Preprints of the 2012 20th Mediterranean Conference on Control & Automation (MED), Barcelona, Spain, July 3-6, 2012

A Solution with Multiple Robots and Kinect Systems to Implement the Parallel Coverage Problem

Hyeun Jeong Min, Duc Fehr, and Nikolaos Papanikolopoulos
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455
Email: {hjmin|fehr|npapas}@cs.umn.edu

***Abstract*— The coverage problem has been traditionally solved for a given number of robots with randomly generated positions. However, our recent work presented a solution to the parallel coverage problem that optimizes the number of robots starting at the same location. The motivations are: *(i)* the number of involved robots affects the total coverage cost, and *(ii)* it requires extra effort to place them at real-world locations. In this work we present a control algorithm for multiple robots with Kinect systems to implement the solution to the parallel coverage problem. Our algorithm utilizes a multi-robot formation. Robots need to localize themselves to know where they are within a map. To localize the robots and to reduce inter-communication, we introduce a technique to place only certain robots in a team. This work also presents an algorithm on how to manage dynamic changes of a group of formations in order to solve the coverage problem. This paper demonstrates the mission, which is to visit every desired position to cover an indoor environment, with a team of real robots and the Kinect system.**

I. INTRODUCTION

Dealing with multiple robots has received more and more attention in the robotics research. The coverage problem using multiple robots has been actively investigated for robotic applications such as searching for specific targets, predator-prey [1], intruder detection [2], sweeping tasks [3], among others. Multi-robot coverage problems are traditionally dealt with a given number of robots starting at randomly distributed locations [4]. Konolige *et al.* have tried to use about a hundred robots for an intruder detection problem (the Centibots project), but the problem is also based on a known number of robots [5]. However, using a given number of robots is not always optimal as discussed in our prior research published in [6]. Besides, it is not practically effective to assume a given number of robots with randomly generated locations since it requires extra effort to place them in certain areas. Howard *et al.* have proposed an algorithm flexible to an unknown number of robots [7]. The algorithm, however, does not provide an optimal number of robots. No research has claimed to find an optimal necessary number of robots prior to our research.

Unlike existing coverage problems, our research has mainly addressed how to find an optimal number of robots for a given environment as well as the corresponding paths. This paper assumes that a team of robots starts at the same area. Ahmadi and Stone have proposed a sweeping task in which a given number of robots could repeatedly visit in a fixed area [3]. The research is about finding a proper partition for a given number of robots in an environment. Our parallel coverage problem, on the other hand, changes the point of view with respect to the existing coverage approaches regarding the area to be covered by the robots. In this paper, we present a solution to the parallel coverage problem.

For the parallel coverage problem, the assumptions are that the environment and the target positions that must be visited by the robots in a team are given. Although we have the information regarding a map and the positions, there are still issues when we attempt to use the robots in a real environment: *(i)* how to control the robots in order to move them towards the desired areas and *(ii)* how to estimate the position of the robots and the target locations.

This paper first considers how to control multiple robots to make them move towards the desired locations. For the implementation of the coverage problem with real robots, this work presents a formation-based algorithm. Since we assume that robots start at the same area, it is reasonable for robots to move together until they encounter a certain position that needs to branch off to other locations. Multiple robots may need inter-communication to share information of desired locations and localization. As we get more robots, more data packets are sent or received through communication. Reducing inter-communication is needed for the robots executing a mission to avoid latent failures that might exist. A vision-based leader-follower formation technique published in [8] does not require inter-robot communications. The concept is that only the leader robot in a formation needs to be localized on a map, and the other follower robots simply follow the leader by using only their camera sensor mounted on themselves while they travel on the same path.

