The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Multi-Robot Boundary Tracking with Phase and Workload Balancing

Michael Boardman, Jeremy Edmonds, Kyle Francis and Christopher M. Clark

Abstract— **This paper discusses the use of a cooperative multiple robot system as applied to distributed tracking and sampling of a boundary edge. Within this system the boundary edge is partitioned into subsegments, each allocated to a particular robot such that workload is balanced across the robots. Also, to minimize the time between sampling local areas of the boundary edge, it may be desirable to minimize the difference between each robots progression (i.e. phase) along its allocated sub segment of the edge. The paper introduces a new distributed controller that handles both workload and phase balancing. Simulation results are used to illustrate the effectiveness of the controller in an Autonomous Underwater Vehicle (AUV) under ice edge sampling application. Successful results from experimentation with three iRobot Creates are also presented.**

I. INTRODUCTION

Robots are increasingly being used to perform a large variety of tasks. Commercial applications give robots the ability to assist the disabled, clean homes, and aid in the manufacturing and processing of products. Military applications give robots additional purpose. They can scour fields for mines, search for snipers in urban combat environments, and even maintain full battlefield awareness for soldiers. They also have great potential in scientific exploration. They have the capability to withstand harsh and unforgiving environments thereby giving them the ability to perform tasks humans cannot perform.

In single robot systems, there is a higher likelihood of mission failure. If the single robot fails, the mission fails. Further, a single robot can only cover so much area in a given time frame. Multi-robot systems typically do not suffer from such possibility of a single point failure. Multi-robot systems also allow for accomplishing tasks of larger size and complexity when compared with single robot systems.

This paper concerns the task of tracking and sampling the (possibly dynamic) boundary of some entity with multiple robots. This is useful for scientific, military, and even commercial applications. For example, boundary tracking can be used to track a crowd of people, survey an oil spill, or detect the edge of a harmful algae bloom with Autonomous Underwater Vehicles (AUVs).

Michael Boardman is a Graduate Student with the Department of Electrical Engineering, California Polytechnic State University, San Luis Obispo, CA 93407, USA mboardma@calpoly.edu

Jeremy Edmonds is an Instructor at the Electronics and Computer Technologies Department, Cuesta College, San Luis Obispo, CA 93405, USA jeremy@jeremyedmonds.com

Kyle Francis is an Undergraduate Student at the Engineering Department, Clark College, Vancouver, WA 98655, USA kyle.francis89@gmail.com

Christopher M. Clark is an Associate Professor with the Department of Computer Science, California Polytechnic State University, San Luis Obispo, CA 93407, USA cmclark@calpoly.edu

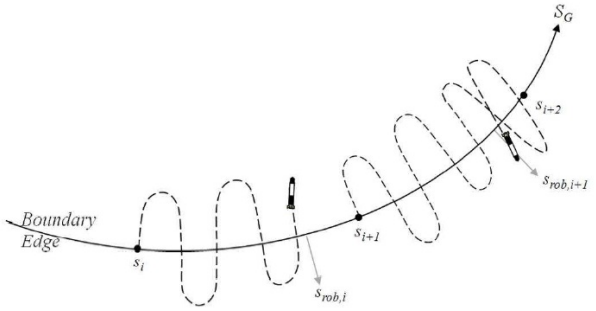


Fig. 1. A candidate coordinate system for distributed control of multiple robots on a boundary edge. In this example, multiple AUVs are distributed along an edge of ice rafts.

A goal of this research is to design a distributed controller in which multiple robots track and follow a continuous boundary edge, while balancing both the phase and workload between vehicles. Controlling the workload will allow the coverage along the boundary to be partitioned equally between the multiple robots. Meanwhile, the phase controller will ensure the robots are at the same location within their partition of the boundary. The robots are assumed to have nonholonomic kinematic constraints and have maximum velocity constraints.

