2005 American Control Conference
June 8-10, 2005. Portland, OR, USA

Cooperative Hybrid Control of Robotic Sensors for Perimeter Detection and Tracking

Justin Clark and Rafael Fierro

Abstract— **In this paper, we present a decentralized coordination algorithm that allows a robotic swarm to locate and track a dynamic perimeter. A cooperative communication scheme is used by the team to rapidly detect a perimeter. Collision-free cycling behavior emerges by composing simple reactive control laws. The decentralized framework could potentially allow the algorithm to scale to many robots. Extensive simulation results and experiments verify the validity and scalability of the proposed cooperative control scheme.**

I. INTRODUCTION

Many applications of multi-robot cooperation have been studied including area coverage, search and rescue, manipulation, exploration and mapping, and perimeter detection [1], [2], [3], [4], [5], [6], [7]. The reader is referred to [8] for a survey on cooperative mobile robotics.

Perimeter detection has a wide range of uses in several areas, including: (1) Military, *i.e.*, locating minefields or surrounding a target, (2) Nuclear/Chemical Industries, *i.e.*, tracking radiation/chemical spills, (3) Oceans, *i.e.*, tracking oil spills, and (4) Space, *i.e.*, planetary exploration. In many cases, humans are used to perform these dull and/or dangerous tasks, but if robotic swarms could replace humans, it could be beneficial.

A perimeter is an area enclosing some type of substance. Consider two types of perimeters: (1) static and (2) dynamic. A static perimeter does not change over time, *i.e.*, possibly a minefield. Dynamic perimeters are time-varying and expand/contract over time, *i.e.*, a radiation leak.

In perimeter detection tasks, a robotic swarm locates and surrounds a substance, while dynamically reconfiguring as additional robots locate the perimeter. Obviously, the robots must be equipped with sensors capable of detecting whatever substance they are trying to track. Substances could be airborne, ground-based, or underwater. If the perimeter moves with a velocity greater than the robots can move, then the perimeter cannot be tracked. Abrupt perimeter changes requiring sharp turns may be difficult to track because of the robots' limited turning radius. See Fig. 1 for an example of a perimeter, an oil spill.[1]

The first author is supported in part by NASA Oklahoma Space Grant Consortium Fellowships. This work is partially supported by NSF grants #0311460 and CAREER #0348637 and by the U.S. Army Research Office under grant DAAD19-03-1-0142 (through the University of Oklahoma).

Authors are with the MARHES Laboratory, School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078, USA {justin.clark, rfierro}@okstate.edu

[1]Courtesy of the NOAA Office of Response and Restoration

Fig. 1. Example perimeter: Oil spill.

A variety of perimeter detection and tracking approaches have been proposed in the literature. Bruemmer, *et al.*, present an interesting approach in which a swarm is able to autonomously locate and surround a water spill using social potential fields implemented with IR, chirps, and light sensing [1]. Authors in [4] show outdoor perimeter surveillance over a large area using a swarm that investigates alarms from intrusion detection sensors. Marthaler and Bertozzi develop a *snake* algorithm to locate and surround a perimeter represented by a concentration function [9]. In [10], potential fields are used to allow a swarm to uniformly surround a target, while avoiding obstacles and evading threats. Authors in [11] use mobile sensing nodes to estimate dynamic boundaries.

In this paper, a decentralized, cooperative hybrid system is presented utilizing biologically-inspired emergent behavior. Each controller is composed of finite state machines and it is assumed that the robots have a suite of sensors and can communicate only within a certain range. A relay communication scheme is used. Once a robot locates the perimeter, it broadcasts the location to any robots within range. As each robot receives the perimeter location, it too begins broadcasting, in effect, forming a relay. Other groups have used the terms *perimeter* and *boundary* interchangeably, but in this paper, there is a distinct difference. The perimeter is the *chemical substance* being tracked, while the *boundary* is the limit of the exploration area.

The rest of the paper is organized as follows: Section 2 details the cooperative hybrid controller. Section 3 presents simulation and experimental results. In Section 4, we give concluding remarks and future directions.

