A Testbed for Coverage Control using Mixed Wireless Sensor Networks

Theofanis P. Lambrou**, Christos G. Panayiotou

KIOS Research Center for Intelligent Systems and Networks, Dept. of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus.

---

*This work is partly supported by the European Structural Funds, the Cyprus Government, the Cyprus Research Promotion Foundation and the European Project CONtrol for COORDination of distributed systems (CON4COORD).

*Corresponding author

Email addresses: faniseng@ucy.ac.cy (Theofanis P. Lambrou), christosp@ucy.ac.cy (Christos G. Panayiotou)

URL: http://www2.ucy.ac.cy/~faniseng/index.html (Theofanis P. Lambrou)
A Testbed for Coverage Control using Mixed Wireless Sensor Networks

Theofanis P. Lambrou**, Christos G. Panayiotou
KIOS Research Center for Intelligent Systems and Networks,
Dept. of Electrical and Computer Engineering,
University of Cyprus, Nicosia, Cyprus.

Abstract

Wireless Sensor Networks (WSNs) is a relatively new technology that has been proposed for several applications including wide area monitoring. Such applications may include stationary or mobile sensor platforms or they may include several stationary and some mobile-robotic sensor nodes that can move in the area in order to achieve certain objectives, e.g., monitor areas that are not adequately covered or assist in the transfer of data to prevent the energy depletion of certain critical nodes. Such networks that consist of both stationary and mobile nodes are referred to as mixed WSNs. This paper presents the development of an experimental testbed for mixed WSNs consisting of stationary and mobile sensor nodes that collaborate to improve the sensing coverage and event detection of the network in a given deployment area. The paper describes the hardware and infrastructure of the testbed as well as a case study for coverage control that was investigated using the testbed. We point out that the developed testbed can be used for the evaluation and validation of different algorithms for coverage control that involve collaboration between stationary and mobile sensors to improve the WSN’s monitoring capabilities. In addition, it can also be used to investigate other objectives as well as other concepts (e.g., network control).

Keywords: Mixed WSN testbed, distributed path-planning, coverage.

1. Introduction

Recent advancements in wireless communication and microelectronics have enabled the development of simple nodes that can communicate wirelessly to form a Wireless Sensor Network (WSN). Such networks have great potential in many applications such as area monitoring, smart buildings, precision agriculture and traffic control. Their unique feature is that they can capture the spatial and temporal dynamics of the environment or process they monitor.

Depending on the applications, a WSN may consists of either stationary sensor nodes or mobile sensors, e.g., sensors installed on robots or aerial vehicles. Another possibility are the mixed WSNs which consist of several stationary nodes together with some mobile nodes that collaborate in order to achieve objectives like improved coverage, improved network connectivity, or prolonged network life. This paper provides an overview of a mixed WSN experimental testbed. The testbed includes several stationary nodes as well as some nodes mounted on mobile robots. A main objective of the testbed is to demonstrate and validate algorithms that allow the mobile nodes to move autonomously in the area in order to achieve their objectives. For the purposes of this paper, the objective is to improve the network’s sensing coverage. Furthermore, the testbed can be used to demonstrate the collaboration between the sensor nodes. Moreover, we emphasize that the testbed can also be used to demonstrate other objectives (e.g., network connectivity) as well as other concepts such as networked control or control over wireless networks, however, these objectives are out of the scope of this paper.

In monitoring applications that involve a large area, coverage holes (areas not sufficiently monitored, where, if an event occurs it may not be detected) are inevitable; either due to an effort to reduce the overall cost, or due to random failures of some nodes. An approach to address the problem of coverage holes is to employ mobile sensor nodes that collaborate with stationary nodes in order to improve the area coverage and/or to detect an event as fast as possible. We are particularly interested in approaches where the mobile nodes (agents) autonomously decide their own path based on “local” information in order to sample areas that have not been sampled by other nodes, stationary or mobile. To achieve this, mobile nodes collaborate by exchanging some information so that they do not sample the same areas. The node collaboration approaches investigated in this paper have been

---

**This work is partly supported by the European Structural Funds, the Cyprus Government, the Cyprus Research Promotion Foundation and the European Project CONtrol for COORDination of distributed systems (CON4COORD).

**Corresponding author

Email addresses: faniseng@ucy.ac.cy (Theofanis P. Lambrou), christosp@ucy.ac.cy (Christos G. Panayiotou)

URL: http://www2.ucy.ac.cy/~faniseng/index.html (Theofanis P. Lambrou)

Preprint submitted to Elsevier Sunday 14th August, 2011
presented in [1, 2]. The underlying principle of these approaches is that the mobile nodes search in a small area around their current location for the points least covered (not adequately monitored by other sensors) and try to move towards those points but, along the way they also navigate such that they avoid areas that are sufficiently covered by other nodes.

