A Multi-Robot System for Continuous Area Sweeping Tasks

Mazda Ahmadi and Peter Stone
Department of Computer Sciences
The University of Texas at Austin
Email:{mazda,pstone}@cs.utexas.edu
http://www.cs.utexas.edu/~{mazda,pstone}

***Abstract*— As mobile robots become increasingly autonomous over extended periods of time, opportunities arise for their use on repetitive tasks. We define and implement behaviors for a class of such tasks that we call *continuous area sweeping* tasks. A continuous area sweeping task is one in which a group of robots must repeatedly visit all points in a fixed area, possibly with non-uniform frequency, as specified by a task-dependent cost function. Examples of problems that need continuous area sweeping are trash removal in a large building and routine surveillance. In our previous work we have introduced a single-robot approach to this problem. In this paper, we extend that approach to multi-robot scenarios. The focus of this paper is adaptive and decentralized task assignment in continuous area sweeping problems, with the aim of ensuring stability in environments with dynamic factors, such as robot malfunctions or the addition of new robots to the team. Our proposed negotiation-based approach is fully implemented and tested both in simulation and on physical robots.**

I. Introduction

Consider a group of robots whose goal is to keep the floors clean in a large office building. This task requires continual execution: by the time the robots have cleaned the entire building once, some parts have become dirty again. A first-cut approach might lead the robots to simply clean the building from top to bottom and then start over again. However, if the rate at which areas of the building become dirty is non-uniform and possibly even non-stationary, a more sophisticated solution is called for. In particular, the robots should ensure that they clean highly-trafficked areas, such as the main entrance and the restrooms, much more frequently than, say, the closets.

We define such a task as an example of *continuous area sweeping* tasks. More generally, a continuous area sweeping task is one in which a group of robots must repeatedly visit all points in a fixed area, possibly with non-uniform frequency, as specified by a task-dependent cost function.

Additional examples of continuous area sweeping tasks include trash removal and the task we consider in this paper, routine surveillance. When performing surveillance, a robot needs to continually traverse its environment in an effort to detect some events of interest, such as gas leaks, water dripping, lights on, open doors, etc. In the surveillance task, a location can be "visited" by observing, rather than by occupying it physically.

The goal of a continuous area sweeping task is not just to sweep the area in minimum time, but to sweep the area in such a way as to minimize the average event detection time, possibly weighted by the importance of different events. *Event detection time* is the time-period between event occurrence and its detection. The definition of *event importance* is problem-dependent. For example, in the trash collection task, the importance of collecting food trash may be higher than that of collecting paper goods. Minimizing the weighted average event detection time will result in the sensible behavior of visiting kitchens and other public areas more often than (most) individual offices. Similarly, for the surveillance task, one may define the importance of identifying gas leaks as being higher than finding lights on.

Continuous area sweeping tasks are closely related to the *security sweep* [1], or *sweeping* [2] task. In the security sweep or sweeping task, the goal is to make the robot(s) visit the whole environment *just once* in minimum time. Continuous area sweeping is also related to *coverage path-planning* [3], which "is a new path planning approach that determines a path for a robot to pass over all points in its free space." [3] The relevant differences are that in continuous area sweeping, the sweep must be performed i) repeatedly (continuously), and ii) non-uniformly, that is with more frequent attention given to some areas than to others. As surveyed by Parker [4], most previous approaches to surveillance assume ideal sensors and no computational bounds. In contrast, in this paper we consider solutions that are fully implementable (and implemented) on physical robots.

We tackle continuous area sweeping by dividing it into two sub-problems:

1) Enabling a *single* robot to autonomously perform a continuous area sweeping task in a sub-region.
2) Partitioning the overall area among the *multiple* robots.

Once the area is partitioned among the robots, each one of them sweeps its part of the environment using the single-robot area sweeping method.

We have previously addressed the first sub-problem of single-robot exploration [5]. In this paper we mainly focus on the second sub-problem: area partitioning. We assume an environment with different dynamic factors, such as addition of new robots, robot malfunctions, change in robot speeds or changing distributions for event appearances.

The remainder of the paper is organized as follows. Section II surveys the previous work most related to our own. In Section III we formalize the class of continuous area sweeping tasks. Section IV introduces an overview of an algorithmic solution to single robot continuous area sweeping tasks. In Section V the negotiation method for adaptive area partitioning is introduced. In Section VI we instantiate the formalism and algorithms on the robot surveillance domain. Our methods

are fully implemented and tested both in simulation and on a physical robot, the Sony AIBO ERS-7 4-legged robot. Section VII discusses future work and concludes.