In Section II, a background of multiple robot systems and boundary tracking is discussed. Section III defines specifics of the problem at hand. Section IV is an overview of the controller method used for the cooperative multiple robot system. In Section V, simulation and real world experimentation implementation and results are discussed. Finally, Section VI gives a conclusion of the information discussed in this paper as well as future work on the topic.

II. BACKGROUND

Several traditional robot navigation strategies have been applied to boundary tracking. To follow boundaries while avoiding obstacles, work in [1] used Artificial Potential Fields. Potential field systems can be used to guide a robot toward a target while avoiding obstacles. Though there is a risk of becoming caught in a local minima, measures can be made to reduce the risk.

Work in [7] describes a method of implementing a global path planner with local sensor data. Instant goals are used to set the path that the robot will follow along with a boundary following algorithm to maintain global boundary following and prevent local minima due to obstacles. This method may detract a robot from observing important elements during observation as an attempt to follow around obstacles.

In [3], the distance of the robot from a discovered boundary becomes minimized as it moves back and forth across a boundary. Multiple robots are used to collect and analyze information to follow the boundary closely in a convoy. The research in [3] did result in false positives due to noisy sensor readings. The focus of this paper was to make all robots traverse the entire area getting as close to the boundary as possible. Another method for coordinating the robots would be to divide the boundary area to be covered.

In [2], a collaborative path planning algorithm assigns one robot as a coordinator and all of the robots comprise a team. The coordinator guides the other robots toward a target. The follower robots are designed to follow the coordinator robot in order to give additional sensing of the environment to the follower robot. Making a single robot the coordinator can lend itself to issues caused by a single robots failure.

To fully cover an unknown area, simplices can partition a 2D region to be covered by multiple robots [6]. The simplices give a path that robots can take to cover the entire unknown area. The path created does not take into account spatiotemporal sampling needs. Further, partitioning the area into separate paths for multiple robots to follow could improve efficiency.

Using an auction system [5], robots are able to travel to all task points faster than just alternating the assignment of points. Though this motion planner can be utilized to patrol a boundary area, localized dynamic events may be missed when performing scientific missions requiring such observation.

The UUV-gas algorithm [8] can be used to perform boundary tracking comprised of circular motion. The focus of this multiple vehicle cooperative tracking is to prevent vehicle collision while having each vehicle following the same boundary. The circular motion in this algorithm is designed to allow the robot to travel only a set distance within the boundary region and outside the boundary region. Using circular motion only could prevent necessary coverage within the boundary region dependent on the application.

In [4], phase balancing is used to maintain distance between a fleet of AUV gliders along a set path. Similar phase balancing can be used for spatio-temporal sampling. This system has all robots covering the entire boundary edge, causing overlap of the same location multiple times. If this is combined with workload balancing, the multi-robot system can efficiently partition the coverage of a boundary area along with improving spatio-temporal sampling of the boundary edge.

III. PROBLEM DEFINITION

Consider a continuous edge segment E defined by two end points s_0 and s_n defined within a coordinate frame where the S_G axis that follows the edge. The problem is to partition E into n sub-segments, each of which is allocated to one of n robots that must track the sub-segment. Hence, the i^{th} robot is designated to sample an interval $\Delta s_i = s_{i+1} - s_i$ along the boundary edge between s_i and s_{i+1}. The n sub-segments may be of different lengths corresponding to the $workload$ associated with each. That is, it may desirable for some robots to have shorter sub-segments (or vice-versa) if the application requires slower tracking and hence slower progression along the edge. The boundary edge itself must be crossed to be detected. (e.g. detecting the edge of an oil spill with an AUV may require the AUV to fly into and out of the oil).

To permit repeated sampling measurements over time, the i^{th} robot will travel from s_i to s_{i+1}, and back to s_i. This motion will constitute one $cycle$, where the location of the robot within this cycle is referred to as the $phase$ ϕ_i and is measured in radians. The robot's phase ϕ_i relates to the position $s_{rob,i}$ along the S_G axis by:

$$\phi_i = \left\{ \begin{array}{ll} \pi \frac{s_{rob,i} - s_i}{\Delta s_i} & if \;\; \dot{s}_{rob,i} > 0 \\ \pi \frac{s_{i+1} - s_{rob,i}}{\Delta s_i} + \pi & else \end{array} \right\} \quad (1)$$

In tracking E, it is desirable to balance both $Workload$ and $Phase$.

