Using Sensor Morphology for Multirobot Formations

Gal A. Kaminka, *Member, IEEE*, Ruti Schechter-Glick, and Vladimir Sadov

***Abstract*—In formation-maintenance (formation control) tasks, robots maintain their relative position with respect to their peers, according to a desired geometric shape. Previous work has examined formation-maintenance algorithms, based on formation control graphs, that ensure the theoretical stability of the formation. However, an exponential number of stable controllers exists. Thus a key question is how to select (construct) a formation controller that optimizes desired properties, such as sensor usage. We present a novel representation of the sensing capabilities of robots in formations, using a monitoring multigraph. We first show that graph-theoretic techniques can then be used to efficiently compute optimal sensing policies that maintain a given formation, while minimizing sensing costs. In particular, separation-bearing (distance-angle) control targets are automatically constructed for each individual robot in the formation, taking into account its specific sensor morphology. Then, we present a protocol allowing control graphs to be switched on line, to allow robots to adjust to sensory failures. We report on results from comprehensive experiments with physical and simulated robots. The results show that the use of the dynamic protocol allows formations of real robots to move significantly faster and with greater precision, while reducing the number of formation failures, due to sensor limitations. We also evaluate the sensitivity of our approach to communication reliability, and discuss opportunities and challenges raised by our approach.**

***Index Terms*—Coordinated movement, mobile robots, multirobot formations, multirobot systems.**

I. Introduction

In formation-maintenance (formation control) tasks, robots maintain their relative position with respect to their peers, according to a desired geometric shape. Various formation maintenance algorithms have been suggested (e.g., [2]–[4], [6], [7], [9], [11], [13], [14], [16]). The algorithms assign each robot with a single or multiple neighboring robots (*targets*) that it must monitor, to maintain the given geometric shape while moving. The set of assigned targets (and their associated controller type) is called *control graph* in [6] and [7].

Previous work has examined constraints on a given control graph, that would ensure the stability of the formation. In particular, one popular method is *separation-bearing control* (SBC) [2], [6]–[9]. In SBC, a single robot is chosen as the leader of the formation. Each robot (but the leader) must maintain a given distance (separation) and angle (bearing) with respect to an assigned target. It was shown that control graphs that induce SBC controllers for each robot, and satisfy other constraints (e.g., connectivity, a single leader, etc.) are sufficient to maintain stable formations.

However, for a given geometric formation, an exponential number of stable possible control graphs may exist [6], [7]. Thus, a key question is how to select (construct) a control graph that optimizes other desired properties in addition to stability. Many such desired properties have to do with each robot's sensor morphology—the type, placement, and configuration of sensors on robot bodies. For instance, a control graph in which one robot must pan its camera backward (relative to the direction of movement) is less likely preferable than one in which the same robot can monitor a target ahead of it. Unfortunately, previous work has often ignored the role of the sensor morphology in selecting between control graphs (see Section II for a detailed discussion, and notable exceptions).

This paper presents an instantiated graph-theoretic framework for control graph selection, based on sensor-morphology consideration. The framework represents alternative sensing schemes in a *monitoring multigraph* in which directed, weighted, edges denote sensing capabilities and their associated costs. By applying graph-theoretic techniques, sensor-optimal control graphs can be constructed efficiently.

We present two contributions. First, we provide an efficient method for automatically constructing sensor-optimal control graphs for SBC control. Second, we present a protocol allowing control graphs to be switched online, to allow robots to adjust to permanent and intermittent sensory failures.

To evaluate these contributions, the monitoring multigraphs framework has been implemented for Sony AIBO robots, and for the player-stage simulator [10]. We show the results of extensive experiments, demonstrating the robustness of the formations resulting from using monitoring multigraphs. We show empirically that use of dynamic switching of control graphs leads to significantly increased precision, better performance, and robustness to changing environmental conditions. We additionally discuss the opportunities and challenges raised by the use of the framework, in particular, in allowing heterogeneous multirobot formations (in which each robot may have a different sensor morphology), and greater robustness to obstacles interfering with robot sensing.

This article is organized as follows. Section II presents related work and background. Section III presents a method for constructing SBC control graphs, given a description of the formation shape and each robot's sensor morphology. Section IV presents a protocol for online dynamic switching of control graphs, for improved robustness. Section V presents the results of the experiments. Section VI discusses opportunities and challenges. Section VII concludes.

Manuscript received July 27, 2006; revised June 2, 2007. This paper was recommended for publication by Associates Editors R. Kelly and I. M. Chen and Editor L. Parker upon evaluation of the reviewers' comments. This work was supported in part by the Israel's Ministry of Science and Technology.

G. A. Kaminka and V. Sadov are with the MAVERICK Group, Department of Computer Science, Bar Ilan University Ramat Gan, 52900 Israel (e-mail: galk@cs.biu.ac.il; sadovv@cs.biu.ac.il).

R. Schechter-Glick was with the MAVERICK Group, Department of Computer Science, Bar Ilan University, Ramat Gan 52900, Israel. She is now with NDS Ltd. 91235 Jerusalem, Israel (e-mail: rutiglick@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2008.918054

II. Background and Related Work

We focus here on most related previous investigations. These have typically made the assumption that sensor configurations match the control graphs. Moreover, existing works often assume all robots are homogeneous, and thus do not generate monitoring rules that are individually tailored to the sensing capabilities of different robots. Our paper addresses these challenges. Other differences with existing work are discussed next.

Perhaps the most closely related work to ours is that of Lemay *et al*. [13] and Michaud *et al*. [14], which present a distributed method by which robots are assigned to positions in a formation. The distributed computation uses a cost function that takes into account distances and angles to the teammates, similar to the cost function we develop next; a bounded search process is used to determine the best (lowest cost) assignment of robots to formation positions. Each robot in a formation determines a cost for assigning its teammates to positions in the given formation, assuming it is the leader (which they refer to as *conductor*). Then, the best (minimal cost) assignment of roles to robots (including the leader) is made. Our approach complements this work. We use computation of sensing costs *after* robots have already been assigned to their positions, to determine the sensor configuration used by each robot to monitor its target. Furthermore, the technique we present allows dynamic switching of control graphs within the same formation, where the work by Lemay *et al*. and Michaud *et al*. allows switching of the formation shapes.

Fierro *et al*. [8] analyzed the stability of SBC and other controllers, and proposed using manually constructed control targets to allow up to three robots to switch between alternative controllers online without relying on communications. Our work complements their results by providing: (1) a method for optimal selection of alternative control graphs, for any number of robots and (2) a protocol using communications for making synchronized control graph decisions.

Desai *et al*. [6], [7] show how a formation can be maintained if each robot monitors an angle and distance to another robot. They use an unweighted control graph to describe monitoring from a global perspective. Desai *et al*. do not discuss selection of a control graph, and assume omnidirectional sensing. However, they discuss switching the geometric shape defining the formations (and their associated control graphs) to tackle terrain changes. Our framework complements such switching.

Balch and Arkin [2] examine three techniques for formation maintenance of up to four homogeneous robots. Two of these (*leader-referenced* and *neighbor-referenced*) techniques are essentially SBC controllers, using static (fixed) control graphs. Fredslund and Matarić [9] describe an algorithm for generating SBC monitoring rules for robots in a given formation. The robots are assumed to have specific sensing capabilities, and the position of the leader is given. The monitoring rules are supplemented by communications for robustness. Our algorithms consider the unique sensor configuration of each robot.

Balch and Hybinette [3] use social potential fields that use attraction and repulsion to position robots within their relative positions in a defined formation. This technique is robust to obstacles in the path of the robots, an important challenge our approach does not yet take into account.

Mourikis and Roumeliotis [15] discuss optimal sensor scheduling policies for formations, in which sensor use for localization within the formation is optimally balanced between resource consumption (e.g., energy) and localization accuracy. The sensors themselves are assumed to be fixed in configuration, but the frequency in which they are used is determined by the policies. Our work is complementary: The frequency in which sensors are used is fixed, but the sensor configuration is dynamically adjusted.

III. Cost-Optimal Formation Control Graphs

We begin by describing the use of monitoring multigraphs to analytically describe various ways in which a robot may monitor its peers by observation (Section III-A). Then, we describe how a multigraph can be used in formation-maintenance tasks to assist in the automatic generation of monitoring rules for robots (Section III-B).

