A Distributed Spanning Tree Method for Extracting Systems and Environmental Information from a Network of Mobile Robots

Brent Beer

Mobile Robotics Lab
Department of Computer Science
SIU Edwardsville
Edwardsville, IL 62026-1656
bbeer@siue.edu

Ross Mead

Interaction Lab
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0641
rossmead@usc.edu

Jerry B. Weinberg

Mobile Robotics Lab
Department of Computer Science
SIU Edwardsville
Edwardsville, IL 62026-1656
jweinbe@siue.edu

Abstract

A multi-robot system, like a robot formation, contains information that is distributed throughout the system. As the collective increases in numbers or explores distant or difficult areas, obtaining collective situational awareness becomes critical. We propose a method for extracting system and environmental information distributed over a collective of robots.

Introduction and Background

Endsley (1988) defines *situational awareness* (SA) as "the perception of the elements in the environment within a volume of space and time, comprehension of their meaning, and the projection of their status in the near future." A robot in a collection of robots, which are coordinating as a swarm (c.f., McLurkin, 2004) or a formation (c.f., Fredslund & Mataric, 2002), can ascertain information about itself and its immediate surroundings. This individual robot SA is useful for tasks, such as maintaining relationships within the collective. However to accomplish group level tasks, such as the collective efficiently pathing around obstacles, the SA of the entire assemblage of robots is necessary; we term this *collective situational awareness* (CSA). CSA can be used to issue commands across the collective to react to the environment or change behaviors to carry out a task. It can also provide human operators with environmental information to create a comprehensive survey.

CSA is attained through the accumulation of information embodied in individual SA, which is naturally distributed information across the collective. Previous work in CSA has focused on recognizing boundaries of the collective. Fekete *et al.* (2005) develops a heuristic to estimate the size of the boundary graphs based on the observation that interior boundaries are often smaller than outer boundary. McLurkin & Demaine (2009) implements A method for boundary detection where a robot determines a "local boundary classification" based on its sensor field-of-view—if no other units are located in this region, the robot is classified as a "boundary"; otherwise, the robot is classified as "non-boundary".

Our approach to CSA works with a formation of robots in which neighborhood relationships are established and maintained between two or more robots (Mead *et al.*, 2009). Having this relationship allows the application of a spanning tree-based algorithm applied similarly to communication networks for optimizing message passing (Pendarkis, 2001).

Algorithm

We define the collective of N robots as a graph $G = \{V, E\}$, in which each robot i is represented as a vertex v_i and communication between robots i and j is represented as an edge $e_{ij} = (i, j)$; therefore, $V = \{v_0, v_1, v_2, \ldots, v_N\}$ and $E = \{e_{ij}, \forall(i, j)\}$. We denote the set of all vertices adjacent to v_i as $[v_i] = \{v_j, \text{for all } e_{ij} \mid i\}$.

We define a message M as having: request data, a request transformation function, a processing function, a response transformation function, response data, and a hop count. *Request data* (from the sender or *parent*), req_{parent}, are input parameters necessary for the receiver (or *child*) to process the message. A *request transformation function*—denoted $req\_trans(req_{parent})$—is used to modify (if necessary) data in the received request; its output contains the appropriate request data, req_{child}, to broadcast to the

vertices adjacent to the child, $[v_{child}]\setminus\{v_{parent}\}$ (i.e., all except for the parent). The bulk of message processing operations is conducted in the *process function*, $proc(req_{parent}, req_{child})$; this function may use information from the parent's request, req_{parent}, as well as the subsequent request from the receiver, req_{child}. The process function returns response data for the child only, denoted res_{child}. A *response transformation function*—denoted $res\_trans(res_{child}, \{res_j, \forall j \mid v_j \in [v_{child}]\})$—is used to process the responses of all relevant adjacent vertices, as well as res_{child}, to produce the appropriate response data for the sender, denoted res_{parent}.

The *hop count* corresponds to the graph distance—$d(i, j)$—of this message from its initiator (referred to as the *message root*, v_{root}); we write the hop count at the parent and child as $hop_{parent} = d(root, parent)$ and $hop_{child} = d(root, child) = hop_{parent} + 1$, respectively. These hop counts are used to discriminate between values that will be factored into the response transformation function. Subsequently, hop_{child} is also used to identify the appropriate parent (if there are multiple) to which the child will respond; specifically, the receiver will only respond to a single parent whose message contained a minimum hop count (of all parents).

