2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Automatic Formation Deployment of Decentralized Heterogeneous Multi-Robot Networks with Limited Sensing Capabilities

Brian Stephen Smith, Jiuguang Wang, Magnus Egerstedt, and Ayanna Howard

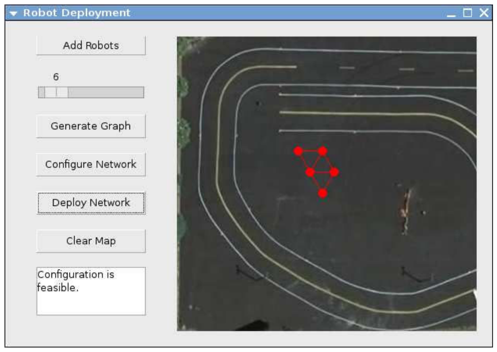
Abstract— **Heterogeneous multi-robot networks require novel tools for applications that require achieving and maintaining formations. This is the case for distributing sensing devices with heterogeneous mobile sensor networks. Here, we consider a heterogeneous multi-robot network of mobile robots. The robots have a limited range in which they can estimate the relative position of other network members. The network is also heterogeneous in that only a subset of robots have localization ability. We develop a method for automatically configuring the heterogeneous network to deploy a desired formation at a desired location. This method guarantees that network members without localization are deployed to the correct location in the environment for the sensor placement.**

I. Introduction

This work is part of a National Aeronautics and Space Administration (NASA) project to implement a multi-robot system for research in Antarctica. A team of NASA scientists will use a network of mobile robots to take sensor readings across ice shelves to better understand the impacts of global climate change. The environment is extremely hazardous and expensive for humans to operate in. Hence, the use of robots is a viable alternative.

The multi-robot network should be able to automatically implement sensing tasks defined by the NASA scientists. To this end, automatic methods are needed to configure the network. The complication of configuring the network is further compounded by the sensing and communication limitations of the network. We consider a decentralized multi-robot network whose members' sensing and communication abilities are limited by a maximum *proximity range*. Here, robots can only estimate the relative position and share information with other robots within proximity range. Furthermore, the network under consideration is *heterogeneous* in that only a subset of agents have localization ability. Such heterogeneous networks can arise due to design, for we show that localization ability for each robot is not required for the goals we consider. However, such heterogeneous networks can also arise due to network failures, such as the failure of the localization sensors on some of the network members. Decentralized networks with limited perception and/or localization ability have seen much recent attention, as in [1], [2], [3], [4], [5].

This paper presents methods for *automatically deploying formations* (e.g., [6], [7], [8]) with a decentralized heterogeneous multi-robot network. Having a prototype multi-robot network composed of mobile agents, a user graphically enters a desired *network deployment* for the network with a Graphical User Interface (GUI), shown in Fig. 1. These coordinates define a set of *deployment positions* which model the desired locations of the robots in the environment and the specific relative geometry that must be satisfied between pairs of robots. We present a method for automatically configuring the multi-robot network to deploy at the desired coordinates despite the limited localization ability of the robots.



Configuration is feasible.

Fig. 1. Graphical User Interface (GUI) for configuring the network.

In Section II, we introduce our multi-robot network and describe the problem of deploying the network to user-specified locations. Section III presents control laws that allow a pair of robots to navigate such that they are never out of proximity range of each other. In Section IV, we present an automatic system utilizing the control laws in Section III to "link" robots in a manner that allows robots without localization ability to navigate to the desired location of the network. As the network members arrive at the deployment positions, the robots uniquely assign themselves positions and navigate to them, as discussed in Section V. Section VI presents experimental results implementing these methods with a prototype multi-robot network. Finally, Section VII concludes.

II. Preliminaries

In this section, we describe how we model the multi-robot network and its desired deployment. We describe how this deployment corresponds to assembling a formation at a specific location in the environment.

