

# Distributed Agent Control with Self-Localizing Wireless Image Sensor Networks

Chris McCormick, Pierre-Yves Laligand, Huang Lee, Hamid Aghajan  
Wireless Sensor Networks Lab, Department of Electrical Engineering  
Stanford University, Stanford, CA 94305  
Email: {chrismc, laligand, huanglee, aghajan}@stanford.edu

**Abstract**—In this work, we propose a technique for network-based control of mobile agents by distributed image sensors. The proposed method is based on tracking the movement of the agents using vision-based processing by the network nodes. Real-time control commands are issued to the agents by the nodes via wireless links, guiding them to desired destinations. A vision-based network localization technique is developed to allow each network node to learn its location and field-of-view in a relative coordinate system. Protocols for hand-over of the agent control between the image sensor nodes are also proposed. The developed platforms for implementing the proposed technique are described, and simulation and experimental results are presented.

## I. INTRODUCTION

Most applications in wireless sensor networks (WSNs) are built upon the objective of monitoring events and measuring parameters of interest in the environment. Information acquired by the sensors is often routed to an observer or a processing unit outside the bulk of the network via a gateway or base station. The observer may send commands to adjust the operation of the nodes, or request updates from a specific region. The availability of powerful, energy-efficient micro-controllers allows for in-node data processing. The network nodes may collaborate to detect and monitor different events and process the acquired data locally, thus providing the observer with processed and higher-level information without the need for a powerful centralized processing unit.

A new direction of research in wireless sensor networks involves the intersection of WSNs and robotics. The project reported in this paper addresses two overlapping areas of research interest: robot navigation, and the broader field of WSNs with actuators and agents.

Robot navigation is a classic research problem with many diverse approaches, most involving sensors mounted directly on the robot. Such sensors provide rich information about the robot's immediate environment, but this information is complicated to interpret and act upon, and generally only includes a nearby portion of the environment. Methods based on a map-learning step by each mobile agent prior to engaging in tracking targets or a Pursuit-Evasion Game (PEG) have been proposed [1], but complications caused by inaccurate measurements made by the agents have called for probabilistic approaches [2], [3], [4], which often result in more complex processing requirements. In other refinements, methods based on an omni-directional camera mounted on the robot [5], [6], [7] and active signaling by markers deployed

in the environment [8] have been proposed. In these methods the moving robot uses a sensing mechanism to find its way around in the environment. The work proposed in [9] relies on a complex feature point extraction and mapping scheme to develop a self-localization technique for robots.

In contrast, a wireless network of sensor nodes can offer simple and relevant navigation data from superior vantage points. For example, ceiling mounted nodes with cameras can provide a simple, two dimensional map of the environment and the relative location of the robot. Mounting sensors in the environment greatly reduces the complexity of the robots, since they no longer need to carry their own navigation sensors or processors, or the extra stored energy required to operate and mobilize this equipment and to process large amounts of sensed data. In fact, as we assume in the current work, the robot can be completely dumb, and simply receive and execute commands. A wireless sensor network approach can also support multiple robots, which greatly reduces the cost of scaling the network.

Techniques based on employing sensing devices mounted in the environment to help guide mobile agents have been explored. The work in [10] uses two red and blue light beacons mounted on the robot to aid a camera in identifying the robot. This is a fairly robust method of robot tracking, and has the added benefit of determining the robot's orientation. However, it employs rather elaborate image processing to recognize the light sources, and hence is not applicable to the case of micro-controller processing scales typically used in wireless sensor networks. In addition, the position of the camera is assumed known at the setup time. Such an assumption may not be valid in wireless sensor network deployments in which large numbers of nodes may be deployed making such manual calibration and localization tasks prohibitively difficult. Additionally, the mentioned work uses the information about the location of the robot only for tracking purposes and does not provide a feedback mechanism to affect the motion of the robot. In a networked approach, a scheme for agent navigation has been proposed utilizing stationary sensors placed in the environment. A grid of wirelessly connected electromagnetic sensors is used in [11] to approximate the location of a robot and aid a pan-tilt camera in tracking the robot. This is a first step in monitoring the location of the robot in its environment, but as proposed in the current work, the localization and control functions can also be accomplished by the camera

alone or by a network of cameras.

Using such a network to navigate a robot is merely the short-term goal of our project, however. A new direction in application design for wireless sensor networks is to enable these networks to affect changes in the environment based on the information they collect from the events. Such changes can be exerted through the use of actuators that are part of the wireless network. Mobile agents can also be included in the network and be commanded to visit specific locations or take on various tasks.

By using local and collaborative processing, the network nodes can issue control commands to mobile agents that can be dispatched and tracked by the network. Such agents may be commanded to provide additional measurements at points within the network or visit hard-to-reach areas and obtain specific measurements. Applications in monitoring and sampling environments used for hazardous material storage are examples of the need to sample the environment with finer granularity than offered by a fixed deployed sensor network. Deployments of mobile robots that communicate with the observer via a set of dedicated radio relay agents have been reported [12]. However, such deployments lack the flexibility for handling multi-agent scenarios, and the task of providing fine-grained control commands to the robots by the operator makes such an approach not scalable. By using a wireless sensor network, direct observations by the nodes and their proximity to the agents allow for a real-time control loop to be established locally, thus avoiding potential communication delays due to multi-hop transmission to and from the outside processing unit or observer. The ability to dispatch and control mobile agents enables the wireless sensor network to also influence the observation environment, and, for example, engage in PEG applications [11].