A. Workload

In this paper, the workload is defined as the area covered by the robot as it traverses the edge. This area will be a function of the distance the robot travels perpendicular to the boundary edge. For example, consider an AUV traversing further under an ice sheet in response to the presence of ice algae growth. Sampling and observing such algae is important to biologists wishing to study the Arctic ecosystem and the effects of global climate change on such ice ecosystems.

Hence, to balance the workload, the error to minimize is the difference between each robot's area covered Ψ_i:

$$\begin{array}{ll} e_{\Psi,i} & = \Psi_{i+1} - \Psi_i \\ & = \int_{s_{i+1}}^{s_{i+2}} d_{des}(s) ds - \int_{s_i}^{s_{i+1}} d_{des}(s) ds \\ & \approx d_{a,i+1} \Delta s_{i+1} - d_{a,i} \Delta s_i \end{array} \quad (2)$$

In eq. 2, the area covered by a robot is approximated as a rectangle and calculated as the product of average depth $d_{a,i}$ and Δs_i. While the d_{des} is a function of the robot position $s_{rob,i}$ along the edge, the boundary values s_i can be controlled by the robot itself.

B. Phase

The second goal of this controller is to the improve the spatio-temporal sampling by minimizing the phase difference between AUVs. This will reduce the likelihood of missing a localized dynamic event. For example, security robots surveying a fenceline should stay in phase to limit the size of the gap between robots created when they move apart. The associated error to be minimized is:

$$e_{\phi,i} = \phi_{i+1} - \phi_i \quad (3)$$

IV. CONTROLLER DESIGN

Since the boundary values s_i can be controlled by the robots themselves, consider the dynamics of the boundary values to be modeled as in eq. 4, with a proposed control input $U_{s,t}$ in eq. 6.

$$S_{t+1} = S_t + U_{s,t} \tag{4}$$

where

$$S_t = \begin{bmatrix} s_0 & s_1 & \dots & s_n \end{bmatrix}_t \tag{5}$$

$$U_{s,t} = \begin{pmatrix} 0 & K_s e_{s,0} & \dots & K_s e_{s,n-2} & 0 \end{pmatrix}^T \tag{6}$$

To understand the error dynamics, consider a three AUV system in which $n = 3$. Considering eigen values of the transition matrix in eq. 7, the error dynamics can be proven stable for $K_s > 0$.

$$\begin{aligned} & E_{s,t+1} \\ &= \begin{pmatrix} e_{s,0} \\ e_{s,1} \end{pmatrix}_{t+1} \\ &= \begin{pmatrix} d_{a,0} & -d_{a,0} - d_{a,1} & d_{a,1} & 0 \\ 0 & d_{a,1} & -d_{a,1} - d_{a,2} & d_{a,2} \end{pmatrix} S_{t+1} \\ &= \begin{pmatrix} 1 - (d_{a,0} + d_{a,1})K_s & d_{a,1}K_s \\ d_{a,1}K_s & 1 - (d_{a,1} + d_{a,2})K_s \end{pmatrix} E_{s,t} \end{aligned} \tag{7}$$

While the controller operates using proportional feedback to control boundary coverage (workload balance), a feedback linearization controller is used for robot location (phase balance). The following phase dynamics were used:

$$\Phi_{t+1} = \Phi_t + \delta\Phi_{s,i,t} + U_{\phi,t} \tag{8}$$

where

$$\Phi_t = \begin{bmatrix} \phi_0 & \phi_1 & \dots & \phi_{n-1} \end{bmatrix}_t \tag{9}$$