A. Multi-Robot Network and Deployment Modeling

We model our multi-robot network as a multi-agent network with $n \geq 2$ agents, $n \in \mathbb{N}$. Here, n defines a set

Brian Stephen Smith, Jiuguang Wang, Magnus Egerstedt, and Ayanna Howard are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. Email: {brian, magnus, ayanna}@ece.gatech.edu, j.w@gatech.edu

of indices for each agent as $N = \{1, \ldots, n\}$. We consider the system over an interval of time $T = [0, \infty)$. Since these are ground robots, the planar position of each agent is represented by a state such that, $\forall i \in N$, $x_i : T \mapsto \mathbb{R}^2$ is the state of agent i and $x_i(t)$ is the position of agent i at time $t \in T$. For all $i \in N$, we define the control for agent i as a single integrator such that $u_i : T \mapsto \mathbb{R}^2$ and $\dot{x}_i(t) = u_i(t)$.

In order to define a desired network deployment, a user graphically inputs the desired *deployment positions* $p_i \in \mathbb{R}^2,\ i = 1, \ldots, N$. Each position corresponds to a location in the environment where we desire a robot to be located. *The goal of the network is to deploy the agents such that each agent navigates to a unique deployment position.* For this problem, we assume there is no preference for specific agents to be located at specific positions.

B. Sensor Limitations

All agents in the network have sensors for estimating the *relative positions* of agents within their *proximity range* $\Delta \in \mathbb{R}^+$. The proximity range is chosen to model the sensing limitations of the robots in the network. Hence, a pair of agents (i, j) can sense and communicate with each other at time t if and only if $\|x_i(t) - x_j(t)\| \le \Delta$. Therefore, this allows us to define $u_i(t)$ as a function of $x_i(t) - x_j(t)$ if and only if $\|x_i(t) - x_j(t)\| \le \Delta$.

This network is heterogeneous in that not all agents have localization ability. We call an agent with localization ability a network *leader*. This localization ability implies that the leaders can estimate the relative position of the deployment positions. We define $N_l \subseteq N$ as the index set of the leaders. Given a specific location in the environment $p_i \in \mathbb{R}^2$, the localization ability of the leaders implies that, $\forall i \in N_l$, we can define u_i as a function of $x_i - p_i$. The remaining agents in the network are *followers* who do not have localization ability. The followers cannot determine their relative positions to any defined goal location in the environment.

C. Formations

Since all agents do not have localization ability, we cannot simply assign agents to positions and have them navigate to each. In [2], [9], we show how the deployment positions define a *formation*, i.e. specific, desired geometric relationships for the agents. We also present methods for determining if the defined formation is *persistently feasible*. If it is persistently feasible, this implies that the network can be automatically configured to assemble the formation, as shown in [3].

While the methods in [3] respect the proximity range of the network, they are based purely on the relative positions of the agents. As a result, the formations are assembled at the initial location of the network. However, the persistent feasibility of the desired formation implies that the location and orientation of the formation is determined by the location of a specific pair of agents [3] [5]. Therefore, our strategy is to assign leader agents to these positions in the formation. Then, once the followers have been led to the deployment positions, they can assemble the formation as presented in [3] using only the relative positions of other agents.

III. Network Deployment Control Laws

In this section, we show how to navigate a pair of agents so they always stay within proximity range of each other. We describe this as *preserving connectivity* between a pair of agents. Utilizing these control laws, a follower agent can follow a leader agent and navigate towards a common position in the environment. These control laws rely only on bounding the maximum velocities of the agents. This is a very reasonable assumption, since most systems have an inherent limit on their maximum velocities due to their design. These control laws are the foundation for our method of automatic network deployment, which we present in the following section.

A. Preserving Connectivity Between Pairs of Agents

The following theorem describes two agents in which one follows the other. By bounding the velocity of the agent being followed to a chosen maximum velocity, we guarantee that the pair of agents never lose connectivity.

Theorem 3.1: Consider a pair of agents l and f as defined in Section II. For a given proximity range $\Delta \in \mathbb{R}^+$ and maximum velocity $u_{max} \in \mathbb{R}^+$, agent f's control laws are defined by

$$K = \frac{u_{max}}{\Delta}, \quad u_f = -K\left(x_f - x_l\right). \tag{1}$$

We assume the following about agent l:

- The control of agent l is continuous and, $\forall t \in T$, $\|u_l(t)\| \le u_{max}$.
- p_l is a globally asymptotically stable equilibrium point of x_l.

