To Appear, ACM/Springer Journal of Wireless Networks (WINET), 2009.

A Networked Mobile Sensor Test-bed for Collaborative Multi-target Tracking Applications

Subir Biswas[1], Sonny Gupta[2], Fan Yu, and Tao Wu

Electrical and Computer Engineering Department

Michigan State University

Abstract This paper presents the design, architecture, implementation, and experimental results from a networked mobile sensor test-bed developed for collaborative sensor tracking applications. The test-bed comprises a fleet of networked mobile sensors, an indoor localization system, a control, debugging and management infrastructure, and a tiered wireless ad hoc network for seamless integration of the above three components and the existing wireless infrastructure. First, the software and hardware architectural details of a Swarm Capable Autonomous Vehicle (SCAV) system for our collaborative applications are presented. Second, the details of an indoor self-localization and Kalman filter based navigation system design for the SCAV platform are presented. Third, as an example multi-sensor application, a collaborative multi-target tracking problem and a heuristics-based networked solution are formulated. Finally, the performance of the collaborative tracking framework is evaluated on the laboratory test-bed for characterizing the impacts of localization and navigation errors on the distributed tracking performance. The experimental study also characterizes the tradeoff between the tracking performance and the consumed wireless bandwidth. The experimental results demonstrate a number of counterintuitive results due to various errors in sensor localization and navigation. .

***Keywords** – Mobile Sensors, Sensor Network, Collaborative Applications, Agent Swarming, Multi-target Tracking, Indoor Localization, Sensor Navigation, Kalman Filter*

1. Introduction

Collaborative mobile sensing is gaining increasing popularity as an enabling technology for a wide range of sensing applications including environmental monitoring, security surveillance, and target tracking. Target tracking is a prevalent military requirement for surveillance and reconnaissance scenarios

[1] Corresponding author (sbiswas@egr.msu.edu); This work was partially supported by grants from National Science Foundation (CMMI-0800103), Air Force Research Laboratory, and Michigan State University's High Assurance Computing Initiative.
[2] This author was with the Networked Embedded and Wireless Systems (*NeEWS*) laboratory at MSU during this work.

including those in urban battlefields and inside large buildings and public places such as airports and train stations. In recent years, the technology of unmanned sensing and tracking [1] has experienced a noticeable shift from using few expensive and feature-rich autonomous sensors, to deploying swarming fleets of large number of relatively inexpensive and smaller autonomous sensors that are wirelessly networked. The advantages of the latter approach are as follows. First, a networked fleet provides a more robust solution in which the loss of few inexpensive sensors does not significantly degrade the overall sensing and tracking capabilities. Second, as outlined in [2] and [3], collaborative tracking can be designed to be more efficient than single-sensor tracking, especially in the presence of multiple mobile targets.

The need for such swarming approach to enable collaborative sensing can be further motivated by observing how a large number of organisms in the nature use swarm dynamics for chasing and evading prays and predators. Schooling fishes, for example, swarm away from a predator using group coordination that relies on simple localized communication. A pack of cheetah uses group coordination to collaboratively track and tackle a prey. Similarly, a swarm of bees uses collaborative mechanisms to localize, track, and chase intruders using group swarming.

Successful deployment of a collaborative mobile sensor system to cater to such applications will require the following critical system components. First, a sensor platform with robust sensing, mobility, and networking capabilities needs to be developed. Second, self-localization abilities will be needed for target tracking. Third, smooth sensor navigation mechanisms will have to be developed in the presence of localization errors and inaccuracies at high platform speeds. Fourth, self-healing mobile ad hoc wireless network protocols will be needed for inter-sensor data dissemination. Finally, multi-sensor collaborative applications will have to be developed by leveraging the underlying localization, navigation, networking and sensing services. Each of these four areas has seen recent research activities [4-11] and progress towards application-driven system integration. This paper contributes towards these integration trends by developing a system that comprises all four subsystems as well as their functional integration in the context of a collaborative multi-target tracking application in indoor settings.

The contributions of the paper are as follows. First, our experience on design and development of the above four subsystems are reported in the context of a laboratory prototype system. Second, a distributed multi-target tracking framework using the above subsystems have been developed and extensively characterized in the presence of various measurement errors. Third, specific algorithmic insights are provided as to how adapting tracking can be implemented to cope with non-ideal localization and networking conditions. It is demonstrated that such non-ideal conditions can cause theoretically sound tracking algorithms to generate unexpected results, and therefore special experimental considerations will be necessary while implementing such mobile sensor systems under non-ideal conditions.

The rest of the paper is organized as follows. Section 2 describes the related work on collaborative multi-target tracking applications. The system architecture is presented in Section 3. The hardware and software system components of the mobile sensor platform are presented in Section 4. Section 5 details an indoor localization framework used by the mobile sensors for self localization, and a filtered navigation framework is presented in Section 6. Design of a collaborative tracking algorithm and its implementation and performance using the proposed test-bed are presented in Section 7. Finally, the paper is concluded in Section 8 with a summary and a list of ongoing work on this topic.

2. Related Work

Indoor sensor localization [20, 23] for moving devices can be performed in an active or in a passive mode. In the active approach, each mobile device has an active transmitter, which periodically broadcasts beacons to a number of external fixed units. The estimated distance of the mobile device from each such fixed unit is fed back wirelessly to the mobile device, which then computes its own coordinate using the locations of the fixed units and individual distances. In the passive approach, it is the external fixed units that broadcast the beacons and the mobile device computes its own coordinate based on the estimates of the individual distances. The impacts of the density and number of external units have been analyzed in [22].

In this paper we use the passive approach with a Kalman Filter [14] based estimation for compensating the localization errors caused due to moving sensors. This approach is similar to what has been taken in [23] (to our knowledge, that is the only other published indoor localization and sensor navigation work) except that, unlike in [23] which uses an Extended Kalman Filter (EKF), we have used a regular Kalman Filter with an assumption of linearity for the sensor dynamics. Experimentally it is demonstrated that the linear KF with PVA (position, velocity, acceleration) state abstraction was sufficient for accurate estimations in the presence of localization errors.

Target tracking using sensor networks has been extensively explored [23, 25, 26, 27, 30] in the recent literature. Both single- [29, 30] and multi-target [25, 28, 31, 32] tracking applications were investigated. The mechanism in [25] explores sensor network based multi-target tracking solutions and their performance in the presence of non-ideal conditions due to packet loss, communication delay variation, and false positive sensor detection. These and the algorithms presented in [28, 31, 32] are designed for systems operating in outdoor fields that are larger than the vehicle dimensions by several orders of magnitude. The field dimensions are also few order of magnitudes larger compared to the GPS based location estimation errors, which could be up to few meters [26]. In contrast, in this paper we investigate the impacts of location and navigation errors on multi-target tracking performance in a very tight indoor environment (i.e. an area of few square meters) that is not as large compared to the vehicle dimensions (i.e. approximately 15cm for the mobile sensor used in our test-bed) and the indoor localization errors, which have an upper bound of 10cm. Such tighter field dimensions (e.g. an area of 3m x 3m has been used for our experimentations) give rise to unique target tracking issues which are dealt with in this paper.

3. Mobile Sensor Test-bed and Developed Services

The developed mobile sensor test-bed and its integration with the existing wireless network infrastructure are shown in Figure 1. The test-bed contains the following main components: 1) a fleet of indoor mobile sensors, 2) an indoor localization system for enabling sensor self-localization in GPS-denied environments, 3) a control, debugging and management infrastructure, and 4) a tiered wireless ad hoc network for seamless integration of the above three components and the existing research infrastructure in various wireless network laboratories in Michigan State University. As for the networked mobile sensors, we have developed a Swarm Capable Autonomous Vehicle (*SCAV*) platform as described in Section 4.

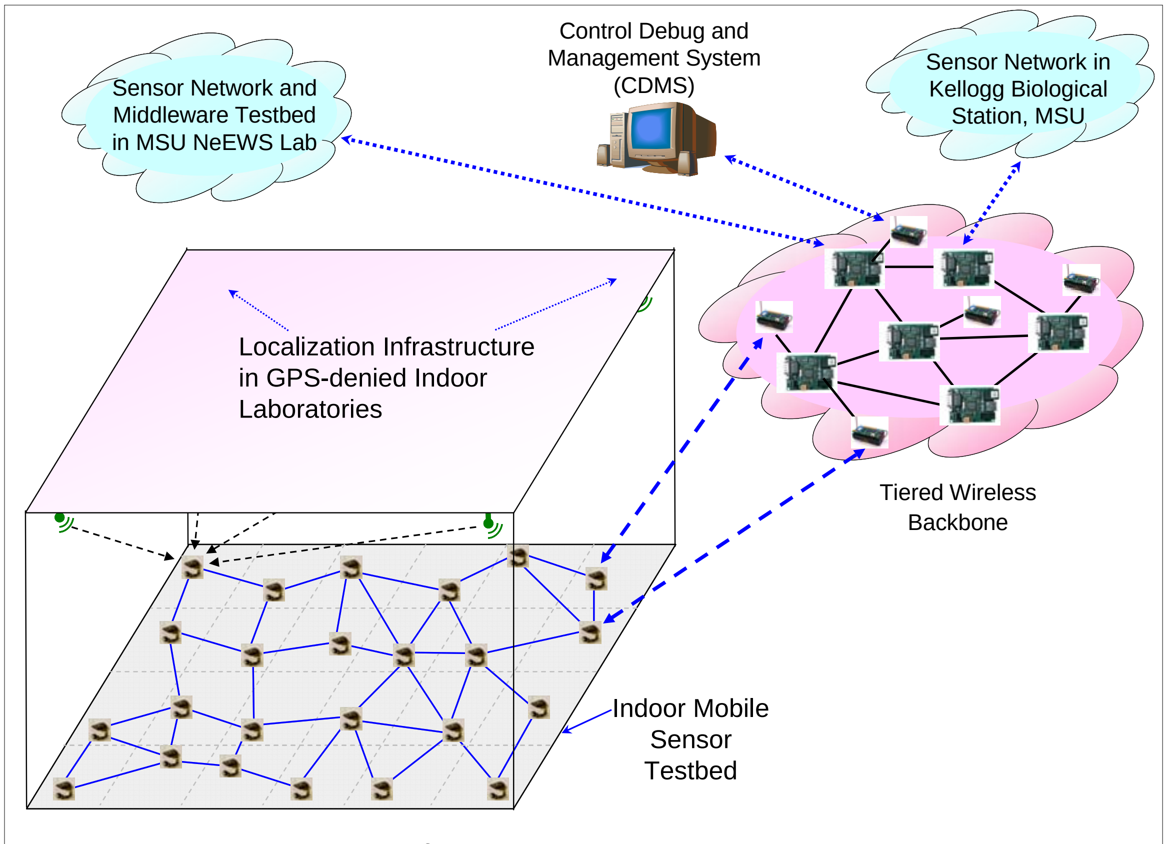

Sensor Network and Middleware Testbed in MSU NeEWS Lab
Control Debug and Management System (CDMS)
Sensor Network in Kellogg Biological Station, MSU
Localization Infrastructure in GPS-denied Indoor Laboratories
Tiered Wireless Backbone
Indoor Mobile Sensor Testbed