0-7803-9098-9/05/$25.00 ©2005 AACC

II. COOPERATIVE HYBRID CONTROLLER

The last few years have seen active research in the development of architectures for multi-robot coordination. These architectures have focused on providing different capabilities to the group of robots. For instance, ALLIANCE [12], a behavior-based software architecture, has focused on fault-tolerant cooperative control.

The theory of hybrid systems [13] offers a convenient framework to model the multi-robot system engaged in a perimeter detection and tracking task. In our previous work [14], we developed an object-oriented software architecture that supports hierarchical composition of robot agents and behaviors or modes. Key features of the software architecture are summarized below.

- **Architectural hierarchy:** The building block for describing the system architecture is an *agent* that communicates with its environment via shared variables and also communication channels. In this application, the team of mobile sensors defines the *group* agent. The group agent receives information about the area *i.e.*, boundary where the perimeter is located.
- **Behavioral hierarchy:** The building block for describing a flow of control inside an agent is a *mode*. A mode is basically a hierarchical state machine, that is, a mode can have submodes and transitions connecting them. Modes can be connected to each other through entry and exit points. We allow the instantiation of modes so that the same mode definition can be reused in multiple contexts.
- **Discrete and Continuous variable updates:** Discrete updates are specified by *guards* labeling transitions connecting the modes. Such updates correspond to mode-switching, and are allowed to modify variables through assignment statements.

The state of a robot agent is given by $\boldsymbol{x} \in \mathbb{R}^n$, its evolution is determined by a set of differential equations:

$$\dot{\boldsymbol{x}} = f_q(\boldsymbol{x}, \boldsymbol{u}) \tag{1}$$

$$\boldsymbol{u} = k_q(\boldsymbol{x}, \boldsymbol{z}) \tag{2}$$

where $\boldsymbol{u} \in \mathbb{R}^m$ is the control vector, $q \in \mathbb{Q} \subset \mathbb{Z}$ is the control mode for the node, $\mathbb{Q}$ is a finite set of control mode indices, $\mathbb{Z}$ denotes the set of positive integers, and $\boldsymbol{z} \in \mathbb{R}^p$ is the information about the external world available either through sensors or through communication channels. The robot agent contains modes describing behaviors that are available to the robot.

For simplicity, the MARHES car-like platform was modeled with the unicycle model:

$$\begin{aligned} \dot{x}_i &= v_i \cos\theta_i \\ \dot{y}_i &= v_i \sin\theta_i \\ \dot{\theta}_i &= \omega_i \end{aligned} \tag{3}$$

where x_i, y_i, θ_i, v_i, and ω_i are the x-position, y-position, orientation angle, linear velocity, and angular velocity of robot i, respectively. Note that v_i ranges from $-2 \leq v_i \leq 2\ m/s$, while ω_i ranges from $-0.3 \leq \omega_i \leq 0.3\ rad/s$. These ranges come from extensive tests of our platform. Refer to Fig. 2 for a view of the platform and model.

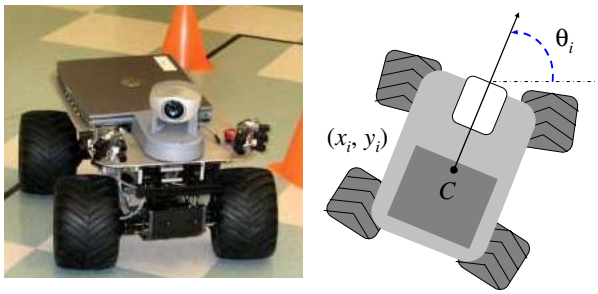

Fig. 2. (a) Platform and (b) Unicycle model.

