Grouper: A Proof-of-Concept Wearable Wireless Group Coordinator

Fayette W. Shaw, Andrew Lawrence, Eric Klavins

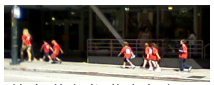
University of Washington, Departments of Mechanical and Electrical Engineering, Seattle, WA 98195

Motivation

Problem Statement: ***Is my group with me?*** Coordination for a group of people often relies heavily on a leader's ability to visually monitor group members. This centralized tracking problem is difficult when an environment is noisy or crowded.

Goal: ***Stay together!*** We present wearable devices, called *Grouper modules*, to aid in group coordination. These devices augment a leader's ability to monitor group members and provides sensory cues for both the *leader* and group members, who we refer to as *followers*.

Applications: There are many scenarios where a group leader would benefit from augmented sensory ability. In a fire, visibility is obscured by smoke. In a collapsed building, visibility is obscured by darkness and debris. In a crowded plaza, visibility is obscured by people. We aim to augment a leader's ability to sense followers by providing visual, audio, and haptic cues the group performs an undesired behavior.



A leader aids visual tracking by dressing followers in bright red shirts.

Background

Group coordination is naturally occurring and engineered. Various tasks include staying together, facing the same way, dividing into subgroups, or creating a particular formation, as shown in the images below. Here we examine the goals of group coordination in the framework of consensus and coordination in multi-agent systems [1].

A duck leads a group of ducklings.

Video still of robots clustered by color. [2]

We represent groups using graphs. A graph describes the interactions between group members, or agents and may be used to indicate communication or proximity. Mathematically, a graph $G = (V,E)$ is a set of vertices V, group members, and edges E, interactions [3].

Staying together in a connected graph

A group that stays together maintains a *connected graph*. In a connected graph, there exists a path connecting any vertex to any other vertex. The upper right graph is a example of a connected graph. In contrast, the lower right graph is an example of a *disconnected* graph.

Connected Graph

Disconnected Graph

Centralized vs. Decentralized: a coordination hierarchy

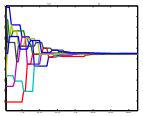
To maintain a connected graph, agents must communicate to one another. One method is to use *centralized* communication and all communicate to a single leader. We use a *star graph*, shown right, to describe centralized behavior, such as when a leader is accountable for the group behavior. Note, however, that a star graph becomes disconnected with the deletion of one edge. Centralized communication is not robust to a single link failure. Thus, we are interested in *decentralized* communication where agents communicate to one another directly.

Star Graph

Distributed Consensus

In a centralized hierarchy, agents simply follow the leader. In a decentralized hierarchy, agents must come to *consensus*. A network of agents may come to consensus on various system states using distributed pairwise averaging. Below is a simple update equation (1) for two agents i and j averaging their states x_i and x_j. At the lower right is a plot representing agents with arbitrary initial conditions using this update equation. Note that they all reach the same value.

$$x_i(t+1) = x_j(t+1) = \frac{1}{2}x_i(t) + \frac{1}{2}x_j(t) \quad (1)$$



Agents reach consensus via Eqn. (1).

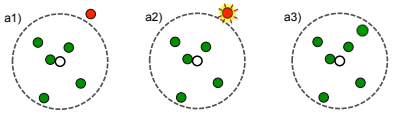
We aim to implement existing consensus algorithms [4] to guide group coordination.

Group Behaviors

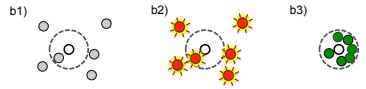
To address the goal to stay together, we designed modules to provide sensory cues to leaders and followers in the event of an undesired group behavior. Below are sketches of these initial or undesired behaviors, sensory cues, and resulting group behavior.

Centralized Behaviors

We begin with two centralized behaviors where follower actions are based about proximity to the leader and followers communicate with only the leader. We model these scenarios as star graphs.



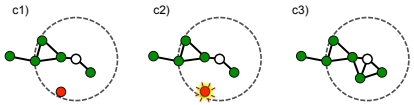
The *cluster* behavior requires all followers to be within a threshold distance from the leader. A follower who has wandered outside the permitted range (a1) receives an alert (a2). Upon returning to a safe distance, the follower's module ceases the alert (a3).



The *global alert* behavior is used when the leader needs to gather the group in close proximity. All followers are alerted regardless of initial proximity (b1-b2). The alert ceases when all agents are within distance (b3).