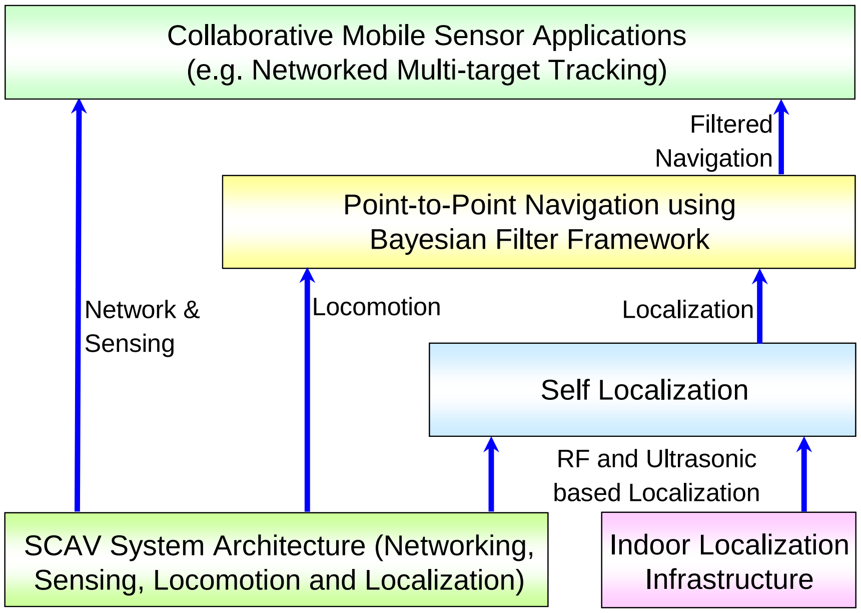
Fig. 1: Components of the mobile sensor testbed and their integration

Developed mobile sensor services and their dependencies are depicted in Figure 2. The SCAV mobile sensor system comprises the following four services: a) wheel or track based locomotion, b) multimodal onboard sensing, c) both ad hoc and access point oriented wireless networking, and d) infrastructure assisted indoor localization using Radio Frequency (RF) and Ultra Sound (US) signals.

Self localization for the mobile sensor nodes within an in-building coordinate system have been achieved using a Time Difference of Arrival (TDOA) based ranging technique [12], coupled with collaborative multi-lateration [13]. A passive localization scheme has been adopted so that a SCAV can self-localize by using the received RF and US signals from preinstalled localization infrastructure.

A point-to-point navigation service is developed using the localization and the SCAV locomotion services. A framework of Bayesian filtering has been incorporated to tackle the intrinsic localization errors contributed by noises such as ambient RF and US interference, vehicle vibration, and inaccuracies introduced by the localization hardware components. Additional inaccuracies are caused when the localization intervals are too large to capture accurate locations of the moving sensors. A Position-

Velocity-Acceleration (PVA) linear Kalman Filter [14] has been used for smooth navigation in the presence of those localization errors.

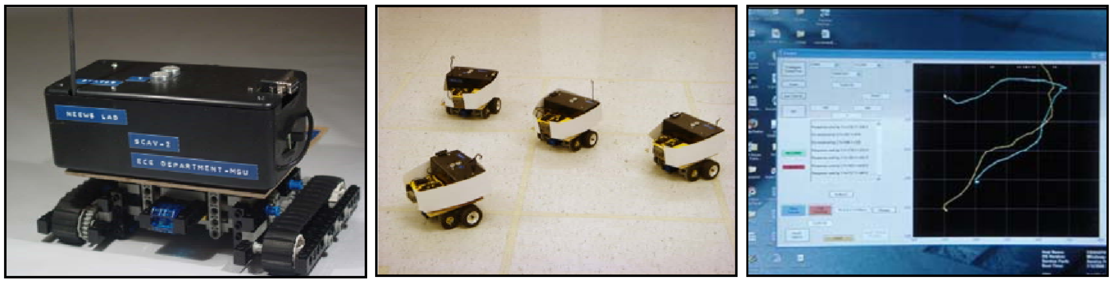

Collaborative Mobile Sensor Applications
(e.g. Networked Multi-target Tracking)
Filtered Navigation
Point-to-Point Navigation using Bayesian Filter Framework
Network & Sensing
Locomotion
Localization
Self Localization
RF and Ultrasonic based Localization
SCAV System Architecture (Networking, Sensing, Locomotion and Localization)
Indoor Localization Infrastructure
Fig. 2: System components of the collaborative tracking using SCAVs

Finally, a set of collaborative sensor applications have been developed using the ad hoc networking and sensing services from the SCAV architecture, and the smooth navigation service is implemented using the Kalman filter. As a representative application, a distributed multi-target tracking system and its variations involving different numbers of tracked and tracking sensors have been presented in this paper. Throughout the rest of the paper, these mobile sensor services components will be further elaborated.

4. Swarm Capable Autonomous Vehicle (SCAV) Architecture

The picture of a mobile SCAV sensor unit is shown in Figure 3. The middle picture shows the front-view of a fleet of four mobile sensors, and the left-most picture shows the back-view of a single sensor without its collision bumper. The following features are currently available [15] in our mobile SCAV fleet. 1) Self-localization with centimeter level resolution, 2) Bayesian filter based point-to-point navigation, 3) Multi-modal onboard sensing, 4) Navigational collision avoidance using infra-red obstruction sensing, 5) Both Ad Hoc and infrastructure based radio communication using 900 MHz RF, and 6) Control, Debugging and Management System (CDMS).

The CDMS software system is designed for managing the SCAV infrastructure locally or through a network as shown in Figure 1. The CDMS system runs on a Windows system and supports the following essential operations for the *SCAV* test-bed: 1) compiled image and software download using wireless links, 2) *SCAV* system debugging by allowing remote print console, 3) command line and graphical interfaces for remote command dispatch to targeted mobile sensors, 4) supporting a variety of sensor and mobile ad hoc network protocols for seamless connectivity with the SCAV test-bed and the backbone mesh network, 5) real-time location tracking for individual sensors and their experimental post-processing, 6) command driven SCAV navigation, 7) sensor specific mobility planning through a graphical user interface, and 8) data upload and telemetry from the *SCAVs* using wireless links. Real-time location tracking screenshot from an example 2-SCAV leader-following experiment is shown in Figure 3.



Physical appearance (6”x6”x4”) Swarm of four sensors CDMS Screenshot

Fig. 3: SCAV mobile sensor platform and the Command Debug & Management System (CDMS)

The internal subsystem level hardware and software components of the SCAV architecture are shown in Figs. 4. A SCAV system contains the following subsystem modules. 1) An Atmel ATmega Localization, Navigation and Tracking (LNT) processor that forms the central processing platform, 2) A Hitachi H 8/3292 based locomotion controller, 3) An Ultrasonic/RF localization card (CRICKET [16], from Crossbow Technologies [17]), 4) A sensor card containing onboard sensors including acceleration, temperature, magnetic field, light, and sound, and 5) An infrared proximity sensor for navigational collision avoidance, and 6) A 900MHz radio interface for network implementation.

Packet routing protocols are implemented over this 900MHz radio interface for ad hoc network operations among the SCAV units, the wireless backbone, and the CDMS modules as shown in Figure 4. The raw data-rate of each link is approximately 20Kbps, and CSMA and AODV are the MAC and the routing protocols that run on these interfaces. Both point-to-point and all-to-all packet communications are supported using AODV and flooding respectively. No link layer protocol is used for hop-level error recovery, and it is left up to the application layer (e.g. the multi-target tracking as presented in Section 7) to deal with packet losses.

As for software, the Hitachi locomotion controller runs an embedded micro-kernel BrickOS [18], a public domain operating system, used for implementing SCAV's locomotion control using geared motors and navigational touch and infrared sensors. A series of motor control and debugging APIs have been implemented between the LNT processor and the locomotion processor using a custom built serial interface. The SCAV mobile sensors are capable of moving with a maximum speed of 10cm per second.

Self-localization in a SCAV is performed within the LNT processor by computing coordinates using the sensor unit's distance from a number of pre-installed localization beacons in known locations. This distance information is supplied by the CRICKET RF/US location sensor cards [16]. Note that the beacons are asynchronously broadcast by four independent CRICKETS at an interval of 0.8 seconds. No clock synchronization was necessary between the ceiling-mounted CRICKET units. In LNT processor, the embedded micro-kernel TinyOS [19] is run for supporting coordinate computation, Kalman Filter based navigation and collaborative tracking services as shown in Figure 2.

The LNT processor is also used for implementing a number of remote commands to and from a Command, Debug and Management System (CDMS) as shown in Figure 4. The CDMS is implemented using a 900MHz RF interface and an Atmel ATmega processor, which is connected to a Windows PC console over RS-232 serial interface. Using this arrangement, commands can be sent to specific SCAVs from a user interface in the Windows PC. Similarly, measurement and system performance parameters from specific SCAVs can be uploaded through the CDMS using the same Windows based user interface.

