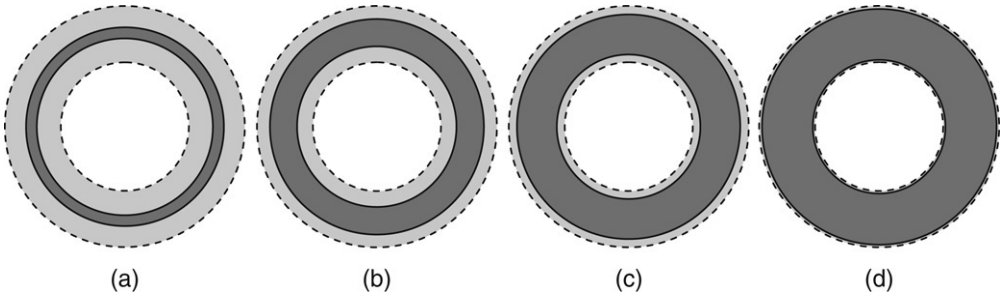

Fig. 7. The four different sets of values for δ used in the experiments. The dashed lines delimit the configuration space represented by $g(q_i, q_j) < 0$, and the continuous lines represent $g(q_i, q_j) = \delta$. The shadowed dark areas represent the safe configuration spaces and the shadowed light regions are the critical configuration spaces. (a) — Set 1, (b) — Set 2, (c) — Set 3, and (d) — Set 4.

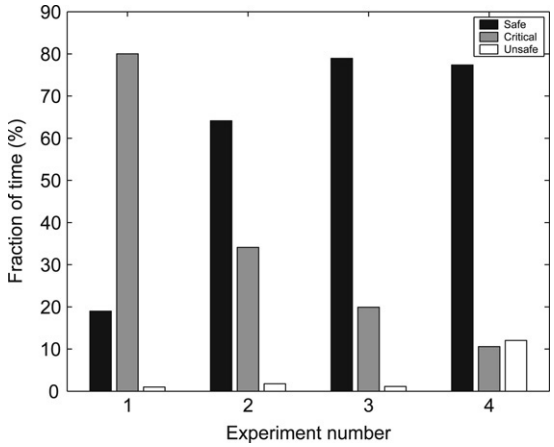

Fig. 8. Effect of δ in the mean time the robots spend in each region of their configuration spaces. The value of δ increases from experiment 1 to 4.

Table 2
Time of completion for each set of experiments

	Set 1	Set 2	Set 3	Set 4
Mean time (s)	38.67	34.90	34.60	41.64
Standard deviation (s)	2.46	1.55	1.74	5.74

The results also show that robots' behaviors change when the relative sizes of their configuration space regions vary. When the critical region is relatively large (set 1), the robots stay a large percentage of time in their critical space. Apparently this is not a big problem since we have proved that the relative control law is still following their navigation functions. However, the time of completion for the whole mission can be compromised, since they are not moving in the best direction, given by the gradient of the navigation function. This can be observed in Table 2 where the mean completion time for each set of experiments is shown. Observe, by comparing experiment sets 1, 2 and 3 that the less time the robots stay in their critical space, the smaller the time required for task completion. By Table 2 one can also notice that time of completion is very much compromised when the robots enter their unsafe configuration spaces, as is the case in the forth set of experiments. It happens because, in this space, the robots completely forget the navigation function they are following in order to rapidly satisfy the constraints. By acting like this, the robots may even move in a direction contrary to the gradient of the navigation function.

4.2. Sensing and communication constraints

In the previous subsection we have demonstrated our methodology with simple, globally localized holonomic robots. We will now exemplify how the proposed methodology can be used in a more attractive, real world application involving non-holonomic robots. Here the formation constraints are directly obtained by the robots' characteristics in applications that involve sensing or communication. We assume that sensing and communication devices have a 360° field of view and can be represented by circles centered at q_i and radius r_i. Therefore, the set of constraints induced by each robot on its neighbors can be represented by a single function $g(q_i, q_k) = (x_i - x_k)^2 + (y_i - y_k)^2 - r_k^2$.