A. Monitoring Multigraphs in Formation Control

We introduce the use of multigraphs to represent the monitoring capabilities of robots in a multirobot system. A *monitoring multigraph* is a tuple $\langle V, E\rangle$ where V is a set of vertices, denoting robots, and E is a *bag* (sometimes called multiset) of weighted edges $\{\langle u, v, w\rangle\}$, each linking two vertices $u, v \in V$, and having a nonnegative weight $w \in \mathbb{N}$. Since E is a bag, an edge linking two vertices may appear more than once (even with the same weight).

Edges denote monitoring capabilities. An edge $\langle u, v, w\rangle$ exists if robot u is able to monitorsense robot v in some distinct fashion, i.e., through a specific sensor. The weight w indicates the monitoring robot's cost of using the sensor, e.g., in terms of energy use, or expected reliability for the given distance and heading [13], [14] (see next). As multiple sensors may exist for one robot to monitor another, multiple edges may exist, with various costs. When a robot can monitor another, the reverse is not always true; thus edges are directed, i.e., $\langle u, v, w\rangle \in E \not\Rightarrow \langle v, u, z\rangle \in E$.

In practice, most tasks require monitoring to be selective. A monitoring multigraph can be useful in such reasoning, and allows the robot to represent monitoring options available to it, and the costs involved. The robots can reason about their monitoring decisions in the context of the global monitoring constraints.

We construct monitoring multigraphs to represent the sensory capabilities of robots in the formation. Vertices (denoting robots) are positioned according to the robots' positions in the formation, relative to their peers. As the vertices represent robots in the formation, the multigraph is embedded in the plane; each vertex has associated Cartesian coordinates in $\mathbb{R}^2$. The initial pose of all robots is to the direction of the movement. Complementary previous work has already addressed issues in the allocation of robots to positions in the formation [13], [14].

TABLE I
TYPE-1 ROBOT SENSOR CONFIGURATION

Attribute	Range	Cost
Distance (mm)	$[0, 450]$	0.4
	$(450, 600]$	0.75
Field of View	$[-30°, 30°]$	0.2
	$(30°, 50°]$	0.4
	$[-50°, -30°)$	0.4
Pan	$[-90°, 90°]$	0.6

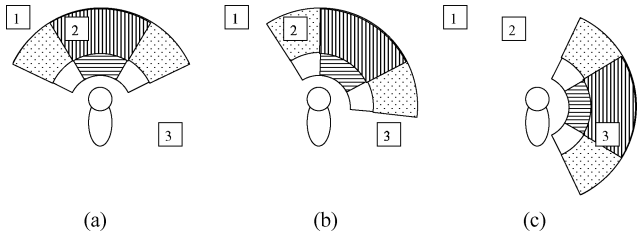


Fig. 1. Monitoring possibilities change based on sensor panning. (a) Pan $0°$. (b) Pan $30°$. (c) Pan $90°$.

For each robot (vertex), we add edges by considering its sensors that can be used for monitoring other robots. We focus on sensors that can provide identification, distance, and bearing to other robots, such as combinations of cameras and distance sensors. We exclude sensors that cannot be used for monitoring others, such as location (e.g., GPS), distances traveled (e.g., odometry), etc.

The weight of an edge is an indication of its *expected* cost-of-usage: Smaller weights indicate lower costs, and thus, greater preference for usage. This cost can be computed based on any number of factors. We empirically found the following three factors to be useful in practice: Sensing distance limits (similarly to [13] and [14]), field of view limits, and panning angle (rotation of the field of view with respect to the center of the robot). We, therefore, focus on these in this paper. However, sensor planning literature (e.g., [1]) reports on additional factors in reasoning about sensing and sensing costs, such as resolution, focus, and feature detectability. These will be addressed in future work.

For each factor, for each relevant range of values, we assign a cost, which reflects our inverse preference for the particular setting of the factor (i.e., to the range of values). For instance, Table I shows an example of a set of such assignments, for a hypothetical robot. The first column (attribute) marks the sensor attribute in question—distance, field of view, or panning angle. The second column (range) marks the values (ranges of values) for which we wish to specify costs. The costs are noted in the final column. Note that several ranges may be possible for each attribute, which may differ in their costs or range of values. We remind the reader that lower costs signify higher preference.

Fig. 1 shows this Type 1 robot using its single sensor at different pan angles. Each curved sub-region denotes monitoring areas with different costs. The two arcs differentiate distance limitations. The numbered squares denote other robots. For instance, Fig. 1(a) shows the robot panning straight ahead (at $0°$). Another robot (1) is outside of the distance range of the monitoring, regardless of the panning angle or field of view. Robot 3 (bottom right) cannot be monitored given the current pan and field of view. The remaining robot (2) is currently within the central field of view. Figs. 1(b) and (c) shows all robots in the same positions, but with different panning angles for the sensor.

An important issue to consider is the origin of sensor costs. With each factor's values, we can associate a number of preferences that would encourage or discourage us from using the sensor in these settings. Factors that come to bear on the cost include: *energy consumption* (some sensors use more energy than others), *latency*, *reliability* (e.g., measurements within some ranges are more reliable than in other ranges), the ability of using the sensor for other purposes (e.g., to detect obstacles), and actual *monetary costs*, in case we are only planning the formations and wish to try different combinations. The combination of all of these different factors (and any others) into a single cost value can be done using existing methodologies for multiple attribute decision-making, once the factors are known [17]. In this paper, we utilized only a single factor (reliability) in our sensor cost models.

To contrast alternative sensing possibilities, we construct edges from the monitoring robot to the other (monitored) robots. An edge is created for each combination of distance, field of view, and pan angle, that can sense the other robot. This is done as follows. First, we compute the area covered by a sensor, given its field-of-view possibilities, distance ranges, and pan options. For a field-of-view range $[f_{\min}, f_{\max}]$, a pan range $[p_{\min}, p_{\max}]$, and a distance range $[d_{\min}, d_{\max}]$, the area covered is a curved region enclosed by the distance range, and defined by the arcs at an angle $[p_{\min} + f_{\min}, p_{\max} + f_{\max}]$. Multiple pan, field-of-view, and distance range options give rise to multiple curved regions, which may overlap. For instance, based on Table I, the leftmost field-of-view range covers the arc $[-50 + -90, -30 + 90] = [-140, 60]°$, the central field of view covers $[-30 + -90, 30 + 90] = [-120, +120]°$, etc.

We then locate robots within each region. For each, we create a directed edge from the monitoring robot. Since the positions of vertices in the multigraph correspond to geometric positions in the formation, the distance between two robots corresponds to the length of the line connecting them, and the angle between any two robots can be computed relative to the initial pose, which faces the direction of movement. For instance, in Fig. 1, robot 1 is outside of the distance range of the monitoring robot. Thus, there would be no edge from the monitoring robot to this left top robot. Figs. 1(a) and 1(b), shows multiple ways in which robot 2 can be monitored—within the central field of view (when the pan angle is set to $0°$) and within the left field of view (pan angle set to $30°$). Thus two edges to it would be created.

Finally, we compute the weight of each edge, as a function of the costs of the distance, field of view, and pan ranges involved. We use a weighted sum function [17] to combine cost factors into a single cost value for the weight of the edge.

In real-world settings, robots may occlude each other. Thus, the last step includes removal of physically occluded edges. As embodied robots occlude each other, any robot x positioned on an edge between a pair of other robots a, b will block their view of each other. In this case, edges $\{\langle a, b\rangle, \langle b, a\rangle\}$ are removed from the monitoring multigraph. When applying this technique with physical robots, we have found it useful to consider

occlusion even if x is not positioned exactly on the edges between a and b, to account for the size of the physical body of an occluding robot.

The result of this process (after it is repeated for all robots, and all sensors) is a weighted, directed, monitoring multigraph, where vertices denote robots and (multi-) edges represent all possible ways in which the robots can monitor each other, given their sensors and limitations.

B. Computing Optimal SBC Control Graphs

Now that the monitoring multigraph is complete for a given formation, it can be used to induce individual controllers for each robot, such that if all robots maintain the distances and angles represented by the selected edges, the formation will be correctly maintained. In particular, we show how to use a version of Dijkstra's single-source shortest paths (S3P) algorithm to construct SBC controllers for each robot, that guarantee optimal sensing-cost formations.