The overall finite automaton consists of three states: (1) *Random Coverage Controller* (RCC), (2) *Potential Field Controller* (PFC), and (3) *Tracking Controller* (TC). These three controllers are composed such that the sensor/robot network is able to locate and track a perimeter. See Fig. 3 for a hierarchical state diagram of the cooperative hybrid system developed herein. In the next section, details of the

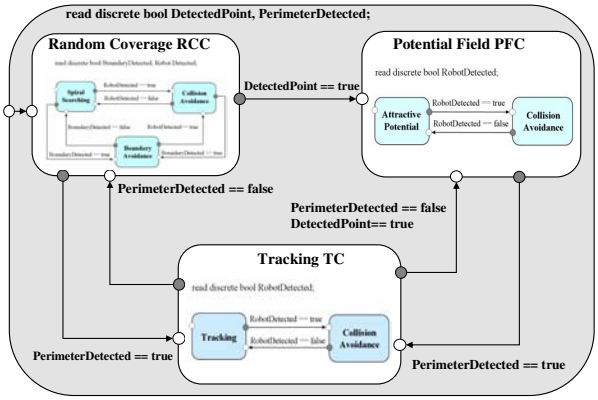

Fig. 3. Overall finite automaton.

controller agents are presented.

A. Random Coverage Controller

The goal of the Random Coverage Controller (RCC) is to efficiently cover as large an area as possible while searching for the perimeter and avoiding collisions. The robots move fast in this state to quickly locate the perimeter. The RCC consists of three states: (1) *spiral search*, (2) *boundary avoidance*, and (3) *collision avoidance*. The spiral search is a random search for effectively covering the area. The boundary and collisions are avoided by adjusting the angular velocity.

The logarithmic spiral, seen in many instances in nature, is used for the search pattern. In [15], a spiral search pattern such as that used by moths is utilized for searching an area.

It has been shown that the spiral search is not optimal, but effective [16]. Some examples are hawks approaching prey, insects moving towards a light source, sea shells, spider webs, and so forth.

Specifically, the linear and angular velocity controllers are:

$$v_i = v_s\left(1 - e^{-t}\right) \tag{4}$$

$$\omega_i = ae^{b\theta_i} \tag{5}$$

where v_s is a positive constant, a is a constant and $b > 0$. If $a > 0$ (< 0), then the robots move counterclockwise (clockwise). Collision and *boundary* (limit of the exploration area here) avoidance are handled in simulation by sharply turning, while in experiments, the robots back up and turn, then go forward.

B. Potential Field Controller

Potential fields have been used by a number of groups for controlling a swarm [10], [7], [6], [1], [3]. In [7], a method using artificial potentials and virtual bodies is shown in which the robot network forms regular polygons upon uniformly surrounding a target. In [6], virtual potential fields and graph theory are used for area coverage. In [3], a potential field method is used that is inspired by an algorithm observed in wolf packs.

The Potential Field Controller (PFC) uses an attractive potential which allows the robots to quickly move to the perimeter once it has been detected. The first robot to detect the perimeter *broadcasts* its location to the other robots. If a robot is within range, then the PFC is used to quickly move to the perimeter. Otherwise, the robot will continue to use the RCC unless it comes within range, at which point it will switch to the PFC. As a robot moves towards the goal, if it detects the perimeter before it reaches the goal, it will switch to the TC. Note that the robots move with constant linear speed when using the PFC. The PFC has two states: (1) *attractive potential* and (2) *collision avoidance*.

The attractive potential, $\mathbf{P_a}(x_i, y_i)$, is [17]:

$$\mathbf{P_a}(x_i, y_i) = \frac{1}{2}\epsilon[(x_i - x_g)^2 + (y_i - y_g)^2], \tag{6}$$

where (x_i, y_i) is the position of robot *i*, ϵ is a positive constant, and (x_g, y_g) is the position of the attractive point (goal). The attractive force, $\mathbf{F_a}(x_i, y_i)$, is derived below.

$$\mathbf{F_a}(x_i, y_i) = -\nabla\mathbf{P_a}(x_i, y_i) = -\begin{bmatrix} \frac{\partial \mathbf{P_a}}{\partial x_i} \\ \frac{\partial \mathbf{P_a}}{\partial y_i} \end{bmatrix}$$

$$\mathbf{F_a}(x_i, y_i) = \epsilon\begin{bmatrix} x_g - x_i \\ y_g - y_i \end{bmatrix} = \begin{bmatrix} F_{a,x_i} \\ F_{a,y_i} \end{bmatrix} \tag{7}$$

Equation (7) is used to get the desired orientation angle, $\theta_{i,d}$, of robot *i*:

$$\theta_{i,d} = \arctan 2(F_{a,y_i}, F_{a,x_i}) \tag{8}$$

Depending on θ_i and $\theta_{i,d}$, the robot will turn the optimal direction to quickly line up with the goal using the following proportional angular velocity controller:

$$\omega_i = \pm\ k\ (\theta_{i,d} - \theta_i), \tag{9}$$

where $k = \frac{\omega_{max}}{2\pi}$ and $\omega_{max} = 0.3\ rad/s$ and θ_i is the orientation angle of robot i.