The overwhelming majority of approaches for solving the sensing coverage problem has been validated and evaluated based only on simulation models. Experimental work with WSNs and teams of cooperative autonomous robots has been limited, primarily due to cost and practical implementation challenges. Nevertheless, experimental validation is particularly important in distributed WSNs and multi-robot systems research since several factors that can significantly affect the behavior and performance of various approaches may not be accurately modeled. Such factors may include asynchronous communication, delayed or dropped packets, hardware limitations, inaccurate sensing etc. While several experimental testbeds involving either stationary sensors or cooperative autonomous robots have been proposed (e.g. [3]−[12]), to the best of our knowledge, none of these specifically addresses the development of mixed WSNs.

This paper presents the development of an inexpensive testbed that enables validation and evaluation of mixed WSNs and multi-robot cooperation algorithms through both simulation and experimentation. A case study for coverage control is also presented.

The remaining of the paper is organized as follows. Section 2 reviews related work. Section 3 presents an overview of the developed testbed and describes its main components. Section 4 defines the coverage control problem while Section 5 presents a case study that is investigated using the testbed and involves the distributed coverage control problem. This section presents all algorithms executed by the mobiles for their collaborative path planning. Section 6 presents the results obtained from the testbed under different scenarios. Finally, the paper concludes with Section 7 where some limitations of the testbed as well as plans for future developments are presented.

2. Related Work

Experimental work with teams of cooperative mobile robots in the context of WSNs has been limited, primarily due to the cost and the implementation overhead required for the experimental validation of the algorithms developed for WSNs. Some experimental testbeds involving only cooperative autonomous robots have been proposed in the literature.

In [3], the authors develop a testbed with multiple mobile khepera robots [13] that form a team which cooperates to visit multiple target points to collect the rewards associated with them. The objective is to maximize the total reward accumulated over a given time interval. This testbed has also been used to address dynamic network deployment in the context of coverage control using only mobile nodes. Khepera robots, though ideal for such testbeds due to their small size and functionality, they are quite expensive (each wireless Khepera robot costs approximately 3000 euros). Therefore, several attempts have been made to develop small, inexpensive, modular and open-source platforms for developing large-scale mobile robot applications [4, 5, 6].

In [7] the authors develop an experimental testbed to study several issues associated with the convergence of control, communication and computation. The testbed consists of remotely controlled cars and each car is controlled by its own dedicated laptop. An overhead camera provides positioning information and communication between laptops is based on IEEE 802.11. The authors have tested many scenarios on this testbed ranging from traffic control to collision avoidance.

Another mobile robot platform that has been incorporated in a testbed is presented in [8]. In this case study authors develop a mobile sensor node based on the MICA2 board developed by Crossbow Technology. Their main objective is to create a network of mobile nodes that trace the boundaries of a diffusion contamination process.

In [9], an experimental testbed for studying multi-vehicle, networked control has been developed. The authors used ducted fans to control the vehicles and develop real-time feedback control algorithms to stabilize the system while performing cooperative tasks. Several other researchers have also demonstrated experimentally leader following cooperation using two Unmanned Aerial Vehicles (UAVs) [10, 11].

Recently, MIT’s Distributed Robotics Laboratory developed an energy efficient four rotor flying robot for indoor and outdoor navigation [14]. Also researches at Utah State University have developed another low cost UAV platform [15] for aerial surveillance missions.

The testbed presented in this paper involves a mixed WSN consisting of stationary and mobile sensor nodes and its objective is to study collaborative algorithms that improve the sensing coverage and event detection. The developed testbed enables also validation and evaluation of mixed WSNs collaboration algorithms through both simulation and experimentation and, to the best of our knowledge, none of aforementioned testbeds specifically addresses the development of mixed WSNs.

3. Mixed WSN Testbed Overview

This section provides an overview of the implementation of the proposed mixed WSN on real hardware. We describe the various components of the testbed which include the stationary sensors, the mobile sensors, as well as the sensor positioning system which is based on an overhead camera. A schematic overview of the testbed is shown in Fig. 1.
We point out that the testbed has been designed and developed using commercial off-the-shelf components in order to reduce the overall system cost. Furthermore, we emphasize that the coverage control problem is important when the area to be monitored is very large, however, it is not possible to confine such a large network in the space of a research lab, thus, an attempt was made to scale down the detection range $r_d$ and the communication range $r_c$ of each node. For the detection range, we assume that an event source can be detected by a node if it is within a distance $r_d$ from the sensor while in practice we use sources that produce very weak signals. For the communication range, every node drops packets that are received from nodes that are further than $r_c$ as explained in the following subsections.