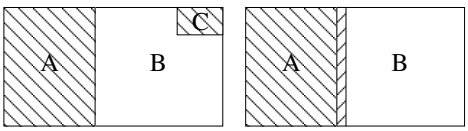
II. Related Work

Most of the methods for area partitioning use fully centralized and static approaches. For example Hert et. al. [6] tries to partition the environment into n equal size parts. Bern et. al. [7] also try to partition the environment into equal size parts but with the additional condition that the parts do not have any acute angles.

Notice that these works assume a heuristic for the notion of best partitioning, such as equal size parts, or parts without acute angles. But in our work, the goal is to minimize the average detection time, and the algorithms will directly try to achieve that goal. In our experimental results section, we will provide an example in which the partitions do not follow any of these heuristics (Figure 6(b)).

There are some other methods that address dynamic area partitioning in different ways, but that are not suitable for continuous area sweeping. For example Min and Yin [8] propose a dynamic area partitioning method in which the robots start with an initial static partitioned area. When a robot finishes its assigned task, it negotiates for more parts of the environment. Since they do not partition the environment permanently, although it is suitable for their one sweep of the environment, it is not good for our continuous sweeping task. For example in Figure 1(a) the robot "a" is responsible for part *A* and robot "b" is responsible of parts *B* and *C*. If by the time that robot "a" finishes part *A*, robot "b" is still sweeping part *B*, then robot "a" will be responsible for part *C*. But a better partition, which our algorithm will achieve is the one in Figure 1(b), where robot "a" gets a little more area close to its original responsibility area. Additionally, Min and Yin assume full and error-free communication, whereas we do not assume full communication between all robots.

Jager and Nebel [9] partition the environment into polygons such that each robot requests to clean a polygon and the others respond if they have cleaned it. This will result in an unpredictable area partitioning, because while a robot is requesting a polygon, it does not consider the whole region that it has and will sweep. Thus, this method is also most suitable for single sweep applications.



(a) (b)

Fig. 1. (a) Partitioning using Min and Kin algorithm [8]. (b) Partitioning using our method.

Schneider and Mataric [10] propose a dynamic method in which all the robots have full knowledge of the positions of the other robots. The only dynamic factor that they can respond to is the addition of a new robot. In addition to the fact that our algorithm handles different dynamic factors (e.g. robots with changing speeds), it does not rely on knowledge of other robots' positions at any given time.

III. Continuous Area Sweeping Formulation

In this section we specify our task in detail. In a continuous area sweeping task, the robot must repeatedly visit all the points in its environment in an effort to detect or react to different types of events $e \in E$. The events can in general have varying degrees of importance, imp_e, and each event may occur in different places with varying frequencies. In the case that all points are equally likely locations for an event of interest, the events are equally important, and the robot needs to be physically present at the point to "visit" it, the problem reduces to the traveling salesman problem. Thus, in general, continuous area sweeping is NP-Hard, and we must rely on approximate solutions.

We begin by dividing the robot's environment into disjoint grid G, with each event occurring in one grid cell. We consider time as a sequence of discrete steps. The orientation $\theta \in O = \{$*North, South, East, West*$\}$ of the robot is also considered as being one of 4 disjoint values. We track the last time a robot has visited each cell $g \in G$ in an array $LV[G]$ by setting $LV[g] =$ current-time whenever the robot visits cell g.

The problem is defined as a tuple $(S, A, T_{sa}, P_{eg}, CF)$, where:

- $S = G \times O \times LV[G]$ is a set of *states*, representing the position and orientation of the robot as well as the array of last-visit times to each cell.
- A is the set of possible actions. The actions in this formulation are specified as going to a point in the environment. In particular, the environment is divided into a *coarse grid* called CG (CG need not be related to G in any way, though in general we expect it to be coarser than G). Each action $a \in A$ is defined as traversing the path between the current position and the center point of one of the coarse grid cells in CG and at the end turning to reach one of the four orientations. That is, there are $|CG| \times |O|$ possible actions from each state. The time complexity of the algorithm is highly dependent on the number of actions, which is why we usually want CG to be coarser than G.
- T_{sa} is the state transition probabilities. Based on the current state and action, it gives the distribution over the states that the robot will transition to. The transition function is stochastic, because based on possible robot localization errors and non-determinism in its movement, the robot may end up in grid cell g_j when aiming for grid cell g_i.
- P_{eg} is the probability of appearance of event e in cell g per cycle. For example, if P_{eg} = 0.1, there is the expectation of event e occurring every 10 cycles in cell g. P_{eg} is a property of the environment and is not observable by the robots.
- CF is the *cost function* of the *policy*. The cost function that we define for the continuous area sweeping problem is the average time elapsed from appearance to detection of the events, weighted by their importance of the event (imp_e). Since the robots should collectively observe the environment in order to detect all the events in minimum time, this criterion is for the whole multi-robot system.