Collisions are avoided in the same manner as in the RCC.

C. Tracking Controller

The Tracking Controller (TC) changes ω and v and in order to track the perimeter and avoid collisions, respectively. Cyclic behavior *emerges* as multiple robots track the perimeter. In [18], cyclic pursuit is presented in which each robot follows the next robot (1 follows 2, 2 follows 3, etc.). The robots move with constant speed and a proportional controller is used to handle orientation. Note that collisions are ignored. The TC differs from [18] in that each robot's objective is to track the perimeter, while avoiding collisions. There are no restrictions on robot order and v is not constant.

The robots' goal in this state is to accurately track the perimeter counterclockwise. The TC consists of two states: (1) *tracking*, and (2) *collision avoidance*. Tracking is accomplished by adjusting the robots' angular velocity. On the other hand, collisions are avoided by changing the linear velocity.

III. RESULTS

The hybrid system has been verified in Matlab, Gazebo, and in experiments, which are described below.

A. Matlab

A Matlab[2] simulation is shown in Fig. 4 with a dynamic perimeter (expanding at 12.5 mm/s). Notice that the robots

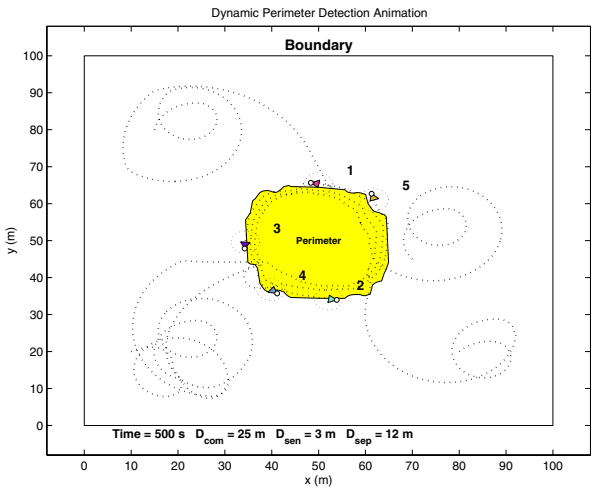

Fig. 4. Matlab simulation showing five robots tracking a dynamic perimeter.

[2]http://www.mathworks.com

are still able to track the expanding perimeter. The bang-bang angular velocity controller is:

$$\omega_i = \begin{cases} -\omega_t & \text{inside perimeter} \\ \omega_t & \text{outside perimeter,} \end{cases} \quad (10)$$

where $\omega_t = 0.1\ rad/s$. A look-ahead distance is defined to be the sensor point directly in front of the robots. If the omnidirectional sensor has detected the perimeter and the look-ahead distance is inside the perimeter, then the robot will turn right. Otherwise, the robot turns left. This zigzagging behavior is often seen in moths following a pheromone trail to its source [19].

B. Gazebo

A Gazebo [20] experiment was developed to verify the simulation results from Matlab. Our platform, the Tamiya TXT-1, has similar characteristics to the ClodBuster model in Gazebo. Each robot is equipped with odometers, a pan-tilt-zoom camera, and a sonar array. Position and orientation are estimated using the odometers. The camera is used for tracking the perimeter. It is pointed down and to the left on each robot to allow the robots to track the perimeter at a small offset. The sonar array is used to avoid collisions.

A simulation is shown in Fig. 5 in which a robot searches for, locates, and tracks a perimeter while avoiding collisions. An environment was created to represent an oil spill in a grassy area. The perimeter and boundary are represented by a black cylinder (oil) and gray walls, respectively.

