Multiplatform Information-Based Sensor Management: An Inverted UAV Demonstration

Chris Kreucher, John Wegrzyn, Michel Beauvais, and Ralph Conti

General Dynamics' Michigan Research and Development Center
1200 Joe Hall Drive, Ypsilanti, MI, USA

ABSTRACT

This paper describes an experimental demonstration of a distributed, decentralized, low communication sensor management algorithm. We first review the mathematics surrounding the method, which includes a novel combination of particle filtering for predictive density estimation and information theory for maximizing information flow. Earlier work has shown the utility via Monte Carlo simulations. Here we present a laboratory demonstration to illustrate the utility and to provide a stepping stone toward full-up implementation. To that end, we describe an inverted Unmanned Aerial Vehicle (UAV) test-bed developed by The General Dynamics Advanced Information Systems (GDAIS) Michigan Research and Development Center (MRDC) to facilitate and promote the maturation of the research algorithm into an operational, field-able system. Using a modular design with wheeled robots as surrogates to UAVs, we illustrate how the method is able to detect and track moving targets over a large surveillance region by tasking a collection of limited field of view sensors.

Keywords: Sensor Management, Multi-target Tracking, Information Theory, Demonstration

1. INTRODUCTION

This paper describes a demonstration of a distributed, decentralized, low communication information-based sensor management algorithm developed at the The General Dynamics Michigan Research and Development Center (MRDC). The method, which is based on a novel combination of particle filtering for predictive density estimation and information theory for action selection, has been illustrated by simulation in a number of realistic model problems.[1,2] In this paper, we focus on a laboratory demonstration of the system, and describe the components needed to produce such a demonstration system and the engineering trades involved.

Sensor management, as defined here, is the process where an agile platform autonomously determines what action it should take. Agility comes in many varieties - for example, platforms may be mobile, may be able to point a sensor, and/or choose which mode to use. We focus on the distributed multi-platform case, where the goal of a sensor management algorithm is to dictate the behavior of individual nodes so that the network as a whole satisfies some global objective. The method we present is specifically designed for large networks of sensor nodes that require limited communication and a decentralized approach. As such, it relies only on communication between neighboring nodes and when communication is done, only a selected subset of a node's measurements are shared.

In previous work,[2] we have focused on a model problem consisting of a collection of mobile unmanned aerial vehicles (UAVs) that interrogate a surveillance region looking for moving ground targets. Each platform is equipped with a down-looking sensor that measures the small portion of the surveillance region immediately below. The purpose of the network is to determine the number of targets and the kinematic state (e.g., position and velocity in 2-D) of each target through repeated interrogation of these small ground patches. The sensor management problem then becomes one of choosing the movement each platform should take to reach the goal as reliably and quickly as possible. Simulation results in those situations show order-of-magnitude type improvement over simple periodic scan when measured by metrics such as time until detect, and the number of resources needed to reliably survey a region.

Further author information: (Send correspondence to Christopher M. Kreucher)
E-mail: {Christopher.Kreucher, John.Wegrzyn, Michel.Beauvais, Ralph.Conti}@gd-ais.com

Defense Transformation and Net-Centric Systems 2007, edited by Raja Suresh,
Proc. of SPIE Vol. 6578, 65780Y, (2007) · 0277-786X/07/$18 · doi: 10.1117/12.719395

In this paper, we describe a demonstration of the method on a related surrogate problem. For logistical convenience, we instead use a collection of ground platforms with up-looking sensors interrogating the sky for moving targets. This "inverted UAV" problem serves to illustrate all of the relevant components of the algorithm and illuminate the engineering challenges, while avoiding flight control and avionics issues that are not directly related to the sensor management solution we wish to illustrate.

The paper proceeds as follows. In Section 2, we give a brief description of the mathematics behind the sensor management algorithm used in the demonstration. Additional details are provided elsewhere.[3,4] In Section 3, we give describe the implementation, including the robotic platforms, the associated sensors, and the signal processing algorithms used to provide platform self localization and sensor processing. Finally, we conclude in Section 4 with some concluding remarks and thoughts on future work.

2. MATHEMATICAL BACKGROUND

This section briefly reviews our method of multiple platform control. More detail is given elsewhere.[3,4] As discussed therein, the approach used here is related to the approach of others[5–10]

Our method of sensor management is based on maximizing information flow through the network. In a network consisting of N agile sensors, we choose the action set $(r_1, r_2, \cdots, r_{N-1}, r_N)$ that is expected to maximize the total utility of the next measurements made. The utility is measured in terms of an information theoretic metric known as the Renyi Divergence which, loosely speaking, values action sets that reduce uncertainty.

The method proceeds as follows. First, sensor models, target models, prior information and sensor measurements are fused together in an entity known as the Joint Multitarget Probability Density (JMPD),[11]

$$p(X, T|Z) \equiv p(x_1, x_2, \cdots, x_{T-1}, x_T, T|Z) = p(x_1, x_2, \cdots, x_{T-1}, x_T|T, Z)p(T|Z) \ . \tag{1}$$

The JMPD captures the uncertainty one has about the state of the system (both the uncertainty about the number of targets T and the states of each individual target x_i). As such, it is an ideal quantity to use for sensor management, as the goal of sensor management is to choose actions which in some sense minimize the uncertainty about the state of the system.