3.1. Stationary Sensor Nodes

MICAz motes [16] are developed by Crossbow Technology and are equipped with an ATMEL ATmega128L [17] processor operating at 8 MHz. The ATmega128 is a low-power 8-bit Microcontroller with integrated Flash, EEPROM, and SRAM memory, it has several internal timers and supports a variety of serial interfaces, including I2C, SPI, USART (Universal Synchronous Asynchronous Receiver Transmitter) and analog inputs through a built-in 8-channel, 10-bit Analog-to-Digital converter which for instance gives the possibility to connect up to 8 analog sensors such as photocells, thermistors, microphones, accelerometers etc. The MICAz mote features an IEEE 802.15.4/ZigBee compliant RF transceiver (Chipcon CC2420 [18]), operating in the 2.4 GHz band with a 250-kbps data transfer rate. The MICAz runs TinyOS [19] and is compatible with existing sensor boards that are easily mounted onto the mote. A photo of a MICAz sensor mote is presented in Fig. 2.

In the context of our testbed, this stationary node is very flexible since it can easily interface with a variety of sensors. However, it has fairly limited computational capabilities, thus it can only execute fairly simple collaboration algorithms. Furthermore, even though it uses a ZigBee transceiver, the version used is not compatible with other ZigBee implementations\(^1\) (e.g., the Xbee Pro RF transceivers which was used as communication devices for the mobile sensor nodes), thus, a TinyOS procedure (component) has been developed to allow interoperability of the Micaz stationary nodes with Xbee Pro RF transceivers.

The stationary sensor nodes have been programmed to receive position messages from the base station. These messages are sent during the initialization phase of the testbed and contain information about the stationary sensor’s actual position. They can also receive position request messages from mobile sensor nodes. When a stationary sensor receives such a packet a decoding procedure extracts the position of the mobile sensor which has sent the packet. The stationary sensor finds its distance to the mobile and decides to respond to the request only if its distance from the mobile is less than $r_c$. In this case, the stationary node responds to the mobile node and sends a position reply message containing its id and position.

Finally, we consider several static event sources that can be placed at various points in the testbed. Such sources include a lamp, a buzzer, a candle or even a chemical source (e.g., an alcohol emitting source). In this case, all sources emit a signal (light, sound, heat or alcohol fumes) that propagates in the testbed environment and can be detected by the appropriate sensor (photocell, microphone, thermistor or chemical sensor [20]). Each stationary sensor periodically samples the environment and when its reading is above a predefined threshold $\tau_d$, called the detection threshold it sends a detection message to the base station containing its position. Depending on the strength of the emitted signal, the characteristics of the environment and the sensor hardware, one can determine the detection range $r_d$ of each sensor. As already mentioned, for this testbed it is desirable to have sources that emit very weak signals such that not all sensors can detect the presence of the source.

3.2. Mobile Sensor Nodes

The mobile sensor node prototypes are based on microchip’s PIC16F877A microcontroller [21] and on the Mark

---

\(^1\)Communication between RF transceivers from different vendors is not straight-forward because of the different ZigBee stacks hosted on transceivers by their vendors.
The PIC (Peripheral Interface Controller) microcontroller is used for information processing (path planning decisions), for controlling the two servo motors [23] (navigation), for sampling the environment via the various interfaced sensors (microphones, photocells, thermistors, chemical sensors) and for exchanging information with other (mobile or static) sensor nodes or the base station through the interconnected RF transceiver [24]. A photo of a mobile sensor node prototype is presented in Fig. 3 while Fig. 4 presents a simplified schematic of the PIC microcontroller circuitry along with the several interfaces.

![Figure 3: The mobile sensor node prototype developed](image)

![Figure 4: A simplified schematic of the mobile sensor node microcontroller circuitry](image)

The developed mobile sensor node has a small size which makes it suitable for experimental investigation for mixed or mobile sensor networks in a laboratory environment. The dimensions \((L \times W \times H)\) of the prototype are only \(10cm \times 9cm \times 7cm\). Another advantage of the developed mobile node is the low development cost. The complete prototype is built entirely from commercial off-the-shelf components and the cost is of the order of 100 euros. On the other hand, as explained in Section 7, the selection of this robot has some drawbacks.

The PIC16F877A is a cheap, popular, powerful and low power 40-pin 8-bit microcontroller. The 40-pin PDIP package allows rich interconnection and easy assembly. The device offers 14KB \((8K \times 14\text{-bits})\) of Flash program memory, 256 bytes EEPROM, and 368 bytes SRAM data memory which limits the size of the cognitive map that can be maintained by the node. The program memory can be reprogrammed and erased up to 100,000 times which makes the device very good for experimental developments. The PIC16F877A features two 8-bit and one 16-bit internal timers that enable a variety of ways to schedule and manage timing efficiently through hardware. It also supports a variety of serial interfaces like I2C, SPI, USART, digital inputs/outputs (I/O) and analog inputs with 8 built-in channels, 10-bit Analog-to-Digital (A/D) converters. This rich collection of interfaces gives the possibility to connect easily a variety of peripherals such as analog and digital sensors (up to 8 analog sensors), actuators (e.g. motors) as well as other devices like wireless communication modules, external memory modules among many others. The microcontroller is operating at 20 MHz with an external ceramic resonator which enables a 200 ns instruction cycle (5 MIPS).