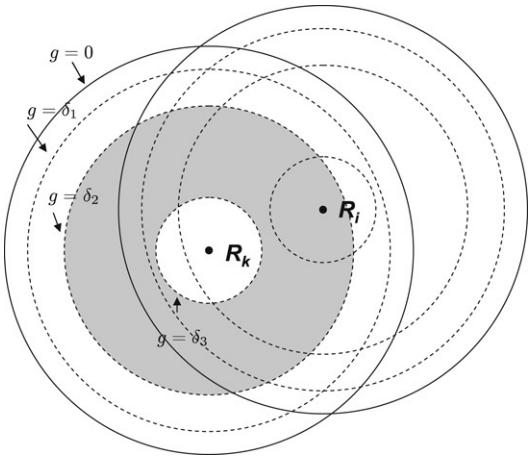

Fig. 9. Sensing/communication constraints: R_k induces constraints on the position of R_i. If R_1 is inside the circumference defined by $g \leq \delta_1$ (outer dashed circle), connectivity with R_k is guaranteed. The shadowed area defined by $g > \delta_3$ and $g < \delta_2$ is a "safe" configuration space for R_i where collisions are avoided and connectivity is maintained.

We will consider that a generic robot R_k induces three constraints in R_i. First, we have a hard sensing or communication constraint given by $g(q_i, q_k) \leq \delta_1$, beyond which the connectivity between R_k and R_i is broken. While δ_1 can be taken to be zero, it is best, in order to be robust to sensing errors, to keep it at a small, negative value. Second, we have a soft sensing or communication constraint given by $g(q_i, q_k) \leq \delta_2$. This is assumed to delineate a range within which the performance of the communication or sensing link is optimal. Finally, we have the collision constraint $g(q_i, q_k) \geq \delta_3$. Observe that $\delta_3 < \delta_2 < \delta_1 < 0$. Fig. 9 shows a picture of R_i's constraints induced by R_k. Each neighbor of R_i induces a similar set of constraints.

The platforms we used in this subsection are car-like robots equipped with omnidirectional cameras as their primary sensors. The robots can be seen in Fig. 10. The communication among the robots relies on IEEE 802.11b networking. A calibrated overhead camera is used to localize the robots in the environment. Because with this camera we are not able to estimate the robots' orientation, we use communication among the robots in order to construct a complete knowledge of the robots configuration. The communication is essentially used for sensing algorithms but is not used for control or decision making.

A limitation of the omnidirectional cameras used by the robots is that their resolution decreases with the distance of the objects. At 2 m, for instance, the projection of an observed robot in the image plane is only one pixel in size. For this reason three robots were programmed to keep sensing constraints with their neighbors and therefore localize themselves with respect to each other.

Fig. 12 shows six snapshots of our first experiment. In this experiment G was specified such that R_1 and R_3 are neighbors of R_2 but they are not neighbors of each other as shown in Fig. 11. The equipotential contours of the navigation function for R_3 is shown in all snapshots. Fig. 12(a) shows that R_1 was initialized outside the sensing region of R_2, which was set to be 1.5 m, and hence in an unsafe configuration space. Observe that this region is smaller than the one where the robots

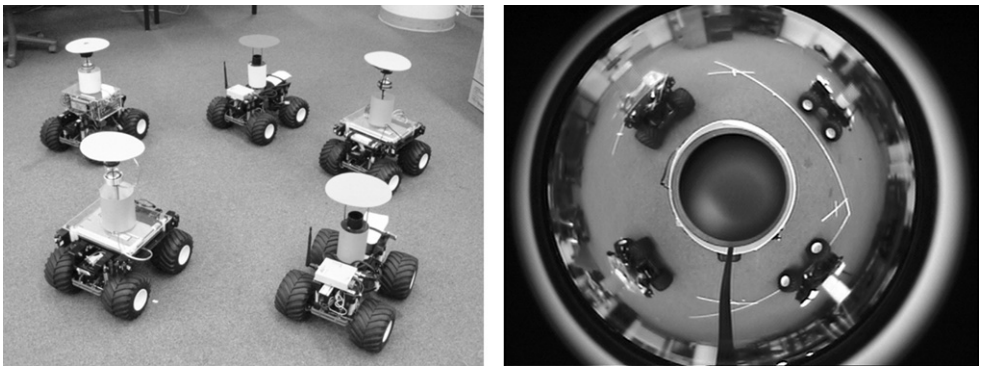

Fig. 10. The GRASP Lab. ClodBusters (left) and an omnidirectional image from their cameras (right).