As shown in eq 10, the proposed control input $U_{\Phi,t}$ for the t^{th} time step consists of several terms, the first of which incorporates the desired phase velocity $\dot{\phi}_{des}$ at which all robot's should maintain, once steady state is reached. Adding the second term $-\delta\phi_{s,i}$ implements feedback linearization to counter the change in phase caused by workload balancing, (i.e. change in s_i. The final term $K_\phi e_{\phi,i} - K_\phi e_{\phi,i-1}$ is used to minimize phase error.

$$U_{\phi,t} = \begin{pmatrix} \dot{\phi}_{des}\Delta t - \delta\phi_{s,0} + K_\phi e_{\phi,0} \\ \dot{\phi}_{des}\Delta t - \delta\phi_{s,1} + K_\phi e_{\phi,1} - K_\phi e_{\phi,0} \\ \dots \\ \dot{\phi}_{des}\Delta t - \delta\phi_{s,n-2} + K_\phi e_{\phi,n-2} - K_\phi e_{\phi,n-3} \\ \dot{\phi}_{des}\Delta t - \delta\phi_{s,n-1} - K_\phi e_{\phi,n-2} \end{pmatrix}_t \tag{10}$$

where

$$\delta\phi_{s,i} = \frac{s_{rob,i,t-1} - s_{i,t}}{\Delta_{i,t}} - \frac{s_{rob,i,t-1} - s_{i,t-1}}{\Delta_{i,t-1}} \tag{11}$$

The resulting error dynamics are shown below. For clarity, and without losing generalization, only the case with $n = 3$ is shown. This system is guaranteed stable if eigen values of the transition matrix are less than 1, requiring the stability condition $K_\phi < 2/3$.

$$\begin{aligned} & E_{\Phi,t+1} \\ &= \begin{pmatrix} e_{\phi,0} \\ e_{\phi,1} \end{pmatrix}_{t+1} \\ &= \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \Phi_{t+1} \\ &= \begin{pmatrix} 1 - 2K_\phi & K_\phi \\ K_\phi & 1 - 2K_\phi \end{pmatrix} E_{\Phi,t} \end{aligned} \tag{12}$$

While the phase cannot be controlled directly, it can be controlled indirectly through the robot's forward and rotational velocities. For example, inputting the controller into equation 8 can yield a desired phase. This phase can be tracked using a linear velocity controller in which the difference between the desired phase and the actual phase of the vehicle are minimized (13).

$$v_i = K_v(\phi_{des,i} - \phi_i) \tag{13}$$

As expected, the K_v term is the proportional control gain.

V. RESULTS

The distributed control system was implemented within MATLAB, and tested with a MATLAB simulator function as well as with actual robots (i.e. iRobot Creates). In both cases, experiments were designed to represent a system of multiple Autonomous Underwater Vehicles (AUVs) tracking and sampling the underside of an ice raft edge, where ice algae commonly grows.

To traverse the boundary edge, each robot uses a repeated series of motions that result in a lawnmower pattern that follows the edge. This series of motions includes 1) the robot moving forward until detecting the entering ice edge using upward facing range sensors, 2) driving forward under the ice as long as the presence of algae is still detected, 3) completing a 180 degree turn along a circular arc, 4) driving forward until leaving the ice edge is detected, and 5) completing another 180 degree turn along a circular arc. Throughout these motions, each robot adjusts its forward and rotational velocity to track a desired phase (see equation 13). To note, if the robot has reached the limit of the edge segment defined by s_i and s_{i+1}, it will change its boundary edge traversal direction.

A. Simulator Implementation

Three different scenarios were simulated, each with a different algae population. The first scenario used has no algae. In this scenario, each robot will travel an equal distance underneath the ice and return out. Here the boundaries should remain equal and all robots should remain in phase. Figure 2a shows the simulated environment. The white area represents the ice and the blue area is the water surrounding the ice raft.