It is worth mentioning that the PDIP package pinouts are fully compatible with more advanced PICs PDIP package pinouts, like the PIC18F4620 which supports up to 10 MIPS and integrates 64KB program flash and 4KB RAM memory. The internal RAM memory hosted by the PIC micro is vital for robotics applications requiring the implementation of cognitive maps since the size of the memory constrains the size of the map which in turn limits the detailed representation of the environment. Furthermore, the limited computational capabilities of the microcontroller limits the complexity of the path planning algorithms that can be implemented on the node.

An XBee Pro [24] RF transceiver is interconnected to the PIC microcontroller through its integrated USART module. The XBee-PRO module operates within the ISM 2.4 GHz frequency band with a 250 kbps data transfer rate. It is easy-to-use, requires minimal power and provides interoperability with ZigBee devices from other vendors (ZigBee PRO feature set). The RF transceiver uses the ZigBee protocol to communicate with other mobile and static sensor nodes and with the base station (vision positioning system). The PIC microcontroller has been programmed appropriately in order to control the Xbee Pro RF module and to encode or decode the data frames which sends or receives. For instance, using the PIC’s USART Received Interrupt, available data frames are directly received from the RF module with negligible delay. Once the PIC receives such a data frame, a decoding procedure extracts the useful information; such information could be...
the position and orientation of the mobile node or information concerning the cooperation with the neighboring mobile and static nodes.

Specifically, the mobile sensor node receives position messages from the base station (vision positioning system) that contain information about its physical position and orientation. These messages are received either periodically or after the mobile explicitly requests the information from the positioning system by sending a position trigger message containing its id. As mentioned previously, the mobile nodes also send position request messages to the other nodes (mobile or static) of the WSN requesting the position of their neighbors. Once a mobile node receives a position request, it replies with a position reply message containing its id and position only if the node that has sent the request is located within a distance \( r_e \) from the mobile. Mobile nodes also send/receive collaboration messages to/from their neighboring mobile nodes containing information which enables them to achieve their objectives (e.g. area coverage). The actual information exchanged is described in section 5.2.

Like stationary sensors, mobile sensors are sampling the environment with their interfaced sensors (microphone, photocell, thermistor, chemical sensor). When their measurement exceeds a predefined threshold \( r_d \) they also send a detection message containing their position to the base station. In a similar fashion to the stationary sensors, there exists a detection range \( r_d \) in which the mobile sensors can provide sensing coverage and reliably detect events. The detection range \( r_d \) could be adjusted to a value that is equal or smaller than the detection range of stationary sensors.

Finally, the PIC microcontroller of the mobile node uses all the information from its neighbors together with information stored in its cognitive map as well as its own measurements to decide its path. The specific algorithms used for a case study are shown in Section 5.

### 3.3. Vision Based Positioning System

The majority of algorithms proposed for sensor networks and cooperative autonomous agents are based on the assumption that each node (agent) must be aware of its position and orientation as well as the position of the nodes in its neighborhood. In the context of coverage control in large areas, it is anticipated that the mobile nodes will be equipped with a positioning system (e.g. GPS) in order to determine their current position. However, since the test-bed is implemented in a limited size indoor environment, an alternative positioning system is needed to provide the id, position and orientation of the mobile nodes.

Among the options available we decided to use a vision based system since RSSI (Received Signal Strength Indicator) based positioning has some disadvantages such as increased noise vulnerability [25] and in addition, it does not easily provide direction measurements. Vision-based tracking is used in many robotic laboratories to extract agent position, orientation and trajectory. However, there is currently no accepted standard software solution available, so many research groups resort to developing and using their own custom software. In contrast to other systems, we have implemented our vision system using MATLAB which makes the implementation easy (some basic routines are available) however, the positioning speed is rather slow.

For the coverage control problem, measuring and improving the area coverage requires the position of all sensors. For stationary sensors the problem is fairly easy since the coordinates of each sensor can be preprogrammed at the initialization phase of the testbed (either using the information provided by the vision positioning system or simply by (manually) measuring the location of each node). However, for the mobile sensors, there is a need to record the position and orientation at every step of the experiment, thus the information provided by the localization system is required periodically (at every step).

The testbed arena is captured with a monochrome camera mounted on the ceiling (see Fig. 1). The developed vision positioning system consists of the following components: a CCD 1392 \( \times \) 1040 pixels camera (Pulnix TM-1325), a camera lens with appropriate focal length (8mm\(^2\)) for capturing the whole arena area, a frame grabber (NI PCI-I426) which connects the camera to a PC workstation using camera link interface, an image post processing algorithm (developed in MATLAB) and finally a ZigBee transceiver (XBee-PRO) for transmitting the position/orientation information to the sensor nodes.