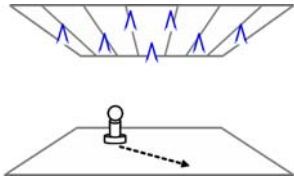


Fig. 1. Simple diagram showing a mobile agent under the control of a network of image sensors.

In this paper, we propose a technique to monitor, track, and command a number of mobile agents by a network of wireless image sensors. In one realization of the concept, the image sensors are assumed to be deployed on the ceiling with their image planes parallel to the ground. Fig. 1 includes a simple schematic showing the system’s physical model. A decentralized and scalable technique is proposed for network localization, which provides estimates for the location of the image sensors, their rotation angle with respect to a common coordinate system, their height from the ground, and each image sensor’s coverage area. The localization technique involves assistance by a mobile object that traverses the area monitored

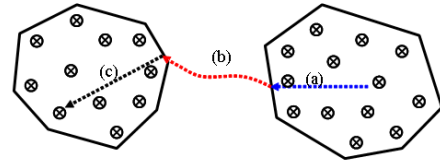


Fig. 2. An example of a partitioned network of image sensors, in which a moving agent is first commanded in (a) by the first partition, then uses dead reckoning in (b) with some errors to get to the second partition, and is commanded by the second partition in (c) to reach the destination point. The agent can deliver packets between the partitions and carries data about its destination during the dead reckoning phase.

by the set of image sensors to define a relative coordinate system. The method is decentralized meaning that each image sensor node makes independent visual observations and calculates its own location with respect to the relative coordinate system. This makes the scheme scalable to large networks since no additional communication overhead is involved in performing this task. Details of the localization technique are provided in Sec. III.

The work presented in this paper aims to incorporate the use of a wireless network of image sensors equipped with low-bandwidth radios to monitor, track, and control a set of mobile agents. This project will allow us to explore pertinent wireless sensor network topics such as:

- 1) Use of a wireless sensor network of image sensors and low-bandwidth wireless communication to monitor and track agents,
- 2) Development of a feedback control mechanism based on visual observations from which motion control commands are issued to the agents over wireless links at regular sampling times,
- 3) Automated localization and coverage area deduction technique for wireless image sensor networks using visual observations of a moving object,
- 4) Development of novel applications for using mobile agents to bridge the communication gap between wireless nodes that are too far apart to communicate. Examples include tasking the agents to make measurements at various geographic points in a sparsely deployed sensor network, dispatching the agents to collect measurements stored at nodes that do not have radio links with the rest of the network, and assigning the task of carrying data packets between two partitioned parts of a wireless network (Fig. 2).

In the remainder of the paper, we first present the model used in setting up the system in Sec. II, and then discuss the technique used for self localizing the network of image sensors in Sec. III. In Sec. IV we provide details of the hardware and system setup for monitoring and communicating with the mobile agents. Sec. V discusses our approach to providing a feedback control mechanism for the motion of the mobile agents. In Sec. VI the results of experiments characterizing the behavior of the system are presented, and Sec. VII offers some concluding remarks.

## II. SYSTEM MODEL

This work is based on a network of devices with wireless communication capability. The system is composed of two types of nodes: image sensor nodes deployed in an adequately large number to provide a visual coverage of the environment, and mobile agent nodes with limited or no on-board sensing capability. The image sensors may operate on batteries or use wired power. However, the communication link between an image sensor and the mobile agents or other image sensors is over wireless channels. The image sensors are deployed on the ceiling looking down at the lab floor (Fig. 1). The only assumption made about the deployment of the image sensors is that their image planes are parallel to the ground. Variations in the location and orientation of the image sensors at deployment time can include a rotation around the axis normal to the image plane, an unknown height for the camera from the ground, and each camera's 2D coordinates relative to a plane parallel to the ground. These parameters are shown in Fig. 3(a). We also assume that the angular field-of-view (FoV) and the number of pixels in the image plane are known. These numbers are fixed physical parameters of image sensors and can be known or measured at manufacturing time. The system can be modeled in the two-dimensional plane, assigning position and orientation parameters to each image sensor. However, each image sensor also has a height parameter, which is unknown and its value is estimated in the localization technique described in Sec. III. Assuming all image planes to be parallel to the system plane, we use a pinhole camera model described by the equation

$$\varphi = \tan^{-1} \left( \frac{2d}{D} \tan \left( \frac{\psi}{2} \right) \right), \quad (1)$$

where  $\varphi$  represents an observed object's angular displacement from the camera's normal view,  $d$  represents the distance from the center of the image plane in pixels,  $D$  represents the image plane dimension in pixels and  $\psi$  represents the FoV angle of the image sensor. This relationship can easily be derived from the pinhole model shown in Fig. 3(b).

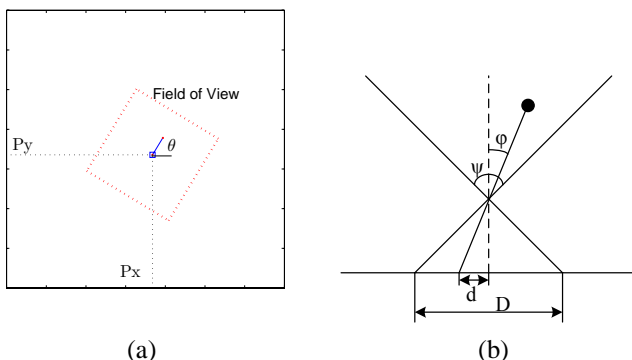


Fig. 3. (a) The coordinates of an image sensor in a plane parallel to the ground (top view). (b) Pinhole camera model. Parameter  $D$  represents the dimension of the image plane.