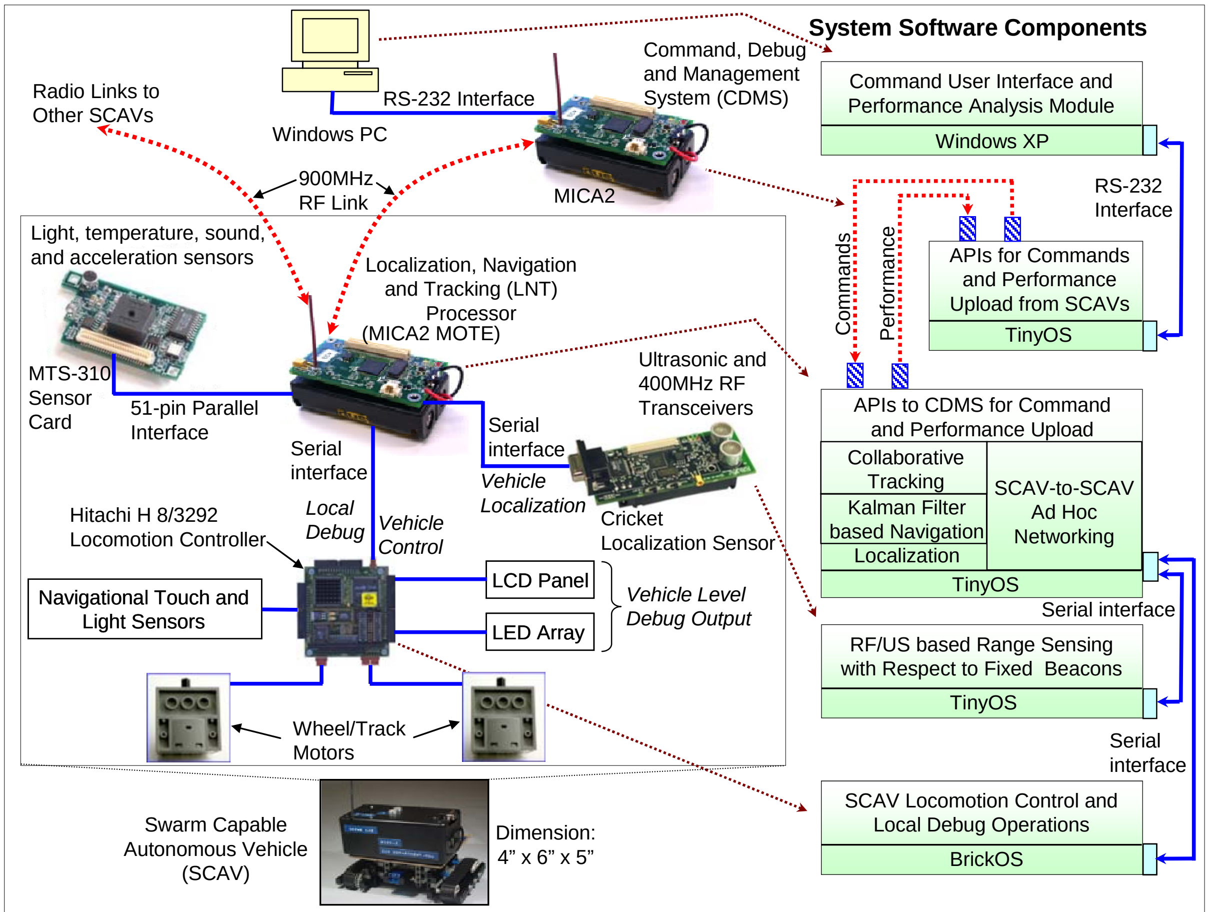

System Software Components
Command User Interface and Performance Analysis Module
Windows XP
RS-232 Interface
Commands
Performance
APIs for Commands and Performance Upload from SCAVs
TinyOS
APIs to CDMS for Command and Performance Upload
Collaborative Tracking
Kalman Filter based Navigation
Localization
SCAV-to-SCAV Ad Hoc Networking
TinyOS
Serial interface
RF/US based Range Sensing with Respect to Fixed Beacons
TinyOS
Serial interface
SCAV Locomotion Control and Local Debug Operations
BrickOS
Radio Links to Other SCAVs
Windows PC
RS-232 Interface
Command, Debug and Management System (CDMS)
MICA2
900MHz RF Link
Light, temperature, sound, and acceleration sensors
MTS-310 Sensor Card
51-pin Parallel Interface
Localization, Navigation and Tracking (LNT) Processor (MICA2 MOTE)
Ultrasonic and 400MHz RF Transceivers
Serial interface
Serial interface
Vehicle Localization
Cricket Localization Sensor
Hitachi H 8/3292 Locomotion Controller
Local Debug
Vehicle Control
Navigational Touch and Light Sensors
LCD Panel
LED Array
Vehicle Level Debug Output
Wheel/Track Motors
Swarm Capable Autonomous Vehicle (SCAV)
Dimension: 4” x 6” x 5”

Fig. 4: System components and software modules in SCAV platform

The primary remote commands invoked from the CDMS to SCAVs’ LNT processors are: 1) SCAN: for scanning the location of specific or all SCAVs present in the experimental field, 2) CHECK_BATT: for checking the remaining battery on specific SCAVs, 3) GO: for sending a SCAV from its current location to a new destination with a specified speed, and 4) STOP: to force a SCAV to stop navigation. And the primary remote commands invoked from a SCAV’s LNT processor to CDMS are: 5) RPRINT: for remote printing from a SCAV to the console PC’s screen, and 6) PERF_UPLOAD: used for remote upload of collected performance data from SCAV sensors to the remote PC’s hard drive. We have constructed a test-bed of three identical CDMS units and five SCAV units which are deployed in a *3m x 3m* experimental field for conducting experiments involving localization, navigation and networked multi-target tracking. Multiple CDMS units are used for printing debugging information from different SCAVs to different CDMS screens.

5. Mobile Sensor Self Localization

The components of the localization system are depicted in Figure 5. When a self-localization is needed, a SCAV computes its absolute distances from a number of static localization beacons pre-installed at known coordinates. Distance from a beacon is computed by measuring the time difference of arrival (TDOA) between an ultrasonic and a 433MHz RF signal simultaneously transmitted by the beacon. Once distances to a sufficiently large number of location beacons are computed, the SCAV computes its own coordinates using the distance values and the known coordinates of the relevant beacons. The MIT CRICKET localization hardware [16] has been used as the static beacons shown in Figure 5, and the localization sensor in the SCAV, as shown in Figure 4.

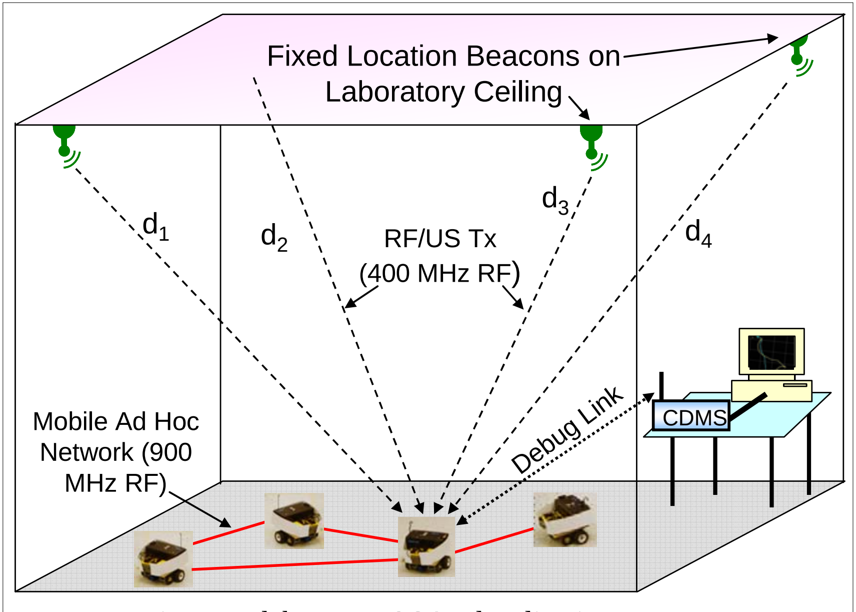

Fixed Location Beacons on Laboratory Ceiling
d_1
d_2
d_3
d_4
RF/US Tx (400 MHz RF)
Mobile Ad Hoc Network (900 MHz RF)
Debug Link
CDMS

Fig. 5: In-laboratory SCAV localization system

5.1 Distance Estimation

The following method is used for distance computation. If V_{RF} and V_{US} are the known speed of RF and ultrasonic signals through air ($V_{RF} >> V_{US}$), and T represents the time difference of arrival at the SCAV, then the distance between the SCAV and the beacon can be written as [20]:

$$d = T/(1/V_{RF} - 1/V_{US}).$$

If the distance to the i^{th} location beacon, installed at coordinate (x_i, y_i, z_i) is computed as d_i, then the measurement error ε_i can be written as: $\varepsilon_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} - d_i$, where (*x, y, z*) is the actual location of the SCAV. With distance measurement from *N* beacons, the localization problem becomes equivalent to estimating the actual coordinate (*x, y, z*), so that the cumulative squared error quantity $\sum_{i=1}^{N} \varepsilon_i^2$ is minimized. This constrained optimization problem is known as Minimum Mean Square Estimate (MSME) [21] in the literature.

5.2 Onboard Coordinate Computation at Real-time

The simplest way to solve this system of *N* 2nd order equations $(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 = d_i^2; i = 1,2,\ldots,N$ is to convert it to *N-1* linear equations by pair-wise subtractions. For example, the quadratic equations for *i* = *i* and *1* can be combined [3] into the linear equation $2x(x_i - x_1) + 2y(y_i - y_1) = (d_1^2 - d_i^2) + (x_i^2 - x_1^2) + (y_i^2 - y_1^2)$ for $i = 2,\ldots,N$. This provides a linear system of equations:

$$\begin{bmatrix} 2(x_2 - x_1) & 2(y_2 - y_1) \\ 2(x_3 - x_1) & 2(y_3 - y_1) \\ \ldots\ldots & \ldots\ldots \\ 2(x_N - x_1) & 2(y_N - y_1) \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (d_1^2 - d_2^2) + (x_2^2 - x_1^2) + (y_2^2 - y_1^2) \\ (d_1^2 - d_3^2) + (x_3^2 - x_1^2) + (y_3^2 - y_1^2) \\ \ldots\ldots \\ (d_1^2 - d_N^2) + (x_N^2 - x_1^2) + (y_N^2 - y_1^2) \end{bmatrix}$$

If this *(N-1)th* order system is represented as $A \times U = V$, then the solution for MMSE can be computed by the following matrix operation: $U = \left[(A^T.A)^{-1} \times A^T\right] \times V$, as long as $N \geq 3$ and the matrix *A* is non-singular.