The goal of each robot is to find a *policy* $\pi : S \mapsto A$ such that the joint policy of all robots minimizes the cost function. For each robot, the policy determines which action is chosen by the robot in each state.

Since the robots do not observe the times of event appearances, they are unable to calculate the cost function (CF) of their executed policies. Thus direct methods to minimize CF even in the single-robot case will not work. A heuristic single-robot algorithm will be presented in the next section.

IV. Exploration Algorithm

In this section, we present a description of our initial approach to single robot continuous area sweeping tasks. We begin by assuming that time is discretized into *cycles* representing the times at which the robot can make action decisions. In our system, cycles begin when the robot receives a new vision input (≈ 25Hz). For the purposes of our algorithm, we define an *expected reward* of each grid cell g at cycle t as the expected sum of *the event importance values* present in grid cell g at time t.

We tackle this problem by dividing it into two sub-problems:

1) Learn the expected accumulation rate of event importance values in each cell (*potential reward*). The *expected reward* of visiting a cell at any given time depends on this rate and the time at which the cell was last visited. **(learning)**
2) Given these expected rewards and knowledge of the robot's (possibly stochastic) transition function, compute a sequence of actions for the robot (*policy*) with minimum cost. **(planning)**

The details of these two steps of the algorithm are presented in our previous paper [5]. We have also theoretically proven that performing the two steps of learning and planning will result in minimizing the cost function [5]. For the purposes of this paper, it is sufficient to know that each individual robot is capable of efficiently engaging in continuous area sweeping within any fixed sub-region of its environment.

V. Cooperative Behavior

We achieve cooperative behavior by partitioning the environment among robots. Partitions are assigned to different robots, and the robots do the exploration autonomously in their assigned partitions. Note that restricting robots to partitions may not necessarily lead to the optimal behavior for multi-robot continuous area sweeping, but doing so allows for a convenient and efficient task decomposition.

A naïve first approach is to statically partition the environment among the robots. However in our environment, with the probability of event appearances changing dynamically plus the possibility of the addition and removal of robots from the environment, static partitioning is not suitable.

Instead, we propose a negotiation model for partitioning the environment among robots. We define RG_x as the set of grid cells that robot x is responsible for. The basic idea behind the negotiation method is: Considering two robots, a and b, if there is a $g \in RG_a$ that robot b can visit — following its own exploration algorithm — more often than robot a, then g should be added to RG_b and removed from RG_a.

Algorithm 1 High level negotiation procedure.

1) **Robot 1 sends** S_1={border line grid cells} of message type 1.
2) **Robot 2** Upon receiving S_1 of message type 1:
t := big negative number;
for all $g \in S_1$ **do**
  t_1 := possible time between visits for g (by Robot 2)
  t_2 := available time between visits for g (by Robot 1)
  if $(t_2 - t_1) > t$ **then**
    t := $t_2 - t_1$;
    g_{max} := g;
  end if
end for
G^* := cells that will be visited because of addition responsibility of g_{max}.
send $S_2 = \{G^*\}$ of message type 2 as an offer.
3) **Robot 2** upon receiving message type 2, accepts the best offer, and sends an acknowledgement (message type 3).
4) **Robot 1** upon receiving acknowledgement, the transfer will be complete.

The high-level negotiation procedure is shown in Fig V. The regular negotiation structure is as follows:

First: in fixed periods each robot,

1) labels the grid cells on the border of its RG as *candidates*. These grid cells are the ones that the robot is considering giving up responsibility for. Note that all the grid cells on the border of a robot's responsibility area, but not on the border of the whole environment are considered candidates.