The camera can capture images of the sensor field either periodically at a rate of 5 frames per second or when triggered (queried by a mobile sensor node via position triggering messages). A typical field image is shown in Fig. 5(a). The testbed arena is 2.20m x 1.40m and covers a size of roughly 1200 by 770 pixels on the image (Fig. 5(a)). The captured image is processed in a MATLAB environment in order to extract the positions of the stationary nodes as well as the position and direction of the mobile nodes. The processing of such a frame takes under 200ms and the results are illustrated in Fig. 5(b). We point out that in order to identify the mobile nodes and their direction, a set of markers are used as shown in Fig. 6. The developed positioning system can provide the position of the sensor nodes with precision of 2cm and the orientation of the mobile sensor nodes with precision of 3° degrees. Once the positioning system identifies the position of each mobile node it sends a unicast position message to each mobile node with its coordinates and orientation. Once each mobile has the position information for every other sensor in its neighborhood it can run the decentralized path planning algorithm in order to determine where to go next. The specifics of the path planning algorithm are described in a following section.

Positioning messages between the base station and the mobile nodes are exchanged in single hop since in coverage control applications it is assumed that the nodes are
3.4. Node Communication Protocols

All nodes communicate using the ZigBee protocol stack, though we have used chips from different manufacturers and thus some adaptations were necessary. ZigBee is a low-cost, low-power, wireless networking standard. A ZigBee Personal Area Network (PAN) is formed by nodes joining to a coordinator or to a previously joined router. Once the coordinator defines the operating channel and PAN ID (preprogrammed initially), it can allow routers and end devices to connect to it. When a node joins a network, it receives a 16-bit network address (associated with the preprogrammed id). Once a router has joined the network, it can allow other nodes to join by connecting to it. ZigBee is built upon the physical layer (PHY) and medium access control (MAC) portion of the data link layer (DLL) defined in IEEE standard 802.15.4 for Low-Rate Wireless Personal Area Network (WPAN). The ZigBee protocol stack supports both beacon and non-beacon enabled networks. In non-beacon-enabled networks, an unslotted CSMA/CA channel access mechanism is used and ZigBee devices typically have their receivers continuously active. In beacon-enabled networks, nodes may sleep between beacons (e.g., beacon intervals may range from 15ms to 250s at 250 kbps) thus lowering their duty cycle and extending their battery life.

The developed testbed has implemented a beacon enabled network with star topology. The module connected to the PC workstation serves as the ZigBee coordinator (ZC), the modules on the mobile robots are set as routers (ZR) and the stationary nodes are set as ZigBee End Devices (ZED). As indicated earlier, the deployment area of the testbed is rather small thus all nodes are generally able to hear transmissions from all other nodes. In order to emulate the limited communication range of each node we use the distance $r_c$ thus a node drops packets that have been received from a node which is at a distance greater than $r_c$. At this point we should also point out that we can extend this packet dropping policy to also emulate non-omnidirectional propagation models (e.g., nodes with directional antennas).

In the ZigBee transceivers there are two types of data transfer transactions. In the first transaction, the data is transferred to the coordinator and in the second transaction the data transfer from the coordinator to the device. When a device needs to transfer data to the coordinator in a beacon enabled network, it listens for the beacon. When the device finds a beacon it synchronizes and transmits the data to the coordinator using slotted CSMA-CA. The coordinator may send an optional acknowledgement frame to complete the transaction. When the coordinator needs to transfer data to a device in a beacon enabled network, it indicates in the network beacon that a data message is pending. The device periodically listens to the network beacon and if a message is pending, transmits a MAC command requesting the data using slotted CSMA-CA. The coordinator acknowledges the successful reception of

Figure 6: Two sample markers coded based on shape compactness. External shape allows the extraction of the mobile node’s ID. Internal shapes (circle and triangle) allows the extraction of robot’s orientation.
the data request from the device by transmitting an optional acknowledgement frame. The requested data frame is then sent by the coordinator using slotted CSMA-CA. The device may send an optional acknowledgement frame. The coordinator will then remove the frame from its list of pending frames in the beacon.

4. Area Coverage Problem

Area coverage is a measure of the effectiveness of the sensor network to monitor the entire field. It measures the percentage of the field area that is monitored by at least one sensor. To compute the area coverage the entire sensor field (arena area) is discretized into an $X \times Y$ grid and thus, the current state of the sensor field is represented by an $X \times Y$ matrix $G_k$, $k = 0, 1, \cdots$ stored in the memory of the base station (PC Workstation in Fig. 1). This $G_k$ matrix corresponds to the system’s “confidence” in detecting an event. If the $(i, j)$th cell falls in the detection range $r_d$ of a static sensor, then the corresponding $G_k(i, j) = 1$, for all $k$ and we are confident that no event will occur in the area of the corresponding grid cell without being detected. If the matrix element has the value 0, then we have no way of knowing if an event has occurred in the corresponding area. This matrix represents the accurate state of the sensor field and is updated as the mobiles move around the field. Thus at every step, we use the following updating rule for every element of matrix $G_k$.