We have implemented this MSME based self-localization mechanism on the onboard LNT processor as shown in Figure 4. A *3m x 3m* SCAV navigation test-bed has been created within a laboratory, at the ceiling of which four static beacons have been installed. The beacons have been positioned within a *0.5m x 0.5m* area right above the center region of the *3m x 3m* test-bed.

[3] If all location beacons are placed at the same height, the z-coordinate cancels out from the equations.

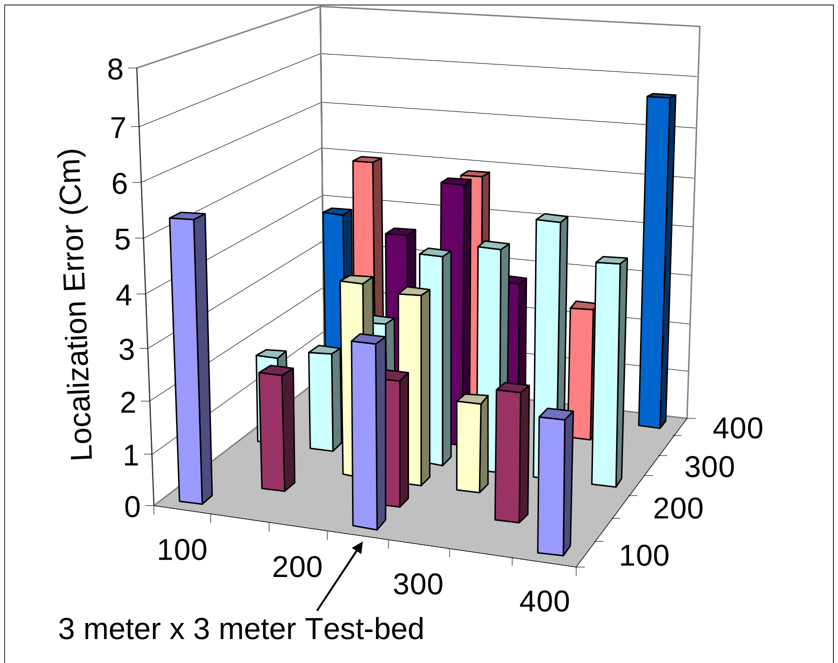


Fig. 6: Performance of sensor self-localization

Performance of SCAV localization is presented in Figure 6, which shows the absolute error represented as the difference between a SCAV's actual coordinate and its location estimated by the localization mechanism described above. The primary source of this error is from faulty distance computation caused by the inaccuracies of the CRICKET hardware used as the beacons as well as listeners within the SCAV units. Other sources of errors were identified as various ambient RF and US interferences, and high frequency mechanical vibration contributed by the SCAV motors.

The measurements were taken by statically positioning a SCAV at different locations in the test-bed as shown in Figure 5. For each location, 50 different location computations were used to report an average error. From Figure 6, observe that the maximum localization error is about 7cm, which is only about 1.7% of the diagonal of the entire test-bed area. Comparing with the localization errors reported in [21] and [22], we consider the implemented indoor location service to be sufficiently accurate for effective navigation and target tracking. Also it can be observed that the errors are generally larger away from the center of the test-bed. This is because the beacons are placed in an area right above the center region, and as the SCAV is moved away from this region the beacon distances increase. The distance computation errors were found to be positively correlated with the distance itself [20]. This explains higher errors near the periphery of the navigation test-bed. Also, the location estimation performance was found to be unbiased.

6. Filtered Sensor Navigation

The most primitive component of the point-to-point SCAV navigation service is to be able to navigate a mobile sensor from its current location to a new destination location in the minimum possible time. This function has been implemented as a remote command to the SCAVs LNT processor. Any remote entity such as the CDMS or another SCAV can issue this command to initiate navigation for a SCAV to a specific destination. A SCAV can also self-invoke the command locally. In both cases, once the command is sent to its LNT processor, a SCAV is required to autonomously navigate to the specified destination.

The navigation latency depends on the accuracy of the underlying localization service as described in Section 4. Although the localization errors in the cases of static SCAV placements, as reported in Figure 6, are very small, the error numbers were found to be significantly larger with moving sensors. While ultrasound interferences due to mechanical vibration and motor noises partially responsible for this inaccuracy, the primary reason for such high errors is as follows. Due to the asynchronous nature of the RF and US transmissions from multiple localization beacons (four in our installation), the simultaneity condition of distance measurement does not hold [23]. In other words, the distances computed to different localization beacons correspond to different SCAV locations, whereas the MSME mechanism for coordinate computation implicitly assumes that all d_i values are measured from the same SCAV location. This contributes to heavy coordinate errors that can translate into slow point-to-point navigation due to frequent erroneous heading changes.

6.1 Error Reduction using Kalman Filter

Since the locomotion dynamics of a SCAV can be deterministically modeled, and is known a priori, it is possible to design a Bayesian filter [24] that can partially compensate for the localization errors by predicting the sensor’s location at successive localization instances according to its underlying locomotion dynamics. The prediction value can then be incrementally corrected using the location computed by the SCAV’s LNT processor. Considering the linearity of SCAV’s movement dynamics, we implement a linear Kalman Filter [14], which implements a special case of the Bayesian filter by assuming that the system and measurement noises are strictly Gaussian. Note than unlike in the indoor filtered navigation approach in [23], which uses an Extended Kalman Filter (EKF), we use a regular Kalman Filter with the assumption of linearity for the SCAV's movement dynamics.

Although an EKF can provide better filtering performance for a wheeled non-holonomic system such as the SCAV, the processing requirement for EKF can be prohibitive for Atmel Atmega 128L (4 MHz) which is used as SCAV's LNT processor platform. This processing constraint is particularly relevant when considering the other system operations such as network protocol processing, sensor navigation, and multi-target tracking as described in later parts of the paper. Considering this resource constraint, which is typical for an embedded sensing platform such as the SCAV, we have chosen a linear Kalman Filter instead of an Extended Kalman Filter for reducing the localization errors.

We use two linear one dimensional PVA (position, velocity, acceleration) Kalman Filters (KFs), one for the *x* dimension with state vectors $\begin{bmatrix} x & \dot{x} & \ddot{x} \end{bmatrix}^T$, and the other for *y* dimension with state vector $\begin{bmatrix} y & \dot{y} & \ddot{y} \end{bmatrix}^T$. These two filters work identically, but independently, and here we present the Kalman equations only for the *x* dimension. Exact same equations apply for the y dimension as well. We have explored both the second order (PV) and the third order (PVA) filters, and based on the experimental accuracies, the third order implementation is chosen. Careful investigations revealed that frequent stop-and-go, and direction changes, and the subsequent changes of SCAV velocity could not be accommodated well by a PV filter in the 3m x 3m experimental environment. A third order filter provided better error estimations. Also, we have tried two approaches, one with coupled *x* and *y* states, and the other with decoupled states as specified above. Experimentally we found that the estimation accuracies from both the models were pretty comparable. The decoupled approach was adopted for its relative computational lightness.

Assuming a constant $\ddot{x}$, and ignoring $\dddot{x}$ and higher order derivatives, the discrete state equations for SCAV locomotion at the k^{th} iteration can be written as:

$$\begin{aligned}
x_k &= x_{k-1} + \dot{x}_{k-1} \cdot \Delta t + \tfrac{1}{2} \cdot \ddot{x}_{k-1} \cdot \Delta t^2 + w_{k-1}^{x} \\
\dot{x}_k &= \dot{x}_{k-1} + \ddot{x}_{k-1} \cdot \Delta t + w_{k-1}^{\dot{x}} \\
\ddot{x}_k &= \ddot{x}_{k-1} + w_{k-1}^{\ddot{x}},
\end{aligned}$$

where Δt is the discrete time step, and *w* represents the system noise. Assuming time invariant system noise, the above system of equation can be written more concisely in vector format as:

$$S_k = A \times S_{k-1} + W \tag{1}$$

The measured value of the *x* through localization at the k^{th} step can be written as: $x_k^m = x_k + v_k^x$, where *v* is the measurement error and x_k is the actual value of *x*. Assuming time invariant measurement error, it can be written as:

$$Z_k = H \times S_k + V \,. \tag{2}$$

The constant matrices A and H in Equations 1 and 2 are: $\begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$. Considering this system and measurement formulation, the Kalman *prediction* equations can be organized as:

$$S_k^{(-)} = A \times S_{k-1}^{(+)} \text{ and } P_k^{(-)} = A \times P_{k-1}^{(+)} \times A^T + Q \,, \tag{3}$$

where $^{(-)}$ indicates a prediction and $^{(+)}$ indicates a Kalman *correction*. The matrix P_k represents the system state covariance, and Q represents the covariance matrix of the system noise parameters. The Kalman correction equations are:

$$\begin{aligned} K_k &= P_k^{(-)} \cdot H^T \cdot (H \cdot P_k^{(-)} \cdot H^T + R)^{-1} \\ S_k^{(+)} &= S_k^{(-)} + K_k \cdot (Z_k - H \cdot S_k^{(-)}) \\ P_k^{(+)} &= P_k^{(-)} - K_k \cdot H \cdot P_k^{(-)} \end{aligned} \tag{4}$$

The matrix K_k is the Kalman *gain*, and R represents the covariance matrix of the measurement error parameters. The LNT processor in SCAV iteratively solves the system of Equations 3 and 4 with Δt set to be 500ms. The covariance matrices Q and R were experimentally tuned at diagonal element values of *0.000005* and *0.1* respectively for the best filter performance.