2) Broadcasts a message consisting of information about the candidate grid cells. The message format is as follows:

$$(g, avg\_time, pot\_reward)$$

Where, g is the grid cell id and avg_time is the robot's current average detection time for that grid cell. pot_reward is potential reward which is the learned expected accumulation rate of event importance values in cell g. pot_reward is used to compute *expected* reward and is sent to other robots for use in the robots' single-robot exploration algorithm (because they have no first-hand experience about the rate of event appearance in other robots' grid cells). These messages are called *type 1 messages*.

Second: Upon receiving type 1 messages, the robot stores them in a list. At fixed intervals, each robot processes its stored messages as follows:

1) For each grid cell g in the stored messages, the robot pretends that it is responsible for it (in addition to its whole current partition), and using the single-robot algorithm finds a new hypothetical path. With that path, it computes *time between visits* for g, and stores it in $time_g$.

2) For each grid cell g in stored messages, using $time_g$ and imp_e the robot computes the weighted detection time ($new\_avg\_time_g$) for that grid cell under the assumption that the robot adds g to its partition.

3) From among all the grid cells mentioned in type 1 messages, each robot finds the cell with the maximum difference

between the computed event detection time ($new\_avg\_time_g$) and the average detection time that the message sender could provide (avg_time_g from the message). That is it finds the cell $maxg$ such that ($new\_avg\_time_g - avg\_time_g$) is maximized. From the local information that the robot has, from among all the candidate cells in type 1 messages, $maxg$ is the best one to add to the robot's partition, because it is the one for which its addition to the partition will most decrease the cost function. Notice that transfer of any cell between two robots may change cells' time between visits for both robots. Since the robots do not have information on those changes, the decisions based on time between visits for transferring more than a single cell could lead to unpredictable transfers, and likely oscillations.

4) $maxg$ which was computed in the previous step, is the cell that the robot will offer to take into its partition. In the new path that the robot has to take in order to visit $maxg$, possibly some additional cells from other partitions will be visited. The robot stores these cells in V_{maxg}. The cells in V_{maxg} will be visited without any further effort, thus the robot offers to take over responsibility for them as well. In particular, it sends a message to take over all the cells in V_{maxg} to the robots currently responsible for them. The message format is as follows:

$$(num, (g_0, avg\_time_0), (g_1, avg\_time_1), ...)$$

Where, num is the number of offered grid cells, while g_i and avg_time_i, for $0 \leq i < num$, are grid cell ids and average detection times for each offered grid cell respectively. These messages are called *type 2 messages*.

Third: Each robot accumulates its received type 2 messages, and then processes them at fixed intervals.

1) The robot has the chance to accept one of the offers, that is, it can give away a set of its cells to one of the robots that has made an offer for them. For this purpose, it assigns a value to each offer. For offer o, its value will be equal to:

$$\sum_{i=0, g_i \in RG}^{i=num-1} avg\_time_i - my\_avg\_time_{g_i}$$

Where num is the number of cells in offer o, avg_time_i is the average time between visits for grid g_i in offer o and $my\_avg\_time_{g_i}$ is the robot's average time between visits for that same grid cell. By accepting cell g_i, avg_time_i will decrease to $my\_avg\_time_{g_i}$. Thus, the offers that decrease the cost function the most have the most value. If the highest value is positive, the offer associated with it is accepted.

In other words, from all the offered grid cells, it finds the set (received from a single robot and in the robot's own *RG*) such that the sum of the difference between the offered detection time and the current average detection time is maximized.

2) The robot then gives up the responsibility for the cells in the accepted offer.

3) Finally, if the robot has accepted an offer it sends an acknowledgement to the robot willing to take responsibility for them (message type 3).

Fourth: When a robot receives an acknowledgement for a set of grid cells (message type 3), it assumes the responsibility of that set of grid cells and the negotiation is considered finished. Each robot then resumes its single-robot sweeping within its (possibly changed) partition.

Notice that the only message type of the three that can cause inconsistency if it is not delivered, is the acknowledgement message (message type 3). If any other message does not get delivered, no change of responsibility will occur. But if the acknowledgement message does not get delivered no robot will assume the responsibility for a set of grid cells and if there is no recovery mechanism, the inconsistency can be permanent. In our current system, we send the acknowledgement message 5 times to reduce the possibility of that inconsistency. In our experiments the maximum of consecutive message losses was 2, and thus no inconsistency occurred.