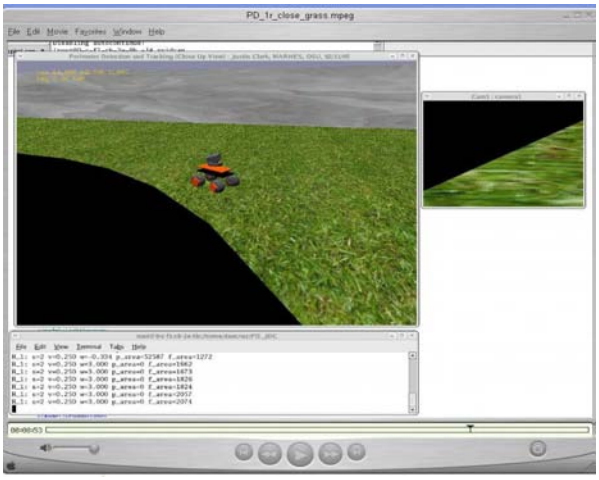

Fig. 5. Gazebo simulation showing camera view on the right.

Gazebo allows real world problems to be debugged that could be difficult to model in Matlab, *i.e.*, a blobfinder. For example, in Matlab a binary sensor (bang-bang controller) was used to track the perimeter. In Gazebo, a blobfinder (proportional controller) was used which allowed for much smoother tracking. In fact, a nearly identical proportional controller was implemented for experiments.

To the best of our knowledge, dynamic perimeters can not be modeled yet in Gazebo. When simulating multiple robots with textures, *i.e.*, the grass and the wall, the simulations started to slow down, whereas without textures, the simulations ran at close to real-time. Gazebo is becoming a powerful tool for verifying cooperative control systems as processing power increases and more models are defined.

C. Experimental

The MARHES multi-vehicle testbed consists of ten Tamiya TXT-1 R/C trucks. Each truck is capable of speeds up to $2\ m/s$ in either direction, has double-steering, and can be operated indoors or outdoors. A suite of sensors are available including a stereo-vision system, a Global Positioning System (GPS), an Inertial Measurement Unit (IMU), IR, odometer wheel sensors, and ultrasonic rangefinders. Each robot is equipped with wireless communication capabilities and an embedded computer (*e.g.*, PC-104 or laptop) running in Linux for high-level control. Low-level control is handled through the Controller Area Network (CAN) sensor/control robot interface.[3]

In Gazebo and in these experiments, an inexpensive camera (blobfinder algorithm) is being used to detect the perimeter. Smooth tracking is accomplished with the following proportional angular velocity controller:

$$\omega_i = k_P\ (\gamma_o - \gamma_i), \quad (11)$$

where $k_P > 0$, and γ_o and γ_i are the areas outside the perimeter and inside the perimeter seen by the blobfinder, respectively. Counterclockwise tracking is assumed which implies that the robot will turn left (right) if the robot is too far outside (inside) the perimeter.

An experiment is shown in Fig. 6 in which three robots search for, locate, and track a perimeter while avoiding collisions. Refer to Figs. 7 and 8 for trajectory and state transitions plots, respectively. Collision avoidance is accomplished through the use of IR sensors while position/orientation information comes from the encoders. R_1 locates the perimeter first and begins tracking. It broadcasts its location to the other robots, who upon receiving the location, enter the PFC. R_2 locates the perimeter next, followed by R_3. Notice in Fig. 7 that the perimeter is not exactly like the perimeter in Fig. 6, but it is fairly

Fig. 6. Indoor experimental setup for perimeter detection.

[3] http://marhes.okstate.edu/

accurate and allows the user to infer the location of the perimeter. Refer to Figs. 9 and 10 for plots of the linear

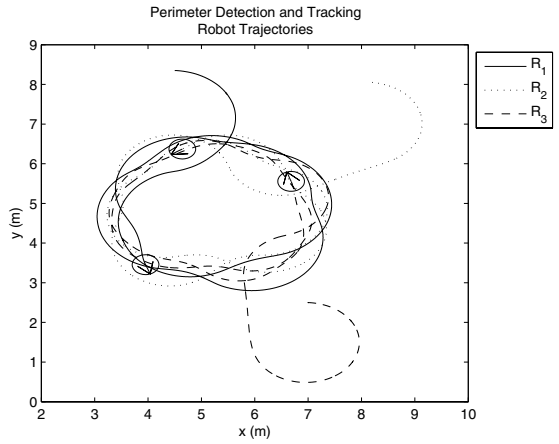

Fig. 7. Three robots defining a perimeter.

