Towards Robust Multi-Robot Formations

Gal A Kaminka and Ruti Glick*
The MAVERICK Group, Department of Computer Science
Bar Ilan University, Israel
{galk, glickr}@cs.biu.ac.il

Abstract— **Robots in formations move while maintaining a predefined geometric shape. Previous work has examined formation-maintenance algorithms that would ensure the stability of the formation. However, for each geometric formation, 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 for robustness. This paper presents a monitoring multi-graph framework for formation controller selection, based on sensor-morphology considerations. We instantiate framework, and present two contributions. First, we show that graph-theoretic techniques can then be used to compute sensing policies that maintain a given formation. In particular, sensor-based control laws for separation-bearing (distance-angle) formation control can be automatically constructed. Second, we present a protocol allowing controllers to be switched on-line, to allow robots to adjust to sensory failures. We report on results from comprehensive experiments with physical robots. The results show that the use of the dynamic protocol allows formations of physical robots to move significantly faster and with greater precision, while reducing the number of formation failures.**

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., [1], [3], [5], [7], [8]). 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 [3].

Previous work has examined constraints on a given control-graph, that would ensure the stability of the formation. In particular, one popular method for such control is *Separation-Bearing Control* (SBC) [1], [3]–[5] 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 each geometric formation, an exponential number of stable possible control graphs exists [3]. Thus a key question is how to select (construct) a control graph that optimizes other desired properties than 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 backwards (relative to the direction of movement) is likely less 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)

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 multi-graph*, in which directed, weighted, edges, denote monitoring capabilities (sensors or communications) and their associated costs. 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 on-line, to allow robots to adjust to permanent and intermittent sensory failures.

To evaluate these contributions, the monitoring multi-graphs framework has been fully implemented for Sony AIBO robots. We show the results of extensive experiments, demonstrating the robustness of the formations resulting from using monitoring multi-graphs. We show empirically that use of the framework leads to significantly increased precision, better performance, and robustness to changing environmental conditions.

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 algorithms. 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 work addresses these challenges. Other differences with existing work are discussed below.

Fierro et al. [4] analyzed the stability of SBC and other controllers, and proposed using manually-constructed control laws to allow up to three robots to switch between alternative controllers on-line without relying on communications. Our work complements their results by providing (i) a method for optimal selection of alternative control graphs, for any number of robots; (ii) a protocol using communications for making synchronized control-graph decisions.

Desai [3] shows how an un-weighted control graph describes the monitoring from a global perspective. Desai does

*This work was supported in part by Israel's Ministry of Science and Technology.

not discuss selection of a control graph, and assumes omni-directional sensing. However, Desai discusses switching formations (and their associated control graphs) to tackle terrain changes. Our framework complements such switching.

Balch and Arkin [1] examine three techniques for formation maintenance of up to 4 homogeneous robots. Two of these (*Leader-Referenced* and *Neighbor-Referenced*) techniques are essentially SBC controllers, using fixed control graphs. Fredslund and Matarić [5] 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 allow for automated selection of the leader, and considers the unique sensor configuration of each robot.

Lemay et al. [7] and Michaud et al. [8] present a method for quantifying the cost of using the sensors to determine distance and angle to a neighbor. They present an algorithm where each robot broadcasts its distance and angle from its peers. Using this knowledge, each robot finds the target to which it has the smallest deviation in angle and distance. However, this is used only in selecting the formation positions of all robots. In contrast, our method uses cost information after the positions have been assigned, to select the optimal target for each robot. We use additional sensory cost factors to allow dynamic switching of control graphs.

III. Cost-Optimal Formation Control Graphs

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

A. Monitoring Multi-Graphs in Formation Control

We introduce the use of multi-graphs to represent the monitoring capabilities of robots in a multi-robot system. A *monitoring multi-graph* is a tuple $\langle V, E\rangle$ where V is a set of vertices, denoting robots, and E is a *bag* of weighted edges $\{\langle u, v, w\rangle\}$, each linking two vertices $u, v \in V$, and having a non-negative weight $w \in \mathbb{N}$. Since E is a bag, an edge linking two vertices may appear more than once.

Edges denote monitoring capabilities. An edge $\langle u, v, w\rangle$ exists if robot u is able to monitor (sense) robot v in some distinct fashion, i.e., through a specific sensor. The weight w indicates the cost of using the sensor. 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.,$\langle u, v, w\rangle \in E \nRightarrow \langle v, u, w\rangle \in E$.