In our application, the mobile agents do not perform data processing for their navigation and do not communicate directly with each other. This means that, from the agents points

of view, the network of image sensors appears as a centralized processing and command entity, to which the agents have a star networking relationship. From the standpoint of the image sensor network, the network is equipped with distributed processing capability, is a peer-to-peer network, and some of the nodes have transient links with new members, which are the agents when they get associated with an image sensor node.

## III. VISION-BASED NETWORK LOCALIZATION

In order to track the agents by a wireless sensor network, the relative positions of the network nodes with respect to the environment need to be known. Node localization is an important step in network initialization, and a plethora of techniques have been proposed in the past to address this. For example, there have been several approaches based on using the received signal strength indicator (RSSI) at the radio to estimate the distance between two communicating nodes [13], [14]. However, obtaining precise ranging information from RSSI measurements is extremely difficult in most environments. Alternatively, schemes have been proposed using radio connectivity [15] and ultrasonic transducers [16]. There has also been increasing interest in sensor networks with image sensors, yet little investigation of localization in wireless networks with imaging capabilities has been done. Recently, an iterative method was proposed in [17] to solve for the image sensor localization problem. However, the method is computationally expensive and requires extensive communication between nodes. Most prior studies of image sensor localization for robotic applications are based on using varying amounts of prior knowledge of the environment. Techniques exploiting landmark-based position recognition [18] and a 3-dimensional knowledge of the environment [19] have been proposed. Related literature relies on the ability to estimate an object's distance from the image sensor using prior knowledge of the object's size and a high-resolution image sensor requiring significant calibration. The method in [20] estimates an image sensor's distance from a marker using fixed-sized markers and the marker's pixel length in an image frame.

In this section an automated and decentralized network localization scheme is proposed based on a few observations of a moving agent that defines a relative coordinate system for the network. The beacon is a simple line-following robot which follows tape placed on the ground with detectable marks at set increments. Each time the beacon reaches a mark, it broadcasts its location relative to its starting point. The distance between marks must be such that the robot makes at least two stops within the FoV of each of the cameras. The tape is laid down in an s-pattern, with a fixed distance between parallel lines, so that the beacon may always know its location. Thus, the  $x$  and  $y$  axes and the unit length of a 2-D coordinate system are defined by the beacon. When the agent stops and broadcasts its coordinates, the image sensors in its communication range each acquire a frame from their respective FoVs. The agent may be observed by a few image sensors, each of which then detects the beacon's location on its image plane by simple frame subtraction using an initial background frame. Fig. 4(b)

shows an example of two camera FoVs in which the agent is observed as it follows the marked line.

More specifically, at time step  $i$ , the robot broadcasts its coordinates denoted by a  $2 \times 1$  vector  $s_i$ . These coordinates can be transferred into the coordinates on the image plane of the image sensor by

$$y_i = \alpha \mathbf{R} (s_i - p) + n_i, i = 1, \dots, N_b, \quad (2)$$

where  $y_i$  is the  $2 \times 1$  observed vector of image plane coordinates, the vector  $p$  includes the coordinates of the sensor, the vector  $n_i$  is the noise modeled by a Gaussian random variable,  $\alpha$  is an unknown scaling factor indicating the height of the camera, and  $\mathbf{R}$  is the rotation matrix with the unknown sensor orientation  $\theta$ , i.e.

$$\mathbf{R} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}. \quad (3)$$

To solve for the unknown variables  $\alpha$ ,  $\theta$ , and  $p$ , we first rearrange (2) to cancel the vector  $p$  and obtain

$$y_{i+1} - y_i = \alpha \mathbf{R} (s_{i+1} - s_i) + (n_{i+1} - n_i), i = 1, \dots, N_b - 1. \quad (4)$$

We define  $2 \times 1$  vectors  $\tilde{s}_i$ ,  $\tilde{y}_i$ , and  $\tilde{n}_i$  by

$$\tilde{s}_i = s_{i+1} - s_i, i = 1, \dots, N_b - 1, \quad (5)$$

$$\tilde{y}_i = y_{i+1} - y_i, i = 1, \dots, N_b - 1, \quad (6)$$

$$\tilde{n}_i = n_{i+1} - n_i, i = 1, \dots, N_b - 1. \quad (7)$$

Then Eq. (4) can be rewritten in the matrix form as

$$\begin{bmatrix} \tilde{y}_1(1) \\ \tilde{y}_1(2) \\ \vdots \\ \tilde{y}_{N_b-1}(1) \\ \tilde{y}_{N_b-1}(2) \end{bmatrix} = \begin{bmatrix} \tilde{s}_1(1) & \tilde{s}_1(2) \\ \tilde{s}_1(2) & -\tilde{s}_1(1) \\ \vdots & \vdots \\ \tilde{s}_{N_b-1}(1) & \tilde{s}_{N_b-1}(2) \\ \tilde{s}_{N_b-1}(2) & -\tilde{s}_{N_b-1}(1) \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} + \tilde{n} \quad (8)$$

where  $\tilde{n} = [\tilde{n}_1(1), \tilde{n}_1(2), \dots, \tilde{n}_{N_b-1}(1), \tilde{n}_{N_b-1}(2)]^T$ , and  $v = \alpha \cos \theta$  and  $w = \alpha \sin \theta$  are the unknown variables.