An outlier rejection mechanism was also embedded within the filter so that whenever a measured coordinate in Z_k is found to be too far from the last predicated value in $S_k^{(-)}$, the Kalman gain is forcibly reduced to deemphasize this new measurement, and thus reducing the effects of overly erroneous location measurements. When the absolute difference between Z_k and $S_k^{(-)}$ is found to be more than 50% of the estimated value $S_k^{(-)}$, the measured Z_k is considered to be an outlier. The implemented gain reduction was 90%, which is experimentally optimized. In other words, the Kalman gain is reduced to one tenth of the original gain in the events of outliers.

6.2 Sensor Navigation

Both the filters for the x and y dimensions are clocked synchronously at 500ms intervals and the corrected x and y coordinates from $S_k^{(+)}$ are used for navigating a SCAV. In case of a missed beacon, the most recently received beacon is used for filter processing at the clocking time points. Such beacon losses were found to be present and they do contribute to the overall navigation error reported in Section 6.3. Figure 7 depicts an integrated localization and filtered navigation approach that is implemented in the LNT processor for point-to-point SCAV navigation. Note that after the Kalman *correction* step is performed, the desired sensor heading is computed by using the corrected coordinates $S_k^{(+)}$ and $S_{k-1}^{(+)}$ from the current and the previous steps. At this stage, the SCAV makes slow turns if it determines that a heading change is necessary to realign its movement towards the current destination.

A constant speed differential between the left motor and the right motor is used for implementing the turns. When a turn needs to be executed, the Atmega LNT processor instructs the Hitachi locomotion controller about the speeds of the individual motors and the duration for which the speed differential should be maintained. These three parameters together determine the amount and the speed of the turn. This integrated localization, filtering, and navigation decisions run continuously till the SCAV reaches its destination specified in the GO command.

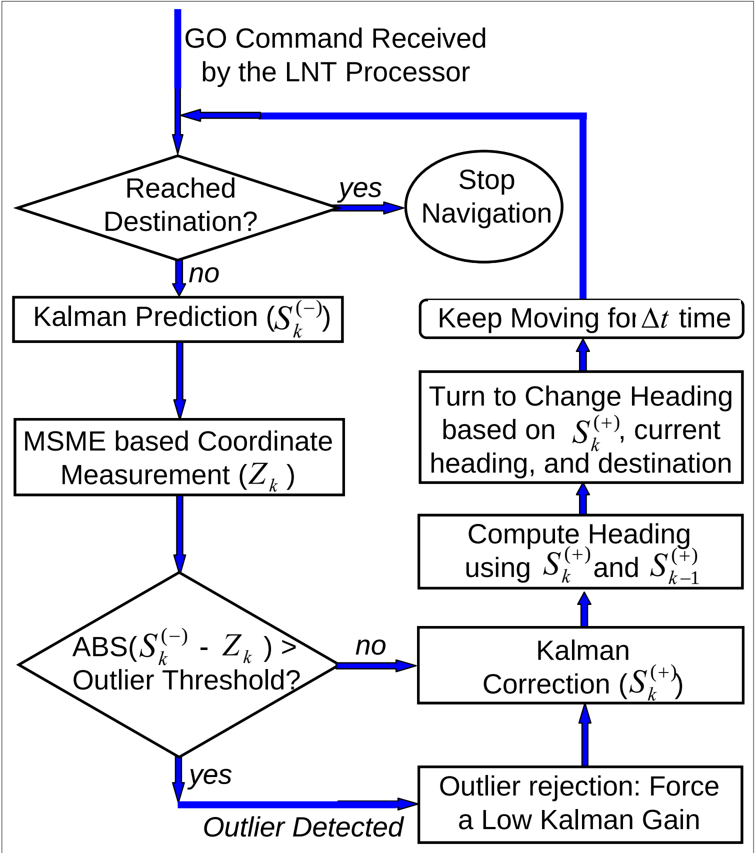

GO Command Received by the LNT Processor
Reached Destination?
yes
Stop Navigation
no
Kalman Prediction ($S_k^{(-)}$)
Keep Moving for Δt time
Turn to Change Heading based on $S_k^{(+)}$, current heading, and destination
MSME based Coordinate Measurement (Z_k)
Compute Heading using $S_k^{(+)}$ and $S_{k-1}^{(+)}$
ABS($S_k^{(-)}$ - Z_k) > Outlier Threshold?
no
Kalman Correction ($S_k^{(+)}$)
yes
Outlier rejection: Force a Low Kalman Gain
Outlier Detected

Fig. 7: Integrated localization and navigation algorithm

It is possible to implement the Kalman process and the navigation process as two asynchronous execution threads, so that the filter produces corrected coordinates driven by a separate clock, and the navigation and other possible application threads can access to those coordinates asynchronously. That way, the clock frequencies of those processes can be set independently.

We have also experimented with navigation implemented in a *stop-and-go* manner so that between the moves, a SCAV waits for some time to compute coordinates using static localization. Since the static localization produces more accurate coordinates, the *stop-and-go* policy is able to offer more accurate navigation than the continuous mode of operation presented in Figure 7. Due to the stops however, the *stop-and-go* was observed to be consistently slower than the continuous mode.

Note that the integrated navigation mechanism in Figure 7 implicitly uses *certainty equivalence principle* to separate navigation from location estimation in that the navigation module assumes the SCAV is at the mean position estimate. These mean position values, in turn, are used for periodic heading correction during navigation.

6.3 Experimental Sensor Navigation Performance

Performance of the SCAV navigation system is reported in Figures 8 and 9. Figure 8:a shows an experimental scenario in which first a GO command was sent to a SCAV to move it from location (*350,350*) to (*150,150*), and then another GO was given from (*150,150*) to (*150,350*). Note that the *300cm x 300cm* area in the graph represents the SCAV test-bed used for the following experiments. After the navigation was completed, the PERF_UPLOA command was remotely invoked from the CDMS (see Figures 1-4) for wirelessly uploading the performance numbers to a PC for data post processing.

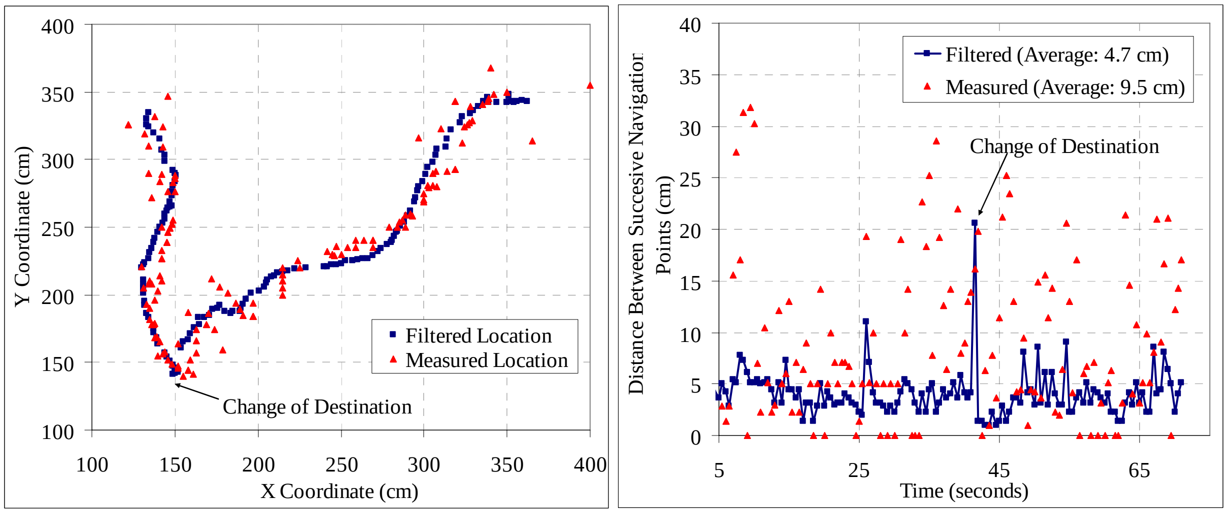


Fig. 8: Navigation performance: a) absolute locations, and b) distance between successive navigation points

The points in Figure 8:a represents the measured coordinates from the vector Z_k and the Kalman corrected coordinates from $S_k^{(+)}$. These points clearly demonstrate how the presented filter is able to smooth out the navigation of a SCAV in the presence of measurement errors. The effects are further quantified in Figure 8:b, in which the distances between successive measured locations (Z_k, Z_{k-1}) and the filtered locations $(S_k^{(+)}, S_{k-1}^{(+)})$ are plotted with time. The average of that distance for the measured locations is approximately 10cm, although the individual measurement goes as high as 40cm. In the ideal case, the average should be around 4.5cm, which is the distance traveled in 500ms (Δt) at a SCAV speed of 9cm/sec used in these experiments. Although there are some high values, the filtered distances average to a pretty close value of 4.7cm. This further confirms the effectiveness of the filter. Note the spike that corresponds to the change of destination caused by the second GO command. Also observe as to how the measurement errors are significantly larger compared to the static localization errors presented in Figure 6. This difference can also explain why a *stop-and-go* navigation, that uses static localization, can deliver more accurate navigation compared to its continuous counterpart used in the experiments here.