Because of its high dimensionality and non-parametric form, the JMPD is estimated using particle filter methods.[12] The JMPD is a hybrid continuous discrete distribution, which is defined over both the discrete number of targets at time k, T^k and the continuous target state space at time k, $X^k \in \Re^{nT}$, i.e., the normalization criteria is $\sum_{T=0}^{\infty} \int dX p(X, T|Z) = 1$, where the single integral sign is to be interpreted to represent the T integrations required. This single probabilistic entity captures all of the uncertainty about a surveillance region, including uncertainty about how many targets there are and what their kinematic states are (e.g., position and velocity).

The second step in our approach is to use the uncertainty, as captured by the JMPD, to choose actions for the sensors to take. Specifically, we we wish to choose the set of sensing actions for the N sensors, $(r_1, r_2, \cdots, r_{N-1}, r_N)$, that lead to the best possible measurements $(z_1, z_2, \cdots, z_{N-1}, z_N)$. Here we define the a measurement as valuable when it reduces the uncertainty present in the JMPD. There is both a theoretical and practical reason for this choice of metric.[13] Of course, we do not know the measurement set that will be received until after the action is taken, so we instead advocate maximizing the *expected* uncertainty reduction an action set will produce. Formally stated, we wish to maximize the expected uncertainty reduction as measured between the JMPD before the latest set of measurements is taken (i.e., the prediction JMPD) and the JMPD after a set of measurements (the posterior JMPD).

A commonly used measure of the difference between two probability densities is the divergence. In this work we employ a particular type of divergence known as the Rényi Divergence, and use it to measure the amount of information flow (or uncertainty reduction) an action set has produced. The Rényi Divergence between densities $p(x)$ and $q(x)$ is denoted $D_\alpha(p||q)$, and its expectation (with respect to the measurements received) will be denoted $\mathbb{E}(D_\alpha(p||q))$. We then seek the action set that maximizes the Divergence between the JMPD before new measurements are made and the JMPD after new measurements are made, i.e., we seek the set

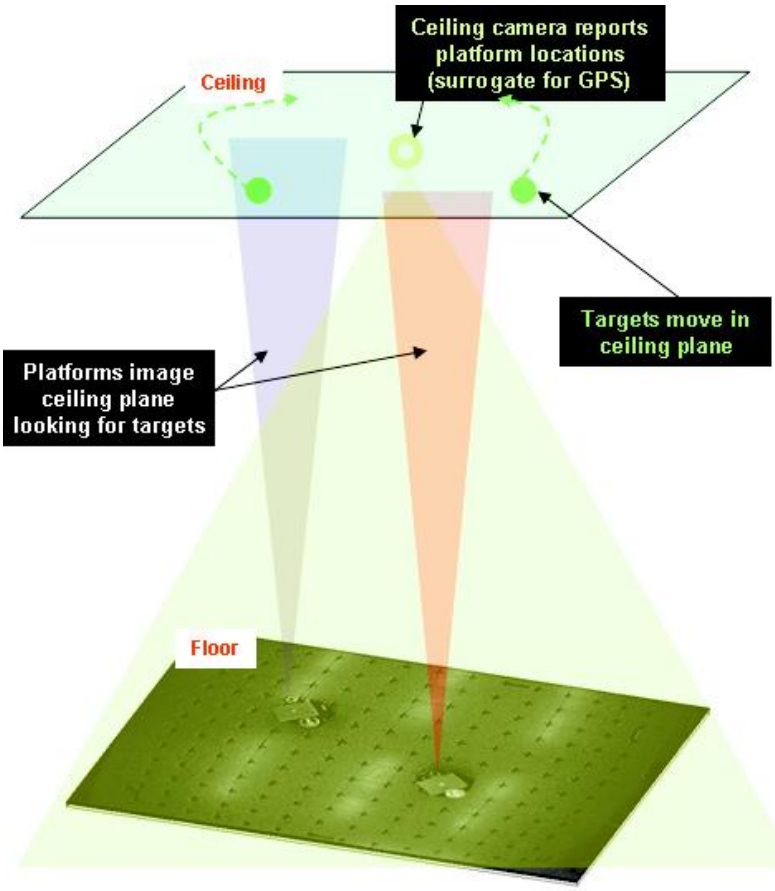

Figure 1. The experimental setup. The demonstration described here is an "inverted UAV" experiment that consists of ground platforms providing surveillance on an air region. Each ground platform has an EO sensor capable of interrogating only a small region above it for targets. The platforms are mobile and can communicate with nearby neighbors thus allowing them to reposition over time to survey the entire area.

$$(\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_{N-1}, \hat{r}_N) = \underset{(r_1,\cdots,r_N)\in \mathbb{C}}{\arg\max}\ \mathbb{E}\big[D_\alpha\big(p(\cdot|Z^{k-1}, z_1, \cdots, z_N, r_1, \cdots, r_N)||p(\cdot|Z^{k-1})\big)\big]\ . \tag{2}$$

Z^{k-1} refers to the set of measurements collected by all the platforms for all time up to and including time $k-1$.