When controlling robots for a coverage problem, we need to localize the robots in order to let them know where they are and to control them to get close to the desired areas. There has been much research on localization; however, it requires special techniques such as a particle filters [9], visual landmarks [10], relative positioning using IR sensors [11], odometry information [12], and vision-based cooperation [13], with various sensors such as a laser scanner, IR sensors, encoders, GPS, and cameras. Special sensors, however, include various assumptions: *(i)* the size of robots should be larger than the sensor, *(ii)* robotic platforms need to have enough power to support the sensing ability, and

978-1-4673-2529-5/12/$31.00 ©2012 IEEE

(iii) the computation for localizing the robots should be fast enough to be applicable to real-world robotic applications. Any type of sensors such as cameras or GPS can be used in the proposed algorithm; however, in this work, we utilize the Kinect sensor for robot localization [14]. The Kinect system is a relatively inexpensive sensing platform, which is able to provide fast 3D estimates of target objects. This work also presents the algorithm to control a team of multiple robots through multiple Kinect systems. It manages dynamic changes of robot formations (from a leader to a follower and vice versa).

Our main contributions in this paper are that *(i)* a practical solution to control multiple robots with Kinect systems to implement a coverage problem is presented, *(ii)* the overhead sensors localize only certain robots to reduce intercommunications, *(iii)* the Kinect is used in a novel way as an experimental testbed, and *(iv)* the algorithm to cover a large area with interconnected sensors (multiple Kinects) is addressed.

We review the coverage problem in Section II. The proposed formation-based algorithm with the Kinects is discussed in Section III. We illustrate the experimental results in Section IV. We finally conclude and consider future work in Section V.

II. PARALLEL COVERAGE PROBLEM

This section reviews the multi-robot parallel coverage problem that we previously investigated in [6]. Our coverage problem originates from the fact that we want to have the robots to initially start at the same location to avoid the complexity of locating them at random positions. In addition, we consider the number of robots that would be best to succeed in a mission in a given environment, instead of optimizing the process with the already allocated resources. Using many robots does not always guarantee faster coverage for a given problem. Also, if we have fewer robots than expected, it will take longer to solve the given problem.

In our coverage problem, we expect robots to visit a set of certain positions in order to search for Objects of Interest (OOIs). The problem is to optimize the required number of robots and the corresponding paths for each robot. Here, the minimum coverage time is the maximum travel time associated with the longest path among the paths covered by each robot. This paper reviews the two terms of the Key Positions Of Interest (KPOI) and the parallel multi-agent coverage problem, and newly define the Turning Position (TP) term.

Definition 1: A Key Position of Interest (KPOI) is the position which a robot in a group must visit. This is to say, a robot is required to visit a location where the respective KPOI is within the active range of the robot's sensor.

Let $V = \{v_1, \cdots, v_n\}$ be a set of KPOIs, and let $P = \{P_1, \cdots, P_r\}$ be a set of subsets of V for r robots corresponding to each path as shown in Table I. Then it satisfies the following conditions: *(i)* $1 < r \leq n$, *(ii)* $\bigcup_i P_i = V$, *(iii)* $\bigcap_i P_i = \emptyset$, and *(iv)* $cost(\pi_i) \leq c_p, \forall i$. Note that we include a KPOI ($v_j \in V$) in P_i only if the i^{th} robot is a leader robot while a group of robots passes through the KPOI (v_j).

Definition 2: The Parallel Coverage Problem limits the space of acceptable solutions as follows. The paths traversed by each robot agent should be characterized by the following: (i) every robot path includes a subset of KPOIs visited only once by the respective robot, and (ii) the robots' KPOIs are complementary and disjoint.

Definition 3: A Turning Position (TP) is the position that has a conjunction of more than two paths that must be visited.

For the Parallel Path Coverage Problem (PPCP), we need to find an optimal number of robots and paths with the minimum coverage cost. The objective of the PPCP is

$$c_p = \min\{\max_i cost(\pi_i)\}, \forall i = 1, ..., r. \quad (1)$$

Here, π_i is the i^{th} path as shown in Table I. The PPCP is to minimize the maximum path cost among paths from a start position to every position in a set of KPOIs.

The proposed strategy for our problem, which finds the minimum path cost and number of robots, is to find a path with the minimum cost achieved by finding the furthest position (v_{max}) from the start. Let σ_i be paths from the start to v_{max}. Then the optimal path cost is $c_{max} = \min_i cost(\sigma_i)$, where i indicates each path from the start to v_{max}. Note that the variable r is the number of the robots which is initially unknown. The optimal number of robots (r) is acquired from the algorithm covered in [6] (refer to the publication for more details).