$$G_{k+1}(i, j) = \begin{cases} 0.5 \cdot G_k(i, j) + 0.5, & \text{if } (i, j) \in D_{r_d}(\bar{s}_s) \\ f \cdot G_k(i, j), & \text{otherwise} \end{cases}$$

(1)

where $\bar{s}_s$ are the coordinates of sensor $s$ in the grid $G_k$ and $D_{r_d}(\bar{s}_s)$ is the set of grid cells covered by sensor $s$ with sensing range $r_d$. Also, $0 \leq f \leq 1$ is the “forgetting” factor. Area coverage is defined as

$$C_k = \frac{1}{X \times Y} \sum_{i=1}^{X} \sum_{j=1}^{Y} G_k(i, j)$$

(2)

If $f = 1$ then $C_k$ represents the area coverage over a time interval $[0, k]$ and it is an appropriate quality metric for applications that require coverage of all locations within some time interval.

5. Case Study: Distributed Collaborative Path Planning

In this section we present a case study that was investigated using the testbed described previously. The case study is motivated by the coverage control problem and involves a collaborative path planning algorithm that is used by the mobile sensors in order to achieve their objective which is to search (cover) areas that are not covered by static sensors or any other mobile sensor.

The path planning algorithm is based on Receding-Horizon approach [26]. At each step, the mobile considers a finite set of future positions where the node can move to. For each candidate position, the mobile’s PIC microcontroller evaluates the cost associated with the position and moves to the one that has the minimum overall cost (defined in (4)).

Suppose that during the $k$th step, the mobile node is at position $x(k)$ and is heading to a direction $\theta$. The next candidate positions are the $\nu$ points $y_1, \cdots, y_\nu$ that are uniformly distributed on the arc that is $\rho$ centimeters away from $x(k)$ and are within an angle $\theta - \phi$ and $\theta + \phi$ (see Fig. 7). The mobile node evaluates a cost function $J(y_i)$ for all candidate locations ($y_1, \cdots, y_\nu$) and moves to the location $x(k+1) = y_{i^*}$, where $i^*$ is the index that minimizes $J(y_i)$.

$$J(y_i) = \min_{1 \leq i \leq \nu} \{ J(y_i) \}$$

(3)

In this model, $\theta$ is the direction that the mobile is heading, $\rho$ is the distance that the mobile can travel in one time step, $\phi$ is the maximum angle that the mobile can turn in a single step, and $\nu$ is the number of candidate positions that are being considered for the next step.

The objective function is of the form

$$J(y) = \sum_{j} w_j J_j(y)$$

(4)

where $J_j(\cdot)$ is a specific objective and $w_j$’s are non-negative weights such that $\sum_j w_j = 1$.

In order to improve the area coverage, the mobiles should move towards larger uncovered regions and on their path, they should avoid (to the extend possible) areas that are covered by static sensors or have been covered by other mobile nodes. After an extensive investigation (see also [1, 2]), two specific normalized functions have been selected: $J_l(\cdot)$ which penalizes positions that are away from large coverage holes and $J_s(\cdot)$ which penalizes positions that are close to static or mobile sensors (i.e., areas covered by other nodes). How $J_l(\cdot)$ and $J_s(\cdot)$ are computed is presented next.

5.1. Path Cost Functions

In this section we present the details for the cost functions that we found to give the best performance (using
5.1.1. Neighboring Sensor Cost Function

The objective of this function is to push the mobile away from areas covered by other sensors. The cost function \( J_c(y) \) used involves a repulsion force that pushes the mobile away from its closest neighbor. The form of this function is given by

\[
J_c(y) = \max_{j \in \mathcal{H}_r(m)} \left\{ \exp \left( -\frac{\|y - x_j\|^2}{r_j^2} \right) \right\}
\]

where \( \mathcal{H}_r(m) \) is the set of all nodes in the communication range \( r_c \) of the mobile \( m \). The detection range \( r_c \) quantifies the size of the region around the mobile \( m \) to be repelled by its neighbors.

5.1.2. Target Cost Function

Every mobile node \( m \) maintains an \( X \times Y \) matrix \( P^m_k \) in the PIC’s internal RAM where it keeps the state of the field (i.e., how well every cell of the field is covered). This matrix is updated using information received from neighboring sensor nodes (static or mobiles). Section 5.2 provides more details on the information exchange between mobiles. Ideally \( P^m_k \) should remain \( P^m_k = G_k \) at all times \( k \), since the matrix \( G_k \) represents the accurate global state of the field which is used for the computation of the area coverage \( C_k \). Clearly, in a dynamic environment where several sensors move, fail or packets are dropped, it is impossible to guarantee that \( P^m_k = G_k \) at all times. Also \( P^m_k \) is restricted by the limited data memory available on mobile nodes. Nevertheless, it turns out that even an approximate state is good enough for the mobiles to achieve their objectives.