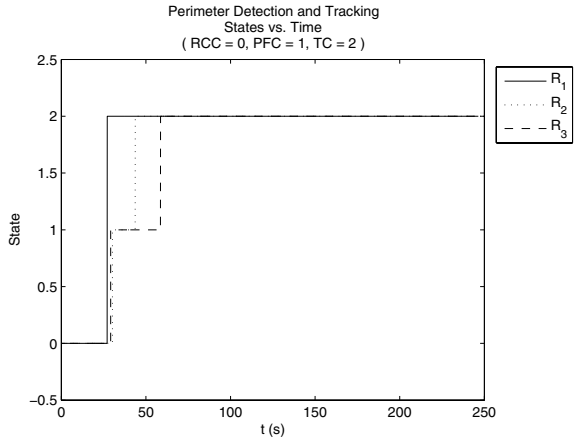

Fig. 8. Discrete state transition plot.

and angular velocities, respectively. Notice that the robots move with almost constant linear speed and the angular speed is sinusoidal, indicating the robots are tracking the perimeter. The large spike in Fig. 9 at approximately $55\ s$ occurs when a robot detects an obstacle and starts to move in reverse.

To verify the accuracy of the low-level control system, the errors between the actual and desired velocities were plotted in Figs. 11 and 12. Note in Fig. 11 that the error is relatively small the majority of the time, except when a robot goes in reverse to avoid an obstacle, *i.e.*, at $55\ s$. There will almost always be a moderate amount of error in the angular velocity because the robots are constantly switching in an effort to track the perimeter.

A second experiment was run in which all of the robots were placed next to the perimeter. The robots start tracking the perimeter. After one lap or so, a section of the perimeter is removed to show that the swarm can still track this

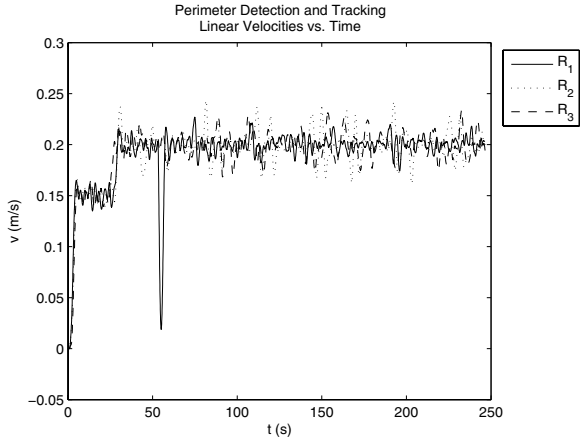

Fig. 9. Robots tracking with nearly constant speed.

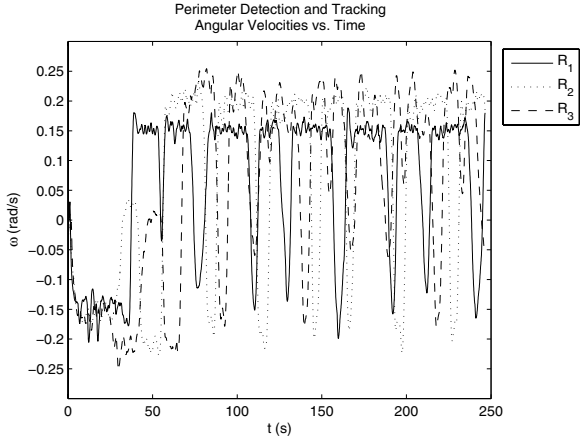

Fig. 10. Angular velocities become sinusoidal indicating the robots are tracking the perimeter.