TABLE I
THE CORRESPONDING RELATIONS BETWEEN ROBOTS, PATHS, PATH COSTS, AND THE SETS OF KPOIS THAT MAKE UP THE PATHS.

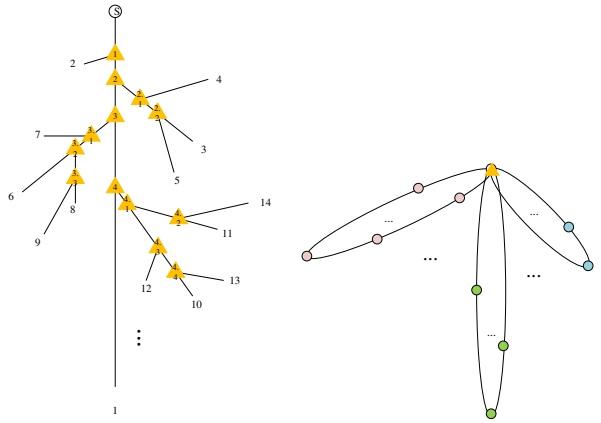
robots	paths	costs	KPOIs
1	π_1	$cost(\pi_1)$	P_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
r	π_r	$cost(\pi_r)$	P_r

III. FORMATION-BASED CONTROL WITH THE KINECT

We have discussed the multi-robot parallel coverage problem in Section II. This section deals with how to control a team of robots in order to visit a set of given locations (the KPOIs). We assume that the overhead sensors (here, the Kinects) have a given map with the KPOIs. The required number of robots and the corresponding paths are acquired by using the algorithm proposed in [6]. For more details about the algorithm, the readers may refer to [6]. We then need to compute the robots' real positions corresponding to the map. The robots may need to follow each other as a team until they come to the fork of their paths. We deal with the robot control algorithms with the localization information acquired from the Kinects and dynamic formations for a team of robots in Paragraphs III-A and III-B, respectively.

A. Robot Localization with the Kinects

This section takes into account the localization algorithms to control a team of multiple robots with the overhead sensors (Kinects). This paper first considers the case of using one Kinect. Fig. 1 shows an example of structures representing paths and tours. The paths and tours are examples which are able to be generated by the algorithms described in [6]. As shown in Fig. 1, the PPCP needs to divide a team of robots at the start node (only one TP). The TPs are represented by the yellow triangles in the figure.



(a) Path. (b) Tour.

Fig. 1. Examples showing (a) the paths and (b) tours generated to cover a set of KPOIs. The leaf nodes on the tree structure in (a) and the small circles in (b) are the KPOIs, and the yellow triangles represent the TPs.

Algorithm 1: Kinect Localization Algorithm

Input: A map (a set of TPs and a set of KPOIs including each position).

Output: The desired position of each leader robot entered in the sensor.

Iterate the following whenever the sensor detects a leader robot:

1) Detect each leader robot in the sub-groups (if several sub-groups exist) shown in the sensor (Kinect).
2) Estimate the positions $P_1, \ldots, P_r$ of the robots.
3) Compute the next desired location for each robot. For $i = 1, \cdots, r$,
   a) If the position (P_i) is a TP, (For $j = 1, \cdots, k$)
      - Compute the number of branches ($B(i)$) connected to the TP and the required number of robots for each branch ($B_j(i) = \#R_j$).
      - Send a message to the robot located at $P_{B_{j-1}(i)}$ (if $j = 1$, then P_1) in order to split the group if there is a request.
      - If $j \neq 1$, change the robot located at $P_{B_{j-1}(i)}$ to the leader robot in the j^{th} sub-group.
   b) If the position is in the middle of a path and there is a request, send the next desired position (a TP or a KPOI) to the robot at P_i.
   c) If the position is a leaf node (a KPOI), we are done.

Algorithm 1 describes how to control multiple robots (the leader robots) from a Kinect system, which produces a 3D depth map. Here, k is the number of branches in each TP. The Kinect sensor has a color camera for images, a projector, and an IR receiver for IR images. By using a structured light technique the Kinect produces a 3D depth map. The sensor is able to be attached to robotic platforms, so it is feasible to use it as a moving platform as well. This work uses it as an overhead sensor in order to localize the robots working on their mission. At each TP, the sensor splits a team of robots into sub-groups, and sends each desired position to each leader of the sub-groups (Fig. 2). The number of sub-groups split at a TP is the number of branches. The number of robots in each sub-group is the required number of robots, which is the number of leaf nodes linked at each branch.