Given the matrix \( P^m_k \), the mobile will search for the center of the biggest coverage hole within a radius \( r_z \) from its current location. This can be done efficiently using the zoom algorithm [27]. The center of the hole becomes the target destination point \( x_t \). The cost \( J_t(y) \) is a function that pulls the mobile towards its target and is a function of the distance between the mobile and the target position. This function should take a smaller value as the mobile moves towards the target destination thus for the purposes of this paper it is given by

\[
J_t(y) = \frac{\|y - x_t\|}{r_z}
\]

It turns out that \( r_z \) is an important parameter of the algorithm. In our previous work [2, 28] it has been shown that the range \( r_z \) must be fairly small compared to the sensor field area which implies that a “local” search is sometimes better than a “global” search. Also note that a smaller \( r_z \) is advantageous since it implies that less information (i.e., less computation and communication) is needed for the coverage hole estimation.

Note that all cost functions used in eq. (4) can be computed by a mobile node using information in its cognitive map or by obtaining information from its neighbors (nodes that fall in distance less that \( r_c \) from the current position of the mobile). Furthermore, the functions are easy to compute and thus all computations can be done even with the limited resources available at each mobile node.

5.2. Distributed Collaboration between Mobile Nodes

Since every mobile determines its path autonomously, a possible problem arises when two or more mobiles are located close to each other. In this case, it is very likely that the information they will use to estimate the next target position will be the same and as a result they will all estimate target locations that are either the same or they are located very close to each other. Clearly, this is not a good collaboration strategy since there is no benefit if they all converge to the same point. To avoid this problem we utilize a collaboration protocol that enables mobile nodes to exchange some information in order to avoid converging to the same point.

As mentioned earlier, at every step a mobile node \( i \) receives the ids and positions of its neighbors (stationary and mobile nodes) using the position reply messages in order to update its \( P^i_k \) cognitive map. When \( i \) discovers other mobiles in its neighborhood, it sends a unicast collaboration request message to these mobile nodes. Once a mobile node \( j \neq i \) is queried for collaboration, it replies with a collaboration reply message which contains its id \( j \), its current target coordinates \( x^j_t(k) \) and possibly its current cognitive matrix \( P^j_k \) depending on a flag value described next.

Each mobile node \( j \) has a small array in its memory where it tracks the ids of the mobile nodes that were in its communication neighborhood during step \( k - 1 \). If node \( j \) was in communication range with node \( i \) then the corresponding flag \( F_i = 1 \) otherwise \( F_i = 0 \). When a mobile node \( j \) receives a collaboration request message from mobile \( i \), \( j \neq i \), it checks its flag value \( F_i \) and replies with a collaboration reply message that contains its current \( P^j_k \) if \( F_i = 0 \). If \( F_i = 1 \) then the collaboration reply message contains only the mobile’s id \( j \) and its current target coordinates \( x^j_t(k) \) (not \( P^j_k \) since it was sent at a previous step).

Thus as the mobiles stay in a neighborhood range \( r_c \), they exchange only their current positions \( x(k) \) (position reply messages) and target coordinates \( x^j_t(k) \) (collaboration reply messages). This protocol significantly limits the communication overhead between mobiles. More information about reducing the collaboration information that needs to be exchanged between the mobiles without serious loss of the performance can be found in [28].

After mobile node \( i \) has exchanged collaboration messages with its neighboring mobiles it has all the necessary information to execute the collaboration protocol. At first, it merges its cognitive map \( P^i_k \) with the cognitive maps \( P^j_k \), \( j \neq i \) it received from its neighbors, so that it does not explore areas already explored by other mobile nodes. Subse-
The path planning algorithm described earlier determines the next point where the mobile should move to. This point is given by the distance $\rho$ and the direction $\theta$ that it has to turn to. These parameters $(\rho, \theta)$ are further processed by the PIC in order to generate the lower level control signals that will drive the two servos motors of the mobile node in order for the mobile to go to the new position. Servomotors have a very simple electrical interface; they have 3 wires, one for power, one for ground and the other for the pulse train. The data wire receives encoded signal in the form of pulse width modulation (PWM). Two PIC’s digital outputs ports have been programmed to output such a PWM signal. The duty-cycle of each pulse is related to the rotation direction of the servomotor, while the number of pulses sent is proportional to the rotation angle of the servomotor. The pulses sent have 20ms period (implemented using a 20 ms Timer 1 interrupt) and the positive pulse width contained within those 20ms varies from 1ms to 2ms (for 1ms pulse, the servo rotates in the clockwise direction, for 1.5ms pulse, the servo holds still and for 2ms pulse, the servo rotates in counter-clockwise direction).

Finally, in order to make all interfaces work properly, several initialization and calibration procedures involving the servos, the sensors and the RF transceiver of the mobile nodes have been developed (e.g. generate the appropriate pulse that will drive the servos in order to navigate the robot to the next desired location).