In practice, most tasks require monitoring to be selective. A monitoring multi-graph can be useful in such reasoning, and allow 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.

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

TABLE I
Type-1 Robot Sensor Configuration.

We construct monitoring multi-graphs to represent the sensory capabilities of robots in the formation. Here, vertices (denoting robots) have an associated position which denotes the associated robot's position in the formation. The initial pose of all robots is to the direction of the movement. Previous work typically assumes that teams are homogeneous, and thus does not address preferences over allocation of robots to places. While we do not make such an assumption (see below), we leave allocation to future work.

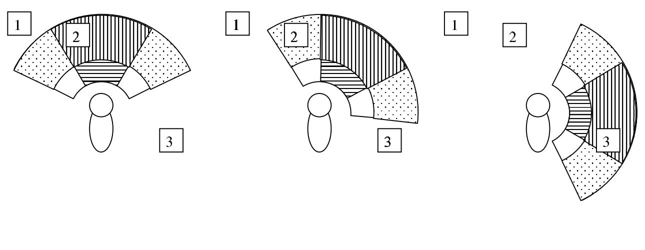
For each robot (vertex), we add edges by considering its sensors that can be used for monitoring other robots. We focus on sensors that can provide identification, distance, and bearing to other robots. For instance, combination of cameras and distance sensors. The weight of an edge is an indication 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, however we empirically found the following three factors to be useful in practice: sensing distance limits, 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.

For each factor, for each relevant range of values, we assign a cost. 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.

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

To contrast alternative sensing possibilities, we will edges from the monitoring robot to the other (monitored) robots. An edge will be created for each combination of distance, field-of-view and pan angle. This is done as follows.

First, we compute the area covered by a sensor, given



(a) Pan 0°. (b) Pan 30°. (c) Pan 90°.

Fig. 1. Monitoring possibilities change based on sensor panning.

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 $[-50 + -90, -30 + 90] = [-140, 60]$ degrees.

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 multi-graph correspond to geometric positions in the formation, the distance between to 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. For instance, In Figure 1 the left top robot is outside of the distance range of the monitoring. Thus there would be no edge from the monitoring robot to this left top robot. Figures 1-a,b, show multiple ways in which the robot ahead of the monitoring robot (and slightly to the left) 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 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. Thus any edges $\{\langle a, b\rangle, \langle b, a\rangle\}$ are removed from the monitoring multi-graph. 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 multi-graph 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 Control Laws

Now that the monitoring multi-graph 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-cost formations.

In SBC, a robot—called *leader*—is responsible for determining the overall global path (e.g., by deferring to a human operator [6], or using a path planner). Each of the other robots (*followers*) is given an individually-tailored control rule, restricting it to maintain a given distance and angle (with respect to the direction of movement) to its *target*—either the leader, or another follower—that in turn monitors its own target. Separation-Bearing control (SBC) thus relies on a single monitoring link for each robot.

We can induce the SBC monitoring rules from the monitoring multi-graph, 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 pre-determined Given the leader, a formation graph can be maintained using SBC under the following conditions:

1) The out-degree of the leader robot is 0.
2) The out-degree 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 above, 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 [2]). 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 backwards.

Another departure from Dijkstra's algorithm is that it must be modified to work with multi-graphs. In particular, its edge-selection policy now must consider multiple edges between any two vertices. It can be shown that this does not change the optimality of the algorithm. We only provide the proof sketch for lack of space: 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. Modifying this step such that it considers multiple edges does not modify its 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 multi-graph $\mathcal{MG}$. Because each edge is specific to the robot in which it origins, the SBC control law of each robot is individually tailured to the monitoring capabilities of the robot. This admits sensor-heterogeneous robots.

IV. Dynamic Switching of Control Graphs

The generation of an SBC control law 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 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 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 pre-selected peer, or needs to updates 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.
2) Each robot that receives the message halts the movement, and adds it to a local list of robots $\mathcal{R}$ that require re-assignment 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 messsages 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 re-execute the modified Dijkstra's algorithm on the monitoring multi-graph $\mathcal{MG}$, to generate a new control graph. However, because only the subset of team-members $\mathcal{R}_k$ is affected, decisions for other robots do not have to be revisited.

The GetVertexesToUpdate algorithm essentially computes all robots that are *upstream* from an affected robot, where upstream is taken to mean traversing the edges in a control graph backwards, 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.