The next scenario has a large amount of algae across half of the ice raft, and no algae across the other half. For this simulation, one robot will be completely submerged in the algae-side as well as half of another robot's boundary. The last robot will only have the axis to patrol. In this case, the boundaries along the patrolling axis should be significantly shorter for one robot, longer for the second robot, and longest for the robot without algae to observe. The algae in (Fig. 2b) is illustrated by the green areas.

Finally, the third scenario involves algae growing to random lengths from the ice edge, (Fig. 2c). To test the phase balancing aspect of this experiment, the robots were first placed in phase with one another. Then, the next three test cases involved initially placing one of the robots 72 degrees out of phase from the other robots. The goal is then for the robots to return to a state of equilibrium with each other while balancing the workload between them.

B. Simulator Results

In the first simulated test scenario, the three robots were initially in phase with each other and followed the boundary of a simulated ice edge with no algae. In Fig. 3, the error in phase and workload is illustrated. In order to compare this nominal error, the actual phase of each robot is displayed in the figure.

It can be seen in Fig. 3 that the phase remains constantly tracked. Also, the workload does adjust slightly and the error approaches zero.

The other simulations gave similar results. The steady state error for the phase can be found in Table I. The steady state error differs between scenarios, but it always decreases over time. This method lends itself to a worst case average of 4.1% steady state error.

TABLE I

PHASE STEADY STATE ERROR ($radians$)

Ice Edge	All Robots Initially In Phase	Robot 1 Initially Out of Phase	Robot 2 Initially Out of Phase	Robot 3 Initially Out of Phase
No Algae	0.073	0.093	0.073	0.073
Half Algae	0.169	0.189	0.206	0.207
Random Algae	0.222	0.240	0.257	0.238

The steady state error for the workload is in Table II. The steady state error does not increase significantly between no algae and half algae scenarios. However, the random algae scenario shows a significant increase compared to the other scenarios. Despite this increase, steady state error is limited to only 0.4% of the total area covered by algae (10 m^2).

C. Robot Implementation

Real-world testing is vital to observe the ability of actual robots to carry out the desired task. Three iRobot Create robots were used to test the functionality of the multiple robot system. These robots use an upward facing infrared sensor (with scalar output) to determine the presence of an "ice raft. Actual AUVs have acoustic range sensors that