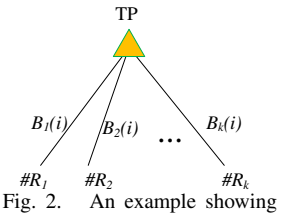


Fig. 2. An example showing a TP and the branches with the number of robot mentioned in Algorithm 1.

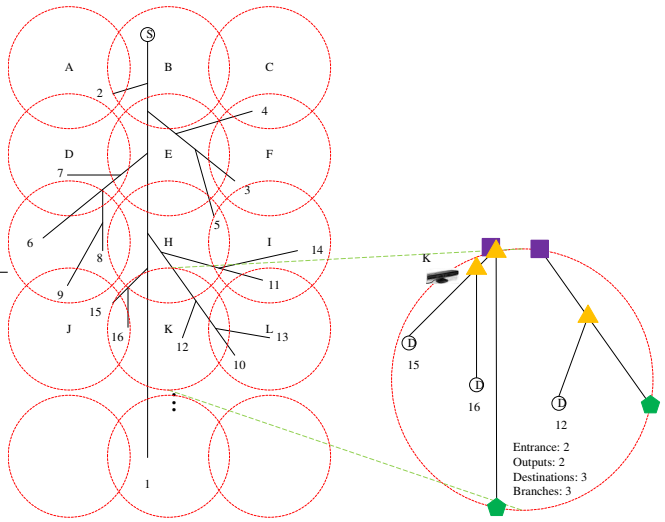


Fig. 3. An example showing the paths represented with a tree and the boundaries uniformly covered with multiple sensors. Each sensor includes the information about the numbers of the inputs (the squares), outputs (the pentagons), destination nodes (the circles), and branches (the triangles).

Algorithm 2: Multi-Kinect Connection Algorithm

Input: A map including K, P, and T with each position

Output: A tree structure to connect the Kinects

1) For each TP, $T_i \in T = \{T_1, \ldots, T_k\}$, $\forall i = 1, \ldots, k$,
   a) Find the number of leaf nodes: $L(i) = \#N_l$
   b) Find the number of branches and the corresponding number of robots: $B_j(i) = \#R_j$.
2) $K_1 =$ the Kinect including the start position.
3) For each Kinect, $K_i \in K = \{K_1, \ldots, K_l\}$, $\forall i = 1, \ldots, l$,
   a) Find adjacent Kinects: having the corresponding positions between the outputs in K_s and the inputs in

other Kinects in K, i.e. For $n_j \in K_i$, $\forall j = 1, \ldots, m$, find K_k, $\forall k$ such that *(i)* (n_j, n_k) are connected, *(ii)* $n_k \in K_k$, and *(iii)* $j \neq k$.

b) Link them (K_ks) to the Kinect (K_i): each link has the information regarding the number of connections (connect K_i and K_k, $\forall k$).

Depending on the mission, a given environment might be larger than the sensing boundary that is able to be obtained from a Kinect sensor. A sensor might have a smaller boundary than the expected one based on the height of the sensor. We examine how to manage multiple Kinect systems for the parallel coverage problem. Fig. 3 shows the use of multiple Kinects that are uniformly distributed to an example tree showing the paths. Algorithm 2 illustrates how to connect the multiple sensors into a tree structure (Fig. 4). For the algorithm we define $T, P,$ and K as sets of TPs, KPOIs, and Kinects, respectively.