Algorithm 1 GetVertexesToUpdate (control graph 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 $\mathcal{R}_k$
```

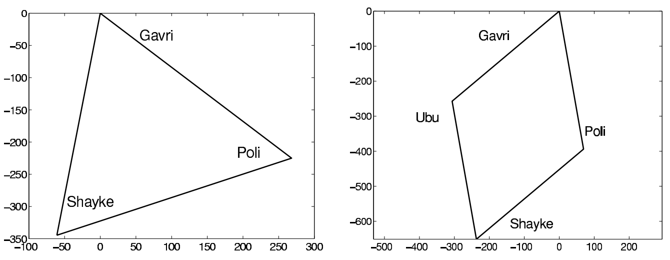
The protocol above can be executed in parallel by all team-members, or using a centralized computation which will distribute the result. When executed in parallel, care must be taken to ensure that (i) the robots begin their decision-making in a synchronized manner (i.e., work on the same initial list of robots $\mathcal{R}$); (ii) arrive at the same choices in the recomputation of the new control graph. The first requirement can enforced in several ways. We chose to enfore 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 ID.

V. Evaluation

To evaluate the use of monitoring multi-graphs 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 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 fixed control graph formation (Section V-B).

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

TABLE II
SENSOR SPECIFICATION, AIBO



(a) Ideal Triangle. (b) Ideal Diamond.

Fig. 2. Ideal formations in fixed control-graph experiments. Robot names are shown.

A. Fixed Control Graphs

The first set of experiments uses fixed control-graphs, generated from the monitoring multi-graphs, to control formations of Sony AIBO ERS-7 robots. Each of these robots has a single camera on its panning head which can be used to detect color blobs in its $\approx 120°$ view-field ($[-59°, 59°]$), although practically, the effective view-field is ($[-35°, 35°]$). Using such color identification, the robot can identify others, when appropriately color marked. The head also contains an infra-red range sensor which can measure distances in the range $[200_{mm}, 1500_{mm}]$, with some uncertainty. We treat the head (camera and distance sensor) as a single logical sensor, providing bearing and distance to another robot 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 $[-35°, 35°]$, is $[-125°, 125°]$. However, in our experience, maintaining the pan angle in the range $[-30°, 30°]$ tends to produce better results. Based on the manufacturer's specifications, and our experience with the robot, we generated a sensor cost specifications for the robot (Table II). Using this table, we used our technique to produce alternative control graphs for the triangular formation specified in Figure 2-a, and the diamond formation in Figure 2-b.

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

In the diamond formation, many control graphs are possible in principle. 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 graph: Shayke monitoring the leader (Figure 3-c), the right follower (3-d), and the left follower (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 six meters in length. The objective was to contrast the stability and robustness of the different control-graphs under reasonable operating conditions.

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

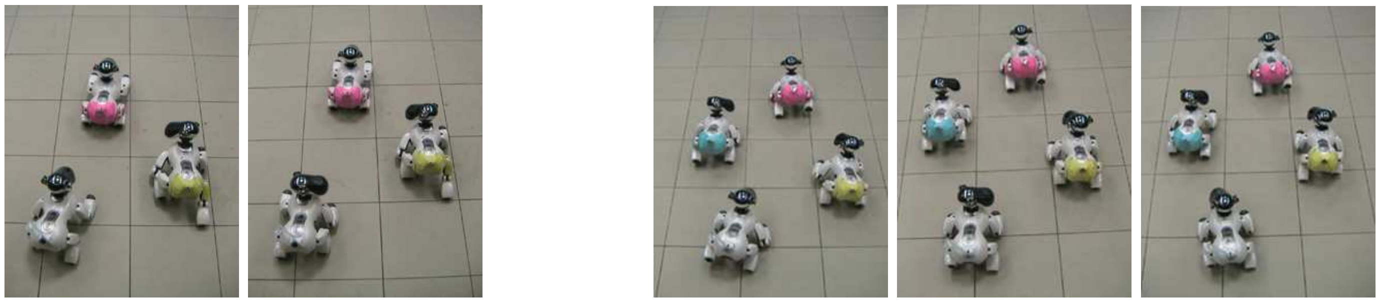