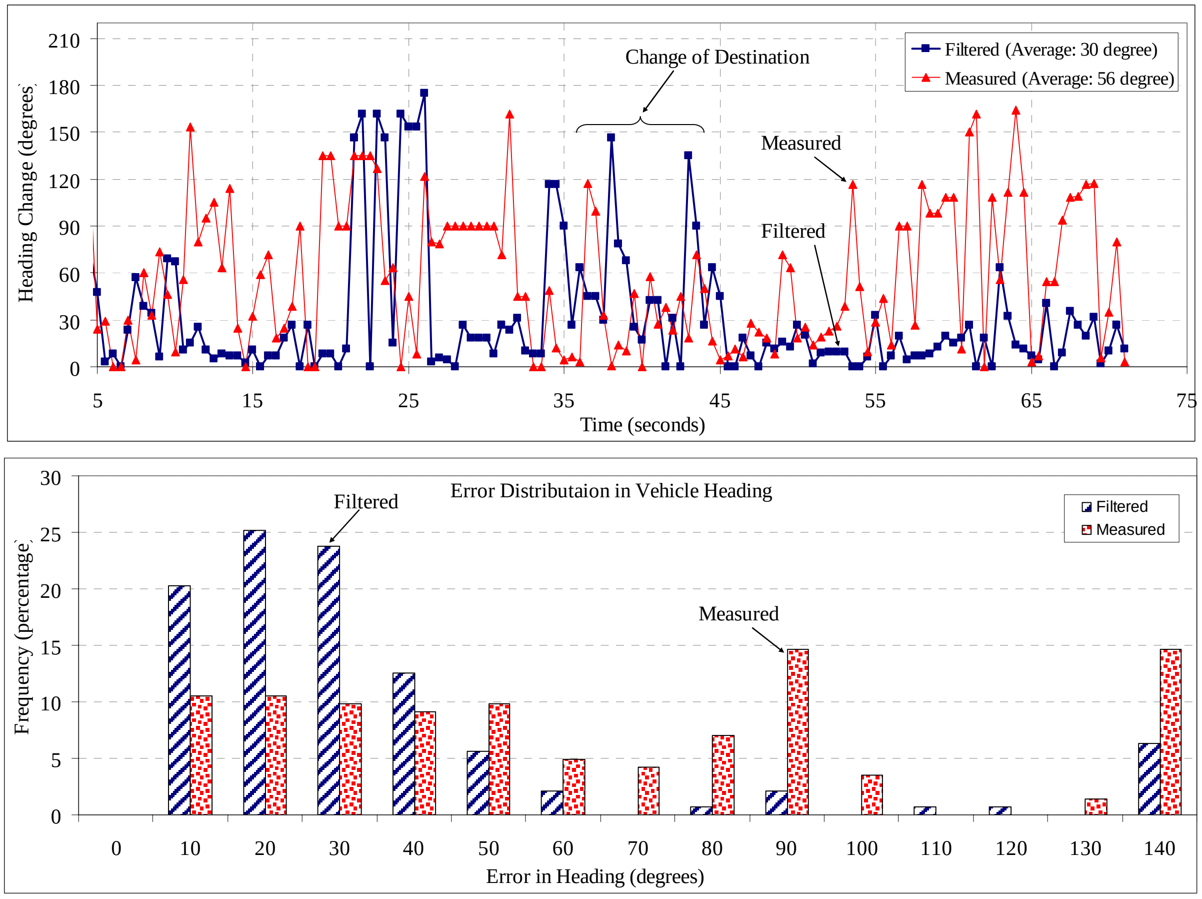


Fig. 9: Navigation performance: a) heading changes, and b) heading error distribution

The filter's effectiveness in alleviating navigation inaccuracies in terms of sensor heading are presented in Figure 9. In Figure 9:a, the absolute difference between the computed headings in successive navigation instances are plotted with time. The measured values correspond to heading computed using Z_k, and the filtered values correspond to heading computed using $S_k^{(+)}$. In ideal situation, the heading should remain constant at an angle of 45^o for the first leg, and 90^o for the second leg of the navigation segments shown in Figure 8:a. The differences in heading, therefore, should remain zero except at the point of destination change at the end of the first leg. The results indicate significantly higher errors and variation in the measured heading changes compared to those computed through filtering.

To quantify these errors further we plot the distributions of the absolute heading errors (difference from 45^o and 90^o in the first and the second legs) in Figure 9:b. Observe how the filtered errors are clustered mostly within an error of 30^o, whereas the raw measurement errors are spread over a much wider range.

These results validate our integrated system in terms of: a) the SCAV platform architecture including its hardware and software components, b) the CDMS based debugging infrastructure, c) indoor self localization system, d) dynamic navigation with a Kalman filter model, and e) integrated operation of all the above subsystems. As shown in Figure 2, all these developed services are leveraged by the collaborative sensor applications including the multi-target tracking framework as presented in the next section.

7. Collaborative Multi-target Tracking

The tracking problem is formulated as follows. There are *M*-number of Mobile Target Agents (MTAs) plying in a reconnaissance area, and the goal is to efficiently track and follow the MTAs using *A*-number of *Autonomous Reconnaissance Vehicles* (ARVs). The ARVs are able to sense MTAs and navigate collaboratively for maximizing their tracking coverage. The problem is to develop distributed algorithms for such collaborative tracking.

Successful tracking for an MTA is defined by the situation when at least one ARV follows the MTA while being within a pre-defined range, termed as the *Tracking Range*. For a sufficiently large time horizon, if D_i represents the cumulative tracking duration for MTA_i, then the overall tracking performance for the entire tracking system can be represented by a Cumulative Tracking Index (CTI), defined as $\sum_{i=1}^{M} D_i$, where *M* is the total number of MTAs in the system. While maximizing the CTI will ensure the best utilization of the ARV fleet, keeping the Coefficient of Variation (COV) of D_i ($i=1,2,\ldots M$) will ensure that certain MTAs will not remain untracked while the others are aggressively tracked. The COV is computed as the ratio of standard deviation of the D_i values and their average. The objective of the tracking algorithms will be to maximize CTI, while keeping the COV small.

7.1 Networked Tracking Algorithm

We propose the following heuristic based collaborative tracking algorithm. The key concept of the algorithm is that the ARVs wirelessly share their individual tracking performance, and individual ARVs *switch* to different MTAs when that is deemed appropriate for meeting the CTI and COV objectives stated above. A pseudo-code for different components of the algorithm is presented in Figure 10.

```
/* Collaborative Tracking Algorithm executed by $ARV_j$ */
Sensing:
$ARV_j$ periodically senses to check if it is within the tracking
range of any MTA
Stored Information:
A D-table with the currently stored $D_i$ values for all $MTA_i$
($i=1,2,...M$); all $D_i$ values are initialized to be zero;
Information Exchange:
$ARV_j$ Periodically sends its D-table to the rest of the ARVs
through an inter ARV ad hoc network

Tracking:
While (for ever) {
   IF (not locked with any target MTA) {
      Remain stationary and wait till an MTA is sensed;
      // $MTA_k$ is sensed by the $ARV_j$
      $ARV_j$ locks to $MTA_k$ and starts tracking it
     $D_k$ in $ARV_j$'s table is incremented by one after
     each constant duration of tracking by $ARV_j$
   } ELSE {
      // $ARV_j$ is currently locked to an $MTA_r$
      // $D_r$ in $ARV_j$'s table is incremented by one after
      // each constant duration of tracking by $ARV_j$
      IF (an $MTA_k$ is sensed by the $ARV_j$) {
         // $ARV_j$ may need to switch from $MTA_r$ to$MTA_k$
         Compute $\hat{D}$, the average $D_i$ over all available MTA
         entries in the local D-table
         IF (($D_r > \hat{D}$) && ($D_r > D_k$ + Switch_Threshold)) {
            $ARV_j$ switches to $MTA_k$ and starts tracking it
         }
      }
   }
}
D-Table Update:
When $RAV_j$ receives a D-table from another RAV, it merges
the received table with its existing table so that: 1) new
MTA entries are created as needed, and 2) for MTA entries,
existing in both the tables, update the $D_r$ field by the
maximum of the two.
```

Fig. 10: Pseudo code for tracking algorithm

The tracking algorithm is decoupled from the D-table dissemination model, and can work with a variety of mechanisms such as controlled flooding, dynamic spanning trees, and similar network protocols. Also note that as long as an ARV remains locked to an MTA, the appropriate D_i value is updated within the local D-table of the ARV. The D_i value in ARV_j's table is increased by one after each constant duration of tracking of MTA_i by ARV_j. Since the D-table merger (see Fig. 10) always use the maximum network wide available value of D_i, this quantity represents the cumulative tracking time of MTA_i by all the ARVs in the network.

According to the presented algorithm, when there are more (or equal) ARVs than MTAs, the algorithm reaches a steady state when at least one ARV gets locked to an MTA. No further target switching is necessary in these scenarios. Even when the MTAs outnumber the ARVs, the Cumulative Tracking Index (CTI) can be maximized just by ensuring that each ARV gets locked to a different MTA. This however does not minimize the Coefficient of Variation of CTI, since it is then possible that few MTAs may remain untracked for the entire time horizon. To avoid this, the concept of switching has been introduced in the algorithm. An ARV, which is currently tracking MTA_r, decides to switch to a newly sensed MTA_k only if it concludes that MTA_r had been better tracked than the network wide average tracking history of all the MTAs in the system, and its MTA_r's CTI is more than that of MTA_k by more than a threshold amount.

A key assumption for this tracking framework is that while sensing an MTA, an ARV is able to detect the identity of an MTA. This enables the ARVs to collaborate for deciding when and how to switch across different MTAs. Without this assumption the problem becomes more complex and that is a topic of our ongoing work.

7.2 Implementation of Tracking on the SCAV Sensor Test-bed

The tracking algorithm was evaluated for varying number of MTAs and ARVs using up to five SCAV units and their point-to-point navigation services as described in Section 5. The MTAs are realized using a set of SCAVs, programmed to move along preset trajectories, and a different set of SCAVs were used as the ARVs within our *3m x 3m* navigation test-bed. The periodic MTA sensing process by the ARVs, indicated in Figure 10, was emulated by programming each MTA to send a periodic wireless LOC beacon containing its location information. Since an ARV is aware of its own location, when it receives a LOC beacon from an MTA within a preset *Tracking Range*, a successful MTA sensing is assumed to be accomplished. Also, an ARV navigates toward the location information found in the most recently received LOC packet from a locked MTA. Due to this location based emulation, the sensing process had similar order of errors as found for the dynamic location errors in Figures 8 and 9. Different levels of

MTA sensing sensitivity were emulated by simply varying the preset *Tracking Range*. Controlled flooding was used for the D-table exchange (see Figure 10) through a 900MHz mobile ad hoc network formed by the ARVs. The ARVs periodically transmit SYNC beacons containing its local D-table information for the controlled flooding.