When a new robot is added to the environment, it sends out a message declaring its presence, and the robots who are close enough to hear its message send out their position information to it. It then takes responsibility for half of the partition of the closest robot to it. The negotiation then continues, and appropriate adjustments are made. Similarly, when a robot is removed from the environment, it sends out a signal notifying others that it is being removed, or if it is crashed, other robots will detect its removal after not hearing from it for an extended period of time. After that closest robot to it takes charge of the its responsibility area, and further negotiations will split the area appropriately.

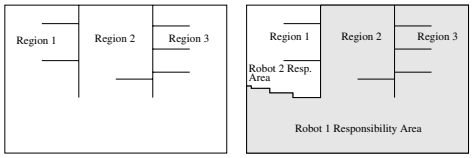
VI. Experimental Results

To test our approach, we have implemented and evaluated our algorithm on a simulated as well as a physical robot in a representation of the routine surveillance task. As our robot, we use Sony ERS-7 four-legged AIBO robots (Figure 2). The robot's sensor device for "visiting" locations in its environment is a camera mounted in the head of the robot. It can capture 208 $\times$ 160 frames of pixels at roughly 30Hz (Due to the computational intensity of image processing, our robots typically make decisions at roughly 25Hz). By turning its head, the robot can gain a 180-degree field of view. It has 20 degrees of freedom and a 576Mhz on-board processor.

Fig. 2. ERS-7 Sony AIBO robot

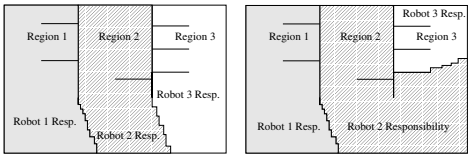
As baseline software, we use our legged team code base [11], which provides robust color-based vision, fast locomotion, and reasonably accurate localization within a 4.4m $\times$ 2.9m area via a particle filtering approach. The field is as specified in the 2004 rules of the RoboCup Four-Legged Robot League: `http://www.tzi.de/4legged`. Even so, the robot is not, in general, perfectly localized, as a result of both noisy sensations and noisy actions. The robot also has limited processing power, which limits the algorithms that can be designed for it. We consider one type of event in the environment, which is the appearance of an orange ball that the robot can recognize from anywhere on the field provided that it has an unobstructed view.

We have tested two different configurations of the world in a custom-built simulator, and one configuration on real robots. The simulator, though abstract with respect to locomotion, provides a reasonable representation of the Aibo's visual



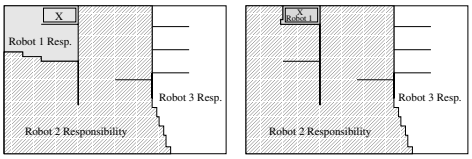
(a) (b)

Fig. 3. (a) Representation of configuration I. (b) The partitioned area between two homogeneous robots in configuration I.



(a) (b)

Fig. 4. (a) The partitioned area for three homogeneous robots in configuration I. (b) The partitioned area for three heterogeneous robots in configuration I. The robot 3 has half the speed of the other two robots.



(a) (b)

Fig. 5. (a) The partitioned area between three homogeneous robots in configuration I, when the chance of ball appearance in area X is 10 times the other cells. (b) The partitioned area between three homogeneous robots in configuration I, when the chance of event appearance is 1000 times more in area X.

and localization capabilities, and allows for more thorough experimentation, particularly in large environments.

In this paper, the focus of the experiments is on the negotiation and the adaptive area partitioning. For detailed results on the single-robot exploration and some videos of a robot in action please see [5].

A. Configuration I in simulation

The configuration of the environment in the first experiment, is shown in Figure 3(a). For this experiment, we divide the world (4.4m $\times$ 2.9m) into a 45×54 grid (G). The coarse grid (CG) is a 15×18 grid. The reported results are averaged over at least 10 trials.

1) Two homogeneous robots: We start with having two homogeneous robots on the field. In the initial partitioning, each robot gets half of the area (divided vertically). The partitioned area, which is achieved after reaching equilibrium is shown in Figure 3(b). Notice that when robot 2 traverses the path between regions 2 and 3, it automatically visits the bottom cells, thus it takes the responsibility for all of the bottom grid cells. They reached this assignment with only one negotiation in which 683 cells were transferred from robot 2 to robot 1.

Conventional area partitioning algorithms will try to divide the area equally between the two robots, which is less efficient than the equilibrium that our robots reached. If the area is divided equally between robots, the average event detection time would be 33.9 ± 0.7 seconds, while with our partitioned area, it is 32.2 ± 0.6 seconds. Since there are just two homogeneous robots in a simple environment, a minor performance enhancement (in this case %5) is all we can expect.