Qualitatively, it can be seen that there exist large variances in the quality of the formations when maintained statically by different control graphs. In the triangle formation, the control graph in which Shayke monitors the leader directly yields much better formation maintenance than the control graph 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

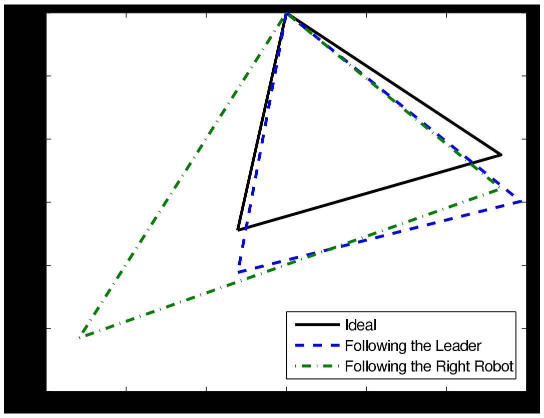
The principal lessson from the first set of experiments is that static formation control graphs can lead to markedly different results. We thus wanted to evaluate the ability of dynamically-switching control graphs to compensate for such limitations, and yield better and more robust formations.

To experiment with this, we re-executed the diamond formation experiments, contrasting a static control graph with that of with a dynamically-switching control graph, using the switching protocol described above. 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 determining whether a robot believed its target to be lost. Smaller number indicates that the robot is very quick to judge its target to be lost, and is thus a good simulation of noisy sensing conditions.

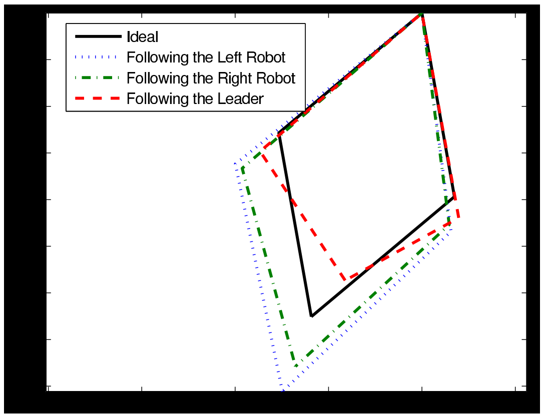


(a) Shayke follows leader. (b) Shayke follows Poly. (c) Shayke follows Leader. (d) Shayke follows Poly. (e) Shayke follows Ubu.

Fig. 3. AIBO robots executing static triangle (Figures a, b) and diamond (Figures c, d, e) SBC control-graphs. Note bottom robot (Shayke) head pose.



(a) Triangle.



(b) Diamond.

Fig. 4. Ideal and actual robot positions.

A large number indicates that the robot is willing to wait a relatively long time before declaring it has lost its target. The numbers actually denote the number of consecutive frames in which the target was not identified. We used values of 4, 20, and 40. Each configuration was repeated 15 times, for a total of 90 trials. Unlike the previous set of experiments, the velocity of the leader was fixed, and thus shorter time for finishing the course indicates improved performance.

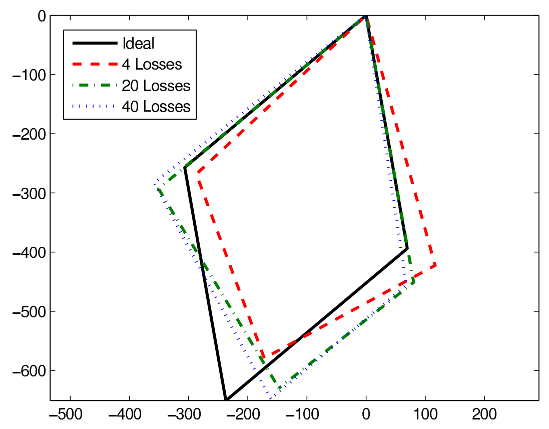
Figure 5 shows the average positions of robots in the diamond formation, in the case of static and dynamic control graphs, for each value of the threshold. As can be seen in the figures, the dynamic control graph yields results that are (i) more consistent across the experimental conditions; and (ii) closer to the ideal formation form. Figure 6 provides quantitative analysis of the same phenomenon. Here, the Y-axis measured the error in placement of Shayke (the robot whose target changed dynamically) in the fixed and dynamic control graph techniques. As can be seen in the figure, in all three conditions, the dynamic switching technique leads to smaller errors.

We examine additional performance measures. Figure 7 shows the time that it took the formation to finish the course, i.e., smaller values are better. The X-axis shows the different threshold settings, simulating different perception errors. The Y-axis measures the time. 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.