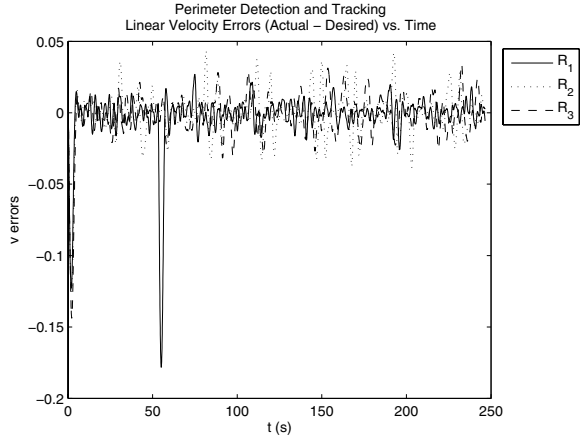

Fig. 11. Accurate linear velocity provided by low-level control unit.

dynamic perimeter. Refer to Fig. 13 for a trajectory plot. Note that the robots were able to adjust as a section of the perimeter was removed. Also, R_1 ran low on batteries

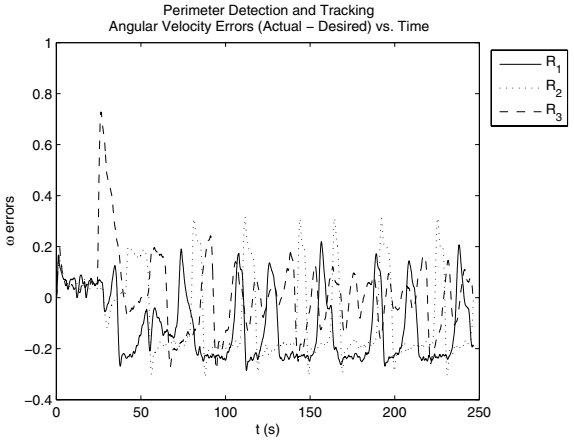

Fig. 12. Moderate error indicating the robots are constantly turning in an effort to track the perimeter.

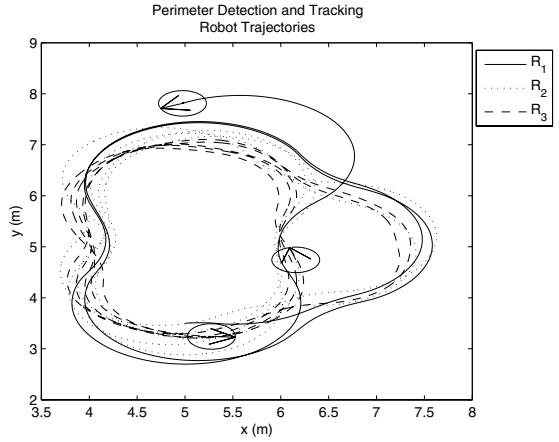

Fig. 13. Three robots tracking a perimeter where part of the perimeter is removed. R_1 runs low on batteries and stops.

and began to have trouble tracking the perimeter until it completely stopped.

IV. CONCLUSIONS

A decentralized, cooperative hybrid system was shown that allows a group of *nonholonomic* robots to search for, detect, and track a dynamic perimeter with limited communication, while avoiding collisions and reconfiguring *on-the-fly* as additional robots locate the perimeter or the perimeter shape changes. The algorithm has been extensively tested in Matlab, Gazebo, and experimentally.

Currently, the performance of the team is being evaluated based on different communication schemes. Also, methods are being investigated to estimate the dynamic perimeter as it evolves and to reduce the searching time.

ACKNOWLEDGMENTS

We would like to thank Daniel Cruz for his help with the Gazebo simulations, the mobile platform, and the experiments. Also, we thank Chris Flesher and Omar Orqueda for their help with vision and Kenny Walling for designing the CAN control unit and CAN sensor nodes.

REFERENCES

[1] D. J. Bruemmer, D. D. Dudenhoeffer, M. D. McKay, and M. O. Anderson, "A robotic swarm for spill finding and perimeter formation," in *Spectrum 2002*, Reno, Nevada USA, August 2002.