When a large number of moving SCAVS (more than 3) are placed within the *3m x 3m* navigation test-bed, frequent sensor collisions were observed. To mitigate such collisions, the following measures were implemented. 1) using an Infra Red based proximity detector for collision avoidance, 2) ensuring a minimum separation between all moving sensors, and 3) a networked "*shadow tracking*" to avoid collisions between multiple ARVs tracking a single MTA. Under such situations, only one ARV directly follows the MTA, and the other ARVs collaboratively choose moving "*shadow*" points of the MTA which are slightly away from the MTA itself. This prevents any undesirable convergence between the tracking ARVs, thus reducing the sensor collision possibilities. The shadow arbitration among multiple ARVs is accomplished based on the unique identifiers of each SCAV units. Also a speed differential has been introduced between the MTAs and the ARVs, so that a speeding ARV can always catch up with a locked MTA by getting closer to it.

7.3 Tracking Performance

The graphs in Figure 11:a show scenarios involving one and two Autonomous Reconnaissance Vehicles tracking a Mobile Target Agent which is programmed to travel along the trajectory: *(350,350)→(150,150)→(350,150)→(150,350)→(350,350)→(150,150)→(150,350)→(350,150)→(350,350)*. For each experiment, 160 LOC packets were sent out by the MTA at the rate of one packet every 2s. The Cumulative Tracking Index (CTI) was measured by counting the number of LOC packets received by an ARV while being within the tracking range of the MTA. All reported results represent an average from three separate tracking runs with the same initial conditions. The ARVs are initially placed such that they are within the sensing range of the MTAs.

For the 1-MTA/1-ARV scenario, the tracking problem reduces to a simple leader-follower strategy without any need for MTA switching as introduced in the general algorithm in Figure 10. In the absence of MTA switching, the algorithm is always expected to provide the best CTI of 160. However, the results in Figure 11:a indicate that the measured CTI can be significantly smaller, especially for short tracking ranges. This is because of the navigations errors, as shown in Figure 8 and 9, which occasionally force the ARV to erroneously drift out the tracking range of the MTA, especially when the range is small. With increased tracking range, since the navigation errors become smaller in a relative term, the CTI value improves, although it finally saturates at a value of 120, which is lower than the best case of 160.

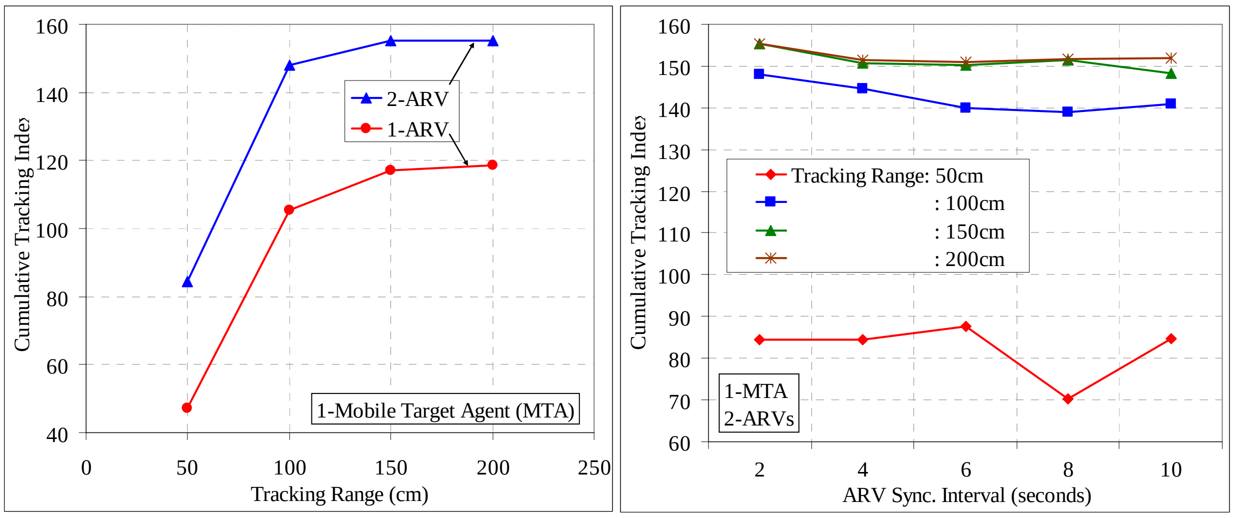


Fig. 11: Tracking performance: effects of a) tracking range, and b) consumed wireless bandwidth

Introducing two ARVs can help improving the situation simply because when one ARV looses the MTA due to navigation error, it is likely that the other ARV is still locked to it. As shown in Figure 11:a, this collaboration effect improves the CTI for all tracking ranges, eventually achieving the near-best case performance of 157, for a tracking range of 200cm. Since two ARVs can simultaneously track one MTA, only one assumes the role of the primary tracker and the other remains as a *shadow tracker* as explained before. These roles however frequently alter when the primary tracker momentarily moves out of the tracking range due to navigation errors.

The effects of variable SYNC beacon frequencies are shown on Figure 11:b. Lower SYNC intervals correspond to higher D-table exchange frequency among the ARVs. With more frequent D-table exchanges, an ARV is expected to take better MTA switching decisions with fresh D_i values received from the other ARVs. The results indicate that, although mild, this effect is there for smaller SYNC intervals (2s to 6s) and for larger tracking ranges. For smaller transmission ranges, however, the navigation errors offset the benefits of fresher D_i values. This results generally indicate that better tracking performance can be obtained at the expense of higher network bandwidth in the form of frequent

D-table exchanges over the inter ARV ad hoc network.

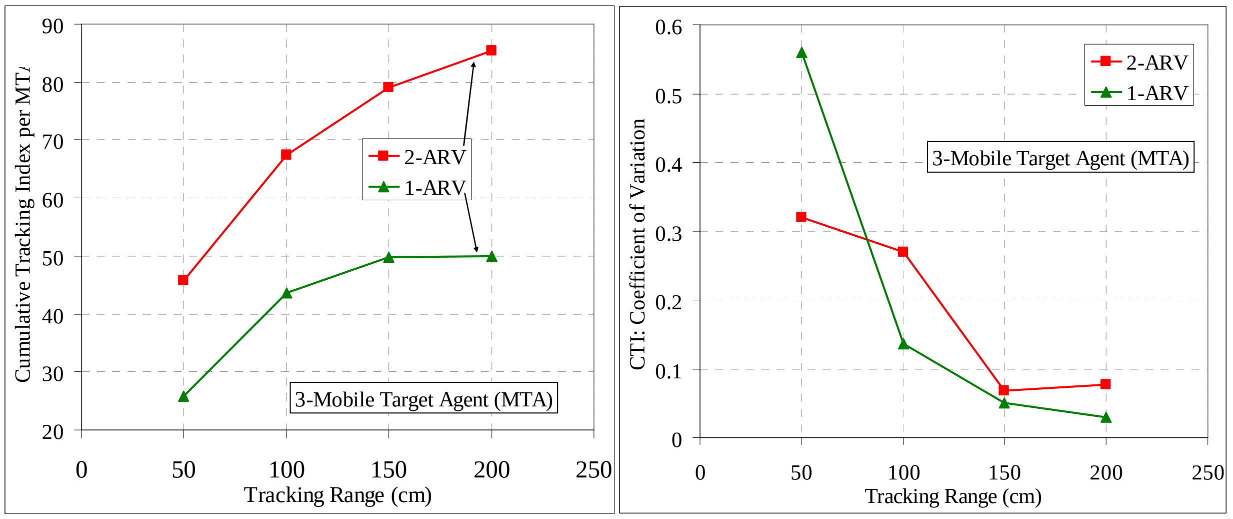


Fig. 12: Tracking performance with more MTAs than ARVs

Performance when the MTAs outnumber the ARVs is shown in Figure 12. The trajectory *(350,350)→(150,350)→(150,150)→(350,150)→(350,350)→(150,150)→(350,150)→(150,350)→(350,350)* is used by three MTAs with sufficient phase difference for avoiding excessive sensor collisions due to SCAV crowding. As shown in Figure 12:a, with increasing tracking range, the CTI values improve due to the same reasons as for the 1-MTA/1-ARV scenario in Figure 11:a. Also, as shown in Figure 12:b, With increasing tracking range, the coefficient of variation for CTI significantly reduces due to the fact that with larger tracking ranges the ARV is able to sense the MTAs more frequently. As a result, the ARV is able to perform MTA switching more effectively, thus reducing the variance of tracking time across the MTAs.

CTI per MTA and the corresponding coefficient of variation, when two ARVs tracking three MTAs, are also reported in Figure 12:a, b. It is evident that introducing a second ARV enables collaboration, and improves the CTI values. Also, the CTI variance across the MTAs is reduced by the fact that an MTA is now less likely to remain untracked.

As shown in Figure 13, we have experimented with different *Switch_Threshod* values (see the algorithm in Figure 10) in order to evaluate its effects on the MTA switching process. With higher threshold values, MTA switching is discouraged, as a result of which the coefficient of variation for CTI increases. Meaning, while few MTAs are well tracked, few others may remain untracked due to the lack of prompt

MTA switching. Although counterintuitive, our experiments showed that the CTI values also degrade with higher *Switch_Threshod* values. Close investigations revealed that the SCAV navigation errors (see Figures 8 and 9) are responsible for this effect in the following manner.