This is a constrained optimization because, among other things, platforms must not collide and platforms must not violate kinematic constraints (i.e., maximum allowable acceleration). Straightforward application of this approach is $O(M^N)$ to compute, where M is the number of possible next-actions and N the number of platforms. Therefore exact implementation of the method is intractable for large N. Therefore, we use a computational method that combines Lagrangian relaxation and a physicomimetic approximation.[2]

3. DETAILS OF IMPLEMENTATION

In this section we describe an experimental demonstration of our multiplatform information based control algorithm. This work complements earlier work[2, 3, 14] which has illustrated the methods purely in simulation. Here, we describe the components necessary to implement the information theoretic sensor management method on real hardware, by discussing each of the components in the architecture.

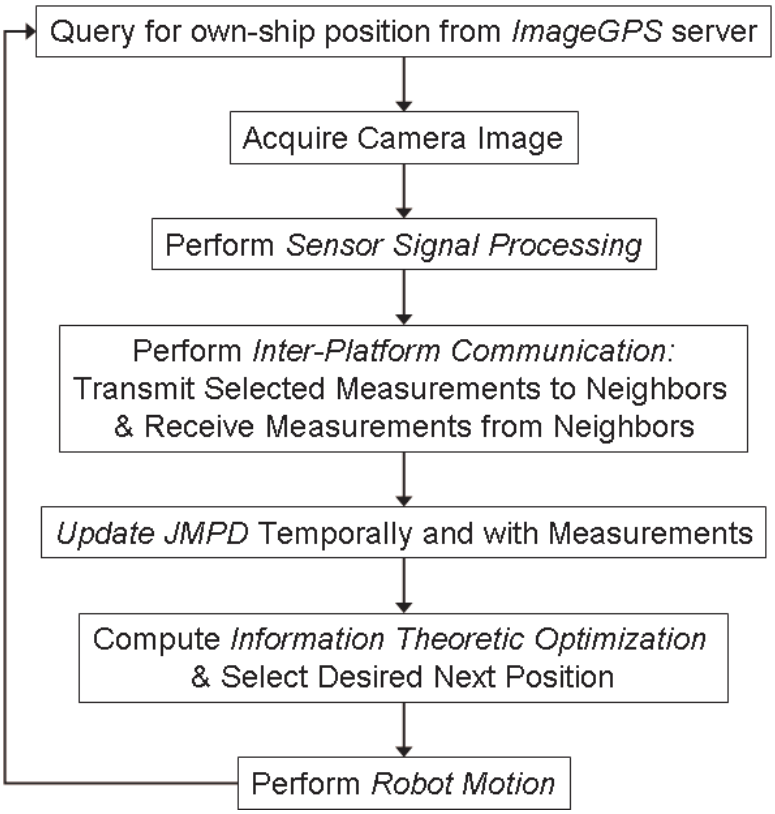

Figure 2. A system flowchart showing the process each platform follows. First, each platform acquires its own-ship position using the ImageGPS capability. Next, each platform acquires and processes a camera image. Selected measurements are then transmitted to neighbors, and measurements are received from neighbors. These measurements are used to update the JMPD and compute the information theoretic optimization used for sensor management. Finally, the required motion is sent to the robot and the robot moves. Each of the modules in this flowchart is described in the following subsections.

The rest of this section proceeds as follows. We first describe the experimental setup and surrogate platform used in the experiment. As discussed earlier, we invert the traditional UAV-based model problem for logistical convenience. We then enumerate and describe in detail each of the major individual elements of the system, from those that do sensor signal processing to those that effect robot motion.

3.1. Experimental Setup: The Inverted UAV

Our earlier simulation work[2] has focussed on a unmanned aerial vehicle (UAV) model problem. In this model problem, a collection of airborne platforms survey a ground area by repeated interrogation. Each UAV can only see a small portion of the ground, but is mobile, so by cooperation amongst the platforms the entire ground can be surveyed over time.

For reasons of convenience, we invert this model problem for the present demonstration. Instead of a collection of airborne sensors interrogating a ground region, we use a collection of ground sensors interrogating an air region. This defers the development and control issues associated with airborne sensors to experts in avionics and flight control, while still allowing illustration of the control algorithm. The air region consists of a number of moving targets (where the number, location, and velocity of each is unknown to the platforms at inception). All of the analogous concerns are present in this demonstration: the network of platforms is to cooperate together to perform surveillance on a region much larger than their instantaneous individual field of view; the number of targets and their kinematic states are time varying and unknown to the network at initialization; each sensor