In SBC, a robot—called *leader*—is responsible for determining the overall global path (e.g., by deferring to a human operator [12], or using a path planner). Each of the other robots (*followers*) is given individually-tailored distance and angle goal values, with respect to its *target*—either the leader, or another follower—that, in turn, monitors its own target. SBC thus relies on a single monitoring link for each robot. The distance and angle are given with respect to the direction of movement.

We can induce the SBC monitoring rules from the monitoring multigraph, by choosing edges that signify sensor choices. The edges' length and angle with respect to the initial pose signify the separation and bearing, respectively. For now, we make the assumption that the leader position has been predetermined. Given the leader, a formation graph can be maintained using the SBC under the following conditions:

1) The outdegree of the leader robot is 0.
2) The outdegree of every follower robot is exactly 1, the outgoing edge pointing to its target.
3) There exists a path from every follower to the leader.

The first condition guarantees that the leader does not have to monitor anyone. The second condition guarantees that each of all other robots has an separation-bearing target to monitor. The final condition guarantees that the formation is connected such that all robots monitoring others will eventually monitor the leader. In other words, it guarantees that the leader robot is indeed positioned such that it is capable of leading, given how it is monitored.

We define a formation graph as optimal, if in addition to the conditions mentioned earlier, it also guarantees that each individual robot is monitoring the leader, directly or indirectly (transitively) using the minimal sensor cost. This is not achievable, in general, by simply selecting the least costly edge out of each robot's position, since such local selection may cause robots to form a cycle, where robots monitor each other, instead of the leader. Moreover, such local selection does not address a key challenge: a robot's overall monitoring cost in the context of a formation depends also on its target's monitoring cost. This is because a robot's position depends on its target, and thus, shorter paths to the leader reduce latency in position update. In other words, it may be better to monitor a robot at a higher local cost, to guarantee that overall, the path from the target to the leader is shorter and less expensive.

Fortunately, graph-theoretic algorithms have already been devised to address such challenges. In particular, we use a version of Dijkstra's S3P algorithm (described in [5]). However, rather than computing the shortest paths from a source vertex to all others, we compute the single-*target* shortest paths. This is easily done by traversing edges backward.

Another departure from Dijkstra's algorithm is that it must be modified to work with multigraphs. In particular, its edge-selection policy now must consider multiple edges between any two vertices, when selecting a minimum-cost edge. It can be shown that this does not change the optimality of the algorithm, in the sense of generating lowest-cost shortest paths. We only provide a proof sketch, as we rely on the standard proof for the optimality of Dijkstra's algorithm [5, Section 25.2]. The optimality of Dijkstra's algorithm depends on a greedy step in which it chooses the lowest-cost edge from a vertex that has already been visited, to a vertex that has not been visited yet. The minor modification we introduced, such that the algorithm considers multiple edges, does not modify the result, which is a single lowest-weight edge from the current vertex.

This step in which edges are selected also touches on a final modification of Dijkstra's algorithm for our purposes. In theory, any ties in alternatives (i.e., edges with same weight) can be broken arbitrarily by the algorithm, since the selection will not affect optimality. In practice, however, we have found it useful to reduce *hops*, the number of edges that leads from a given robot to the leader. This is done by breaking ties in such a manner as to prefer edges that minimize hops

Using the modified Dijkstra's algorithm, a single edge is selected optimally for each robot but the leader. These edges form an SBC control graph to be executed by the robots, i.e., the algorithm induces a control graph $\mathcal{G}$ out of the monitoring multigraph $\mathcal{MG}$. Because each edge is specific to the robot in which it originates, the separation and bearing targets of each robot are individually tailored to the monitoring capabilities of the robot. This admits sensor-heterogeneous robots (see Section VI-A).

To deploy the formation, the robots are assumed to possess controllers capable of reaching and maintaining the separation and bearing target values. On one hand, the assumption of capable controllers makes this technique potentially useful for many different robots. On the other hand, this abstract view of the individual controllers also removes the possibility of addressing important details, such as movement constraints of the robots. We leave further exploration of individual controllers—and their effect on the control graphs—to future investigations.

IV. Dynamic Switching of Control Graphs

The generation of SBC control targets for each robot is done automatically, based on the *expected* cost of using the robots sensors. However, during deployment, sensors may act differently from what was anticipated, due to catastrophic or intermittent failures. For instance, a camera may get stuck in a particular

Algorithm 1 GetVertexesToUpdate (multigraph G, robots $\mathcal{R}$)

1: $\mathcal{R}_k \leftarrow \emptyset$
2: $V \leftarrow \mathcal{R}$
3: **while** $\exists v \in V$ **do**
4: Remove v from V, put into $\mathcal{R}_k$
5: **for all** $e_{j,v}$, edges from robot j to v **do**
6: Insert vertex j to V
7: Return $\mathcal{R}_k$

angle, or lighting conditions may inhibit the ability to track specific colors.

To address this, we propose a distributed protocol that allows robots to dynamically switch control graphs while maintaining the formation. Such a protocol (by explicit or implicit communications) is required, to coordinate the robots in their switching of control graphs. Uncoordinated switches may result in two or more robots following each other, cyclically, instead of the leader. The protocol works in several steps

1) If a robot fails to monitor its preselected peer, or needs to update its cost estimates for its peers, it first broadcasts a message to all team members, to let them know that a recomputation of the graph is needed. During this phase, any number of robots may broadcast in parallel. Messages include revised edge costs. Note that such revised costs can reflect increases (e.g., due to sensor failures, as discussed next), as well as decreases (e.g., due to greater availability because of changes to body pose).
2) Each robot that receives the message halts the movement, and adds it to a local list of robots $\mathcal{R}$ that require reassignment of targets and sensors. The robot receiving the message will not report on any readjustment it wishes to make while within this step of the protocol.
3) All robots make sure that all messages have been received and processed. This can be done either by having receivers acknowledge received communications, or more simply (but less reliably) by having a timeout mechanism that ensures no new messages are generated.
4) All robots call on Algorithm 1 to determine the set $\mathcal{R}_k$ of robots in the team that are potentially affected (i.e., transitively) by a change in the initial list of robots' target assignments.
5) All reexecute the modified Dijkstra's algorithm on the monitoring multigraph $\mathcal{MG}$, to generate a new control graph. However, because only the subset of team-members $\mathcal{R}_k$ is affected (by increases or decreases in edge costs), decisions for other robots do not have to be revisited.

The GetVertexesToUpdate algorithm computes all robots that are *upstream* from an affected robot, where upstream is taken to mean traversing the edges in the multigraph backward, from the leader to the outmost followers. The algorithm essentially follows all edges backward, from the initial set of robots, adding additional affected robots as it goes. It halts when no new affected robots can be discovered.

The protocol shown earlier can be executed in parallel by all teammembers, or using a centralized computation that will distribute the result. When executed in parallel, care must be taken to ensure that 1) the robots begin their decision-making in a synchronized manner (i.e., work on the same initial list of robots $\mathcal{R}$); 2) arrive at the same choices in the recomputation of the new control graph. The first requirement can be enforced in several ways. We chose to enforce it by introducing a timeout mechanism: Once a robot announces that a recomputation is necessary, other robots have a certain time period in which they can add to the list. To prevent parallel execution of Dijkstra's algorithm from making different decisions, any ties are arbitrarily broken by preferring the robot with lower identifying number (ID).

The reliance on communications to synchronize the set of affected robots and revised edge costs raises several challenges with respect to communications. Most importantly, the protocol for dynamically switching between control graphs, as proposed in this section, can clearly be sensitive to communication failures. For instance, loss of a message announcing a revised edge cost (and the need for a recomputation) may cause the formation to break, as the sender might stop while the intended receivers, never having received the message, continue to move on. More reliable protocols may be used (e.g., requiring acknowledgment of any message, and retransmission after a time-out), but they may affect overall performance, and bandwidth usage. We empirically evaluate the effects of message loss on the dynamic switching method, in the next section. We also note in passing that, of course, the calculation of a static sensor-optimal control graph is not reliant on such communications.

Another challenge raised by the reliance on communications is the issue of range and bandwidth usage. In environments with limited bandwidth, and/or limited range of communications, the protocol described earlier may become unsuitable for use. For instance, the leader and one of the last followers, some edges (hops) away, may be too far apart to communicate directly. As a result, a message by the follower, informing of a change in edge cost, may fail to directly reach the leader. For such cases, the communication layer used by the robots would have to be revised to allow for indirect communications (i.e., by routing). Such a mechanism falls well beyond the scope of this paper, and we raise the issue here to emphasize the reliance of dynamic switching on the communication capabilities of the robots.