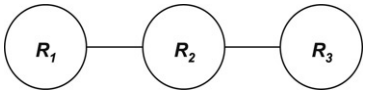

Fig. 11. Cooperation graph for the experiment in Fig. 12.

cannot actually detect the others (a circle of approximately 2 m), and therefore immediately outside its critical region of the configuration space they still can perform localization. The next snapshot shows that the robots move to satisfy this constraint. Fig. 12(c) shows R_2 and R_3 very close to each other. The activation of the avoidance constraints is then followed by a repulsion (Fig. 12(d)).

In another experiment we consider the deployment of a sensing network of 5 robots. One of the robots is static and is considered to be a fixed base. Fig. 13 shows four snapshots of the experiment where all the robots follow the same navigation function. Observe that because of the sensing constraints, a chain-like formation is obtained from the base to the goal.

5. Conclusions and future work

We developed a suite of decentralized reactive controllers for the cooperative motion control of a group of mobile robots. Our approach is based on the online modification of pre-computed navigation functions in order to satisfy formation

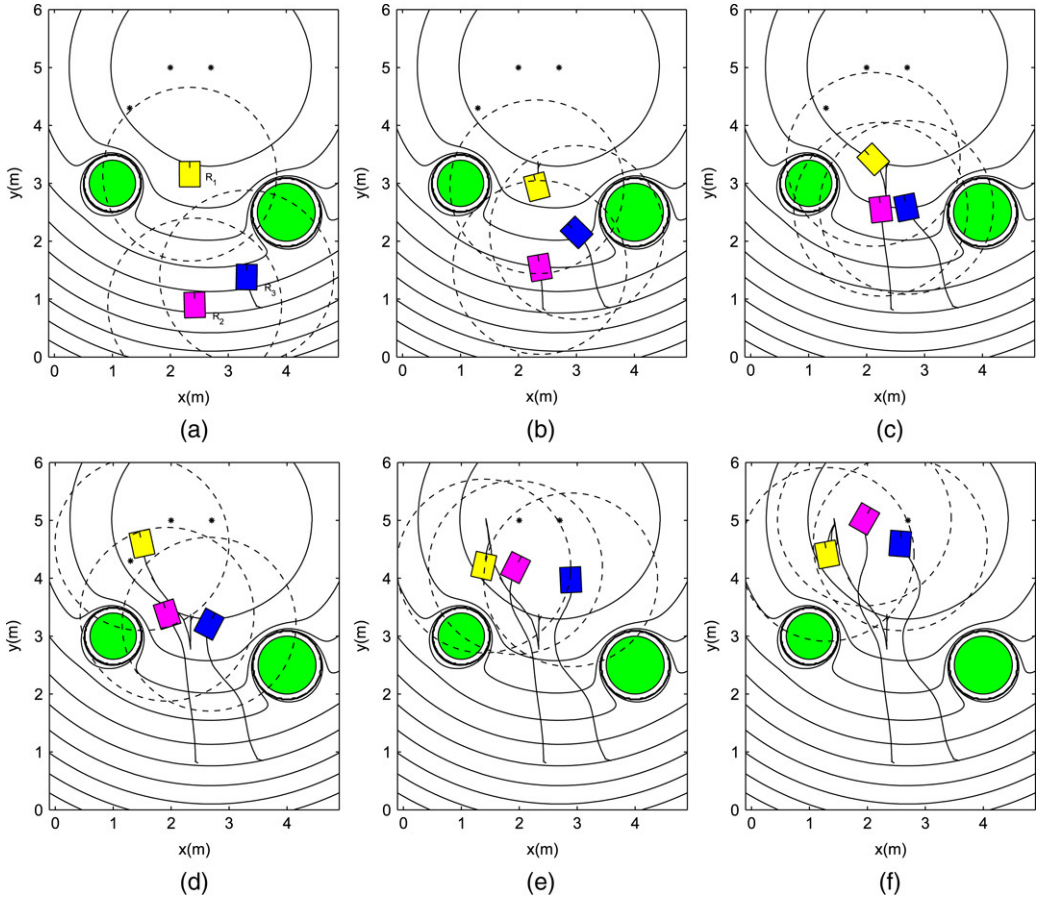

Fig. 12. Three robots following their navigation functions while maintaining sensing constraints with at least one other robot. Ground truth data is overlaid on the equipotential contours of the navigation function for R_3.