Then, for every initialization of the pair such that $\|x_f(0) - x_l(0)\| \le \Delta$,

- $\|x_f(t) - x_l(t)\| \le \Delta \ \forall t \in T$, and
- p_l is a globally asymptotically stable equilibrium point of x_f.

Proof: By our assumptions, x_l is continuously differentiable and Lipschitz continuous, since its first derivative is defined and bounded over the entire domain. From (1), x_f is continuously differentiable.

First, we show that connectivity is preserved. If $\|x_f(t) - x_l(t)\| \ge \Delta$, then this implies that

$$\begin{aligned}
&\frac{d}{dt}\left(\frac{1}{2}\|x_f(t) - x_l(t)\|^2\right) \\
&= -\frac{u_{max}}{\Delta}\|x_f(t) - x_l(t)\|^2 - \dot{x}_l(t)^T\left(x_f(t) - x_l(t)\right) \\
&\le -u_{max}\|x_f(t) - x_l(t)\| + u_{max}\|x_f(t) - x_l(t)\| \\
&\le 0.
\end{aligned} \tag{2}$$

In other words, (2) implies that the distance between x_l and x_f never increases while their distance is greater than or equal to Δ. Therefore, if we assume for some $t \in T$ that $\|x_f(t) - x_l(t)\| > \Delta$, we always have a contradiction.

If $t = 0$, then the initialization assumption is violated. If $t > 0$, then the continuous differentiability of x_f implies that, for some $t_1 < t$, that $\|x_f(t_1) - x_l(t_1)\| \geq \Delta$ and $\frac{d}{dt}\left(\frac{1}{2}\|x_f(t_1) - x_l(t_1)\|^2\right) > 0$. In other words, this assumption implies that, for some time before t, the distance between x_l and x_f is greater than or equal to Δ and their distance is increasing. This violates (2). Therefore, $\|x_f\,(t) - x_l\,(t)\,\| \leq \Delta \;\forall t \in T$.

To show that p_l is a globally asymptotically stable equilibrium point of x_f, we first define the translated system $\tilde{x}_f = x_f - p_l$. The control laws in (1) imply that $\dot{\tilde{x}}_f = \dot{x}_f = -K(x_f - x_l)$. Similarly, we define the translated system $\tilde{x}_l = x_l - p_l$. Substitution implies that

$$\dot{\tilde{x}}_f = -K\left(x_f - (\tilde{x}_l + p_l)\right) = -K(\tilde{x}_f - \tilde{x}_l).$$

Taken together, $\tilde{x}_f$ and $\tilde{x}_l$ are a cascade system [10]. This implies that, since $\tilde{x}_l$ has a globally asymptotically stable origin, then $\tilde{x}_f$ does as well. This implies that p_l is a globally asymptotically stable equilibrium point of x_f. ■

The following corollary shows that, when following another agent with a constant, bounded velocity, the relative position of the leading and following agents stabilizes to a position Δ apart from each other, with the follower directly "behind" the leader.

Corollary 3.1: Consider again the pair of agents described in Theorem 3.1. Assume that $\hat{u} \in \mathbb{R}^2$ is a constant unit vector. Assume that agent l has a constant velocity in the direction of $\hat{u}$ with a magnitude of $u_{max} \in \mathbb{R}^+$ such that $\dot{x}_l(t) = u_{max}\hat{u} \;\forall t \in T$. This implies that $-\Delta\hat{u}$ is a globally asymptotically stable equilibrium point of $\tilde{x}_f = x_f - x_l$.

Proof: We define $\hat{x}_f = x_f - x_l + \Delta\hat{u}$. This implies that

$$\begin{aligned}\dot{\hat{x}}_f &= \dot{x}_f - \dot{x}_l = -K(x_f - x_l) - u_{max}\hat{u} \\ &= -K(x_f - x_l + \Delta\hat{u}) = -K\hat{x}_f\end{aligned}$$

The system $\dot{\hat{x}}_f = -K\hat{x}_f$ has a globally exponentially stable origin. This implies that $-\Delta\hat{u}$ is a globally exponentially stable equilibrium point of $\tilde{x}_f$. ■

Corollary 3.1 implies that we should choose a "safe" proximity range for the network. Ideally, the chosen proximity range should be well enough within the actual limits of the robot sensors to allow for the noise and potential error of the system.

