Anomaly Detection in Swarm Robotics: What if a member is hacked?

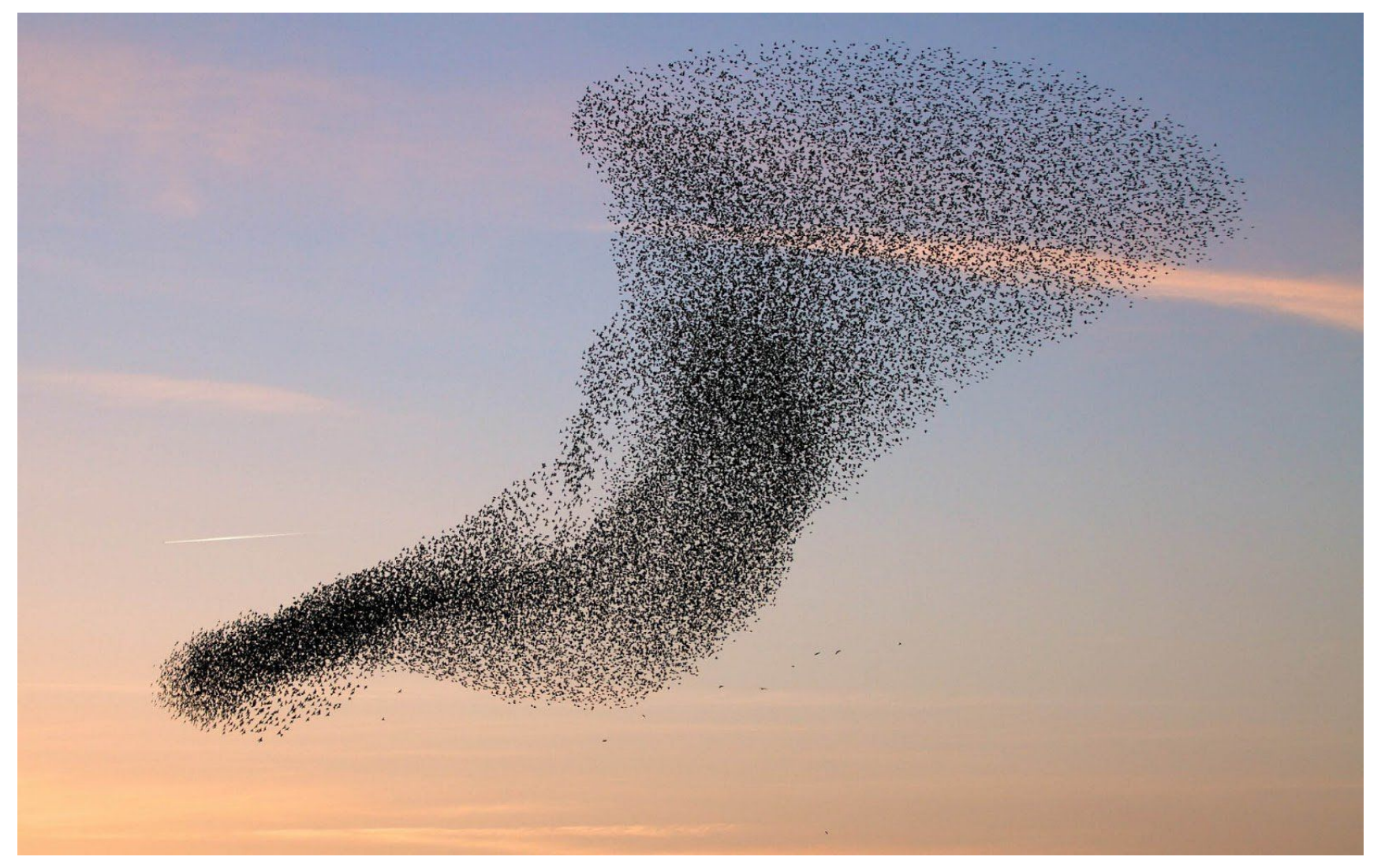
Dan Cronce, Dr. Andrew Williams, Dr. Debbie Perouli