We can solve for  $v$  and  $w$  in Eq. (8) using a standard least-squares technique, which leads to a closed-form solution here since only a  $2 \times 2$  matrix inversion is needed in our model. After obtaining the estimation results  $\hat{v}$  and  $\hat{w}$ , we can estimate  $\hat{\alpha}$ ,  $\hat{\theta}$  and  $\hat{\mathbf{R}}$ . The sensor coordinates  $p$  then can be derived by

$$\hat{p} = \frac{1}{N_b} \sum s_i - \frac{1}{\hat{\alpha}} \hat{\mathbf{R}}^{-1} \hat{y}_i. \quad (9)$$

Knowing estimates for the image sensor's coordinates in the relative coordinate system defined by the movement of the agent, along with the estimates for the camera's height and orientation allows us to create a visual coverage map of the environment. After each image sensor has calculated its own location parameters, the nodes take turn to broadcast their parameters to all nodes within their communication range. This information is used later in the agent control protocol for handing over the control of the agent as it leaves one image sensor's FoV and enters another one's. Fig. 4(a) shows a simulated network of 9 image sensors deployed randomly

with random orientations for their image planes. In Fig. 4(b), the path and the locations where the moving agent makes stops to be visually observed by the cameras are shown. Each image sensor needs to observe the agent on at least two stop points to be able to solve for its own location parameters. The results of self-localization by the image sensors are shown in Fig. 4(c), in which the estimates for the coordinates and the orientation are superimposed on the actual camera positions, and each camera's FoV is also drawn based on the estimated orientation and the estimated height of the camera.

#### IV. SYSTEM AND HARDWARE SETUP

Our hardware platform consists of an expandable network of wireless nodes based on demonstration boards with Chipcon's CC2420 radios [21] that operate under the IEEE 802.15.4 protocol. These demonstration boards include Atmel Atmega128L microcontrollers (16MHz with 128KB of Flash memory) [22], 2 serial ports, and simple features such as LEDs, pushbuttons, a small joystick, and a potentiometer.

##### A. The Robot

The robot used in our test deployment (Fig. 5(a)) is built around Parallax's Boe-Bot robotics kit, which uses two continuous rotation servos as motors. Unlike typical electrical motors with only positive and negative terminals, servos are controlled by a Pulse-Width Modulation (PWM) signal. This allows for precise motor control and variable motor speeds.

We mounted a Chipcon demonstration board on the robot to provide the agent with the ability to receive and process movement commands via the radio link. To interface the Chipcon board with the robot kit, we wrote an API for the board's microcontroller to translate simple movement commands into PWM signals for sending to the motors. The API includes move forward / backward, and rotate left / right commands as well as commands for adjusting speed, moving a specified distance (in inches or centimeters), and rotating a specified number of degrees to the left or right. These last two commands are a tribute to the merits of placing sensors in the environment instead of on the robot. Accurate movement commands are easily achievable with the reasonable consistency and accuracy of the small servos. The robot's relatively small size (4.5 inches wide by 5.5 inches long), and light weight (due to the absence of bulky sensors and heavy-duty processing units on the robot), allow for the use of these servos instead of larger motors. Also, considering the operating distances from the cameras in indoor environments (3 to 4 meters from cameras on the ceiling), the robot's size could easily be reduced further while its image plane size would still allow for it to be tracked.

##### B. User Interface

We developed a User Interface in Java to monitor the position of the agent as it moves in the FoV of the image sensors, and to send commands to the robot using a Chipcon wireless mote as a transmitter. The UI communicates with the transmitter over a serial connection (Fig. 5(b)). The screen

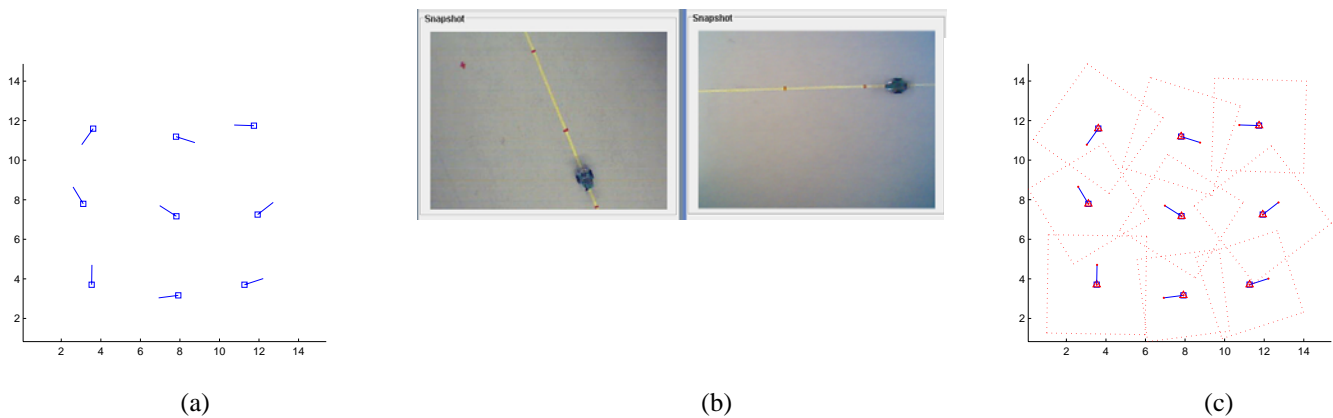


Fig. 4. Results of the localization algorithm simulated for a network of 9 image sensors randomly deployed (top view). (a) Image sensor locations and orientations. (b) The path the moving agent follows and the observation points. (c) Algorithm results showing the estimated coordinates, orientation of the image plane, and the FoV of each image sensor.

capture in Fig. 5(c) shows how the detected position and trail of the agent are superimposed on the captured image frame. The UI includes controls to send basic commands to the robot, as well as run the control algorithm to autonomously navigate the robot to a destination specified in pixel coordinates. Ultimately, the UI serves as a test bed for developing control algorithms before porting them to the microcontrollers.