2) Three homogeneous robots: Later in the experiment we added a new robot in the middle of the field. While usual non-adaptive area partitioning methods cannot adapt to the addition of the new robot, our robots reached a new equilibrium which is showed in Figure 4(a). It took the robots two negotiations to reach this partitioning. 421 cells were transferred from robot 3 to robot 2 and 283 cells from robot 2 to robot 1.

A conventional static partitioning for three robots could achieve a similar partitioning. The average event detection time in this case was 28.3 ± 0.6 seconds.

3) Three heterogeneous robots: Following the previous experiment, we slowed down robot 3 to half of its original speed. That condition can happen in the real world as the result of a joint failure. It took the robots one negotiation with 457 cells transfered to reach the new partitioning which is shown in Figure 4(b).

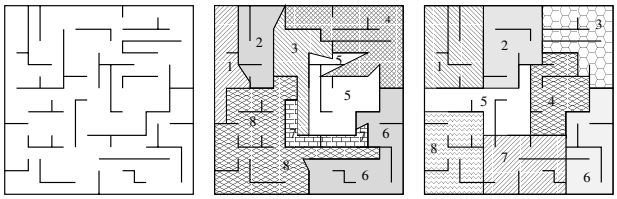
If no new negotiation were performed after slowing down robot 3, the average event detection time would have been 30.1 ± 0.4 seconds. However, after the negotiation and the resulting new partitioning, the average event detection time was 29.3 ± 0.2 seconds. Notice that here we only have 3 robots, and only one of them slows down. If there are more robots, the speedup will be more significant. To our knowledge, previous work in the area could not adapt to this new situation.

4) Non uniform distribution of event occurrences: Continuing with three homogeneous robots, we next consider the case in which the chance of event appearance in area X (Figure 5(a)), which consists of 45 grid cells, increases by a factor of 10. The robot can learn the distribution based on the event occurrence by itself [5], though it requires time to notice the change. In this case, to speed up the experiment, we manually increased the potential reward value in the robot's internal algorithm to represent the new distribution. With only one negotiation and 497 cell transfer, the new partitioning is formed, which is shown in Figure 5(a). The average event detection time with the original partitioned area, (shown in Figure 4(a)) was 34.7 ± 0.8 seconds, while with the new partitioning it reduces to 29.4 ± 0.5 seconds.

If the chance of the ball appearance in area X is multiplied by 1000, one of the robots ends up constantly staying and watching area X, while the other two robots divide the environment as shown in Figure 5(b). With the original partitioned area (Figure 4(a)), the average detection time would be 45.3 ± 0.2 seconds, but with the new partitioning it reduces to to 0.2 ± 0.0 seconds. Notice that with the new partitioning, one of the robots constantly watches area X and thus most of the events are observed in no time.

B. Configuration II in simulation

The aim of this experiment is to show that the cooperation algorithm can scale up to more complex situations. The environment in this experiment is shown in Figure 6(a). It is 8m $\times$ 8m and is divided into a 80×80 grid G. CG is a 20×20 grid.



(a) (b) (c)

Fig. 6. (a) Representation of configuration II. (b) Task decomposition of 8 robots in configuration II, which is achieved by our partitioning algorithm. (c) A typical and reasonable task decomposition in configuration II.

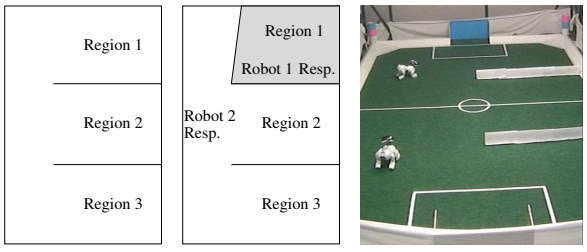
There are 8 robots with different speeds, as shown in Table I. The partitioned area after negotiation is shown in Figure 6(b). It took the robots 67 negotiations and in total 24535 cell transfers to reach this partitioning. Although some of the shapes looks irregular, each robot can observe its whole partition while following a simple path. Using a perfectly space-equivalent partition (as some approaches do), leads to average event detection time of 15.3 ± 1.0 seconds. In Figures 6(c) we show a heuristic partition chosen so as to roughly equalize space, but in a way that follows borders and appears to be reasonable. The average detection time in the heuristic reasonable partition (Figure 6(c)) is 9.0 ± 0.8, while with our partitioning (Figure 6(b)), it decreases to 6.1 ± 0.5. This data is averaged over 10 trials. This significant improvement over the static area partitioning suggests that with higher number of robots, the advantage of our method is more significant.