V. Evaluation

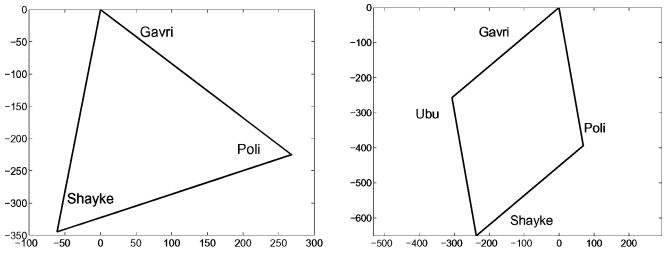
To evaluate the use of monitoring multigraphs in practice, we first show a series of experiments on physical robots (Sony AIBOs), in which automatically-generated, static control graphs are used (Section V-A). The results show that static (fixed), non-switching, control graphs can result in diverse performance quality. Then, we show that the use of the dynamic switching of control graphs solves this problem: In extensive experiments, dynamically-switching formations proved more robust and better-performing than a static control graph formation (Section V-B).

A. Static Control Graphs

The first set of experiments uses static control graphs, generated from the monitoring multigraphs, to control formations of Sony AIBO ERS-7 robots. Each of these robots has a single

TABLE II
SENSOR SPECIFICATION, AIBO

Attribute	Range	Cost
Distance	$[200, 1500]$	0.4
Field of View	$[-35, 35]$	0.5
Pan	$[-90, -30)$	0.7
	$[-30, 30]$	0.2
	$(30, 90]$	0.7



(a) (b)

Fig. 2. Ideal formations in static control graph experiments. Robot names are shown. (a) Ideal triangle. (b) Ideal diamond.

camera on its panning head that can be used to detect color blobs in its $\approx 120^\circ$ view-field ($[-59^\circ, 59^\circ]$), although practically, the effective view-field is ($[-35^\circ, 35^\circ]$). Using such color identification, the robot can identify others, when appropriately color marked. The head also contains an infrared range sensor that can measure distances in the range $[200_{\mathrm{mm}}, 1500_{\mathrm{mm}}]$, with some uncertainty. We treat the head (camera and distance sensor) as a single logical sensor, providing bearing and distance to another robot (identified in the camera image, at computable angle to the body of the observer). The head pans 90° left and right, and thus, the maximal practical angle range for its vision, when combined with the practical field of view of $[-35^\circ, 35^\circ]$, is $[-125^\circ, 125^\circ]$. However, in our experience, maintaining the pan angle in the range $[-30^\circ, 30^\circ]$ tends to produce better results. Based on the manufacturer's specifications, and our experience with the robot, we generated a sensor cost specification for the robot, based on reliability (Table II). Using this table, we used our technique to produce alternative control graphs for the triangular formation specified in Fig. 2(a), and the diamond formation in Fig. 2(b).

We discuss the triangle formation first. In the first (normal) case, the resulting SBC formation had Gavri as the leader (see Fig. 2 for robot names), and the other two robots monitoring it directly [Fig. 3(a)]. To experiment with different monitoring rules, we modify the sensor specification table for one of the robots (Shayke), such that the cost of panning in the range $[50^\circ, 90^\circ]$ is 0.7, and infinity anywhere else (forcing Shayke to look sideways, with only 40° of leeway). This case simulates, for instance, a failure where Shayke's camera pan motor is stuck. Providing this modified input to our algorithm produces a different monitoring graph, where Shayke is monitoring robot Poli, which, in turn, monitors robot Gavri [Fig. 3(b)].

In the diamond formation, many control graphs are possible in principle. For the purposes of the experiments, we have restricted the algorithm to control graphs in which the last robot, Shayke, is the only one to select different targets. This is done by tweaking its associated cost table, in effect rendering it heterogeneous from its peers. We experimented with three control graphs: Shayke monitoring the leader [Fig. 3(c)], the right follower [Fig. 3(d)], and the left follower [Fig. 3(e)]. All control graphs were generated automatically.

We ran 15 trials with each of these alternative SBC formations (a total of 75 trials). In these trials, the leader was controlled manually to determine an obstacle-free straight line of about 6 m in length. The objective was to contrast the stability and robustness of the different control graphs under reasonable operating conditions.

Fig. 4 shows the resulting formations, as represented by the average positions of robots with respect to each other. Fig. 4(a) shows the two triangle formations, while Fig. 4(b) shows the three diamond formations. Each figure also plots the ideal formation for comparison. The formations are shown in an X, Y coordinate system measuring millimeters. For the purpose of comparison, the leader is positioned at $(0, 0)$.

Qualitatively, it can be seen that there exist large variances in the quality of the formations when maintained statically by different control graphs. In the triangle formation, the control graph in which Shayke monitors the leader directly yields much better formation maintenance than the control graph in which Shayke is monitoring the leader indirectly, through another robot. This is likely the result of latency in the responses of Shayke to the movements of the leader, as they are filtered by the intermediate robot's actions and perceptions.

However, in the diamond case, the reverse is true. Here, two control graphs prove to yield good results; both of these control graphs monitor the leader indirectly. In contrast, the control graph in which Shayke monitors the leader directly shows that the formation is not maintained. We believe that this is due to the effective sensor range of the robot, which causes Shayke to believe that the leader is farther than it really is, thus leading Shayke to move more quickly to get closer to the leader.

B. Dynamically Switching Control Graphs

The principal lesson from the first set of experiments is that static formation control graphs can lead to markedly different results. We thus wanted to evaluate the ability of dynamically-switching control graphs to compensate for such limitations, and yield better and more robust formations.

To experiment with this, we reexecuted the diamond formation experiments, contrasting a static control graph with that of a dynamically-switching control graph, using the switching protocol described in Section IV. In both cases, the robots began with the same control graph, but in the dynamic cases, they were allowed to switch to a different target. To control and trigger such switches, we varied a threshold in the vision system, determining whether a target was lost. A lower threshold causes the vision system to quickly give up on a target, simulating noisy sensing conditions (in which a target would often be lost). A larger threshold causes the vision system to wait to reacquire the target before giving up. The numbers actually denote the number of consecutive frames in which the target was not identified. We used values of 4, 20, and 40. As the typical frame-rate was 28–30 frames/s, this translates into giving up after approximately 140 ms, 690 ms, or 1370 ms, respectively.

(a) (b) (c) (d) (e)

Fig. 3. AIBO robots executing static triangle (a, b) and diamond (c–e) SBC control graphs. Note bottom robot (Shayke) head pose. (a) Shayke follows leader. (b) Shayke follows Poli. (c) Shayke follows leader. (d) Shayke follows Poli. (e) Shayke follows Ubu.

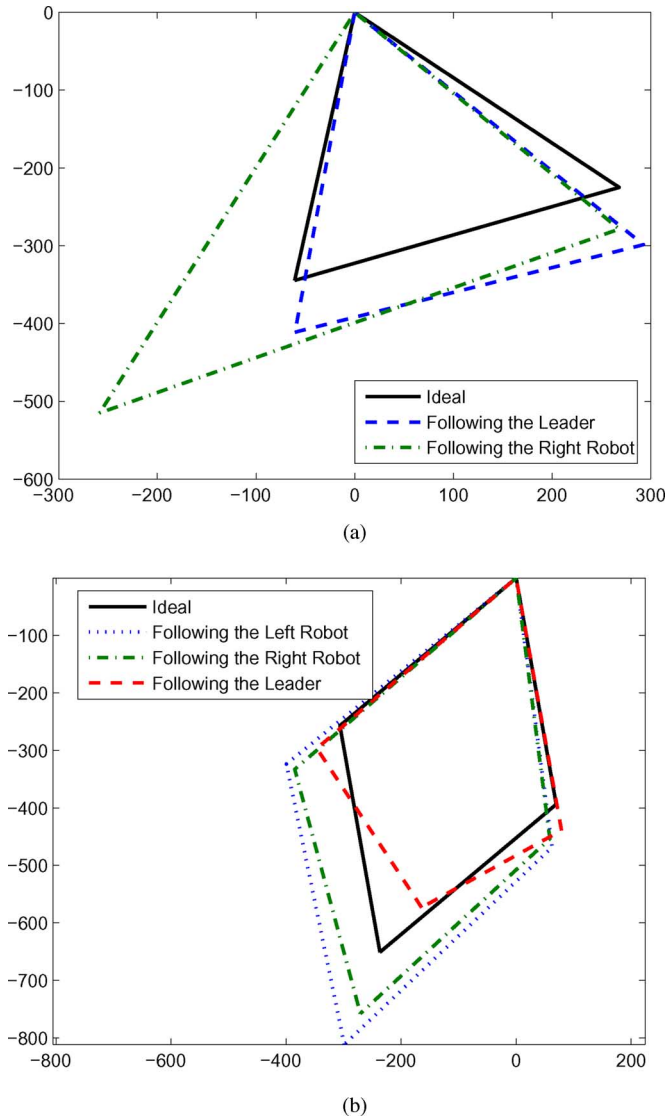


Fig. 4. Ideal and actual robot positions. (a) Triangle. (b) Diamond.