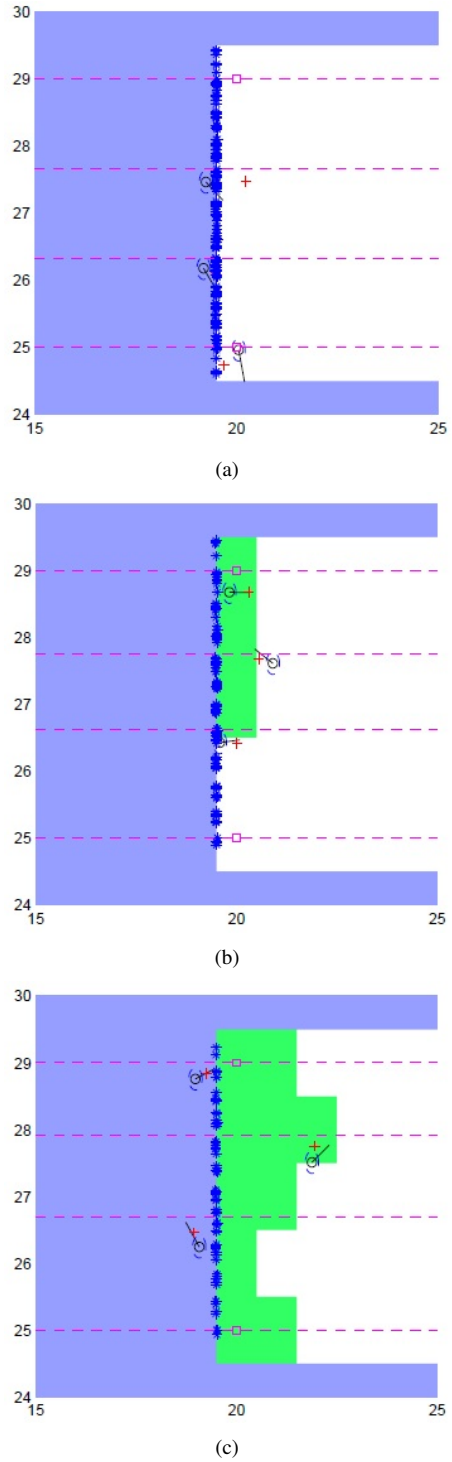
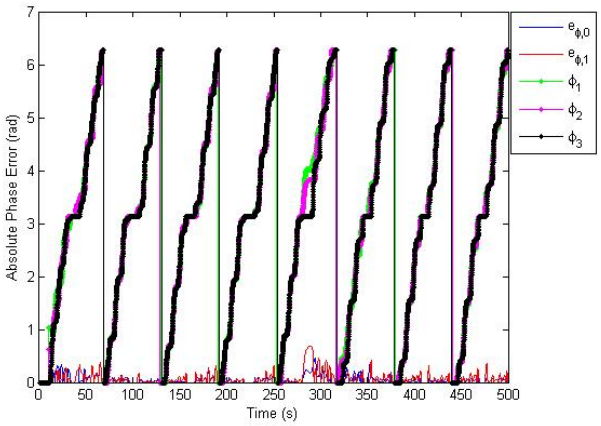
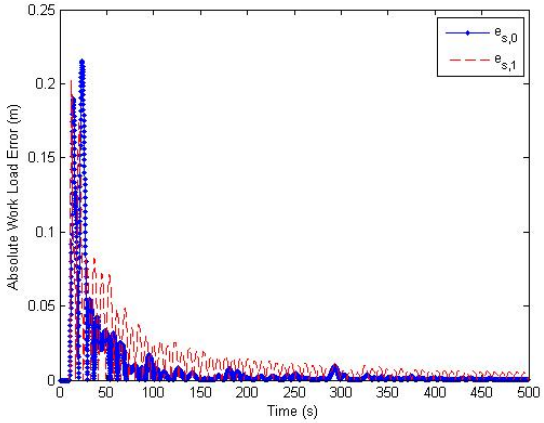


Fig. 2. The Simulator GUI: A top down view of three robots (circles) navigating water (light blue) and under ice (white). Asterisks (blue) indicate edge detections, squares (pink) indicate edge end points, and plus marks (red) indicate current desired locations of robots. Horizontal dashed lines indicate each robot's sub-segment end points. The algae (green) covers part of the under ice edge in some scenarios. In (a), there is no algae on ice edge. In (b), algae is covering half of ice edge. Scenario simulation (c) has a random coverage of algae.



(a)



(b)

Fig. 3. Simulation results for the case with no algae under the ice and the three robots starting in phase. The phase error is plotted in (a) and the workload error is plotted in (b).

TABLE II

WORKLOAD STEADY STATE ERROR (m^2)

Ice Edge	All Robots Initially In Phase	Robot 1 Initially Out of Phase	Robot 2 Initially Out of Phase	Robot 3 Initially Out of Phase
No Algae	0.010	0.013	0.015	0.012
Half Algae	0.015	0.015	0.016	0.019
Random Algae	0.037	0.036	0.043	0.043

can detect the presence of ice above in a similar fashion. The ice is simulated using foam boards hanging above the ground. Due to laboratory space constraints, it was not possible to simulate the algae underneath the ice edge for actual robot implementation. Despite the lack of simulated algae, workload balancing can still be seen in the robot implementation due to errors from robot localization.

The Create robots utilize the same MATLAB controller application as the simulator. The robots communicate with a computer running the MATLAB program via bluetooth wireless communication. The robots receive wheel velocities from the computer and return odometry and IR measurements. The main computer may act as a centralized system, however, the program architecture is decentralized.