### C. Image Sensor

We designed a hardware interface between the Chipcon wireless mote board and the image sensor model ADCM 1670 by Agilent Technologies to equip the network motes with image sensing capability. Initially, we employed USB webcams to develop the user interface and the agent tracking and control algorithms. The ADCM 1670 image sensor has a CIF-scale resolution and supports both serial and UART interfaces. A new wireless image sensor mote with on-board image sensing, IEEE 802.15.4 radio, and ARM processor is being designed at our lab, which we plan to use in the next phase of deployment for the proposed distributed agent control technique.

### D. Robot Tracking

A simple image differencing algorithm is employed to track the robot in the camera's FoV. This algorithm yields coordinates of the moving agent in the environment's relative coordinate system (as discussed in Sec. III). Control commands are issued to correct for the movement of the agent after it makes a movement (linear or rotational). Tracking of multiple agents is discussed in Sec. V.

## V. CONTROL MECHANISM

If the robot could execute movement commands with perfect precision, the network could simply decide on a course for the robot to take and then rely on the robot to arrive at the intended destination. The network would not then need to continuously monitor the robot's movement to see whether or not it requires correction. However, such accuracy assumptions are unrealistic, and, moreover, the position of the destination

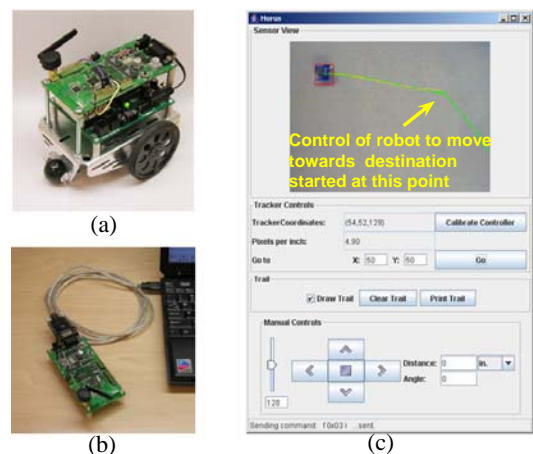


Fig. 5. (a) The robot is equipped with a Chipcon 2420DB Demonstration Board with a 2.4 GHz radio and an Atmel ATmega128L micro-controller. (b) A wireless mote (Chipcon 2420 Demonstration board) is interfaced with a computer used for monitoring and giving commands to the robots by the user. (c) The user interface allows for defining a destination for the robot and shows the view from the monitoring image sensor and the robot's path.

may change during the robot's journey, requiring interaction between the network and the robot along the way.

The robots in [23] relied on accurate odometry to arrive at their destination, but showed an average error of 16% over 10 trials on a 300cm path. In that work, the robot polls the network for sensed data and makes its own decision, thus the need for the network to track the robot is eliminated. However, without knowledge of its actual coordinates relative to those of the destination, the accuracy of the system is limited as odometry errors accumulate. Moreover, the technique cannot be used when the destination moves as in a PEG application.

Without any sensing or self-correction capability on the agent, the network must process the visual information it obtains from the agent at every sampling instance and issue a new command to correct the course of the robot. The control algorithm can use one of two approaches to correct the robot's

movement:

- 1) Re-orient the robot towards the destination, or
- 2) Re-orient the robot towards the desired path to the destination.

The second option is better suited for navigating the robot around obstacles, where it must adhere to a specific path to reach the destination. Since our network has not yet been configured to handle obstacles, we have opted for the first option, as it offers the quickest path to the destination.

The platform outlined in this paper has been created to study the challenges involved in the operation of a multi-agent, multi-camera network. Closing a feedback control loop using image frames, position tracking, and issuing wireless control commands when the agent is monitored within the FoV of an image sensor poses a set of such issues. In a distributed monitoring and control situation, a key challenge is handling the distributed forms of observed data and designing a protocol to allow for efficient hand-over of control between different network nodes. In the case of visual sensing as in our platform, multiple camera nodes observe the environment, and their FoVs may overlap, or there may be observation gaps between the FoVs of the image sensors that are not covered by any nodes. In order to properly delegate the control of the robot between the nodes, the image sensor nodes must have an understanding of the relative locations to their neighbors. The network localization scheme presented in Sec. III establishes a technique for finding the visual coverage area of the network and the relative positioning of the FoVs of the sensors.

To design an algorithm for controlling multiple agents using a network of sensors, the problem of tracking and identification of the agents as they move across the network poses another challenge. A mechanism we adopted to facilitate tracking in the case of image sensor nodes is to employ a vision-based handshaking protocol when an agent enters into a camera's FoV. The protocol calls for a signaling scheme in which the sensor separately commands each robot entering into its FoV to turn on its LED for a period of time in a stationary position. This enables the sensor node to make association between the identity of each agent communicated through the radio and its corresponding detected blob in the image plane.