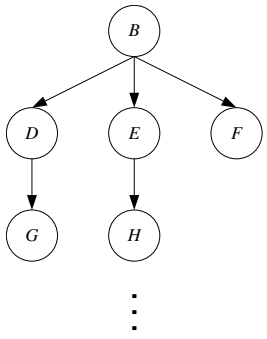
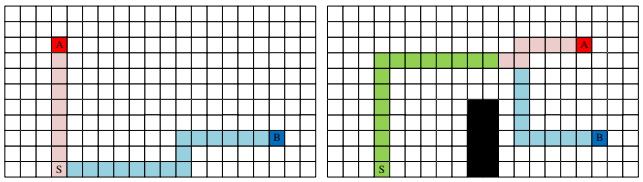


Fig. 4. A construction of a tree for multiple sensing boundaries generated from the information in Fig. 3.

B. Multi-robot Formations for Robot Control

This paper discussed how to localize the robots' current and desired positions in Section III-A, we now need to find control velocities in order to approach to the target locations. This section deals with the control of the robots through information procured from the Kinect systems. For the multi-robot formation control, differential mobile robots are considered. Instead of having all the robots communicate with the Kinect, only the leader robots attempt to use intercommunication with it. To do this we apply a vision-based formation scheme. Fig. 5 shows simple examples of two robots moving towards the target locations, named A and B in the cell-based maps. In Fig. 5(a), two robots are distributed from the start location (represented as S). Fig. 5(b), on the other hand, shows that the two robots need to move together up to the turning position (the path is represented by the green cells), and separate towards their desired positions (A and B).

We consider dynamic changes of the roles of robots in a formation to adapt to the paths in the parallel coverage problem as shown in Algorithm 1. The robots are initiated as one formation at the start position. They then separate as sub-groups at the TPs they encounter. Each leader robot is manipulated by acquiring the current and desired positions from the Kinect, while the follower robots in each team are autonomously controlled by using the camera sensor mounted on them. The control law regarding the follower robots is based upon the techniques in [8]. A robot formation branches at the TPs, and the role of each leader robot is to move to their goals through the information obtained from the Kinect.



(a) No formation is required. (b) A formation is necessary from S to the TP.

Fig. 5. Examples showing that the robots distribute towards their goal positions (a), and need to move together until the turning position (TP) and then separate to their goals (b).

1) Leader Robot: The control velocities for each leader robot are computed by the information acquired from a Kinect system. Let $P_L(t) = \begin{pmatrix} x_L & y_L \end{pmatrix}^T(t)$ and $P_L(t + \Delta_t) = \begin{pmatrix} x_L & y_L \end{pmatrix}^T(t+\Delta_t)$ be the current position and the desired next position of the leader. Then the control velocities are computed by the robot as follows:

$$\begin{aligned} x_L(t + \Delta_t) &= x_L(t) + \nu_L \cos\theta_L(t)\Delta_t \\ y_L(t + \Delta_t) &= y_L(t) + \nu_L \sin\theta_L(t)\Delta_t \\ \theta_L(t + \Delta_t) &= \theta_L(t) + \omega_L\Delta_t, \end{aligned} \tag{2}$$

where $\mathbf{u}_L = \begin{pmatrix} \nu_L & \omega_L \end{pmatrix}^T$ is the control velocity of the leader robot. We can find the control velocities for a leader robot using the position information.

2) Follower Robot: The control velocities for the follower robots are based on a relative geometric information, and are described as follows:

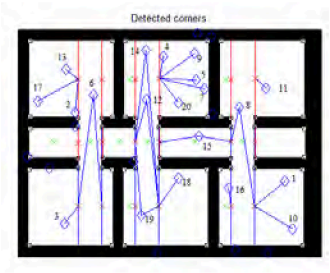
$$\mathbf{u}_F = \begin{pmatrix} \nu_F \\ \omega_F \end{pmatrix} = \begin{pmatrix} -\cos\alpha & 0 \\ \frac{1}{l}\sin\alpha & 0 \end{pmatrix} \begin{pmatrix} c_1\tilde{l} - \dot{x}_L\cos(\theta_F + \alpha) + \dot{y}_L\sin(\theta_F + \alpha) \\ c_2\tilde{\alpha} - \frac{1}{l}(\dot{y}_L\cos(\theta_F + \alpha) - \dot{x}_L\sin(\theta_F + \alpha)) \end{pmatrix} \tag{3}$$

where l, α, l_d, and α_d are the relative depth, orientation, the desired depth, and orientation between a leader (L) and the follower (F), respectively. Here, c_1 and c_2 are the user-selected controller gains, and $\tilde{l} = l - l_d$ and $\tilde{\alpha} = \alpha - \alpha_d$. More details regarding the follower's control law are described in [8].