Theorem 3.1 shows us that, as long as the velocity of the leader agent is defined and bounded by our chosen maximum velocity, the distance between the pair of agents cannot exceed the proximity range. A corollary of Theorem 3.1 is that the state and dynamics of agent f also satisfy the same assumptions made about agent l.

Corollary 3.2: Consider the pair of agents described in Theorem 3.1. Given the same assumptions and initialization as in Theorem 3.1, then the velocity of the follower agent is bounded by u_{max} such that $\|u_f(t)\| \leq u_{max} \;\forall t \in T$.

Proof: The control law for agent f in (1) implies that, $\forall t \in T$,

$$\|u_f(t)\| = |-K|\|x_f(t) - x_l(t)\| = \frac{u_{max}}{\Delta}\|x_f(t) - x_l(t)\|.$$

Since $\|x_f(t) - x_l(t)\| \leq \Delta \;\forall t \in T$, then, $\forall t \in T$,

$$\|u_f(t)\| \leq \frac{u_{max}\Delta}{\Delta} = u_{max}.$$

■

Corollary 3.2 implies that, while agent f is following agent l, another agent within proximity range of agent f could follow agent f in the same manner and never loose connectivity with agent f. This suggests that we can connect agents together to follower a single leader using the control laws in Theorems 3.1. We present an automatic system for accomplishing this in the next section.

IV. An Embedded Graph Grammar System for Deployment

Here, a network graph is defined to represent the system. We describe how to connect agents using an Embedded Graph Grammar (EGG) system [11] such that their connectivity is preserved as they follow the leader agents.

A. Network Graph

We define a *network graph* to represent the modes of the agents and the topology of the control laws of the network. We denote this vertex-labeled graph by $G(t) = (V, E(t))$, where $V = \{v_1, \ldots, v_n\}$ is the vertex set, $E(t)$ is the edge set (at time t), and l assigns a label to each vertex. Each vertex is associated with its corresponding agent such that v_i is the vertex of agent i. The label function l assigns labels to each vertex that correspond to the control laws of each agent. Thus, $l(v_i)$ indicates the control laws of agent i. The edges in this graph represents *constraints* between robots. Each edge is an ordered pair, where the order defines the direction of the edge. Thus, $(v_i, v_j) \in E(t)$ is an edge from v_i to v_j, indicating that agent i's control laws are dependent on the position of agent j. If a path from v_i to v_j exists, we say that agent i is a *predecessor* of agent j. If $(v_i, v_j) \in E(t)$ we say that agent i is an *immediate predecessor* of agent j. This graph represents what the network is doing at any given instant in time. Hence, the graph is dynamic, and the membership in E changes as the system evolves.

B. Embedded Graph Grammar System for Deployment

Here, we present a method to automatically generate *Embedded Graph Grammar (EGG) systems* for deploying agents using the control laws from Section III. An EGG is a formalism that encodes dynamic, geometric, and network properties of a multi-agent system in a unified manner. It extends the notion of a *graph grammar* that takes as inputs vertex-labeled graphs, and produces other vertex labeled graphs according to a given rule set. Through the application of the rules in the rule set, edges may be removed or added to the graph, and the vertex labels may change.

In order to characterize how robots should navigate and establish and maintain constraints with other robots, as well as the corresponding change in network topology, we define graph-transition *rules*. Each rule consists of a vertex-labeled *left graph* $\mathcal{L}$ (the input to the rule), a vertex-labeled *right graph* $\mathcal{R}$ (the output to the rule), and a *guard* that defines specific conditions under which the rule is applicable. These rules define the desired interactions between robots and are given to each robot, along with the corresponding control laws for each position, in order to execute the formation.