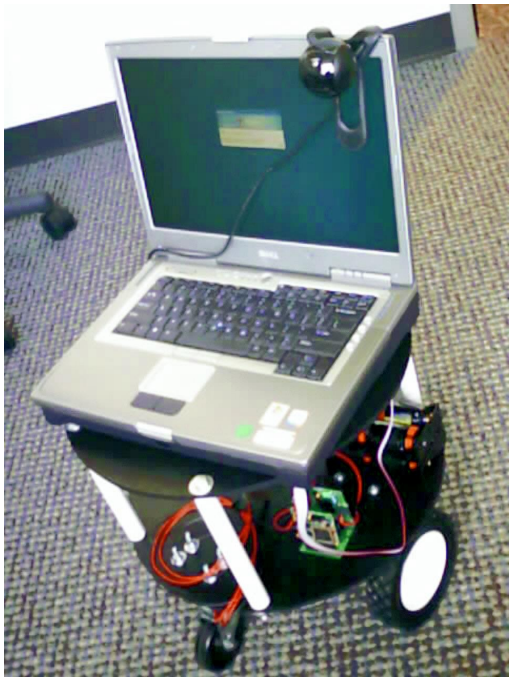

Figure 3. The Inverted UAV experiment described in this paper uses a wheeled ground robot that carries a standard laptop. The laptop executes the the sensor management algorithm described earlier and produces commands which dictate the movement of the robot. The low-level robot motion is controlled via a 16-bit CPU system which communicates with both the motor controller board and the laptop.

is imperfect (i.e., characterized by a detection probability and a false alarm probability); and each platform has both the ability to communicate with neighbors and the ability to move. By repeated interrogation and cooperation amongst the sensors, the surveillance region can be completely interrogated and moving targets detected and tracked. Figure 1 illustrates the laboratory setup graphically, and Figure 2 shows a flowchart which describes the actions each platform performs. Each important component in the illustration will be described in future sections.

3.2. The Surrogate Platform

A collection of hobby-shop type ground robots with EO sensors was used to illustrate the methods. The robot platform, shown in Figure 3, consists of a 16-bit CPU system with a serial port interface, an expansion backplane, a dual-motor controller backplane card, two DC-driven motors attached to two wheels through gear boxes, and two encoders to measure rotational movement of each wheel. A standard laptop computer sits on top of the robot chassis and communicates to the 16-bit controller over a serial interface.

Decisions about robot movements are made using the information theoretic control algorithm outlined earlier. The sensor used to gather measurements is a simple EO camera attached to each of the robots and pointing directly up. It generates measurements that provide evidence as to the presence or absence of targets immediately overhead. This information is used to construct the JMPD and compute the most valuable actions to take next. This algorithm is run on the laptop computer in MatLab^{TM}.

Desired movements are translated into commensurate requirements on how the robot wheels are to move, as measured by encoder ticks. This is done taking into account the robot geometry, current heading angle, and the relationship between encoder ticks and distance and angle movement. These requests are passed via serial communication to the robot controller board. Software written for the robot controller interprets these requests and does the low level control to actually make the wheels move in the prescribed fashion, complete with feedback and error checking.

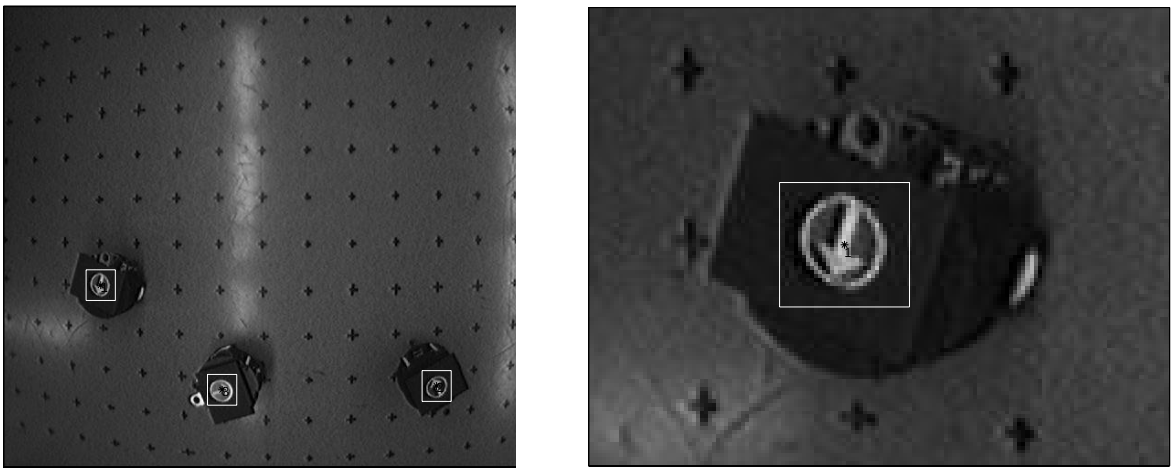

Figure 4. Left: The view of the entire laboratory from above. Right: The image zoomed at the location of one of the robots. The boxes drawn on each image show the result of the image GPS algorithm which has used signal processing algorithms to extract the location of the platforms. Information about own-ship position and orientation is provided to a platform upon request, analogous to traditional GPS.

3.3. The Image GPS Module

An important requirement of our method is that each platform knows its own location. In a fielded system this will be done using GPS, inter-sensor communication techniques,[15] dead reckoning or some other technique.

In our present laboratory implementation, however, there are several issues with GPS. The most significant issue is that GPS signals will not penetrate into our lab environment - as line-of-sight to the orbiting satellites is required. Additionally, given the relatively small area being used for the demonstration (approximately 15 feet by 15 feet) and the relatively large size of the robots (16 inch diameter cylinders), a highly accurate position estimate is required to avoid collisions - on the order of an inch.