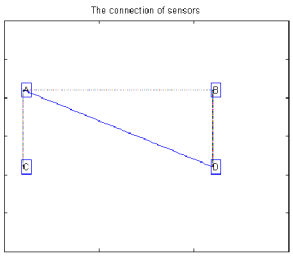
IV. EXPERIMENTAL RESULTS

In this section we demonstrate: *(i)* how to connect KPOIs and TPs with multiple sensors in simulation, *(ii)* how to extract a leader robot from a Kinect image, *(iii)* how to localize the robot using a Kinect, and *(iv)* how to control robots in a real environment through the intercommunication between a robot and a Kinect.

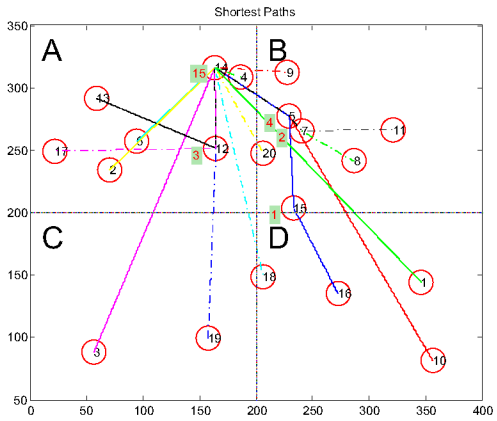
We first show a simulation using a generated map with a set of given KPOIs. Fig. 6(a) shows the map used in our previous work in [6]. Fig. 6(c) shows a tree starting at node 14. In the graph each node and edge include the information about the position and distance, respectively. There are fifteen leaf nodes at the root node, and four TPs at nodes $5, 7, 12,$ and 15. The numbers denoted around the TPs are the required



(a) Map.



(b) The connected tree with multiple sensors.



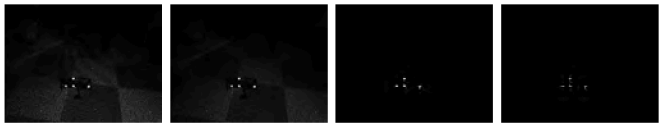
(c) The KPOIs and TPs.

Fig. 6. The simulation results showing (a) the given KPOIs in a map, (b) the connection of sensors, and (c) a generated tree (the root node is 14) with the leaf nodes and TPs.

number of robots. In this simulation we assume that there are four sensors to cover the environment, and name them A, B, C, and D. Fig. 6(b) shows the result of the generated tree in order to connect the four sensors. Based on the edges shown in Fig. 6(c), the sensor A is connected with B, C, and D, and B is connected with D.

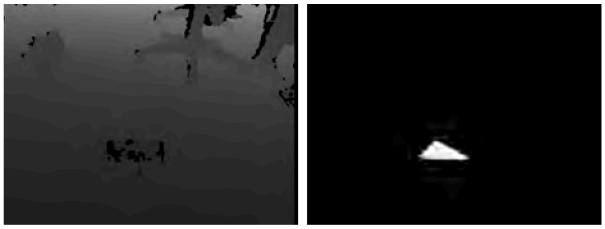
In order to extract the position of the leader robot from the Kinect, we are placing IR reflective markers on the leader robot which we detect from the IR image (Fig. 7(a)). The idea is to use intensity thresholding on this image to extract the marker blobs. Looking at the IR image, we notice that there are a lot of high intensity pixels. In order to smooth these out we pass the image through a median filter (Fig. 7(b)) the result of which we threshold (Fig. 7(c)). So far a hard set threshold is working well enough. If in the future adaptive thresholding becomes necessary we will look into algorithms like the Otsu's method. In Fig. 7(c) we still capture some of the IR spots from the Kinect projector. These we filter out with morphological operations on the image by processing the image with a 3x3 element. Then we process the image with a 5x5 element to remove the possible holes in the tracked blobs. The final result is given in Fig. 7(d).