Background on Swarms

- *Swarm* - a collection of homogeneous individuals who locally interact
- *Swarm Intelligence* - the collective behavior exhibited by a swarm

Swarm intelligence algorithms enjoy many benefits such as scalability and fault tolerance. However, in order to achieve these benefits, the algorithm should follow four rules:

- *Local interactions only* - there cannot be a global store of information or a central server to command the swarm
- *Simple* - the members should not be overly complex
- *Homogeneous* - all members should be identical
- *Emergent behavior* - the collective behavior exhibited by the swarm


A flock of birds exhibiting swarm intelligence.

http://www.howitworksdaily.com/wp-content/uploads/2015/07/wallpaper-of-a-flock-of-flying-birds-hd-bird-wallpapers.jpg

The Algorithm Tested On: Boids Flocking

The Boids flocking algorithm emulates the flocking of animals. Each Boid only influences other Boids who are in the same *neighborhood*. A neighborhood is simply a radius around the Boid. We achieve our swarm by summing the vectors of three forces:

- *Cohesion* - this force attempts to move the Boid in the center of all other Boids in its neighborhood
- *Separation* - this force attempts to keep a specified distance from all other Boids in its neighborhood
- *Alignment* - this force is the average of all the vectors in which the other Boids are heading

The Problem

In swarm intelligence, each member locally contributes to the global behavior. If a member were to be maliciously controlled, its behavior could be used to control a portion of the global behavior. This research attempts to determine how one member can determine anomalous behavior in another. However, a problem arises:

If all we're detecting is anomalous behavior, how do we tell the difference between a fault and being hacked?

From an outside perspective, we can't tell if there's a problem with the sensors, the motor, or whether the device is being manipulated. Therefore, this problem is basically reducible to fault detection. What makes it different from standard fault detection is the inclusion of a constraint:

We can't trust the member we're looking for faults in.

If that member is hacked, it could lie about its position, current task, or general health status.

Monitoring Every Member

In order to detect anomalies, we have to be actively looking for them. To decentralize and parallelize the load, each member attempts to monitor exactly one other member. If a member does not have a "suspect," it will select one. It then broadcasts its choice. If another member is already monitoring that suspect, that member will send a back-off signal. The suspectless member will exhaustively search for an unmonitored suspect.

Detecting Anomalies

We detect anomalies by comparing the suspects value to what it should be. The Boids flocking algorithm can be linearly computed, and we can retrieve the inputs without having to reconstruct them; this simplifies the detection algorithm. We simply define some rules that are indicative of anomalous behavior:

- *If the suspect stops sending messages, it is immediately suspicious.*
- *If the suspect claims to be where it is not, it is suspicious.*
- *If the suspect is where it should not be, it is suspicious.*

Finding Absolute Positions of Others

The messages broadcasted from each swarm member contain positions from its odometry. However, we cannot trust the member we're monitoring, so we must have a another way of finding the position of the suspect. Our method is to find the distance from three points using the signal strength of its WiFi and then using trilateration to determine its current position.

Experimental Setting

Currently, we have a large, empty space for the robots to roam. We plan to set up devices to act as wifi trilateration servers, and eventually, develop an application to introduce an anomalous member.

Results So Far

The flocking algorithm has been tested in both simulation and experimentation. Additionally, the wifi trilateration service has been shown to correctly judge distance and position within tolerance.

Acknowledgements

I would like to acknowledge the Humanoid Engineering & Intelligent Robotics (HEIR) Lab for their equipment and support and my cohorts for their assistance.

This work was supported in part by a National Science Foundation REU Site grant, #ACI-1461264, 'Computation Across the Disciplines,' and NSF National Robotics Initiative grant #1427399 at Marquette University.