Robots	1	2	3	4	5	6	7	8
Speed (cm/s)	10	20	10	30	40	40	20	50

TABLE I

SPEED OF ROBOTS FOR CONFIGURATION II EXPERIMENT

C. Configuration III with real robots



(a) (b) (c)

Fig. 7. a) Representation of configuration III. (b) Task division for real robots in configuration III. (c) Picture of configuration III with two robots.

To show that the system also works on real robots, we present an experiment with real robots in a simple environment. An overview of the configuration is shown in Figure 7(a) and the picture of the actual environment with the robot is shown in Figure 7(c). The robots know the locations of the walls in the environment, but must decide for themselves how to move so as to perform surveillance.

In this case, the grid G was 15×18, and CG was a 5×6 grid. The resulting task division between the robots is shown in Figure 7(b). For the same reason as discussed in Section VI-A.1, this is the optimal task division. For a movie of this experiment, please visit `http://www.cs.utexas.edu/~AustinVilla/?p=research/surveillance`

VII. CONCLUSION AND FUTURE WORK

In this paper, the problem of *multi-robot continuous area sweeping* is examined. The problem is defined as one in which robots must repeatedly visit every part of the environment in order to detect a set of events of interest. The frequency of the events can possibly be non-uniform. Thus the robots should visit the points with non-uniform frequency. Examples of continuous area sweeping tasks are surveillance and cleaning.

The focus of this paper is area partitioning, while the robots are continuously doing their tasks. The area partitioning is done by a negotiation method, which is adaptive to dynamic environments. The adaptive area partitioning is especially important if the rate of event appearance is non-uniform in the environment, or if the robots are heterogeneous in their capabilities.

Our on-going research agenda includes expanding the robot behavior to include non-greedy planning and to find an optimality bound for the area partitioning method. Also we are working on designing recovery mechanisms for the cases where unbounded consecutive messages get lost, and an inconsistency occurs.

ACKNOWLEDGMENTS

The authors would like to thank the members of the UT Austin Villa team for their efforts in developing the software used as a basis for the work reported in this paper, and also Bikramjit Banerjee, Roozbeh Mottaghi, Ali Nouri and Mohan Sridharan for their comments on earlier versions of this paper. This research was supported in part by NSF CAREER award IIS-0237699 and ONR YIP award N00014-04-1-0545.

REFERENCES

[1] N. Kalra, A. T. Stentz, and D. Ferguson, "Hoplites: A market framework for complex tight coordination in multi-agent teams," Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-04-41, 2004.

[2] J. A. T. Y. E. Kurabayashi, D. Ota, "Cooperative sweeping by multiple mobile robots," in *Proc. of IEEE International Conference on Robotics & Automation (ICRA)*, 1996.

[3] H. Choset, "Coverage for robotics; a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, 2001.

[4] L. E. Parker, "Distributed algorithms for multi-robot observation of multiple moving targets," *Autonomous Robots*, vol. 12, no. 3, pp. 231–255, 2002.

[5] M. Ahmadi and P. Stone, "Continuous area sweeping: A task definition and initial approach," in *The 12th International Conference on Advanced Robotics*, July 2005.

[6] S. Hert and V. Lumelsky, "Polygon area decomposition for multiple-robot workspace division," *Special Issue of International Journal of Computational Geometry & Applications on Applied Computational Geometry*, vol. 8, no. 4, pp. 437–466, 1998.

[7] H. Bast and S. Hert, "The area partitioning problem," in *Proceedings of the 12th Canadian Conference on Computational Geometry*, 1995.

[8] T. W. Min and H. K. Yin, "A decentralized approach for cooperative sweeping by multiple mobile robots," in *International Conference on Intelligent Robots and Systems (IROS)*, 1998.

[9] M. Jager and B. Nebel, "Dynamic decentralized area partitioning for cooperating cleaning robots," in *ICRA*, 2002.

[10] M. Schneider-Fontan and M. Mataric, "Territorial multi-robot task division," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, 1998.

[11] *Peter Stone et al., "The UT Austin Villa 2004 RoboCup four-legged team: Coming of age," The University of Texas at Austin, AI Laboratory, Tech. Rep. UT-AI-TR-04-313, October 2004.*