A second, but equally important positioning constraint is the requirement to know the orientation (or compass direction) of the robot. Several solutions are available in the real world including the use of a compass and the accumulated movement vector information from GPS. As traditional GPS is not viable and the small package size of the robot platform precludes the easy electromagnetic isolation of a compass required for an accurate reading, another solution was required to address determining instantaneous orientation.

To deal with both position and orientation issues, a method of signal processing based on images collected from an overhead camera, which we refer to as "ImageGPS" was employed. By affixing color coded arrows (see Figure 4) to each robot oriented to point in the 'forward' movement direction, a camera mounted on the ceiling of the lab space can be used to find both the position of a specific color-coded robot and its relative orientation. A 'position server' module was written to run on a computer connected to both the ceiling camera and a wireless router. This server constantly examines the images from the ceiling camera and determines the position and orientation of each robot. PCs on each robot that control movement can query their position information via the wireless connection through the wireless router. Note that the only information provided to a platform from the ImageGPS is own-ship position. As such, the ImageGPS works similarly to a traditional GPS as it merely provides a platform with its own-ship location when desired.

3.4. Sensor Signal Processing

In this "inverted UAV" problem, the sensor is mounted on the robot and interrogates the area directly above it for targets. Targets are assumed to lie in a horizontal plane a fixed distance above the sensor. By knowing the sensor's position and orientation, each position in the sensor's field of view can be mapped to location in the target plane by a simple geometric transformation.

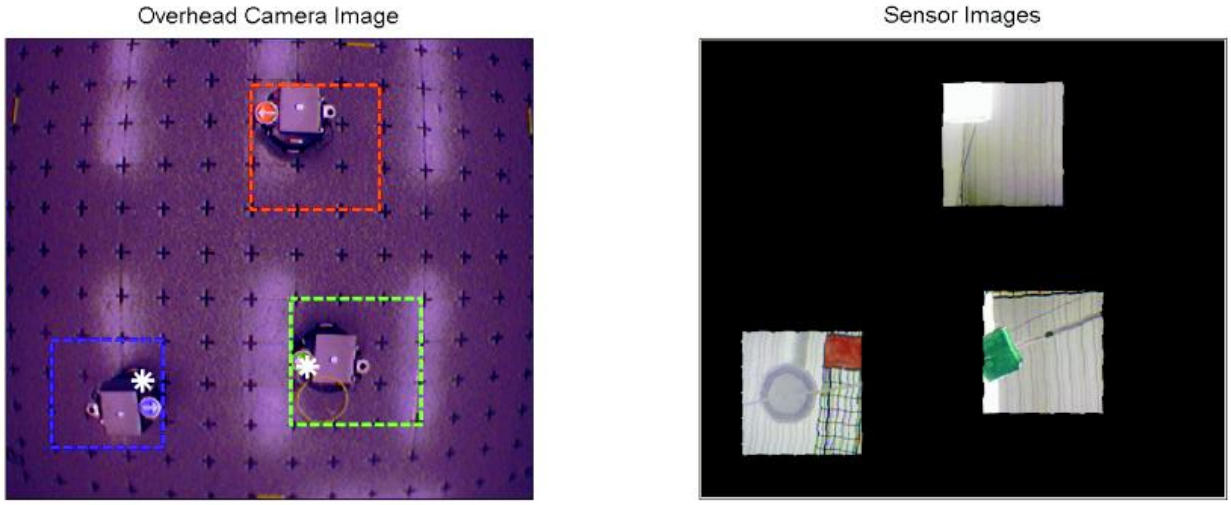

Figure 5. An illustration of the color camera images acquired by three platforms. Left: the location of a the platforms as viewed from an overhead camera. Right: the ceiling images acquired by the platforms, rotated and placed relative to the overall ceiling (note that each platform only knows about its own image and not those of others). The target is a homogeneous colored square (here only partially in the camera field of view), and the background contains lighting fixtures, duct work, ceiling, cords, and wire runs.

Detection of targets is color-based. This simplified method was chosen purposely as illustration of sensor signal processing methods is not the main goal of this demonstration. Rather, the demonstration seeks to illuminate the information theoretic method of control, so in that sense sensor signal processing is a nuisance parameter. The sensor chosen is an inexpensive color camera that generates an image where each pixel contains a three vector (R, G, B). Each target is a uniform color, distinct from the background (ceiling, lights and other clutter in the ceiling plane). An illustration of a captured image is given in Figure 5. A linear transform is used to convert the three color camera data into a single value per pixel detection map such that the target is emphasized. A target measurement is made by choosing those pixels of the detection map that correspond to the area to be measured, and averaging the values over those pixels. The threshold is chosen based on the desired (p_d, p_{fa}). An empirical ROC curve, which traces out the (p_d, p_{fa}) relationship for this sensor is shown in Figure 6.