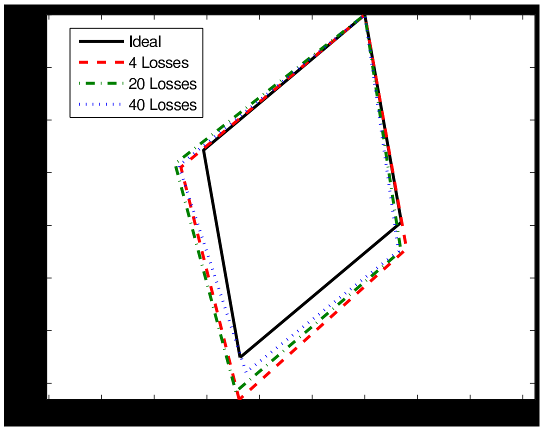
Finally, we examine the percentage of time the formation was maintained, i.e., both angles and distances were within their tolerance levels (10 degrees, 15cm). In Figure 8 The Y-axis here shows the percentage of time, and thus larger values are better. The figure shows two benefits to the dynamic-switching approach. First, the percentage of time the formation was maintained was significantly higher than with the static control graph (a two-tailed 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.

VI. Discussion and Future Work

We presented a novel representation for reasoning about formation contorl graphs, based on directed weighted monitoring multi-graphs. We have shown that the approach allows the use of graph-theoretic techniques, to address key open problems. In particular, we have provided novel techniques that (i)



(a) Fixed.



(b) Dynamic Switching.

Fig. 5. Average Formation

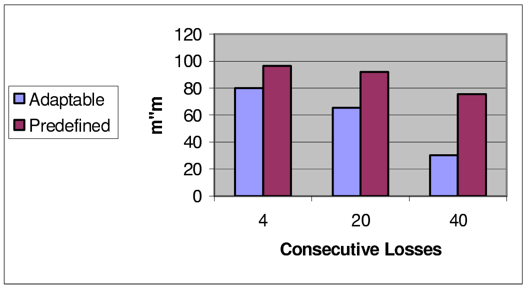


Fig. 6. Position Errors.

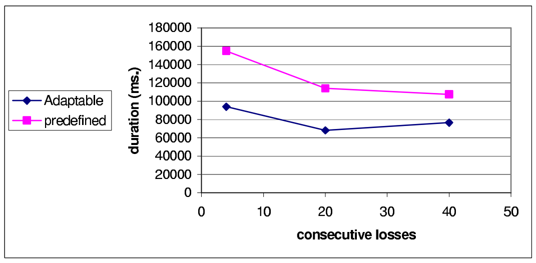


Fig. 7. Time to complete course.

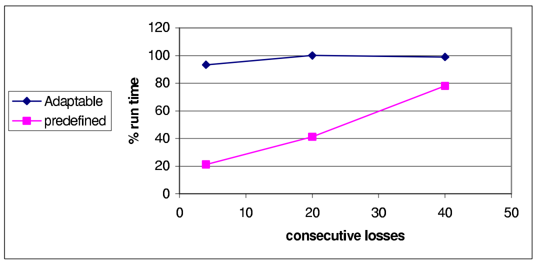


Fig. 8. Percentage of time formation maintained.

optimally account for sensor morphology and constraints in generating distributed formation-maintenance SBC controllers; (ii) allow robots to dynamically switch formation control graph for added robustness. We demonstrated the use of the technique in systematic experiments with physical robots, and have shown that the use of our techniques leads to significant increases in both performance and robustness to environmental conditions.

Acknowledgments. We thank Noa Agmon for useful comments, and K. Ushi for support.

REFERENCES

[1] T. Balch and R. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.
[2] T. T. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. MIT Press, 1990.
[3] J. P. Desai. A graph theoretic approach for modeling mobile robot team formations. *Journal of Robotic Systems*, 19(11):511–525, 2002.
[4] R. Fierro, A. K. Das, V. Kumar, and J. P. Ostrowski. Hybrid control of formations of robots. *IEEE International Conference on Robotics and Automation*, 2001.
[5] J. Fredslund and M. J. Mataric. A general algorithm for robot formations using local sensing and minimal communications. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 10 2002.
[6] G. A. Kaminka and Y. Elmaliach. Experiments with an ecological interface for monitoring tightly-coordinated robot teams. In *ICRA-06*, 2006.
[7] M. Lemay, F. Michaud, D. Létourneau, and J.-M. Valin. Autonomous initialization of robot formations. *IEEE International Conference on Robotics and Automation*, pages 3018–3023, 2004.
[8] F. Michaud, D. Létourneau, M. Gilbert, and J.-M. Valin. Dynamic robot formations using directional visual perception. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.