By selecting a single parent, we eliminate cycles in message responses throughout G; topologically, the set of response paths (edges), E', can be viewed as a tree, in which E' spans G, with v_{root} as its root. The approach is distributed, and does not require a stored representation of the tree itself; however, it can be shown that the selection of v_{root} impacts the performance of message passing. Specifically, the resulting spanning tree minimizes the hop count from v_{root} to all leaf nodes, and is, thus, considered minimal with respect to the operation, but not necessarily with respect to the graph as a whole. Strategic selection of v_{root} for a given message is crucial in its performance, and will be analyzed in subsequent studies.

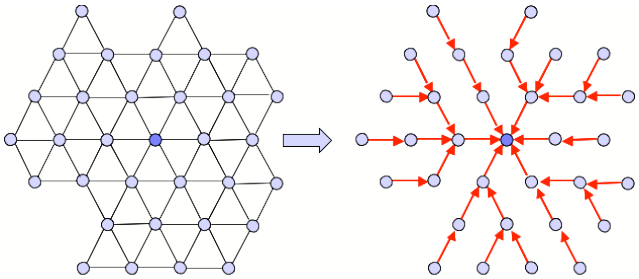

Figure 1: In response to a message originating from v_{root} (dark blue; left), child vertices respond to their respective parent vertices; the accumulated message paths of all responses form a spanning tree subgraph (right).

Implementation and Applications

The approach was implemented in simulation (Beer *et al*., 2010); details of messages useful for CSA can be found at http://roboti.cs.siue.edu/projects/formations/props_ops.php. Real-world applications could benefit from collective situational awareness such as volumetric monitoring, environmental exploration, and target tracking. For example, consider the Spring 2010 Deepwater Horizon oil spill in the Gulf of Mexico (Avery, 2010); a large collection of robots could be deployed to monitor the location and distribution of the oil in the region, and perhaps traverse the spill for extraction and cleanup. Though data is sampled and shared locally, the distributed spanning tree communication method provides a means to propagate and process this data to a collective situational awareness a guide group-level control strategies.

References

Avery, H. 2010. The Ongoing Administration-Wide Response to the Deepwater BP Oil Spill. Whitehouse.gov. http://www.whitehouse.gov/blog/2010/05/05/ongoing-administration-wide-response-deepwater-bp-oil-spill. Posted May 5, 2010. Retrieved October 8, 2010.

Beer, B., Mead, R., & Weinberg, J.B. 2010. A Distributed Method for Evaluating Properties of a Robot Formation. Student Abstract and Poster Program of the 24th AAAI Conference on Artificial Intelligence (AAAI-10), Atlanta, Georgia.

Endsley, M.R. 1988. Situation Awareness Global Assessment Technique (SAGAT). Proceedings of National Aerospace and Electronics Conference, 789-795.

Fekete, S.P., Kaufmann, M., Kröeller, A., & Lehmann, K. 2005. A New Approach for Boundary Recognition in Geometric Sensor Networks. Canadian Conference on Computational Geometry, 84-87.

Fredslund, J. & Matarić, M.J. 2009. Robots in Formation Using Local Information. 7th International Conference on Intelligent Autonomous Systems, Marina Del Rey, California, 100-107.

McLurkin, J.D. 2004. *Stupid Robot Tricks: A Behavior-Based Distributed Algorithm Library for Programming Swarms of Robots*. Master's Thesis, Massachusetts Institute of Technology.

McLurkin, J.D. & Demaine, E.D. 2009. A Distributed Boundary Detection Algorithm for Multi-Robot Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2009), St. Louis, Missouri.

Mead, R., Long, R., & Weinberg, J.B. 2009. Fault-Tolerant Formations of Robots. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2009), St. Louis, Missouri, 4805-4810.

Pendarakis, D., Shi, S., Verma, D., & Waldvogel, M. 2001. ALMI: An Application Level Multicast Infrastructure. 3rd Usenix Symposium on Internet Technologies & Systems (USITS).