Decentralized Behaviors

Unlike the centralized behaviors, these do not rely on direct interaction between the leader and each follower. Instead, some interactions are indirect and communicated through other followers.



The *connected graph* behavior is similar to the cluster behavior, except that the followers must be a threshold proximity from any other group member who is connected to the leader. A follower that is not connected to the group (c1) is alerted (c2). This alert ceases when the follower rejoins the group.

The *divide into groups* behavior is inspired by task assignment for robots [5] and ants [6]. Often, groups have subgoals and thus, must be divided into subgroups. While task assignment is not inherently decentralized, this behavior is envisioned so that uninitialized followers (d1) choose the closest leader (d2). In the event that groups are sufficiently uneven, followers would reassign themselves to groups (not pictured). Lastly, the subgroups would cluster together.

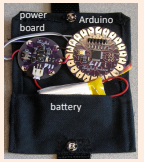
Demo

Welcome!

Visitors are invited to act as followers in the *cluster* and *global alert* behaviors.

Grouper follower module

Module face



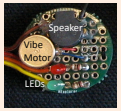
Module inside

Instructions

Pick up a follower module and walk away from demo table.
What happens? When do you receive an alert?
What are you inclined to do when you receive an alert?

Module States

- Safe state green LED on
- Unsafe state red LED on vibe motor on speaker alarm



Q board detail

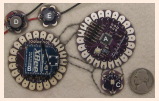
Leader module

When a follower performs is in an unsafe state, the leader is notified via the LCD screen. The leader can change between the two group behaviors via the mode switch.

How It Works

The Xbee radio packets are associated with received signal strength indication (RSSI), measurements that are approximately proportional to the square root of distance. This measurement is then run through a low-pass filter to reject noise and a state-dependant threshold is applied.

LilyPad Prototyping



The LilyPad Arduino [7] and associated components constitute a sewable, washable, open source hardware and software toolkit. This platform of numerous sensors and actuators enables fast prototyping of small, wearable devices.

Above is an early prototype of Grouper module. Parts labeled: A) LilyPad Arduino, B) XBee shield, C) Tri-LED, D) vibration motor, E) speaker. Components are connected via conductive thread (except D and E). Nickel for scale.

Future Work

Design refinement
Integrate LilyPad components on a single circuit board, optimize communication protocol

System characterization Characterize radio signal strength as a function of distance in various environments (outdoors, hallway, room, crowd, etc.), and leader/follower interaction time

Implement Decentralized Algorithms
Stay together using peer-to-peer communication, estimate the global graph using pairwise by averaging local graphs [4]

User testing
Observe user behavior in response to sensory cues, design intuitive cues

Applications
Education: Field trips, school bus transportation, special-needs students
Recreation: Team sports, hiking/climbing/mountaineering
Rescue: Fire-fighting, mining, natural disasters
Tactical/Military

Acknowledgements: E. Klavins and F. Shaw are partially supported by NSF Grant #0735953: "EFRI ARESCI: Controlling the Autonomously Reconfiguring Factory." The authors thank A. Leone, N. Martin, T. Minor, and L. Meernik for hardware and software support.

References: [1] R. Olfati-Saber, J. Fax, and R. Murray. Consensus and Cooperation in Networked Multi-Agent systems. Proceedings of the IEEE, 2007. [2] J. McLurkin, Analysis and Implementation of Distributed Algorithms for Multi-Robot Systems. PhD thesis, June 2008. [3] C. Godsil and G. Royle, Algebraic Graph Theory. Springer, 2001. [4] F. Shaw and E. Klavins, Distributed Estimation and Control in Stochastically-Interacting Robots. Proceedings of the 47th IEEE Conference on Decision and Control, 2008. [5] J. McLurkin and D. Yamins, Dynamic Task Assignment in Robot Swarms. Proceedings of Robotics: Science and Systems Conference, 2005. [6] M. Dorigo, E. Bonabeau, and G Theraulaz, Ant Algorithms and Stigmergy. Future Gener. Comput. Syst., 2000. [7] L. Buechley, M. Eisenberg, J. Catchen, and A. Crockett, The LilyPad Arduino: Using Computational Textiles to Investigate Engagement, Aesthetics, and Diversity in Computer Science Education. Proceedings of SIGCHI Conference on Human factors in Computing Systems, ACM, 2008.