3.5. Inter-Platform Communication

For networked sensors, an important real-world issue is communications with other sensors. To emulate the real-world fact that sensors have a limited communication capabilities (due to a number of factors including line-of-sight, transmission power limitations, and receiver noise), each sensing platform has a module that only allows it to communicate with neighboring sensors. By knowing where it is at any given time and by including the current location in each message sent to each platform, other platforms can do a simple distance calculation to decide if the message should be 'heard' and processed. In more complex implementations, probability models can be used to vary the communications success rate as a function of distance. In the laboratory this function is accomplished with wireless cards on each computer and a wireless router that handles the transmission of messages between platforms.

Furthermore, when sensors communicate, the method adopted here is to send only selected measurements. Sending measurements has the virtue of avoiding incest involved in sending probability densities as well as only requiring a small bandwidth. Measurements are selected to be sent from the most recently acquired measurements based on the likelihood (locally computed) that the measurement has originated from a target. Therefore, at each time step, there may be zero or many measurements transmitted to neighboring platforms depending on

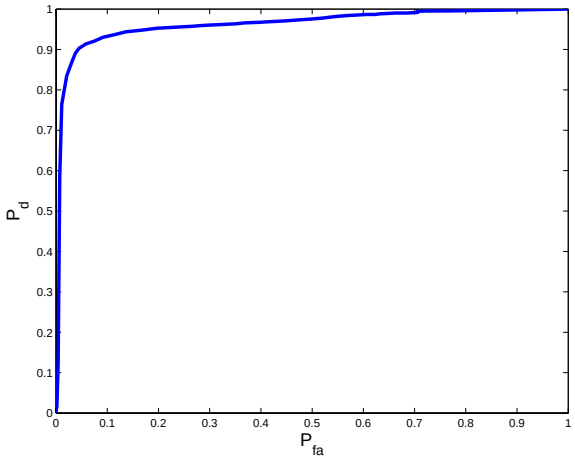

Figure 6. The Sensor ROC, which describes the detection/false alarm tradeoff for our EO sensor looking for moving targets in the ceiling plane. In our demonstration, we typically operate at $P_d \approx 0.85$.

the locally computed probability that the measurement has originated from a target. In our experiments, we find that approximately 5% of measurements are transmitted from each platform.

Finally, each platform also includes its own-ship position when sending messages to neighbors. This information allows a platform to compute a local optimization of its actions that approximates the joint information theoretic optimization (see Reference[2] for details). Again, only those platforms that are within the communication radius of the transmitting platform receive this information. Those that are farther than the communication radius hear nothing.

3.6. JMPD Update and Information Theoretic Movement Computation

The mathematics described in Section 2 is implemented on the laptop computers riding on each of the robot chassis. Each robot does a local computation using only information known locally (i.e., measurements made by the platform and those received from others) to compute a local JMPD. This is done by first updating the JMPD temporally using models on target kinematics and arrival probability. This includes an prediction of the probability density over kinematic states of known targets as well as prediction as to the number of targets. Next, the most recently received measurements are used to measurement update the JMPD using Bayes' rule (see Reference[12] for further discussion of this update).

As described earlier, the JMPD captures all of the uncertainty about the surveillance region. The sensor management method advocated here works by choosing actions that are expected to maximally reduce uncertainty about the surveillance region (see Reference[3] for further discussion of the information theoretic approach). In the multiplatform low communication setting, this computation is effected locally using only the local JMPD and the estimate of nearby neighbor positions (see Reference[2] for further discussion of this optimization). What results is a desired next position for a platform that is approximately the best next position in terms of maximizing overall information flow through the network, taking into account the position of other platforms.

Figure 7 provides an illustration of the inner workings of the algorithm. Each platform uses a local estimate of the JMPD to compute a local information gain surface. This local surface for the three platforms shown in the figure is shown in the sub-figures. While qualitatively similar, the surfaces are slightly different, representing the different measurement sets available at each sensor. This local gradient of this information gain surface (evaluated at the platform location) is combined with physicomimetic forces derived from knowledge of neighboring platform positions to generate a desired movement.