The algorithm to observe and command an agent within the FoV of an image sensor consists of the following steps:

- 1) If the user has specified a destination point within the FoV of the image sensor, a box is drawn at the specified coordinates to highlight the destination.
- 2) The algorithm uses the direction that the robot entered into the FoV to determine the robot's orientation.
- 3) With the robot's orientation known, the algorithm calculates the angle the robot needs to rotate to face the destination, and sends the appropriate command.
- 4) The ratio between pixels and inches is known from the localization of the network, and is used to calculate the distance to the destination after the correction in the agent's orientation.
- 5) Rather than issuing a single command to have the robot

TABLE I  
PSEUDO-CODE FOR IMAGE-BASED CONTROL ALGORITHM

```

1: // Receive command from user interface:
2: goToCoordinates( $x_{dest}, y_{dest}$ )
3: // Control algorithm:
4:  $x_i =$  getRobotXCoordinate()
5:  $y_i =$  getRobotYCoordinate()
6:  $d_{pixels} =$  getDistanceToDestInPixels( $x_i, y_i$ )
7:  $d_{inches} =$  convertPixelsToInches( $d_{pixels}$ )
8: while  $d_{inches} > d_0$  do
9:    $x_i =$  getRobotXCoordinate()
10:   $y_i =$  getRobotYCoordinate()
11:  moveForward( $d_0$ , inches)
12:  wait()
13:   $x_f =$  getRobotXCoordinate()
14:   $y_f =$  getRobotYCoordinate()
15:   $\theta_{robot} =$  calculateRobotOrientation( $x_i, y_i, x_f, y_f$ );
16:   $\theta_{dest} =$  calculateAngleToDest( $x_f, y_f, x_{dest}, y_{dest}$ );
17:  rotateTowardsDest( $\theta_{robot}, \theta_{dest}$ );
18:  // Check if robot is in hand-off zone:
19:  if robotInHandoffZone( $x_f, y_f$ ) &
    robotLeavingView( $x_f, y_f, \theta_{robot}$ ) then
20:    nextNode = calculateNextNode( networkTopology, robotCo-
    ordinates, robotOrientation)
21:    nextNodeAddress = getAddress(nextNode)
22:    passRobotControl(nextNodeAddress)
23:  end if
24: end while
25: // Move forward the remaining distance:
26: moveForward( $d_{inches}$ , inches)

```

drive all the way straight to the destination, the path is broken up into increments (currently 6 inches long). Each time the robot moves forward another 6 inches, the algorithm uses the initial and final coordinates from the move to recalculate robot's orientation.

- 6) The algorithm continues to issue commands to the robot to move 6 inches at a time and to re-orient itself towards the destination using simple image processing operations, until the robot's coordinates are found to be within less than 6 inches of the destination, at which point the robot is commanded to move forward the exact remaining distance after a possible correction in its orientation.

Table I includes a pseudo-code for the image-based control mechanism. The algorithm has proven effective in guiding the agent to the desired destination. This is despite the errors caused by the robot's lack of full adherence to the exact execution of each command. The latter effect is due partly to an internal calibration ratio that translates the length of linear motion commands to turns in the motors. An example of how this ratio affects the response of the robot is presented in Sec. VI. We opted to not tightly adjust the calibration ratio in order to allow the control mechanism to prove its effectiveness in face of a bias in command execution by the robot.

An important property of a control algorithm is its sampling rate - the frequency with which the algorithm re-evaluates the robot's course and sends a new command. The factors affecting the decision on sampling rate selection in our network-based

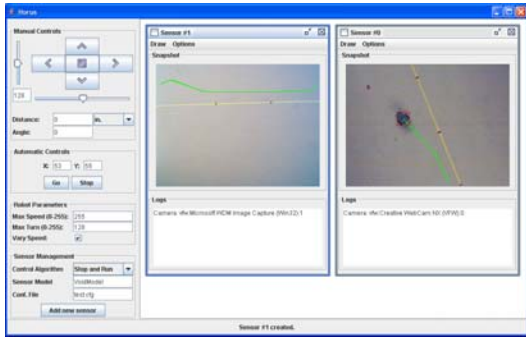


Fig. 6. Example of multi-camera view.

approach include the cost of image capture and processing, the energy used in radio transmission to issue control commands, and the relatively low moving speeds in mobile robots as compared to image frame rates available in typical image sensors. Based on these factors, we opted to base the system's sampling rate on the distance rather than choosing a time-based rate. The control algorithm issues new commands after a set distance is traversed by the agent, while a more frequent image sampling rate is employed with simple image differencing to track the robot. This approach greatly simplified the robot's control algorithm for the preliminary development of our system. As an extension to our development effort, we plan to develop a hybrid sampling rate scheme based on both time and distance in which the network would adjust its radio communication rate for each agent based on the granularity of the tasked commands, and according to a priority table defined by the application.

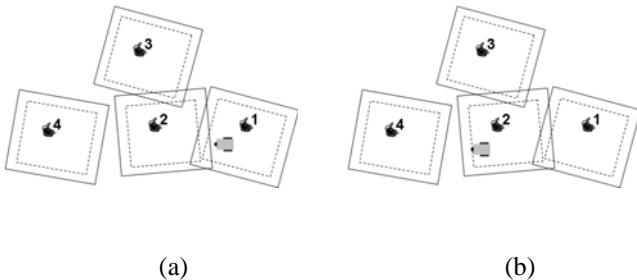


Fig. 7. Examples of cases for the hand-over of robot's control between image sensors with (a) overlapping FoVs, (b) non-overlapping FoVs.

With multiple cameras managing the robot, as in Fig. 6, conflicts between commands from different cameras might arise. Based on the knowledge of the network topology provided by the localization algorithm of Sec. III, the following scheme is used to enable a hand-over mechanism for when the agent needs to move across the FoVs of several image sensors:

- 1) When the robot is about to leave a camera's FoV, it enters a *hand-over* zone around the edge of the image. Based on the neighbor topology information, the node determines whether another image sensor has an overlapping FoV at the edge of its image plane.
- 2) If such an overlapping FoV exists (See Fig. 7(a)), the

node notifies that neighbor to prepare to take over the control process when the robot enters into the overlapping zone.

