CECS 574 Final Project: *Movement Algorithm*

AUTHORS:
Ahmed Arbi[†] Erick Ortiz[†] Justin Lee[†] Moris Galvez[†] Tate McGeary[†] Zack Obeid[†]
May 9, 2018

Abstract

This document encapsulates details of a Distributed Computing project. The project is designed to demonstrate the capabilities of kilobots shown in figure 1. Concepts such as localization, coverage, broadcast and spanning tree formation are used to create a pattern shown in figure 2. The combination of these principles allowed for the movement and placement of individual nodes at equidistant points using a seed of three kilobot as a point of reference.

CECS 574 Final Project: *Movement Algorithm*

Ahmed Arbi, Erick Ortiz, Justin Lee, Moris Galvez, Tate McGeary, and Zack Obeid

I. Introduction

Kilobots are mobile agents which communicate to each other by reflecting messages in infrared (IR) light off the ground surface. The agents can be used to model complicated distributed system models before they are translated into real world problems. Applications of mobile agents such as Coverage, Localization and Pattern Formation were also explored in this project. Coverage in mobile agent means that nodes are placed at maximum distance from each other to cover the largest distance. Localization implies that each node is aware of the location of all the other nodes in the system. Pattern Formation involves a specific pattern being formed by mobile agents in a system. This project utilized Localization and Coverage along with other distributed computing concepts to form the pattern seen in figure 1.

Figure 1: Depiction of kilobots.

II. Motivation and Description

The purpose of this project was to implement distributed computing concepts presented in CECS 574, and to adapt these concepts to produce a unique idea that has never been done before with kilobots. Inspiration was also drawn from past implementation of algorithms and the thought of combining them to achieve our project proposal. In order to meet the scope of this project, the following requirements must be satisfied: This projects requires eight kilobots. Three are used to create a seed, and five nodes are randomly placed, which will eventually land in their designated location. This was achieved with a leader election algorithm.

A seed is created which is used to coordinate the movement of the bots. The following algorithm is used to create a seed:

```
Each node v cycles through neighbours, u
    If there are two neighbours within threshold
        Set v status as potential leader
    If v’ s status is potential leader
        Share v’s id with all u neighbours
and set v’s election flag true
```

Upon receiving a message each node v executes the following code:

```
    If v’s status is not seed or status is seed and
received message is from a potential leader with id
less than the current potential leader
        Set received message id to leader
        Update gradient if necessary and
status to potentially safe
    If all neighbours of larger gradient than v
have the same potential leader of v and all are safe
neighbours
        Set v’s status to safe
    If v is potential leader and all neighbours
are safe
        Set v’s status to final leader
```

Once a seed has a been created, the movement can be coordinated with the assumption that all nodes are set up within communication distance of one another. Additionally, all of the nodes are structured as a tree. To initiate movement, a message is generated by the seed and propagates through the tree. If the target_id does not match the current node’s id, then the message is forwarded to the next neighbor. Otherwise, the target node orbits its nearest neighbor in clockwise fashion. This movement continues orbiting static nodes until the target node encounters the Leader node. When this condition is met, the target node begins to orbit the inner edge of the communication boundary formed by the Leader. The node will stop when it reaches the designated position relative to the Leader. This generates a message destined for Leader. The Leader issues a new movement command destined for the new target node. The new target nodes enter an avoidance movement state when they encounter nodes that have been set in their appropriate location or if they encounter nodes with a lower gradient. This action alters the trajectory of the target node and forces it to travel beyond the inner Leader communication edge. Upon reintroduction the target node continues to orbit the inner Leader communication edge.

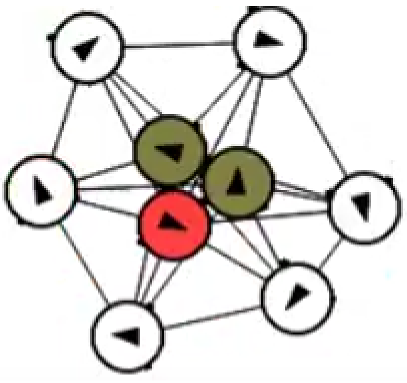

Figure 2: Final configuration of movement algorithm.

III. Implementation Problems

One of the most prominent issues encountered during the development of the algorithm was communicating to all nodes to transition to different phases. Numerous flags were used to indicate the current phase, phase completion, and the start of new phases. Forced delays were introduced to allow for the propagation of flag updating messages through broadcast.

Another hurdle to overcome was to implement a networking protocol that would allow the kilobots to send and receive messages with specific targets. A forwarding function was added that would allow a kilobot to resend a message not intended for it. A check was introduced to prevent forwarded messages from flooding the system and a flag was used to disable the forwarding function to allow them to be cleaned from the system. Current networking practices use a “time to live” value that is decremented until the message dies out naturally; this method was not implementable in our system because of limitations imposed by the kilobot message payload size.

The test on the physical kilobots was met with unforeseen obstacles that negatively affected the overall performance of the movement and placement functions. In several iterations of testing the communication between the kilobots was unstable leading to node placements that were not defined by the placement algorithms. Distances had a margin of error greater than in the simulation and the stability of the messages between bots may have been compromised by outside factors and signals.

IV. Results

The resulting work from the application of the project in a simulated environment demonstrated the conceptual belief on a series of algorithms and ideas strewn together to form a cohesive plan for node placement. The simulation revealed that the overall idea is feasible and realizable. Through a series of trials, the limitations of the idea was thought to be revealed and resolved.

The transference from a simulated environment to a physical world, outside variables started to show that further work needed to be done. The idea behind the algorithms were used and demonstrated on the kilobots in piecemeal, however the overall system confirmed not all aspects were accounted for. Differences between the simulation and physical world became apparent when compared side by side and the changes between the two were glaring.

V. Conclusion and Future Work

When using the Kilobots for implementation a difference was determined between real world and simulated application. During the simulation it was determined that all debugging and testing was done under ideal scenarios as well as stable communication for messages. All variables were able to be accounted for in its development. When applying the ideal scenario to the real world, the algorithms did not account for the actual movement of the bots. Nor did the system account for the interference with message passing factor in the surface material that the bots were placed on. Further issues were prevalent during the actual use of movement for the bots; the team did not have enough experience. The aforementioned issues compounded when the team had a lack of visibility into the internal logic of the system due to lack of debugging tools, and the inherent nature of distributed systems at a debug phase.

With some planning, some of the above issues could be mitigated, but not resolved. For a stabilization of the bots in a real-world the environmental factors could be monitored to an observer estimation controller. Such a controller would allow the bots to estimate its current state and disturbances to stabilize the movement and decision making to behave similar to the simulation. The controller would have an additional advantage through the modeling of the bots behaviors and allowing for some outside effects to be reduced. Movement could be further stabilized through the characterization of each motor used on every bot. The characterization of the individual bot’s motors would allow for the calibration to be performed to provide the optimal power for movement. From motor characterization, the system would further benefit from modeling the message passing and possible disturbances that could occur. Fault tolerance would then be built into code base; the aftereffect would be a more robust and accurate communication system. The system would have a reduce interference from the surfaces used and more reliable behavior based on message passing. Finally, scalability can be improved through the use of fractal formation; allowing the nodes to naturally place themselves without a leader giving designated coordinates would imply an infinite growth in scalability.