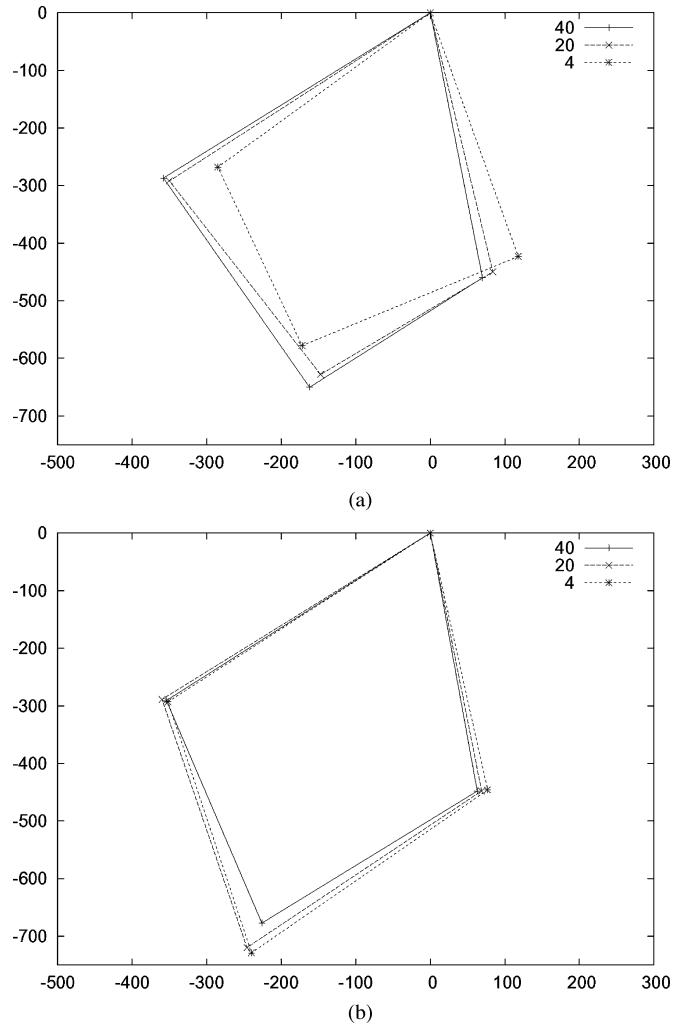


Fig. 5. Average Formation. (a) Static control graph. (b) Dynamic control graph.

Each of the three noise settings was repeatedly tried with a static and dynamic control graph. Each configuration was repeated 15 times, for for a total of 90 trials (45 with the static control graph, 45 with the dynamic control graph). In the static control graph settings, all robots used the leader as the target. Unlike the previous set of experiments, the velocity of the leader was constant, and thus, shorter time for finishing the course indicates improved performance.

Fig. 5 shows the average positions of robots in the diamond formation, in the case of static and dynamic control graphs, for each value of the threshold. As can be seen in the figures, the dynamic control graph yields results that are 1) more consistent

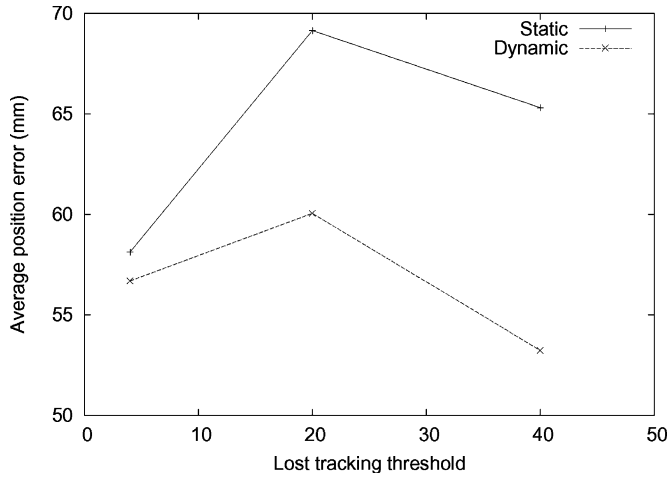


Fig. 6. Position errors.

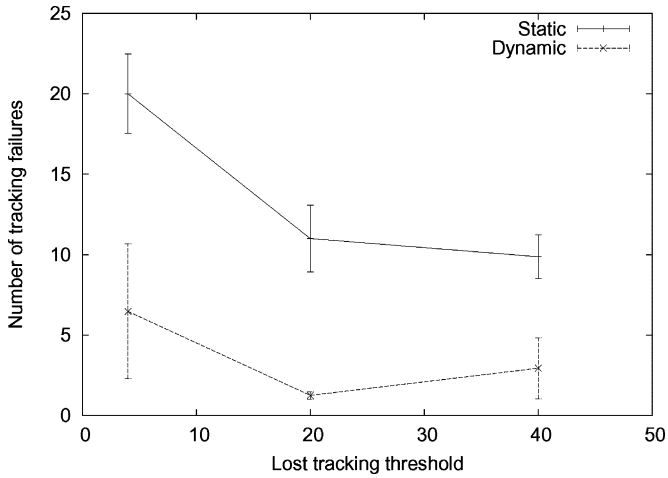


Fig. 7. Number of target tracking losses. Error bars show one standard deviation.

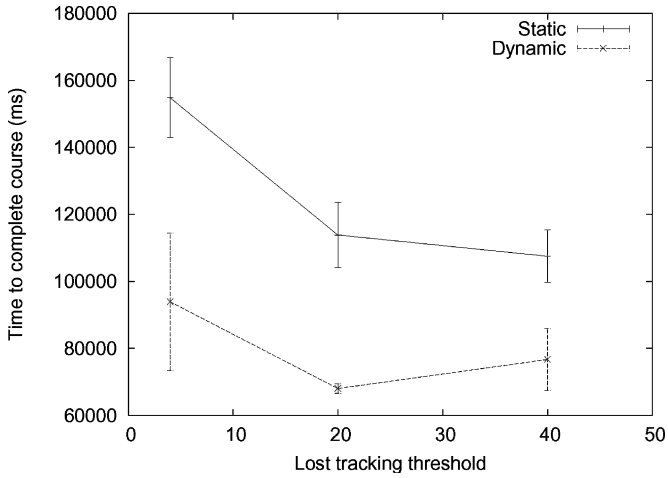


Fig. 8. Time to complete course. Error bars show one standard deviation.

across the experimental conditions and 2) closer to the ideal formation form.

Fig. 6 provides quantitative analysis of the same phenomenon. Here, the *X*-axis measures the different threshold settings, simulating different perception errors, as described earlier. For each of the robots, we recorded its actual trajectory, and compared it to the ideal trajectory with respect to the leader. This allows us to measure, for each robot, its actual average position in the formation. The *Y*-axis measured the average error in position of the robots with respect to the ideal formation, i.e., the distance between their actual positions, and their ideal positions. The two curves show the average position errors in the static and dynamic control graph techniques. As can be seen in the figure, in all three conditions, the dynamic switching technique leads to smaller errors.

The reason for this difference between the static and dynamic control results is that in the static control settings, the robots cannot switch to a different target, and are thus, more significantly affected by noise settings. In particular, a lower threshold (simulating higher noise) causes followers to lose their targets more often. As a result, they often have to stop, and reacquire their targets. This frequent stopping and starting ends up causing the formation to be deformed. In different noise settings, this does not happen as often, and thus, the formation is not as deformed. In contrast, in the dynamic control settings, the robots adjust the control graph in response to frequent failures to track their target, until they settle on a control graph that is more stable (in terms of tracking). Thus their formation is not deformed, regardless of the noise settings.