Assume that $\mathcal{V}_{\mathcal{L}}$ is the vertex set of the left graph $\mathcal{L}$ of a rule r. As in [11], rule r is applicable only if there exists a label-preserving isomorphism between the vertices $\mathcal{V}_{\mathcal{L}}$ of the left graph $\mathcal{L}$ and the vertices of $G(t)$. The guard function g determines whether or not the rule can be applied and evaluates to $true$ or $false$.

If a rule is applicable, the subgraph of $G(t)$ isomorphic to $\mathcal{V}_{\mathcal{L}}$ can be replaced in $G(t)$ by the right graph $\mathcal{R}$ in the rule. A guarded rule is represented by the triple $r = (\mathcal{L} \rightharpoonup \mathcal{R}, g)$. In [3], we describe how to implement these EGG systems with a multi-robot network.

For the EGG we present here, we define the label set of each vertex such that each label has two parts: the *mode* and the *go flag*. The mode indicates what control law the agent is implementing, while the go flag is a boolean indicating whether or not the agent can implement the control law. If the go flag is $false$, the agent must sets its control to zero and stay at the same location. We use the notation that $l(v_i).mode$ is the mode of agent i, and $l(v_i).go$ is the go flag of agent i.

Since this is a heterogeneous network, the leaders and the followers each have different modes. We define mode L as *leader mode*. When in leader mode, the agent's control law is designed to stabilize the agent to a given goal position for deployment. Thus, $l(v_i).mode = L$ and $l(v_i).go = true$ if and only if $i \in N_l$ and agent i is moving towards its assigned deployment location.

The follower agents initially are assigned mode U, which is *unassigned mode*. This mode corresponds to a follower that has no one to follow, and does not move. Once it has been assigned someone to follow, it changes its mode to F, which is *assigned follower mode*. Agents in assigned follower mode have a single edge in the network graph directed towards the agent they are following. Thus, $l(v_i).mode = F$ and $l(v_i).go = true$ indicates that agent i is following the agent at the head of its directed edge. If $l(v_i).go = false$, the agent does not move.

C. Initialization

To describe the required initial conditions of this EGG system, we define a *proximity graph* $\mathbb{G}(t) = (\mathbb{V}, \mathbb{E}(t))$. Here, $\mathbb{V} = V$, the vertices in our network graph, and $\exists(v_i, v_j) \in \mathbb{E}(t)$ if and only if $\|x_i(t) - x_j(t)\| \leq \Delta$, and we say that agents i and j are *connected*. Hence, edges in our proximity graph indicate which pairs of agents can sense and communicate with each other.

We initialize the leader agents in mode L, the follower agents in mode U, and the go flags of all agents are set to $false$. Thus, at $t = 0$, the network graph $G(0)$ has no edges. We initially require that a path exists in the proximity graph $\mathbb{G}(0)$ between each follower and at least one leader. As long as this condition is satisfied, the initial proximity graph can be disconnected. The following describes the EGG rules for "linking" pairs of agents.

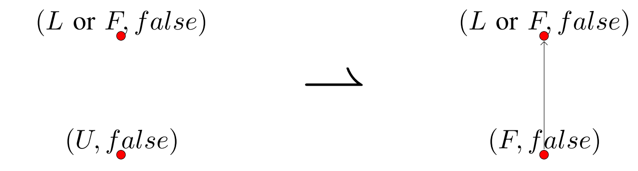


Fig. 2. Linking rules.

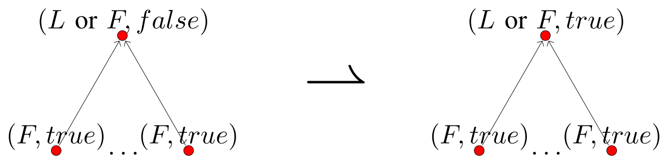


Fig. 3. Go rules.

D. Linking Agents with Linking Rules

To establish edges, we first use *linking rules* shown in Fig. 2. There are two linking rules. In the first linking rule, the left graph consists of two agents. One is a leader agent, and one is an unassigned follower. The left graph has no edges. If they are within proximity range, the follower can switch to follow mode and add an edge to the network graph directed towards the leader. The follower then implements the control law described in (1). The other linking rule is identical except that, instead of a leader agent, the left graph includes an assigned follower. By repeatedly applying this rule, all unassigned followers are assigned to follow either a leader or a predecessor of a leader.