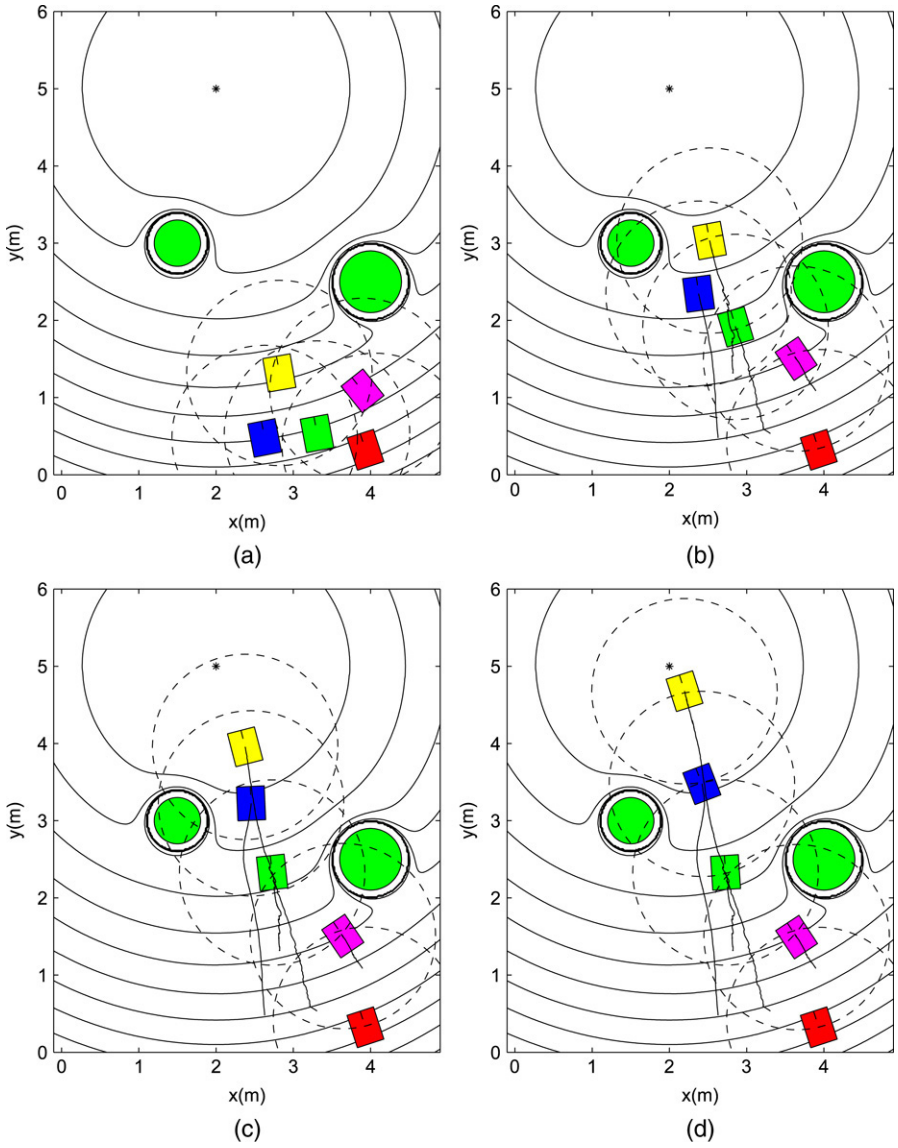

Fig. 13. Deploying a mobile sensor network with 5 nodes. Figures (a)–(d) show four snapshots of the same experiment. One of the robots is static and is considered a basis. Ground truth data is overlaid on the equipotential contours of the navigation function for the robots.

constraints. Proofs of convergence are presented in the case of holonomic robots. Although we have only presented here results with sensing and communication constraints, we have shown in previous work that the approach can be extended to other constraints such as those derived from manipulation [1] and formation control [21].

There are two main advantages of our approach. The decentralized algorithms mainly rely on the robots' ability to estimate the positions of their neighbors. Because robots are easily instrumented (in our case, this is done by tagging them with colored collars), this is relatively easy even in an unstructured environment. Also, no communication is required, allowing the operation in communication faulty situations. Therefore, our methodology is potentially scalable for larger groups of robots operating in unstructured environments.

