Lab 4: Configuration Control

Due: February 12, 2009

The goal of this lab is to design several different configuration control algorithms. All of the algorithms should exhibit correct operation, be scalable to variations in population sizes, and be self-stabilizing to the desired configuration.

1 Setup

Before you start, update your copy of the SwarmBotAPI subversion repository, unzip the lab 4 source, and download the new SwarmOS to the robot. Review the documentation's main file, index.html, located in the lab4\doc folder.

There are new IR remote commands, `sir` and `smir`. Use these instead of `s` and `sm` to send commands to your group's mini-swarm. Continue to use `dwir` to download software using the IR transmitters.

There is no way to do data collection over the IR system, and no easy way to modify the operating system to do data collection on a subset of robots. Therefore, only one team can do data collection at a time. I will keep the data collection tasks small to minimize interference.

2 Tree-Based Configuration Control

Design any two of these tree-based configuration control algorithms:

Cluster Design an algorithm that clusters all the robots in the network at the root of the tree. This algorithm should be self-stabilizing: if the root is removed, a new one should be elected. If the root is moved, the robots should re-cluster to the new location.

Disperse Design an algorithm that disperses all the robots from the root of the tree. The dispersion should be from the root, and should form concentric rings around the root. It is ok to specify a fixed distance of dispersion, but just moving away from nearby robots without using the tree structure is not an acceptable design.

Tree Navigation Design an algorithm that uses the tree to guide one robot from anywhere in the network to the root of the tree. If the root is moved during the process, the navigating robot should re-navigate.

3 General Configuration Control

Design any one of these configuration control algorithms:

Uniform dispersion Design an algorithm that will provide a uniform dispersion in a bounded environment, like the playpen. The algorithm should be insensitive to population changes, and self-stabilize to a uniform dispersion if robots are removed from or added to the configuration. It should not use a fixed dispersion radius, instead relying on the physical constraints of the environment to contain the dispersion, and the algorithm to spread the robots uniformly.

Line Formation Design an algorithm that will produce a "straight" line of robots. The algorithm should be insensitive to population changes, and self-stabilize to a straight line if robots are removed from or added to the configuration.

4 Check-off and Write-up

4.1 Check-off (Complete as a group)

1. Demonstrate two tree-based algorithms from section 2. Show proper operation, scalable population sizes, and self-stabilization. Use the lights to indicate which robot is the leader.
2. Demonstrate one algorithms from section 3. Show proper operation, scalable population sizes, and self-stabilization. Use the lights to indicate key behavioral states.

4.2 Experiments and Write-up (Complete individually)

Write a **1 page** report with five paragraphs: introduction(1), algorithms(3), and conclusion(1). For each algorithm, clearly state the problem definition, and your algorithm design. Explain what invariant(s) your algorithms strive to maintain, and what configurations cause them to struggle.