E. Moving Agents with Go Rules

The *go rules* specify when the leaders and assigned followers can begin implementing their assigned control laws, as shown in Fig. 3. For either a leader or an assigned follower, if all its immediate predecessors have $true$ go flags and if no agents within proximity range to it are unassigned followers, the agent switches its go flag to $true$ and begins implementing its control laws. These rules guarantee that all unassigned followers are linked to follow a leader or a predecessor of a leader before the agents within their proximity range move.

F. Break Rules

While the previous rules ensure that agents become predecessors of leaders, and that agents do not leave unassigned followers behind, it is possible for multiple followers to follow the same agent. If that agent has a constant velocity, then Corollary 3.1 implies that the followers will stabilize to the same location, directly behind the leading agent. For an actual mobile robot system, this is not desirable, since the robots must avoid colliding. Therefore, we define *break rules* that reduce the immediate predecessors of a single agent.

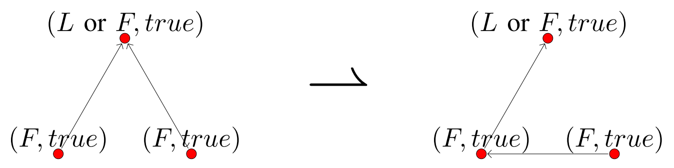


Fig. 4. Break rules.

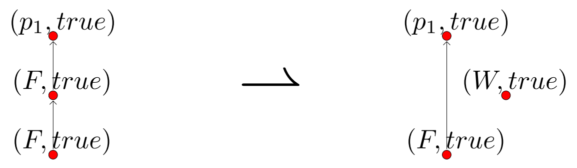


Fig. 6. Wander rules.

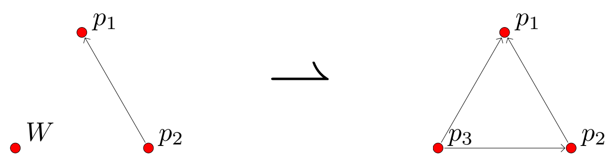


Fig. 5. Vertex addition rules.

The left graph has a agent being followed by two follower agents. If all of these agents are within proximity range of each other, one of the following agents is switched to follow the other follower. For any agent with multiple immediate predecessors, the repeated application of this rule ensures that, eventually, it will only have one immediate predecessor. Fig. 4 represents these break rules.

When implemented with our multi-robot network, this EGG results in "chains" of robots, as shown in Fig. 7. In general, this EGG system for deployment can be used with only one network leader to allow any number of followers to navigate to a desired location.

V. Formation Assembly

In this section, we discuss how to assemble formations as the network arrives at the deployment coordinates in the environment. We show how to automatically generate an EGG system for assembling persistent formations in [3]. In this previous work, the network graph is initially unassembled, and each agent begins in a *wander* mode. An initial edge is added between a *leader* and *first-follower* pair of agents. The positions of these agents specify the location and orientation of the formation in the environment. Then, *vertex addition rules* attach wanderers to the leader and first-follower, as shown in Fig. 5. These vertex addition rules assign the agents their unique positions. These agents then navigate to the correct locations using only the *relative positions* of other agents. When all vertex addition rules have been applied, the formation is assembled.

The initial conditions of the EGG presented in [3] are that all wander mode agents must be within proximity range of either the leader or first follower agent in the formation. While our EGG for linking agents allows us to navigate the network with only one leader, we utilize two leader agents in our current implementation. In this implementation, we order the deployment positions such that p_1 is the *leader position* and p_2 is the *first-follower position*. Both leaders are initially assigned to navigate to p_1. When a leader has successfully navigated to this position, it assigns its mode to p_1 such that $l(v_i).mode = p_1$. The remaining leader, when encountering the agent assigned to p_1 will then switch and begin converging to the first-follower position p_2. Therefore, the network deploys as two chains, one which converges to the leader position in the formation graph, and one that converges to the first-follower position.