Since the depth map generated by the Kinect is dependent on the IR image, having bright markers on the IR image confuses the Kinect's depth computation. Most of the time there is no direct depth data available at the points masked by the markers (Fig. 9(a)). In order to circumvent this problem



(a) IR Image (b) Median Filtering (c) Thresholding (d) Opening

Fig. 7. These figures show the steps taken to extract the markers on the leader robot from the Kinect IR image.



(a) Depth Image (b) Convex Hull

Fig. 9. These figures show the depth image generated by the Kinect (a) and the convex hull (b) built from the markers that is used to get a depth estimate of the leader robot.

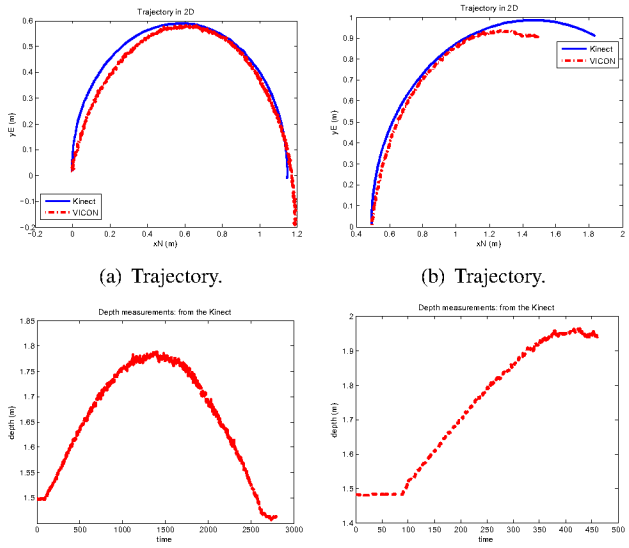
we are computing the convex hull of the detected markers (Fig. 9(b)), which we then use as a mask on the depth image. The distance estimate is then the mean of all the depth measurements contained in this region of interest.

Fig. 8. The experimental environment. A camera in the VICON system, a Kinect, and a robot are shown.

We try to compare the localization result from a Kinect sensor with the VICON system. The VICON system is a relatively expensive and very accurate camera motion capture system with six cameras. To compare the results we used one Explorer robot, developed at the University of Minnesota, with four markers attached on top of the robot and a Kinect system. Fig. 8 is the experimental setup showing one of six cameras in the VICON system, the Kinect, and one Explorer robot. Fig. 10 shows the comparative results. In comparison with the accurate capture system, the Kinect provides fair estimation results regarding the 2D position of the robot. Besides, it is relatively cheap, small, and attachable to a moving platform.

Finally, we implement a search mission in a real environment. Fig. 11(a) shows a layout of our office. The circle denoted with S is the starting position for a leader robot, and

the yellow triangle is a TP position. There are two KPOIs in the layout. The size of the environment is $3.2 \times 4.6(m)$. We use two Explorer robots in a formation, and they separate from the TP. Fig. 11(b) shows the trajectories of the two robots localized from the Kinect sensor.



(a) Trajectory. (b) Trajectory.

(c) The depth estimated from the Kinect corresponding to (a). (d) The depth estimated from the Kinect corresponding to (b).

Fig. 10. The comparison results of the trajectories estimated from the Kinect (the red dashed line) with the VICON system (the blue line). The ground truth trajectory is estimated with the relative information from the Kinect and the transformation of the coordinate system of the Kinect. The linear and angular velocities (m/s and rad/s) of the robot in (a) and (b) are $(0.015, 0.023)$ and $(0.08, 0.08)$, respectively.