The main limitations of the algorithms used here include (i) the requirement of a high level, centralized planning for neighborhood assignment; (ii) guarantees for up to two neighbor robots; and (iii) the assumption of identical navigation functions for two neighbor robots. All these are important theoretically but have not been a problem in our practical implementations. The neighborhood assignment problem, for example, will only be a problem when we consider groups of hundreds or thousands of robots.

There are important directions for future work. Firstly, it is necessary to explicitly model the non-holonomic constraint of the robots. Although we have presented encouraging results with non-holonomic robots following the holonomic inputs generated by our method, our proofs are not applicable to these systems. Secondly, it is important to develop decentralized algorithms for numbering the robots and selecting the cooperation graph, what currently has been made by a centralized planner. Finally, it will be valid to test the robustness of the methodology in outdoor environments where communication and sensing constraints are really important for the task completion.

Acknowledgments

This work was partially supported by FAPEMIG and CNPq of Brazil, and DARPA of USA.

References

[1] G.A.S. Pereira, V. Kumar, M.F.M. Campos, Decentralized algorithms for multi-robot manipulation via caging, International Journal of Robotics Research 23 (7–8) (2004) 783–795.
[2] J.R. Spletzer, C.J. Taylor, Sensor planning and control in a dynamic environment, in: Proc. IEEE Int. Conf. on Robotics Automation, 2002, pp. 676–681.
[3] B. Aronov, M. de Berg, A.F. van der Stappen, P. Svestka, J. Vleugels, Motion planning for multiple robots, in: Proc. Int. Symp. Computational Geometry, 1998, pp. 374–382.
[4] J.E. Hopcroft, J.T. Schwartz, M. Sharir, On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the Warehouseman's Problem, International Journal of Robotics Research 3 (4) (1984) 76–88.
[5] S.M. LaValle, S.A. Hutchinson, Optimal motion planning for multiple robots having independent goals, IEEE Transactions on Robotics and Automation 14 (6) (1998) 912–925.
[6] T. Simeon, S. Leroy, J.-P. Laumond, Path coordination for multiple mobile robots: A resolution complete algorithm, IEEE Transactions on Robotics and Automation 18 (1) (2002) 42–49.
[7] Y. Guo, L.E. Parker, A distributed and optimal motion planning approach for multiple mobile robots, in: Proc. IEEE Int. Conf. on Robotics Automation, 2002, pp. 2612–2619.
[8] R.C. Arkin, Behavior-Based Robotics, MIT Press, Cambridge, MA, 1998.
[9] T. Balch, R.C. Arkin, Behavior-based formation control for multi-robot teams, IEEE Transactions on Robotics and Automation 14 (6) (1998) 1–15.
[10] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, International Journal of Robotics Research 5 (1) (1986) 90–98.
[11] E. Rimon, D.E. Koditschek, Exact robot navigation using artificial potential functions, IEEE Transactions on Robotics and Automation 8 (5) (1992) 501–517.
[12] J.M. Esposito, V. Kumar, A method for modifying closed-loop motion plans to satisfy unpredictable dynamic constraints at runtime, in: Proc. IEEE Int. Conf. on Robotics Automation, 2002, pp. 1691–1696.
[13] A. Howard, M.J. Mataric, G.S. Sukhame, Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem, in: Proc. Int. Symp. Distributed Autonomous Robotic Systems, 2002, pp. 299–308.
[14] J. Reif, H. Wang, Social potential fields: A distributed behavioral control for autonomous robots, in: Proc. Int. Workshop Algorithmic Foundations of Robotics, A. K. Peters, Wellesley, MA, 1995, pp. 431–459.
[15] J.P. Desai, J. Ostrowski, V. Kumar, Controlling formations of multiple mobile robots, in: Proc. IEEE Int. Conf. on Robotics Automation, 1998, pp. 2864–2869.
[16] Y.Q. Chen, Z. Wang, Formation control: a review and a new consideration, in: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2005, pp. 3181–3186.
[17] T. Eren, P.N. Belhumeur, A.S. Morse, Closing ranks in vehicle formations based rigidity, in: Proc. IEEE Conf. on Decision and Control, 2002, pp. 2959–2964.
[18] R. Olfati-Saber, R.M. Murray, Graph rigidity and distributed formation stabilization of multi-vehicle systems, in: Proc. IEEE Conf. on Decision and Control, 2002, pp. 2965–2971.
[19] N.E. Leonard, E. Fiorelli, Virtual leaders, artificial potentials and coordinated control of groups, in: Proc. IEEE Conf. on Decision and Control, 2001, pp. 2968 –2973.
[20] A. Das, J. Spletzer, V. Kumar, C. Taylor, Ad hoc networks for localization and control, in: Proc. IEEE Conf. on Decision and Control, 2002, pp. 2978–2983.
[21] G.A.S. Pereira, A.K. Das, V. Kumar, M.F.M. Campos, Formation control with configuration space constraints, in: Proc. IEEE/RJS Int. Conf. Intelligent Robots and Systems, 2003, pp. 2755–2760.
[22] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementations, The MIT Press, 2005.
[23] L.C.A. Pimenta, A.R. Fonseca, G.A.S. Pereira, R.C. Mesquita, E.J. Silva, W.M. Caminhas, M.F.M. Campos, On computing complex navigation functions, in: Proc. IEEE Int. Conf. on Robotics Automation, 2005, pp. 3463–3468.
[24] B.B. Lowekamp, D. O'Hallaron, T. Gross, Direct queries for discovering network resource properties in a distributed environment, Cluster Computing 3 (4) (2000) 281–291.
[25] P.D. MacKenzie, Q.F. Stout, Optimal parallel construction of hamiltonian cycles and spanning trees in random graphs, in: Proc. 5th ACM Symp. Parallel Algorithms and Architectures, 1993, pp. 224–229.
[26] C. Reynolds, Flocks, birds, and schools: A distributed behavioral model, Computer Graphics 21 (1987) 25–34.