In order to switch followers from follow mode to wander mode, we employ *wander rules* that assign followers to wander mode once they have reached the deployment location. If a follower is following a leader agent, and all its immediate predecessors are within proximity range of that leader, it switches to wander mode, and all its predecessors begin following the leader or first-follower. Fig. 6 depicts wander rules. The application of this rule implies that all wanderers are within proximity range of either the leader or the first-follower, which are sufficient conditions for successfully assembling the formation [3].

VI. Implementation

This section discusses implementation results of the automatic EGG generated for formation deployment, which can be seen in the video submission that accompanies this paper.

In order to approximate the Antarctic robots in the pre-Antarctic stages of this project, we use the prototype network. For mobility, each robot has a wheeled platform base and is dynamically similar to the tracked platform for the Antarctic. Each has an onboard computer, and communication between robots is achieved by wireless communication modules on each robot. GPS receivers on each robot estimate the location and heading of each robot.

By sharing GPS values, the robots can obtain *relative* position information of other robots (i.e., the range and bearings of other robots relative to their own heading). Since we have global communication ability, and, thus, global information, we can arbitrarily limit the information each robot is allowed to use. This allows us to verify our methods with a variety of sensing limitations.

We implement the deployment depicted in Fig. 1, where a user has chosen $n = 5$ deployment positions using the GUI with satellite imagery of our test field. The GUI is able to compare the positions with the known coordinates of reference positions in the satellite image to estimate the desired deployment coordinates for the network. Using the methods in [2] and the defined proximity range $\Delta = 6$ m, the software determines that the deployment positions can be assembled as a persistent formation. The network members are automatically configured to assemble the formation using the methods in [3]. All network members are configured to implement the EGG system for linking agents discussed in Section IV. Then the network members are given the command to deploy.

Fig. 7(a) shows the initial state of the network. The leaders are in leader mode, labeled L, and the followers are initially in unassigned mode, labeled U. The robots begin applying the EGG rules described in Section IV and in [3]. By applying linking rules, two followers begin following the leader on the left, and one follower begins following the leader on the right. Since two follower are following the left leader, the breaking rule is applicable, and is applied, allowing the followers to follow as a "chain", seen in Fig. 7(b). The leader on the right arrives at deployment position p_1 first, the leader position for the formation. It changes to the appropriate mode, allowing its follower to switch to wander mode, as seen in Fig. 7(c). Since the first-follower position has not been filled, this wander cannot perform a vertex addition yet. In Fig. 7(d), the remaining leader sees that a robot is already assigned the leader position. It switches and navigates to the first-follower position p_2. This allows one of its followers to switch to wander mode W, while also allowing a vertex addition to take place. Here, the first wanderer is assigned to position p_3. In Fig. 7(e), another vertex addition is applied, assigning position p_4 to the wanderer. There is one more wanderer, since the last follower has also switched to wander mode. However, there is no vertex addition rule possible for where this last wanderer is located. As described in [3], the remaining wanderer begins moving through the formation towards a position that remains to be assigned. Eventually, it is in range of both robots assigned to p_2 and p_3, allowing a vertex addition rule to assign it to p_4, as shown in Fig. 7(g). Fig. 7(h) shows the final formation. Here, the error of each robot is within the limitations of our platform, which estimates the relative positions between robots with an error of approximately 2 m.

VII. Conclusions

We have presented automatic tools for deploying heterogeneous multi-robot networks as a mobile sensor network. While only a subset of robots have localization ability, the robots with localization can lead the robots without localization such that all robots arrive at the user-defined deployment locations. As the robots arrive, they begin executing an automatically generated system for assembling a persistent formation at the deployment location. The result is that all robots are assigned a unique deployment position, and robots without localization ability still converge to their assigned positions. This has been demonstrated with an actual multi-robot network.

VIII. Acknowledgements

This was supported under a contract with NASA.