Fig. 7 shows the number of tracking failures in the static and dynamic settings. The *X*-axis shows the tracking failure thresholds, as described earlier. The *Y*-axis shows the number of tracking failures, i.e., the number of times the thresholds were passed. In the static case, each such tracking failure would result in the formation stopping to let the robots reacquire their lost target. In the dynamic case, the formation would stop to let the robots switch (acquire) a different robot. The figure shows that the number of tracking failures is significantly reduced in the dynamic case. This is because after switching, tracking is much more successful, and thus, very few additional tracking failures occur.

We examine additional performance measures. Fig. 8 shows the time that it took the formation to finish the course. The *X*-axis shows the different threshold settings, simulating different perception errors. The *Y*-axis measures the time; smaller values are better. The figure shows that the dynamically-switching technique leads to significantly smaller durations for finishing the task. A two-tailed *t*-test (assuming unequal variances) of this data results in a null-hypothesis probability value $p < 0.0001$ in all settings. Here again, the cause for the greater time to complete the course, in the static case, is due to the frequent stopping and restarting of the formation, which occurs whenever the robot loses its target and must reacquire it.

Finally, we examine the percentage of time the formation was maintained, i.e., both angles and distances were within their tolerance levels (10°, 15 cm). In Fig. 9 the *Y*-axis shows the percentage of time, and thus, larger values are better. The figure shows, two benefits to the dynamic-switching approach. First, the percentage of time the formation was maintained was significantly higher than with the static control graph (a two-tailed

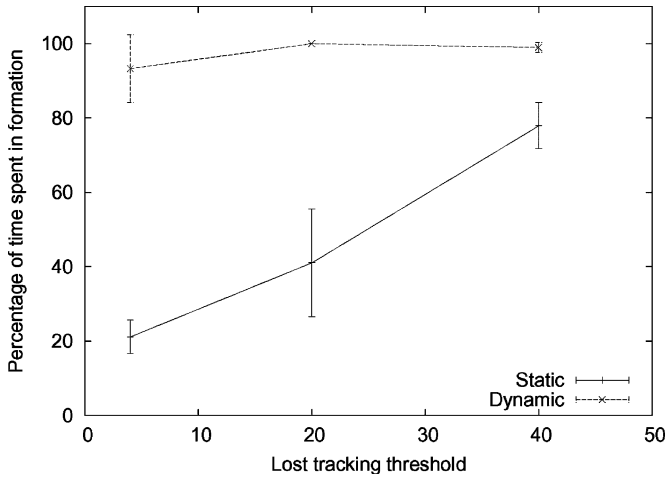


Fig. 9. Percentage of time formation maintained. Error bars show one standard deviation.

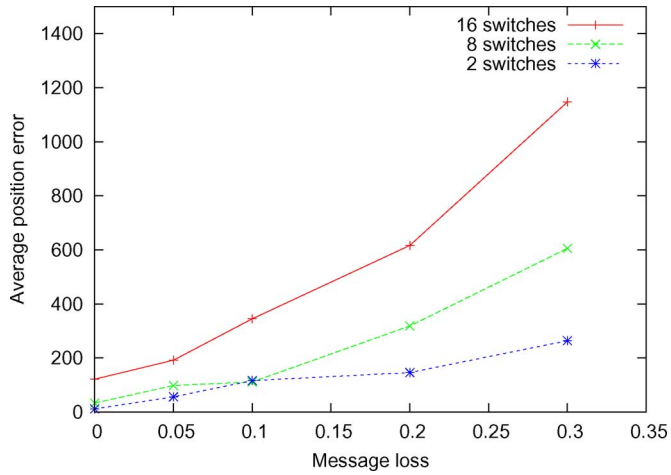


Fig. 10. Position errors (in millimeters) due to message loss, for different message loss rates, and different rates of switching (requiring communications).

t-test assuming unequal variance shows $p < 2.40109 \times 10^{-15}$). Second, and perhaps more importantly, the percentage essentially remains constant despite the significant change to environmental conditions. This is in contrast to the static control graph approach, whose performance was lower, and also inconsistent across the controlled conditions.

C. Dynamic Switching and Communication Reliability

The previous section has demonstrated the efficacy of dynamic switching over static control graphs. However, dynamically switching between control graphs is a procedure that relies on reliable communications, as previously discussed. Here, we evaluate the effects of message loss, a common cause for unreliability, on formations utilizing dynamic control.

To carry out this evaluation, we utilized simulated AIBO robots in the player-stage API [10]. The robots were placed in a diamond formation, facing in the direction of the movement. A fixed 5 m course was to be traversed by the robots, as measured by the last robot in the diamond formation. We then tested several configurations of two independent variables: First, we controlled the rate of message loss by randomly dropping point-to-point messages at a rate of 0%, 5%, 10%, 20%, and 30%; 0% message loss implies perfect communications (as earlier), and 30% message loss means almost a third of all messages are lost. Second, we controlled the number of times the switching procedure was triggered (i.e., the opportunity for utilizing dynamic switching). We tested with 2, 8, and 16 switches of the last robot in the formation, within the 5 m course). For all 15 combinations, we carefully measure the average position error of the robots as they finish the course, i.e., the average of the distances between each of the robots' actual (ground truth) locations, and their ideal locations according to the formation shape. Each of the 15 combinations was run 15 times. Robots utilized an approximate 800 ms timeout to wait for an acknowledgment of each sent message (the exact value varied somewhat, as it was tied to the perception rate of the robots in question). If the acknowledgment was not received (e.g., because the sender did not receive the message triggering the acknowledgment, or because the acknowledge message itself was lost), the sender would retransmit the message. During such times, the formation would be deformed, as robots that did not receive a message to recompute the graphs might continue to move.

Fig. 10 shows the results for each combination, averaged over 15 trials. The X-axis shows the message loss rate. The Y-axis shows the average position error in millimeters. Each of the curves corresponds to a different setting of the number of switches (2, 8, and 16). Two immediate results are evident: First, that given a message loss rate (i.e., a specific value on the X-axis), greater opportunity for communications generally leads to greater position errors. Second, that given a fixed number of switches (i.e, a curve), greater message loss leads to greater errors. Thus, one conclusion is that indeed the dynamic switching technique is very much affected by the reliability of communications; the more communications are required, the greater the effect of message loss on the formation.

However, a more subtle qualitative conclusion may be drawn from these results. None of the curves show linear increase in the position errors. Instead, the position errors seem to increase nonlinearly with message loss rates. For low message loss rates (up to 5%), there seems to be only slight degradation in the performance of the formation for each curve. At higher message loss rates (e.g., above 20%), the effect on position error is dramatic. This suggests that the simple protocol we utilized (acknowledgment with timeout) is capable of handling only limited message losses, and a more robust protocol should be sought. Thus, while generally, dynamic switching assumes reliable communications, it would be more accurate to say that dynamic switching is reliant on the protocol used to exchange messages.

Indeed, a second experiment demonstrates that dynamic switching is sensitive to the choice of protocols. Here, we repeated the configuration of 30% message loss, with 16 switches along the 5 m course (the most challenging of the communication experiment settings). However, in these runs, we changed the protocol slightly, to cut in half the time-out (waiting for an acknowledgment on a sent message). The intuition here is that

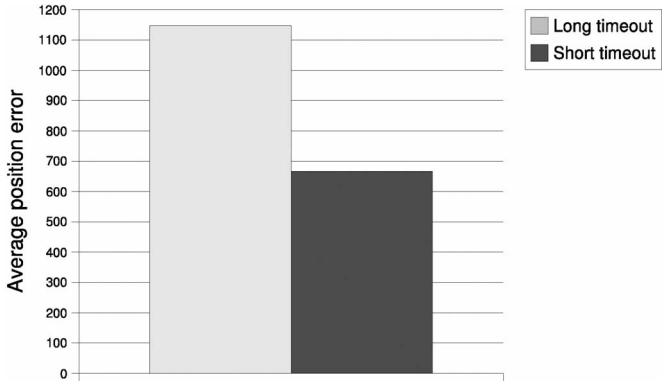


Fig. 11. Average position error (in millimeters) with normal (light) and shortened (dark) time-out in the communication protocol.

if robots wait less time for peers to acknowledge receipt of a message, they would more quickly resend the message, and thus be more quick to overcome message losses. This, in turn, would lead to shorter opportunities for robots to continue to move, despite their peers halting (to recompute the control graph).