- 3) If the node determines that there is a gap between its FoV and a neighbor's FoV along the path that the robot is supposed to move along (See Fig. 7(b)), it issues a rotation command so the robot's orientation is headed towards the center of the intended neighbor's FoV, and then issues a linear movement command for the robot to move a long enough distance to enter the FoV of that neighbor. It also notifies the neighbor about this effect and provides an estimate of the robot's time of arrival based on its recent speed history.

## VI. EXPERIMENTAL RESULTS

In this section we present experimental results of using image sensor-based tracking and control of the mobile agent. As we shall see, even though the robot does not execute the movement commands issued to it exactly and makes errors in the steps it takes, the control mechanism can successfully guide it to reach the destination. This is an expected benefit of using closed-loop feedback to control dynamic systems. Fig. 8(a) shows the results of the agent control algorithm used to guide the robot from 3 different initial points to a fixed desired destination in a camera's FoV. In each case, before starting to issue the actual control commands to move the agent towards the destination, the agent is commanded to move a specific short distance so that its orientation is determined by the image sensor from the direction of its path. Knowing the heading of the robot, the subsequent commands are issued to make corrections to its course and move it forward until it reaches the destination. In our control strategy, if the distance of the robot from the destination is more than 6 inches, a command to move 6 inches is issued to the robot. After the robot finishes executing the command, it notifies the controller. Then, the image sensor calculates a correction for the heading angle of the robot and issues a command if needed to correct the angle towards the destination. After that command is executed by the robot, the controller issues another linear movement command and the mentioned steps are repeated until the agent's distance to the destination is within a fixed tolerance range. Fig. 8(b) shows an actual snapshot taken by the image sensor on which the actual path of the agent and the set of linear movement commands are superimposed. As it can be observed in this figure, the robot makes errors in executing some of the commands and deviates from the direct path to the destination. However, its path is repeatedly corrected until it reaches the destination. Fig. 9(a) shows the distance between the robot's actual path and the ideal straight line between the starting point and the destination for 5 experiments. The robot's large deviance from the straight-line path is inherent to our control approach—the control algorithm prioritizes getting the robot to the destination, not adhering to a specific path.

Fig. 9(b) presents the average error in the robot's execution of a command over 4 experiments. Notice that the robot consistently undershoots the command. This is actually a

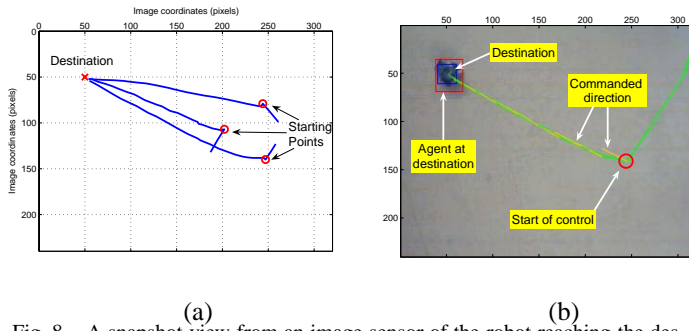


Fig. 8. A snapshot view from an image sensor of the robot reaching the destination with actual path and the linear movement commands superimposed.

controllable aspect of the robot: there is a fairly constant ratio between the number of cycles the motors are on and the number of centimeters the robot travels. This ratio can be fine-tuned to improve the robot's accuracy. However, we decided to maintain this undershoot bias, since commands with an overshoot bias and too low of a sampling rate often result in unstable oscillations in the feedback control loop.

Finally, we measured the error in reaching the destination in 5 tests with the same initial and destination points. The average error in reaching the destination was 0.73 inches, which was equivalent to 1.7% of the length of the direct path. An example video of the mobile agent being guided to reach a destination point can be found at [http://wsnl.stanford.edu/demos/agent\\_control\\_demo12.avi](http://wsnl.stanford.edu/demos/agent_control_demo12.avi).

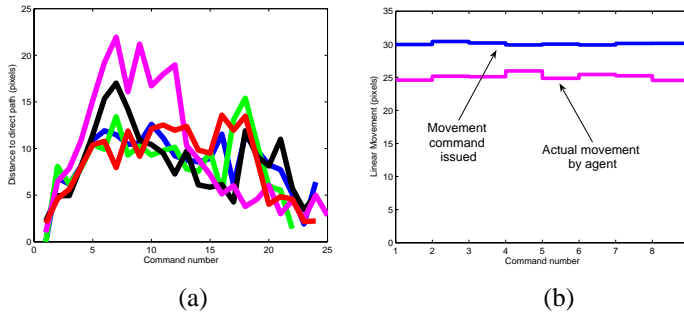


Fig. 9. (a) Results of 5 tests showing deviation of the robot's actual path from the straight line between the starting and destination points. (b) Average of 4 commands and actual distances traversed by the robot.

## VII. CONCLUSIONS

We have developed a technique for distributed control of mobile agents by a network of image sensors. The network visually tracks the movement of the agents using simple image processing operations, and guides them to desired destinations by issuing real-time control commands via wireless link, forming a closed control loop. A vision-based network localization technique was developed allowing each network node to learn its location and field-of-view in a relative coordinate system. Protocols for hand-over of the agent control between the image sensor nodes were also proposed. Extensions of the work include guiding the agents along specified paths and implementing a network-based obstacle detection and avoidance technique.

## VIII. ACKNOWLEDGMENTS

We would like to thank Chipcon for their donation of IEEE 802.15.4 wireless motes used in this project.