Guilherme Augusto Silva Pereira received the B.S. and M.S. degrees in Electrical Engineering and the Ph.D. degree in computer science from the Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil, in 1998, 2000, and 2003, respectively. Dr. Pereira received the Gold Medal Award from the Engineering School of UFMG for garnering first place among the Electrical Engineering students in 1998. He was, from November 2000 to May 2003, a Visiting Scientist at the General Robotics, Automation, Sensing and Perception (GRASP) Laboratory, University of Pennsylvania, Philadelphia. Since July 2004, he is an Assistant Professor of the Electrical Engineering Department at the Federal University of Minas Gerais (DEE/UFMG), Belo Horizonte, Brazil, where he is the director of the Computer Systems and Robotics (CORO) Laboratory. His research interests include cooperative robotics, robot navigation, control engineering, computer vision, and distributed sensing. Dr. Pereira is a Member of Sociedade Brasileira de Automática.

Vijay Kumar is the UPS Foundation Professor and the Chairman of Mechanical Engineering and Applied Mechanics at the University of Pennsylvania. He has a secondary appointment in the Department of Computer and Information Science. He has been on the Faculty at the University of Pennsylvania since 1987. He served as the Deputy Dean of the School of Engineering and Applied Science from 2000–2004 and directed the GRASP Laboratory, a multidisciplinary robotics and perception laboratory, from 1998–2004.

Dr. Kumar is a Fellow of the American Society of Mechanical Engineers (ASME) and the Institution of Electrical and Electronic Engineers (IEEE). He has served on the editorial boards of the IEEE Transactions on Robotics and Automation, the ASME Journal of Mechanical Design, and the IEEE Transactions on Automation Science and Engineering. He is the recipient of the 1991 NSF Presidential Young Investigator award, the 1997 Freudenstein Award for significant accomplishments in mechanisms and robotics and the 2004 IEEE International Conference on Robotics and Automation Kawamori Best Paper Award. He is also a Distinguished Lecturer in the IEEE Robotics and Automation Society.

Mario Fernando Montenegro Campos, Ph.D., is an Associate Professor of Computer Vision and Robotics in the Department of Computer Science at the Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil. He holds a B.S. degree in Engineering, and an M.S. in Computer Science, all from the Universidade Federal de Minas Gerais, and a Ph.D. in Computer and Information Science from the University of Pennsylvania. His research interests include cooperative robotics, robot vision, sensor information processing. His main contributions are in haptics, multi-robot cooperation and robot vision. He is the founder and director of the Vision and Robotics Lab — VeRLab, UFMG, Brazil. He is a Distinguished Lecturer in the IEEE Robotics and Automation Society.