6. Obtained Results for the Experimental Case Study

In this section we present a representative scenario of the movement of the mobile sensor nodes to illustrate and validate the behavior of the proposed path planning algorithm.

![The experimental mixed WSN scenario setup consisting of twelve stationary and two mobile sensor nodes](image)

The experimental setup consists of a sparse wireless sensor network with 12 stationary nodes and 2 mobile sensor nodes (see Fig. 9). The monitored region (arena) has dimensions $220\, \text{cm} \times 140\, \text{cm}$. For the purposes of the experiment, the detection radius of all sensors is set to $r_d = 12\, \text{cm}$ and the dynamic search area range is selected as $r_z = 38\, \text{cm}$. The neighborhood range is set to $r_c = r_z + r_d = 50\, \text{cm}$ and it is also fixed in the sensor node firmware such that any node (mobile or static) should drop any packets received from nodes that are located at a distance greater than $r_c$. The constant weights are set to $w_1 = w_2 = 0.5$ and the mobile maneuverability parameters are set to $\rho = 5\, \text{cm}$ and $\phi = 60^\circ$ while for every decision $\nu = 5$ candidate next positions are considered.

Fig. 10(a) shows the paths that the mobile nodes followed in the experimental testbed and Fig. 10(b) shows the paths followed in the simulation environment. The same

---

**Figure 8:** A simplified flowchart of the firmware running on each PIC microcontroller.

**Figure 9:** The experimental mixed WSN scenario setup consisting of twelve stationary and two mobile sensor nodes.
set of parameters was used for both the simulation environment as well as the testbed environment. The two mobile nodes navigate collaboratively through the field, sampling points that are not adequately covered by the stationary sensors. As seen from the paths followed, there is collaboration between the mobile and the stationary sensors in the sense that the mobiles have found two different paths that are least covered by the stationary sensors. Also notice how the two mobiles collaborate and select different paths at the beginning of their journey.

Figure 10: The paths followed by two mobile sensor nodes to collaboratively improve the sensor area coverage of the sparse stationary sensor network deployment. The same set of parameters used in both testbed and simulation. The sensing range of each sensor node is indicated by circles with dotted line. The big (blue) circle indicates is the communication range $r_c$. A video clip of the motion of the two mobile nodes can be found at http://www2.ucy.ac.cy/~faniseng/senetslab/robots.html

Fig. 11 presents a comparison between the area coverage improvement achieved by the mobile nodes in the testbed and simulation environments. The area coverage improvement in the simulation environment is clearly larger compared to the area coverage improvement in the testbed. As indicated in Fig. 10(b), in the simulation environment the mobile nodes avoid almost perfectly the regions covered by stationary nodes. In contrast, in the testbed Fig. 10(a), there exists some overlap between the area covered by static and mobile sensors which are attributed to various unmodeled parameters (dropped packets, uncertain motion of the mobile nodes).

Finally, for a given scenario (fixed positions of the static nodes and fixed initial positions for the mobiles), we run the path planning algorithms twenty times and recorded the average area coverage due to the navigation of the mobiles. In the simulation environment, the paths of the mobiles for all twenty repetitions are identical (there is no randomness involved). In the experimental testbed however, the paths vary significantly as a result of unmodeled phenomena (dropped packets, inconsistencies in the robot motion).

The average area coverage improvement over all repetitions together with the recorded standard deviation are depicted in Fig. 12. These results indicate that the area
coverage improvement obtained in the simulation environment is rather “optimistic”. The discrepancy between the results obtained through simulation and testbed is due to several factors ranging from hardware limitation of the mobile platform (uncertain motion and processing accuracy) to asynchronous communication faults (delayed or dropped packets) which are currently being further investigated. These factors affect the path of the mobiles in a rather random way and thus the variance recorded from the twenty repetitions is significant.

7. Conclusion

Motivated by the need for an experimental platform to validate distributed path planning algorithms in mixed sensor networks, we have developed a new testbed for studying collaborative path planning algorithms for multi-robot systems and distributed area monitoring techniques for mixed WSNs. This paper presents an overview of the testbed developed together with details on the static and mobile sensor nodes used and the overall infrastructure developed. The testbed has been used to validate a coverage control case study where mobile nodes collaboratively sample areas not adequately monitored by the static sensor network. The case study has also indicated some of the limitations of the testbed mainly due to the selected mobile platform which turned out to be a little difficult to accurately control.

In the future we plan to upgrade the hardware and software of our testbed in order to make it faster and more robust. Particular attention will be given to the controlled motion of the mobile platforms. In the current version, it was evident that the mobile nodes could not accurately implement the path suggested by the path planning algorithm. In addition, we plan to use the testbed to come up with more accurate simulation models such that the discrepancies between simulation and experimentation are eliminated. This approach can lead to better simulation models but more importantly to more robust algorithms that can tolerate unexpected events.

References