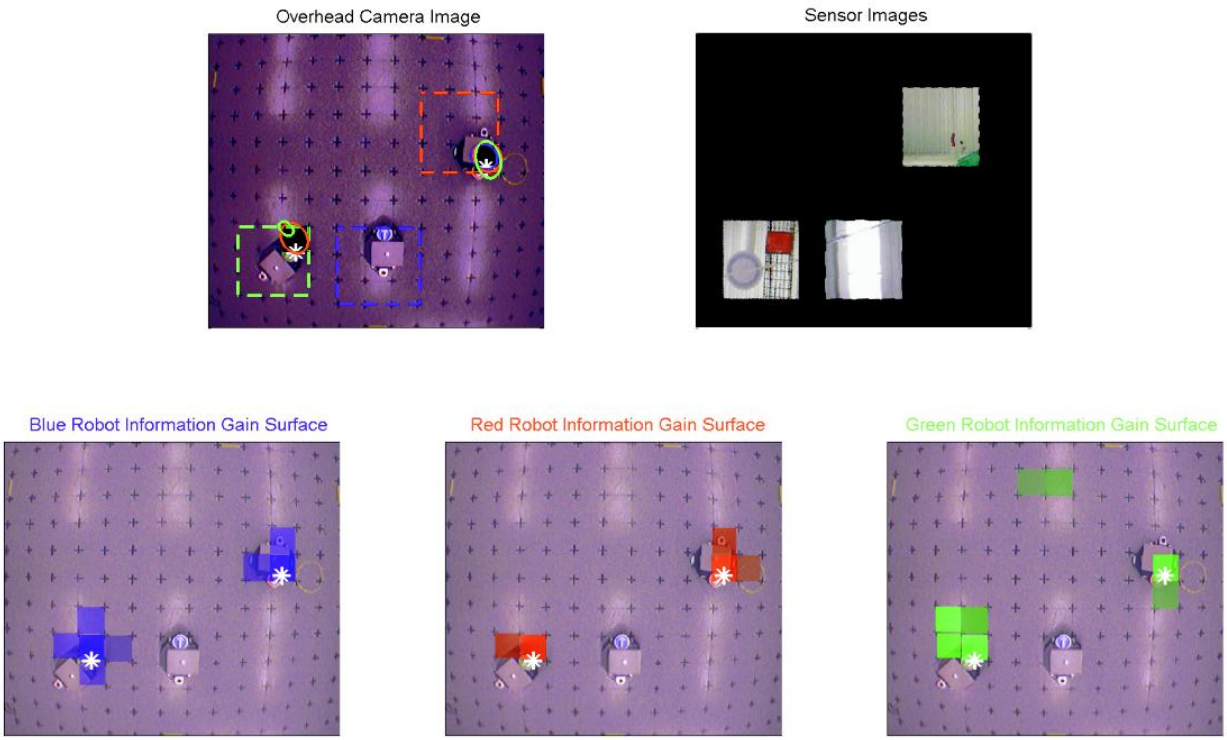

Figure 7. A single frame illustration of the demonstration. Top left: the overhead camera image showing the location of the three platforms. The FOV of each sensor is highlighted. Covariance ellipses are drawn to illustrate each platforms estimate of target location. Top right: The sensor images acquired by each platform. Bottom: The expected information gain surfaces computed by each of the three sensors. While qualitatively similar (as they should be as they are in neighbor to neighbor communication), each platform has a more precise estimate of information gain in its local area.

3.7. Robot Motion

The decision as to the next position of the robot is made using the information theoretic sensor management algorithms outline earlier. In short, a (local) potential field is computed which approximates the joint information theoretic optimization. The robot then moves in the direction of the gradient, thereby making the movement that maximizes expected gain in information. This computation is done on the laptop riding on the robot chassis. Decisions first take the form of desired next positions of the robot and are next translated a pair of movement commands (one indicating the angle the robot should turn and the second the distance to move forward once reaching that angle) using simple geometry. These commands are converted into units of encoder ticks (related to revolutions of each robot wheel) and send via serial cable to the robot controller.

Low-level robot motion is controlled by a Motorolla HC11 16-bit microprocessor. Software running on this processor receives movement commands in the form of requested encoder ticks for each wheel via serial cable with the laptop. This software than effects the required movement and provides feedback as to the successful completion to the laptop. Several challenges present themselves when attempting to execute movement commands. During forward movement, it is not sufficient that both wheels turn the same number of rotations, they must in fact turn at the same rate to ensure the robot moves straight. This requires the software to to constantly monitor the relative motion of the wheels and adjust voltage levels (actually PWM on-time). Furthermore, an analogous requirement is present during robot rotations. We have chosen to have the robot rotate by fixing one wheel and moving the other (this choice, where the robot does not rotate around its center, is compensated for in geometric calculations earlier).

There are a number of error sources that may cause the robot to not arrive in the exact position desired

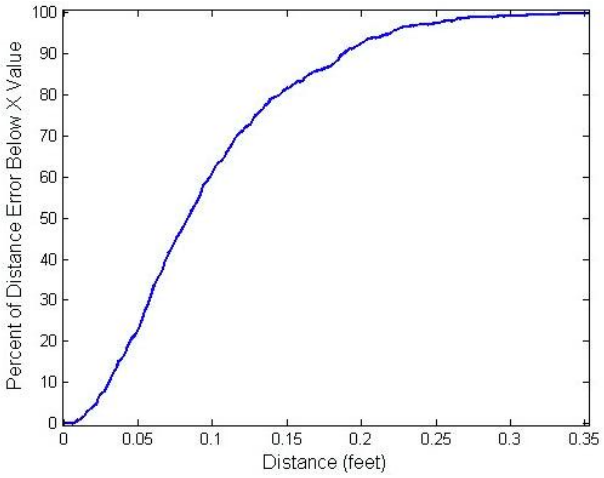

Figure 8. Empirical CDF of robot movement errors. For the movement sizes and rotations used in our experiments, the total position error is typically on the order of an inch.

by algorithm. First, battery charge and temperature effects the translation of movement in feet or angle to movement in encoder ticks. Second, robot momentum makes it difficult to precisely stop the robot after a certain number of encoder ticks (the control module may turn off power to the motors, but the robot still glides a bit). Third, the finite rate of the motor control processor makes counting encoder ticks less than perfectly precise, and hence the correction for non-uniform wheel spin rate is imperfect. This may result in slightly curved motion when straight motion is desired. Finally, there are a number of smaller effects, such as wheel slippage (one wheel spinning and registering encoder ticks, while not actually moving the robot) that happen in practice. All of these issues are addressed with algorithmic compensations, but cannot be expected to be exactly eliminated.