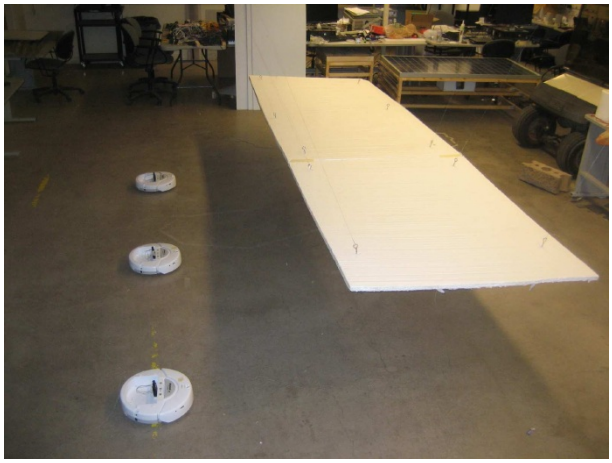


Fig. 4. The hardware setup: Three iRobot Creates navigated underneath an overhanging foam block used to simulate an ice raft.

The robots are initially placed 1m from the foam board's edge, and are aligned with each other at varying distances parallel to the S_G axis. Four different experiments were performed utilizing the same algae populations as in simulations. The first experiment begins with robots first reaching the foam's edge in phase with each other. The next experiment is initialized with the first robot 72 degrees out of phase with the other two robots. Another experiment begins with the middle robot 72 degrees out of phase with the other two robots. The final experiment sets the third robot 72 degrees out of phase with the other two robots. The results from this data should illustrate that the robots can recover from being out of phase quickly while sampling the underside of the overhanging foam block and evenly disperse the workload between them.

D. Experimental Results

Three iRobot Creates were tested to track the boundary of a 4m long foam board without any simulated algae. Figure 5 displays the resulting boundary following with robot trajectories.

In Fig. 6, the error in phase and workload is displayed. In order to compare this nominal error, the actual phase of each robot is displayed in the figure.

It can be seen in Fig. 6 that the phase remains constantly tracked. Also, the workload does adjust slightly and the error reduces to zero just after one cycle of the robot between its boundaries.

Other cases produced similar results. Table III displays the steady state error for the phase in all cases. In the worst case the steady state error remains below 2.5%.

The steady state error for the workload is also seen in Table III. The steady state error for the workload remains constant. This makes sense since the workload should not have to change across the various cases.

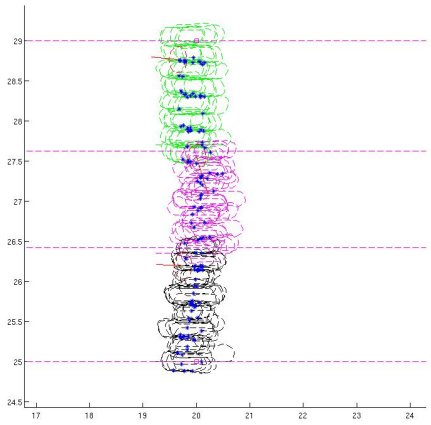
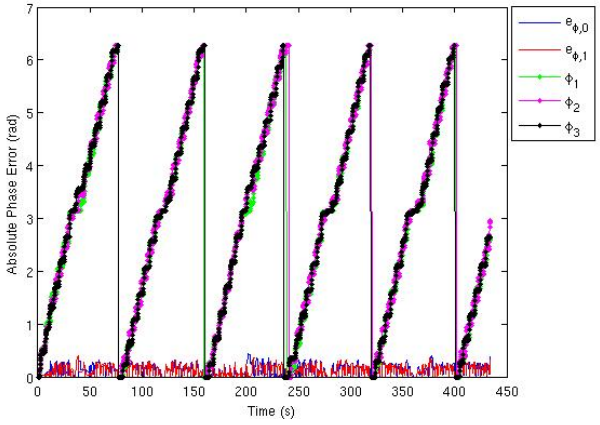