V. CONCLUSIONS AND FUTURE WORK

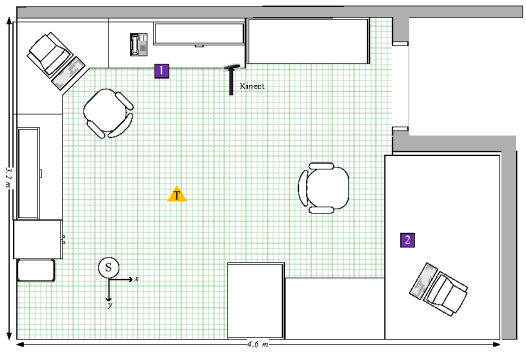
This work presented a new solution for the parallel coverage problem dealing with finding an optimal number of robots starting at the same location. It contributes a new algorithm based on multi-robot formation to reduce waste of inter-communication signals, and a method for real-robot implementation using Kinect sensors. We demonstrated our proposed algorithms to implement a search mission by using real robotic platforms (Explorers) and the Kinect sensor. For future work, we will consider placing Kinect sensors on mobile platforms in order to facilitate the cooperation between dynamic robots. The access to a good onboard sensing capability will improve the possibilities of a team of robots to execute a coverage mission.

ACKNOWLEDGEMENTS

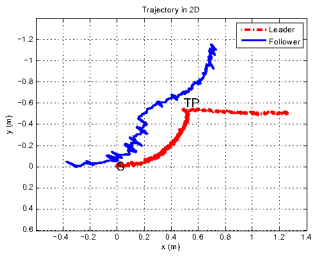
This material is based upon work supported by National Science Foundation through grants #IIP-044348, #IIP-0726109, #CNS-0708344, #IIP-0934327, #IIP-1032018, #IIS-1017344, #CNS-1138020, #IIP-1127938, and #IIP-1237259.

REFERENCES

[1] A. Weitzenfeld, "A prey catching and predator avoidance neural-schema architec ture for single and multiple robots," *Journal of Intelligent and Robotic Systems*, vol. 51, pp. 203–233, 2008.



(a) The layout.



(b) The trajectories.

Fig. 11. (a) The layout of our real experiment and (b) the trajectories of the two Explorer robots executing a mission. The layout is part of our laboratory, and the circle with S is the start position. The squares located at the upper left corner (1) and the lower right corner (2) are the desired positions for each robot.

[2] M. Moors, T. Röhling, and D. Schulz, "A probabilistic approach to coordinated multi-robot indoor surveillance," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005, pp. 3447–3452.

[3] M. Ahmadi and P. Stone, "A multi-robot system for continuous area sweeping tasks," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 1724–1729.

[4] N. Agmon, N. Hazon, and G. A. Kaminka, "The giving tree: constructing trees for efficient offline and online multi-robot coverage," *Annals of Mathematics and Artificial Intelligence*, vol. 52, pp. 143–168, 2000.

[5] K. Konolige, D. Fox, C. Ortiz, A. Agno, M. Eriksen, B. Limketkai, J. Ko, B. Morisset, D. Schulz, B. Stewart, and R. Vincent, *Centibots: Very Large Scale Distributed Robotic Teams*, experimental robotics ix ed. Springer-Verlag Berlin Heidelberg, 2006, pp. 131–140.

[6] H. J. Min and N. Papanikolopoulos, "The multi-robot coverage problem for optimal coordinated search with an unknown number of robots," in *IEEE Int. Conf. on Robotics and Automation*, 2011.

[7] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Proc. of the 6th Int. Symp. on Distributed Autonomous Robotics Systems*, 2002, pp. 299–308.

[8] H. J. Min, A. Drenner, and N. Papanikolopoulos, "Vision-based leader-follower formations with limited information," in *IEEE Int. Conf. on Robotics and Automation*, 2009.

[9] D. Fox, S. Thrun, F. Dellaert, and W. Burgard, *Particle filters for mobile robot localization*. NY, USA: Springer Verlag, 2000.

[10] S. Se, D. Lowe, and J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks," *The Int. Journal of Robotics Research*, pp. 735–758, Aug. 2002.

[11] J. Pugh and A. Martinoli, "Relative localization and communication module for small-scale multi-robot systems," in *Int. Conf. on Robotics and Automation*, 2006, pp. 188–193.

[12] J. M. O'Kane, "Global localization using odometry," in *Int. Conf. on Robotics and Automation*, 2006.

[13] R. Madhavan, K. Fregene, and L. E. Parker, "Distributed cooperative outdoor multirobot localization and mapping," *Autonomous Robots*, vol. 17, pp. 23–39, 2004.

[14] (2011, Mar.) Kinect. [Online]. Available: http://en.wikipedia.org/wiki/Kinect