Fig. 11 shows the results from these experiments. The *Y*-axis marks the average position errors in millimeters. The two bars mark the results for the time-out used in the previous experiments (here, called "long time-out"), and the use of a shorter time-out (half the longer one). The figure shows that by cutting the waiting period, robots are able to reduce the formation position errors, as the formation is less deformed.

These results support the conclusion that the efficacy of dynamic switching, in the presence of message loss, is dependent on the reliability and speed of the communication protocol utilized to trigger the recomputation of the control graph. The exploration of novel communication protocols is beyond the scope of this paper. We thus leave this to future work.

VI. Discussion

The techniques presented (and evaluated) in the previous sections carry several implications for the design and deployment of multirobot formations. In this discussion section, we raise several of these implications and attempt to discuss the opportunities and challenges offered by the techniques presented.

A. Multigraphs for Heterogeneous Teams

Not all robots are created equal—at least not in terms of their sensor morphologies. One key opportunity raised by the technique we present is for automatically generating SBC control graphs for heterogeneous teams. Because the control graphs are automatically generated from the sensor-morphology descriptions of each robot, it is possible, using the algorithms in this paper, to generate control graphs that are optimized for each individual robot. For instance, the techniques in this paper would automatically generate SBC control graph for a triangular formation composed of one robot with an omnidirectional camera (with a shorter range), one Sony AIBO robot (with a limited panning capability, and limited field of view), and a Pioneer robot using an onboard pan-tilt-zoom camera with a 180° field of view.

TABLE III
Type 2 Robot Sensor

Attribute	Range	Cost
Distance	$[0, 100]$	0
	$(100, 800]$	0.1
Field of View	$[-36^\circ, 36^\circ]$	0.1
Pan	$[-5^\circ, 5^\circ]$	0.1

TABLE IV
Type 3 Robot Sensor

Attribute	Range	Cost
Distance	$[0, 310]$	0.4
	$(310, 675]$	0.7
Field of View	$[-30^\circ, 30^\circ]$	0.2
Pan	$[-90^\circ, -5^\circ)$	0.6
	$[-5^\circ, 5^\circ]$	0.4
	$(5^\circ, 90^\circ]$	0.6

TABLE V
Type 4 Robot, Multiple Sensors

Sensor	Attribute	Range	Cost
1	Distance	$[0, 400]$	0.1
		$(400, 900]$	0.2
	Field of View	$[-20^\circ, 20^\circ]$	0.3
	Pan	$[0^\circ, 10^\circ]$	0.1
2–8	Distance, F.o.V	Same as above	
	Pan°	$[-180, -170], [-82, -72]$	
		$[-36, -26], [36, 46], [72, 82]$	0.1
		$[108, 118], [114, 124]$	

We demonstrate the use of the technique in producing control graphs for heterogeneous teams of robots, where hypothetical robots vary in their sensor morphologies. We define three hypothetical types of robots, with which we create a variety of formations. The sensor specification for these appears in Tables 1 (Type 1) and III–V (Types 2–4). All tables follow the same format as that of Table 1. Table V reports on multiple sensors. The details are shown for the first sensor, and the rest replicate its distance and field of view ranges, though at different panning values.

These hypothetical robots are not meant to necessarily correspond to any existing realistic robots. Rather, we defined their sensor morphologies to demonstrate a range of sensor-optimal control graphs that emerge, given different combinations of robots with diverse sensor morphologies. For example, Table IV describes a robot that can more easily pan its camera forward (within a 10° angle, -5° to $+5^\circ$) than to the sides, and cannot at all turn it backward. The camera's field of view is 60°-wide, and the effective distance measurement length is 0–310 mm. Slightly further, it is still possible to measure distance, but at a larger cost (stemming from the decreased accuracy of the sensor's measurements).

We have experimented with different formations and different combinations of these robots, and produced monitoring multigraphs and resulting formation graphs. All graphs were produced automatically, using the algorithms described earlier. These show the diversity of control graphs resulting from the use of the algorithms described earlier.

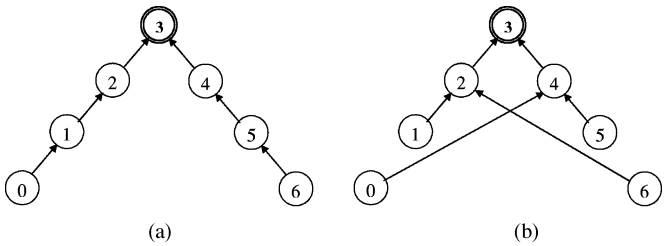


Fig. 12. Triangular, seven robots. (a) All type 3. (b) All type 4.

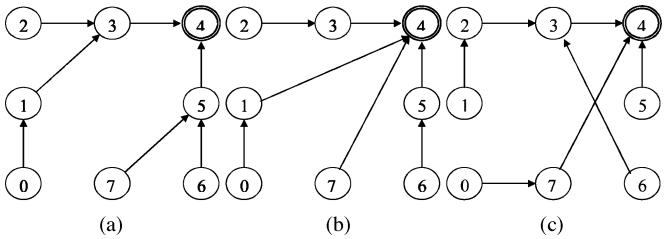


Fig. 13. Square, eight robots. In (c), the robots in positions 0,2,5 are of type 1, positions 1,4,6,7 are of type 2, and position 3 holds a robot of type 3. (a) All type 1. (b) All type 3. (c) Mixed types 1,2,3.

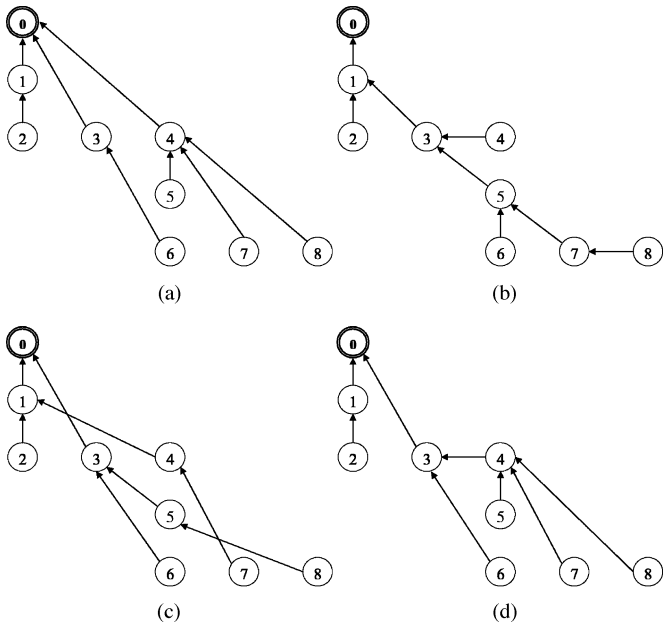


Fig. 14. Stairs, nine robots. In (d), positions 1,4 have robots of type 1; positions 0,2,5,7 hold robots of type 2; positions 3,6 hold robots of type 3, and finally position 8 holds a robot of type 4. (a) All type 4. (b) All type 1. (c) All type 3. (d) Mixed all types.

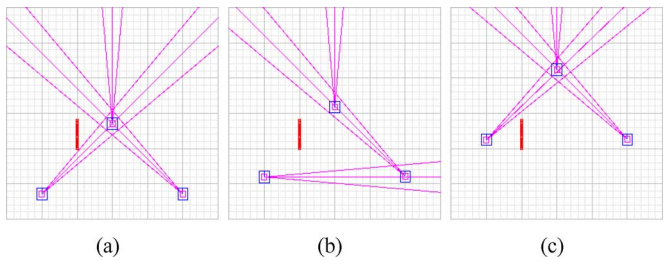


Fig. 15. Triangle formation overcoming an obstacle. The obstacle is a short wall (vertical in the images earlier), which can interfere with the left robot's monitoring of its teammates.

Fig. 12 shows the formation graphs for triangular formation with seven robots, of type 3 [Fig. 12(a)] and type 4 [Fig. 12(b)]. In the latter, given the possibility of using a sensor at a pan angle unavailable to the first, it became cheaper for a trailing robot to directly monitor a robot farther from it, but only once removed from the leader, as opposed to monitoring a closer robot that is much more removed from the leader. This shows the importance of using individual monitoring cost as the optimality criterion, and thus, the choice of the Dijkstra's algorithm.