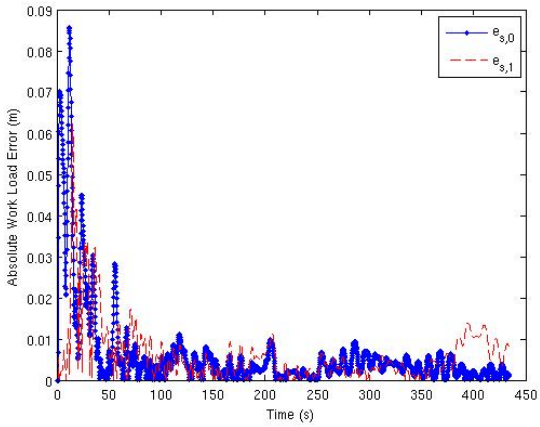
Fig. 5. Top down view of trajectories of iRobot Creates in following the simulated ice raft (i.e. the overhanging foam block.) Different colored paths are associated withe different robots, (i.e. green, pink, black). Blue asterisks indicate detected boundary edge crossings. Horizontal dashed lines indicate end points of each robots segment on the boundary edge. Actual foam block edge lies at x=19.75m. Units are in meters.

TABLE III
EXPERIMENT STEADY STATE ERROR

Steady State Error	All Robots Initially In Phase	Robot 1 Initially Out of Phase	Robot 2 Initially Out of Phase	Robot 3 Initially Out of Phase
Phase $(radians)$	0.129	0.1474	0.1381	0.152
Workload (m^2)	0.014	0.018	0.016	0.015



(a)



(b)

Fig. 6. Experimental results for the case with no algae under the ice and the three iRobot Create robots starting in phase. The phase error is plotted in (a) and the workload error is plotted in (b).

VI. CONCLUSION AND FUTURE WORK

This work presents a distributed boundary edge tracking controller for multi-robot systems. The controller balances workload and phase. It balances workload with a proportional controller that adjusts the boundaries that each robot works within in order to appropriately disperse the total coverage area between each robot. It balances the phase using a feedback linearization controller that allows robots to match their edge traversal progression within their individual boundaries.

The controller is provably stable to drive differences between robot workloads and phase differences to zero. It is also scalable since robots only need state information from nearest neighbors, and not the entire group. This was also demonstrated with simulations and real robot experiments. Future work for this project will ideally involve placing these controllers on Autonomous Underwater Vehicles (AUVs) deployed in Arctic expeditions where ice is abundant.

REFERENCES

[1] S. Charifa and M. Bikdash, "Adaptive boundary-following algorithm guided by artificial potential field for robot navigation", *IEEE Workshop on Robotic Intelligence in Informationally Structured Space*, pp.38-45, May 2009.

[2] J. Chen and L. Li, "Path Planning Protocol for Collaborative Multi-Robot Systems", *in IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp.721-726, June 2005.

[3] A. Joshi, et. al., "Experimental Validation of Cooperative Environmental Boundary Tracking with On-board Sensors", *in American Control Conference*, pp.2630-2635, June 2009.

[4] D.A. Paley, et. al., "Cooperative Control for Ocean Sampling: The Glider Coordinated Control System", *IEEE Transactions on Control Systems Technology*, vol. 16, no.4, pp.735-744, July 2008.

[5] X. Wang and V. Syrmos, "Coverage Path Planning for Multiple Robotic Agent-Based Inspection of an Unknown 2D Environment", *17th Mediterranean Conference on Control and Automation*, pp. 1295-1300, June 2009.

[6] S.S. Ge, et. al., "Boundary Following and Globally Convergent Path Planning Using Instant Goals", *IEEE Transactions on Systems, Man, and Cybernetics*, vol.35, no.2, pp.240-254, April 2005.

[7] B. Chow, et. al., "Assigning Closely Spaced Targets to Multiple Autonomous Underwater Vehicles", *Proceedings of the Unmanned Untethered Submersible Technology*, 2009.

[8] C.H. Hsieh, et. al., "Experimental validation of an algorithm for cooperative boundary tracking", *Proceedings of the American Control Conference*, pp. 1078-1083, 2005.