References

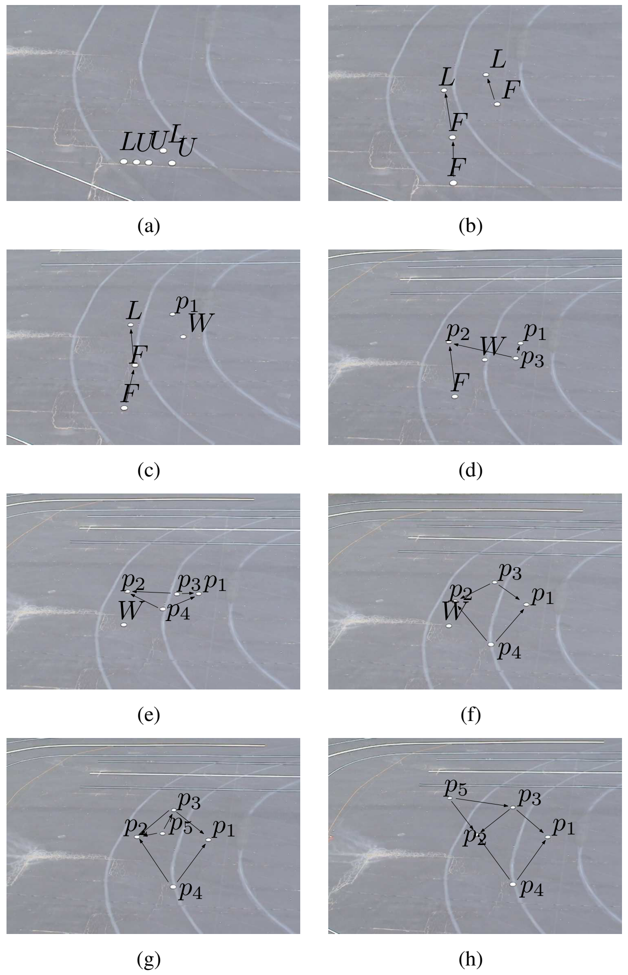


Fig. 7. Formation deployment and assembly with the multi-robot network. This corresponds to the desired deployment shown in Fig. 1.

[1] M. Ji and M. B. Egerstedt, "Distributed coordination control of multi-agent systems while preserving connectedness," *IEEE Trans. Robot.*, vol. 23, no. 4, pp. 693–703, Aug. 2007.

[2] B. S. Smith, M. Egerstedt, and A. Howard, "Automatic generation of persistent formations for multi-agent networks under range constraints," in *Proc. of the First Int. Conf. on Robot Commun. and Coordination*, 2007.

[3] ——, "Automatic deployment and formation control of decentralized multi-agent networks," in *Proc. of the IEEE Int. Conf. Robot. Autom.*, Pasadena, CA, USA, May 2008, pp. 134–139.

[4] F. Zhang and S. Haq, "Boundary following by robot formations without gps," *Proceedings of the IEEE Int. Conf. Robot. Autom.*, pp. 152–157, May 2008.

[5] B. S. Smith, J. Wang, and M. B. Egerstedt, "Persistent formation control of multi-robot networks," in *Proc. of the IEEE Conf. on Decision and Control*, Cancun, Mexico, Dec. 2008, pp. 471–476.

[6] J. Desai, J. Ostrowski, and V. Kumar, "Control of formations for multiple robots," in *Proc. of the IEEE Int. Conf. Robot. Autom.*, May 1998, pp. 2864–2869.

[7] G. A. Kaminka and R. Glick, "Towards robust multi-robot formations," *Proc. of the IEEE Int. Conf. Robot. Autom.*, pp. 582–8, May 2006.

[8] L. Vig and J. A. Adams, "Multi-robot coalition formation," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 637–49, August 2006.

[9] B. S. Smith, M. Egerstedt, and A. Howard, "Automatic generation of persistent formations for multi-agent networks under range constraints," *Mobile Networks and Applications*, 2009, to appear.

[10] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ 07458, USA: Prentice Hall, 2002, ch. 4, pp. 179–180.

[11] J.-M. McNew and E. Klavins, "Locally interacting hybrid systems with embedded graph grammars," *Proc. of the IEEE Conf. on Decision and Control*, pp. 6080–6087, 2006.