Fig. 13 shows the results for a square formation, with eight robots. Here, we show both homogeneous teams [Figs. 13(a) and (b); all robots of same type], and nonhomogeneous teams [Fig. 13(c)].

Finally, we demonstrate the use of the technique with the complex stairs formation, with nine robots. Fig. 14(a)–(c) show homogeneous teams (type 4, 1, and 3, respectively). Fig. 14(d) shows the result of using a mixed team.

We note that the allocation of robots to different positions within the formation was made arbitrarily in these examples: First, the robots were assigned to roles, and only then was the algorithm run to generate the control graph. It is possible, however, to utilize the algorithms of Lemay *et al*. [13] and Michaud *et al*. [14], to generate an improved assignment of roles, even prior to execution of our algorithms.

B. Obstacle Avoidance

The dynamic switching technique does not specifically address obstacles, any more than other SBC methods. However, under some conditions, dynamic switching can, in fact, result in somewhat greater robustness to obstacles. To demonstrate this, we carried out a simple experiment in the player-stage simulation environment [10], shown in Fig. 15.

The figure shows a triangle formation overcoming an obstacle to visibility (but not to movement), by relying on dynamic switching. In (a), the left robot is about to lose visibility of leader, as the obstacle is about to enter its field of view, and occlude the leader. In (b), it switches to monitoring the right robot, in response to losing sight of the leader. In (c), it switches back to monitoring the leader, in response to losing sight of the right robot. Note that no static control graph could have overcome the visibility constraints imposed by the obstacle.

This demonstration shows that using dynamic switching may result in additional robustness, under specific conditions. However, an in-depth exploration of these conditions and the scope of the obstacle avoidance characteristics of dynamic switching is beyond the scope of this paper.

C. Sensor Interference

The issue of sensor interference is rarely, if ever, raised in the context of multirobot formations. We believe that one reason for this is that, in most cases, SBC control graphs are generated by hand, by a designer who takes sensor interference into account. However, when automatically generating control graphs—as

we do in this paper—sensor interference should ideally be addressed automatically, at least to some extent.

In general, sensor interference can occur when a combination of sensors—here employed by different robots—is activated in parallel, and their measurements actively interfere with each other. For instance, ultrasonic (sonar) sensors may suffer from interference when the signals generated by one sensor bounce back from the target, and are picked up by another sensor. Careful timing of the sensing operations is one solution approach to this problem. In general, however, sensor interference is difficult to model in a sensing multigraph, as we construct in this paper. In particular, this is because sensor interference is dependent on two (or more) sensors being active at once, and thus, requires modeling the dependency between edges in the multigraph, contrary to the (implicit) assumption of independence between edges, which is used by the extended Dijkstra algorithm. Modeling edge dependence seems to require a different representation, and a significant change to the algorithms presented earlier.

However, at least a limited possibility exists to address sensor interference using the current technique. In particular, just as the multigraph construction procedures eliminates edges that are realistically impossible due to robots occluding each other, it is possible to add additional edge-removal heuristics, which would remove edges corresponding to sensing modes that have a high risk of interference, such as sensing modes that cross each other.

In practice, we did not experience sensor interference issues in the formations and Sony AIBO robots used in the experiments of this paper, despite the use of infrared sensors for measuring distance. We, thus, chose to leave this challenge for future work.

VII. Summary

We presented a novel representation and algorithms for automatically generating multirobot formation control graphs, based on directed weighted monitoring multigraphs. We have shown that the approach allows the use of graph-theoretic techniques, to: 1) optimally account for sensor morphology and constraints in generating distributed formation-maintenance SBC controllers and 2) allow robots to dynamically switch formation control graph for added robustness. We demonstrated the use of the technique in systematic experiments with AIBO robots, and have shown in these experiments that the use of our techniques leads to significant increases in both performance and robustness to environmental conditions. In addition, we discuss a number of challenges and opportunities in multirobot formations, raised by the techniques we presented, and open for future research.

Acknowledgment

We thank N. Agmon, L. Parker, and the anonymous reviewers for useful comments, and D. O'Leary for asking inspiring questions. As always, K. Ushi deserves special thanks for her support.

References

[1] S. Abrams, P. K. Allen, and K. Tarabanis, "Computing camera viewpoints in a robot work-cell," *Int. J. Robot. Res.*, vol. 18, no. 3, pp. 267–285, 1999.

[2] T. Balch and R. Arkin, "Behavior-based formation control for multi-robot teams," *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 926–939, Dec. 1998.

[3] T. Balch and M. Hybinette, "Social potentials for scalable multirobot formations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA-2000)*, vol. 1, pp. 73–80.

[4] S. Carpin and L. Parker, "Cooperative leader following in a distributed multi-robot system," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 2994–3001.

[5] T. T. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1990.

[6] J. P. Desai, "A graph theoretic approach for modeling mobile robot team formations," *J. Robot. Syst.*, vol. 19, no. 11, pp. 511–525, 2002.

[7] J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Trans. Robot. Autom*, vol. 17, no. 6, pp. 905–908, Dec. 2001.

[8] R. Fierro, A. K. Das, V. Kumar, and J. P. Ostrowski, "Hybrid control of formations of robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA-2001)*, vol. 1, pp. 157–162, 2001.

[9] J. Fredslund and M. J. Mataric, "A general algorithm for robot formations using local sensing and minimal communications," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 837–846, Oct. 2002.

[10] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc. Int. Conf. Adv. Robot.*, Coimbra, Portugal, 2003, pp. 317–323.

[11] G. Inalhan, F. Busse, and J. How, "Precise formation flying control of multiple spacecraft using carrier-phase differential GPS," presented at the *AAS/AIAA Space Flight Mech.*, San Diego, CA, 2000.

[12] G. A. Kaminka and Y. Elmaliach, "Experiments with an ecological interface for monitoring tightly-coordinated robot teams," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA-2006)*, pp. 200–205.

[13] M. Lemay, F. Michaud, D. Létourneau, and J.-M. Valin, "Autonomous initialization of robot formations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA-2004)*, vol. 3, pp. 3018–3023.

[14] F. Michaud, D. Lćtourneau, M. Gilbert, and J.-M. Valin, "Dynamic robot formations using directional visual perception," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2002, vol. 3, pp. 2740–2745.

[15] A. I. Mourikis and S. I. Roumeliotis, "Optimal sensor scheduling for resource constrained localization of mobile robot formations," *IEEE Trans. Robot.*, vol. 22, no. 5, pp. 917–931, Oct. 2006.

[16] D. J. Naffin and G. S. Sukhatme, "Negotiated formations," in *Proc. Eigth Conf. Intell. Auton. Syst. (IAS-8)*, Amsterdam, The Netherlands, 2004.

[17] K. Yoon and C. Hwang, *Multiple Attribute Decision Making: an Introduction*. Thousand Oaks, CA: Sage, 1995.

Gal A. Kaminka (M'06) received the Ph.D. degree from the University of Southern California, Los Angeles.

He had a 2-year stint as a Postdoctorate fellow at Carnegie Mellon University, Pittsburgh, PA. He is currently a Senior Lecturer in the Department of Computer Science, Bar Ilan Universiy, Ramat Gan, Israel, where he also leads the MAVERICK Group, supervising 15 M.Sc. and Ph.D. students. His current research interests include teamwork and coordination, multiagent and multirobot systems, behavior and plan recognition, and modeling social behavior.

He was the recipient of an IBM Faculty Award and top places at international robotics competitions.

Ruti Schechter-Glick received the M.Sc. degree from the Department of Computer Science, Bar Ilan University, Ramat Gan, Under the guidance of Israel Dr. Gal Kaminka of the MAVERICK Group.

Currently, she is a Software Engineer at NDS Ltd., Jerusalem, Israel. Her current research interests include multi-robot systems and artificial intelligence.

Vladimir Sadov is currently working toward the M.Sc. degree in the Department of Computer Science Bar Ilan University, Ramat Gan, Israel. His current research interests include humanoid robots and multi-robot systems. He is currently a member of the MAVERICK Group, under the direction of Dr. Gal Kaminka. He is also a freelance Software Engineer.