[2] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2002, pp. 1327–1332.

[3] H. V. D. Parunak, S. A. Brueckner, and J. Odell, "Swarming pattern detection in sensor and robot networks," in *ANS 10th Int. Conf. on Robotics and Remote Systems for Hazardous Environments*, Gainesville, Florida USA, March 28-31 2004.

[4] J. T. Feddema, C. Lewis, and D. A. Schoenwald, "Decentralized control of cooperative robotic vehicles: Theory and application," *IEEE Trans. on Robotics and Automation*, vol. 18, no. 5, pp. 852–864, October 2002.

[5] R. Fierro, P. Song, A. Das, and V. Kumar, "Cooperative control of robot formations," in *Cooperative Control and Optimization*, R. Murphey and P. Pardolos, Eds. Kluwer Academic Press, 2002, vol. 66, ch. 5, pp. 73–93.

[6] J. Tan and N. Xi, "Peer-to-peer model for the area coverage and cooperative control of mobile sensor networks," in *SPIE Symposium on Defense and Security*, Orlando, Florida USA, April 12-16 2004.

[7] P. Ögren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Trans. on Automatic Control*, vol. 49, no. 8, pp. 1292–1302, August 2004.

[8] Y. U. Cao, A. S. Fukanaga, and A. Kahng, "Cooperative mobile robotics: Antecedents and directions," in *Autonomous Robots*, vol. 4, no. 1. Kluwer Academic Publishers, March 1997, pp. 7–27.

[9] D. Marthaler and A. L. Bertozzi, "Tracking environmental level sets with autonomous vehicles," in *Recent Developments in Cooperative Control and Optimization*. Kluwer Academic Publishers, 2004.

[10] J. S. Baras, X. Tan, and P. Hovareshti, "Decentralized control of autonomous vehicles," in *Proc. IEEE Conf. on Decision and Control*, Maui, Hawaii USA, 2003, pp. 1532–1537.

[11] A. Savvides, J. Fang, and D. Lymberopoulos, "Using mobile sensing nodes for boundary estimation," in *Workshop on Applications of Mobile Embedded Systems*, Boston, Massachusetts, June 2004.

[12] L. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *IEEE Trans. on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[13] R. Alur, T. Dang, J. Esposito, R. Fierro, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G. Pappas, and O. Sokolsky, "Hierarchical hybrid modeling of embedded systems," in *EMSOFT 2001*, T. Henzinger and C. Kirsch, Eds. Berlin Heidelberg: Springer-Verlag, 2001, vol. 2211 of LNCS, pp. 14–31.

[14] R. Fierro, A. Das, J. Spletzer, J. Esposito, V. Kumar, J. P. Ostrowski, G. Pappas, C. J. Taylor, Y. Hur, R. Alur, I. Lee, G. Grudic, and J. Southall, "A framework and architecture for multi-robot coordination," *Int. J. Robot. Research*, vol. 21, no. 10-11, pp. 977–995, October-November 2002.

[15] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Distributed odor source localization," *IEEE Sensors*, vol. 2, no. 3, pp. 260–271, 2002, special Issue on Artificial Olfaction.

[16] D. W. Gage, "Randomized search strategies with imperfect sensing," in *Proceedings of SPIE Mobile Robots VIII*, vol. 2058, Boston, Massachusetts USA, September 1993, pp. 270–279.

[17] B. Bayazit, "Potential field methods," Washington University in St. Louis, St. Louis, Missouri USA, Tech. Rep., 2003, course Notes CS 522A.

[18] J. A. Marshall, M. E. Broucke, and B. A. Francis, "Unicycles in cyclic pursuit," in *Proc. American Control Conference*, Boston, Massachusetts USA, June 30-July 2 2004, pp. 5344–5349.

[19] S. J. Phillips and P. W. Comus, *A Natural History of the Sonoran Desert*. Arizona-Sonora Desert Museum: University of California Press, December 1 1999.

[20] R. T. Vaughn, B. P. Gerkey, and A. Howard, "On device abstractions for portable, reusable robot code," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Las Vegas, Nevada USA, October 2003, pp. 2121–2427.