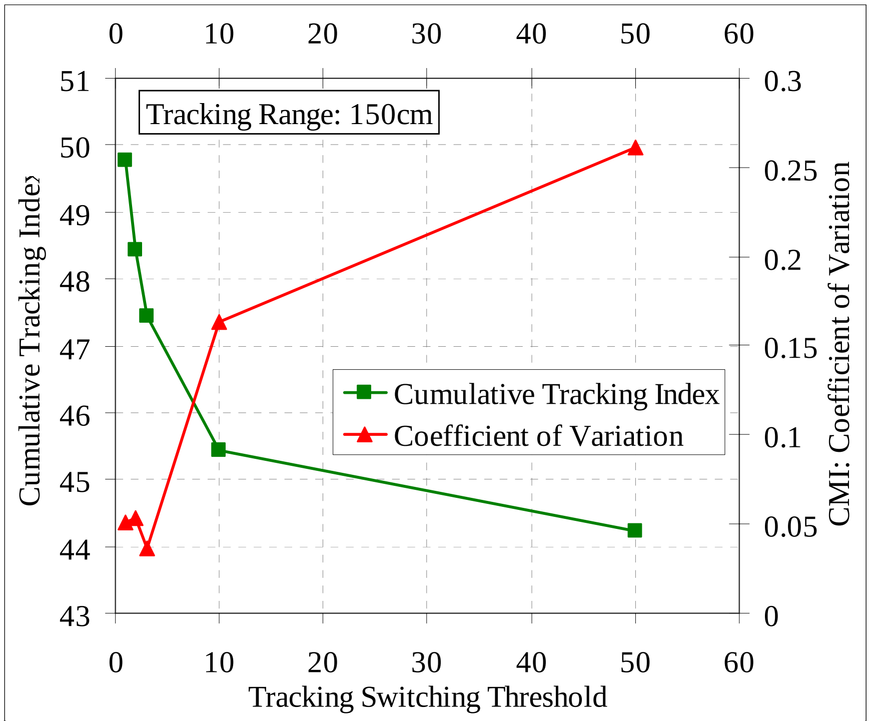


Fig. 13: Effects of varying switching thresholds

As seen in Figure 11, certain amount of loss of tracking happens even in 1-MTA/1-ARV scenario simply due to the navigation errors even in the absence of MTA switching. Similar effects are present here for a large *Switch_Threshod*, when very few MTA switching are happening. With more frequent switching, this loss of tracking gets compensated because an MTA now spends less time not being tracked, thus improving the CTI per ARV values. In the presence of ideal localization and navigation with no or very little error, we expect the CTI per ARV to in fact increase up to a point with higher *Switch_Threshod* values.

These target tracking results demonstrate that non-ideal localization and navigation conditions can cause theoretically sound algorithms to generate unexpected results, and therefore special experimental considerations will be necessary while implementing such mobile sensor systems under non-ideal conditions.

8. Summary and Future Work

We have presented the design, architecture, implementation experience, and experimental results from a networked mobile sensor test-bed developed for collaborative sensor applications. The test-bed comprises a fleet of networked mobile sensors, an indoor localization system for enabling sensor self-localization in GPS-denied environments, a control, debugging and management infrastructure, and a tiered wireless ad hoc network for seamless integration of the above three components and the existing wireless network infrastructure. A multi-target sensor tracking framework has been implemented on the mobile sensor test-bed as a representative collaborative application. Joint networking and tracking mechanisms are developed for tracking multiple mobile targets using a team of networked mobile sensors.

In the first part of the paper, the architectural and implementation details about the software and hardware design of the developed mobile sensors and its various supported services are presented. In the second part, first the problem of multi-target tracking using collaborative mobile agents has been introduced. Then a networked distributed tracking algorithm has been formally proposed. Finally, we report the experimental performance of the proposed tracking framework implemented in our mobile sensor test-bed. In addition to valuable implementation insights about the localization, navigation, Kalman filtering, and ad hoc networking processes, the experimental results lead us to the following conclusions about the overall system. First, localization and navigation errors, which are usually present in real-world scenarios, can significantly affect the tracking performance, even for a simple 1-leader/1-follower scenario. Navigation errors are found to have more complex performance impacts in scenarios with multiple targets and tracking sensors. Second, there exists a tradeoff between the tracking performance and the consumed wireless bandwidth. It is found that in most scenarios, better tracking can be achieved at the expense of increased communication through the mobile ad hoc networks formed by the tracking sensors. Finally, the frequency at which a tracking sensor switches its target is an important design parameter in scenarios with higher number of targets than the tracking sensors. Careful tuning of target switching will be needed for striking a balance between acceptable overall tracking and the variance of tracking across individual targets.

Future work on this topic includes developing a number of sensor swarming algorithms for collaborating attack localization and search strategies using the baseline tracking mechanisms developed in this paper. More simulations will be conducted to compare the different navigation and tracking algorithms with exiting ones. We are also experimenting with higher order Bayesian filters for improved navigation and tracking in the presence of higher localization errors.

9. References

[1] B. T. Ludington, L.tang, and G. J. Vachtsevanos, "Target tracking in an urban warfare environment using particle filters," In Proceedings of Aerospace, 2005 IEEE Conference.

[2] T. Yu and Y. Wu, " Decentralized multiple target tracking using netted collaborative autonomous trackers," Computer Vision and Pattern Recognition, 2005.

[3] F. Zhao, J. Shin and J. Reich, "Information-Driven Dynamic Sensor Collaboration for Tracking Applications," IEEE Signal Processing Magazine, March 2002.

[4] B.H. Kim, C. D'Souza, R. Voyles, J. Hesch, and S.I. Roumeliotis, " A Reconfigurable Computing Platform for Plume Detection with Mobile Sensor Networks," In Proc. SPIE Conference on Unmanned Systems Technology VIII, Orlando, FL, Apr. 2006.

[5] J. Friedman, D. C Lee, I. Tsigkogiannis, S. Wong, D. Chao, D. Levin, W. J Kaiser, M. B Srivastava, " RAGOBOT: A New Platform for Wireless Mobile Sensor Networks", in Proc. of the 1st IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2005), June-July 2005.

[6] C. Wu, W. Sheng, and Y. Zhang, "Mobile Self-Localization using Multi-Dimensional Scaling in Robotic Sensor Networks", The International Journal of Intelligent Control and Systems, vol. 11, no. 3, Sept., 2006

[7] G. Jang, S. Kim, W. Lee, and I. Kweon, "Robust Self-localization of Mobile Robots Using Artificial and Natural Landmarks", Proc. 2003 IEEE Int. Symp. On Computational Intelligence in Robotics and Automoation, pp. 412-417

[8] C. Chang, "Improving hallway navigation in mobile robots with sensor. habituation," in Proc. 2000 IEEE Int. Joint Conf. Neural Networks", July 2000

[9] F. Iida, "Goal-Directed Navigation of an Autonomous Flying Robot Using Biologically Inspired Cheap Vision," Proceedings of the 32nd ISR (International Symposium on Robotics), 19-21 April 2001

[10] J. Hahner, C. Becker, K. Rothermel, "A protocol for data dissemination in frequently partitioned mobile ad hoc networks". Proc. of the 8th IEEE Intl. Symp. on Computers and Communication (ISCC) 2003

[11] S. Goel, M. Singh, D. Xu, B. Li. "Efficient Peer-to-Peer Data Dissemination in Mobile Ad Hoc Networks," in the Proceedings of the International Workshop on Ad Hoc Networking (IWAHN 2002), Vancouver, British Columbia, August 18-21, 2002

[12] A. Miu, "Design and Implementation of an Indoor Mobile Navigation System," MS Thesis, Computer Science and Engineering, Massachusetts Institute of Technology, Jan. 2002.

[13] A. Savvides, H. Park and M. Srivastava, "The *n*-Hop Multilateration Primitive for Node Localization Problems," Mobile Networks and Applications vol. 8, 2003, pp. , 443–451.

[14] I. Petersen and A. Savkin, "Robust Kalman Filtering for Signals and Systems With Large Uncertainties,". Springer Verlag, 1999.

[15] Swarm Capable Autonomous Vehicles (SCAV): a networked micro-robotic system for collaborative tracking applications: www.egr.msu.edu/~sbiswas/Research_n/SCAV.htm

[16] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," In Proceedings of. 6th ACM MOBICOM.

[17] Crossbow inc: www.xbow.com/

[18] BrickOS project page: http://www.cs.ecu.edu/~hochberg/fall2002/brickOS/index.html

[19] TinyOS Project page : http://sourceforge.net/projects/tinyos

[20] N. Priyantha, "The Cricket Indoor Location System," Ph.D. Thesis, Computer Science and Engineering, Massachusetts Institute of Technology, June 2005.

[21] A. Savvides, C. Han and M. Strivastava, "Localization in Ad Hoc Networks of Sensors," In Proceedings of ACM SIGMOBILE, July 2001, Rome, Italy

[22] N. Balasu, J. Heidemann, S. Estrin, and T. Tran, "Self-Configuring Localization Systems: Design and Experimental Evaluation" ACM Transactions on Embedded Computing Systems, Vol. 3, No. 1, February 2004, pp. 24–60.

[23] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha, "Tracking Moving Devices with the Cricket Location System," *MobiSYS'04,* June 6, 2004, Boston, Massachusetts, USA.

[24] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian Filters for Location Estimation," IEEE Pervasive Computing, Vol. 3, pp 1536-1268, 2003.

[25] Songhwai Oh, Luca Schenato, Phoebus Chen, Shankar Sastry, "Tracking and Coordination of Multiple Agents Using Sensor Networks: System Design, Algorithms and Experiments," Proceeding of IEEE, Jan, 2007.

[26] P. Corke, R. Peterson and D. Rus, “Localization and Navigation Assisted by Networked Cooperating Sensors and Robots,” Int. J. Robotics Research, vol. 24(9), September 2005.

[27] C. M. Cianci, J. Pugh, and A. Martinoli, “Exploration of an Incremental Suite of Microscopic Models for Acoustic Event Monitoring Using a Robotic Sensor Network,” In IEEE International Conference on Robotics and Automation, 2008.

[28] R. R. Brooks, D. Friedlander, J. Koch, and S. Phoha, “Tracking multiple targets with self-organizing distributed ground sensors,” J. Parallel Distrib. Comput, vol. 64, 2004.

[29] J. Liu, J. Reich, and F. Zhao, “Collaborative in-network processing for target tracking,” J. Applied Signal Processing, Apr 2003.

[30] D. Li, K. Wong, Y. H. Hu, and A. Sayeed, “Detection, classification and tracking of targets,” IEEE Signal Process. Mag., vol. 19, no. 2, Mar 2002.

[31] J. Shin, L. Guibas, and F. Zhao, “A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks,” in Proc. 2nd Workshop Information Processing in Sensor Networks, Apr 2003.

[32] J. Liu, J. Liu, M. Chu, J. Reich, and F. Zhao, “Distributed state representation for tracking problems in sensor networks,” in Proc. 3rd Workshop Information Processing in Sensor Networks, Apr 2004.