## REFERENCES

- [1] X. Deng, T. Kameda, and C. Papadimitriou, "How to learn an unknown environment, I: the rectilinear case," *Journal of the ACM*, pp. 215–245, V.(45)2, 1998.
- [2] S. Thrun, W. Burgard, and D. Fox, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learning / Autonomous Robots*, pp. 1–25, V.(31)85, 1998.
- [3] J. Kim, R. Vidal, H. Shim, O. Shakernia, and S. Sastry, "A hierarchical approach to probabilistic pursuit evasion games with unmanned ground and aerial vehicles," in *Proc. of 40th IEEE Conf. Decision and Control*, 2001, pp. 634–639.
- [4] J. Hespanha, H. J. Kim, and S. Sastry, "Multiple-agent probabilistic pursuit-evasion games," in *Proc. of 38th Conf. on Decision and Control*, 1999, pp. 2432–2437.
- [5] R. Cassinis, D. Grana, and A. Rizzi, "Self localization using an omnidirectional image sensor," in *Proc. of SIRS96 4th Int. Symp. on Intelligent Robotic Systems*, 1996. [Online]. Available: [http://www.ing.unibs.it/~cassinis/docs/papers/03\\_015.pdf](http://www.ing.unibs.it/~cassinis/docs/papers/03_015.pdf)
- [6] C. F. Marques and P. U. Lima, "A localization method for a soccer robot using a vision-based omni-directional sensor," in *RoboCup 2000: Robot Soccer World Cup IV*, 2001, pp. 96–107.
- [7] A. Bonarini, P. Aliverti, and M. Lucioni, "An omnidirectional vision sensor for fast tracking for mobile robots," *IEEE Trans. on Instrumentation and Measurement*, pp. 1–25, V.(49)3, 2000.
- [8] R. Cassinis, F. Tampalini, and R. Fedrigotti, "Active markers for outdoor and indoor robot localization," in *Proc. of DEA-Unibs*, 2005. [Online]. Available: [http://www.ing.unibs.it/~cassinis/docs/papers/05\\_009.pdf](http://www.ing.unibs.it/~cassinis/docs/papers/05_009.pdf)
- [9] H. Hajjdiab and R. Laganieri, "Vision-based robot localization," in *Proc. of 2nd IEEE Int. Workshop on Haptic, Audio and Visual Environments and Their Applications*, 2003, pp. 19–24.
- [10] R. Cassinis, P. Meriggi, and M. Panteghini, "A very low cost distributed localization and navigation system for a mobile robot," in *Proc. of ECMR*, 2003, pp. 171–176. [Online]. Available: [http://www.ing.unibs.it/~cassinis/docs/papers/03\\_036.pdf](http://www.ing.unibs.it/~cassinis/docs/papers/03_036.pdf)
- [11] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. S. Sastry, "Distributed control applications within sensor networks," *Proceedings of the IEEE*, pp. 1235–1246, V.(91)8, Aug. 2003.
- [12] H. G. Nguyen, N. Pezeshkian, A. Gupta, and N. Farrington, "Maintaining communication link for a robot operating in a hazardous environment," in *Proc. of 10th Int. Conf. on Robotics and Remote Systems for Hazardous Environments*, 2004.
- [13] P. Bergamo and G. Mazzini, "Localization in sensor networks with fading and mobility," in *Proc. of IEEE PIMRC*, 2002, pp. 750–754.
- [14] F. Mondinelli and Z. M. Kovacs-Vajna, "Self localizing sensor network architectures," *IEEE Trans. on Instrumentation and Measurement*, pp. 277–283, April 2004.
- [15] Y. Shang, W. Ruml, Y. Zhang, , and M. P. J. Fromherz, "Localization from mere connectivity," in *Proc. of 4th ACM int. symp. on Mobile ad hoc networking & computing*, 2003, pp. 201–212.
- [16] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Mobile Computing and Networking*, 2000, pp. 32–43.
- [17] W. E. Mantzel, "Distributed alternating localization estimation of camera networks," in *M.S. thesis, Rice University*, 2004.
- [18] M. Werman, S. Banerjee, S. D. Roy, and M. Qiu, "Robot localization using uncalibrated camera invariants," in *Proc. of IEEE CVPR*, 1999.
- [19] A. Arsnio and M. I. Ribeiro, "Absolute localization of mobile robots using natural landmarks," in *Proc. of 5th IEEE Int. Conf. on Electronics, Circuits and Systems (ICECS)*, 1998.
- [20] G. Adorni, G. Destri, M. Mordonini, and F. Zanichelli, "Robot self-localization by means of vision," in *Proc. of Euromicro Workshop on Advanced Mobile Robots (EUROBOT)*, 1996, pp. 160–165.
- [21] Chipcon AS, "SmartRF CC2420DBK demonstration board kit," *User Manual, Revision 1.3*, November 2004. [Online]. Available: [http://www.chipcon.com/files/CC2420DBK\\_User.Manual.L.3.pdf](http://www.chipcon.com/files/CC2420DBK_User.Manual.L.3.pdf)
- [22] Atmel Corporation, "ATmega128(L) 8-bit AVR Microcontroller," *Data Sheet, Revision 2467M-AVR-11/04*, November 2004. [Online]. Available: [http://www.atmel.com/dyn/resources/prod\\_documents/doc2467.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf)

- [23] G. T. Sibley, M. H. Rahimi, and G. S. Sukhatme, "Robomote: A tiny mobile robot platform for large-scale sensor networks," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2002, pp. 96–107.