The algorithm, as designed, is able to tolerate errors in positioning very well. In fact, as each platform receives its actual position after movement is completed from the ImageGPS capability these errors are nearly transparent to the algorithm. Obviously, if the positioning errors are significant, the total performance of the algorithm will be worse than ideal. However, there is not a ripple effect of the error that leads to unstable behavior. If the robot moves to a position different than requested, the ImageGPS reports the true actual position and the next step in the information theoretic optimization computes the gradient at that new true point.

Despite the error sources outlined here, the ultimate performance of the motion algorithm is in fact very good. Figure 8 is an experimental CDF illustrating the position error (in inches), which shows the majority of the movements result in an error less than or equal to 2 inches. This is a combination of all of the error sources outlined above, including errors in turning the robot, keeping the robot moving straight, momentum and things like slippage.

4. CONCLUSIONS

This paper has described a laboratory demonstration of a distributed, decentralized, low-communication sensor management algorithm. The sensor management algorithm, described in detail elsewhere,[2] is a novel combination of particle filtering for predictive density estimation, information theory for action selection, and a physicomimetic relaxation for computational tractability. In simulation studies on a model problem reported previously,[1–3] this algorithm has shown order-of-magnitude type performance improvement over simple periodic scan. The work reported in this paper adds a level of fidelity that simulations alone cannot provide by illustrating the algorithm in the lab using a collection of robots detecting and tracking an unknown number of moving targets.

We have focused here on describing the engineering challenges associated with transitioning an algorithm which is successful in the simulation environment to the laboratory. These include the logistics of reporting own-

ship position, the mechanics of making a robot physically move from one position to another (and the associated errors) and communication between platforms. What has resulted is a successful demonstration of the control technology, providing a springboard for transition into a real field-able system.

REFERENCES

1. C. Kreucher, K. Kastella, and A. Hero, "Information-based sensor management for multitarget tracking," in *Signal and Data Processing of Small Targets 2003*, O. E. Drummond, ed., *Proc. SPIE* **5204**, pp. 480 – 489, August 3 - 8 2003.
2. C. Kreucher, K. Kastella, J. Wegrzyn, and B. Rickenbach, "An information based approach to decentralized multi-platform sensor management," in *Defense Transformation and Network-Centric Systems*, R. Suresh, ed., *Proc. SPIE* **6249**, pp. 134–145, April 17 - 21 2006.
3. C. Kreucher, K. Kastella, and A. Hero, "Sensor management using an active sensing approach," *Signal Processing* **85**, pp. 607 – 624, March 2005.
4. C. Kreucher, K. Kastella, and A. Hero, "Multi-platform information-based sensor management," in *Defense Transformation and Network-Centric Systems*, R. Suresh, ed., *Proc. SPIE* **5820**, pp. 141 – 151, March 28 - April 1 2005.
5. L. D. Stone, T. L. Corwin, and C. A. Barlow, *Bayesian Multiple Target Tracking*, Artech House, 1999.
6. K. J. Hintz and E. S. McVey, "Multi-process constrained estimation," *IEEE Transactions on Man, Systems, and Cybernetics* **21**, pp. 434–442, January/February 1991.
7. M. I. Miller, A. Srivastava, and U. Grenander, "Conditional mean estimation via jump-diffusion processes in multiple target tracking/recognition," *IEEE Transactions on Signal Processing* **43**(11), pp. 2678–2690, 1995.
8. S. Mori, C. Y. Shong, E. Tse, and R. P. Wishner, "Tracking and classifying multiple targets without a prior identification," *IEEE Transactions on Automatic Control* **AC31**(5), pp. 401–409, 1986.
9. R. E. Bethel and G. J. Paras, "A PDF multisensor multitarget tracker,," *IEEE Transactions on Aerospace and Electronic Systems* **34**, pp. 153–168, January 1998.
10. W. Spears, R. Heil, D. Spears, and D. Zarzhitsky, "Physicomimetics for mobile robot formations," *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-04)* **3**, pp. 1528–1529, 2004.
11. K. Kastella, "Joint multitarget probabilities for detection and tracking," *Proceedings of SPIE Acquisition, Tracking and Pointing XI* , 1997.
12. C. Kreucher, K. Kastella, and A. Hero, "Multitarget tracking using the joint multitarget probability density (general dynamics medal paper award winner, 2005)," *IEEE Transactions on Aerospace and Electronic Systems* **41**, pp. 1396–1414, October 2005.
13. C. Kreucher, A. Hero, and K. Kastella, "A comparison of task driven and information driven sensor management for target tracking," in *The 44th IEEE Conference on Decision and Control*, pp. 4004–4009, December 12-15 2005.
14. C. Kreucher, K. Kastella, and A. Hero, "Tracking multiple targets using a particle filter representation of the joint multitarget probability density," in *Signal and Data Processing of Small Targets 2003*, O. E. Drummond, ed., *Proc. SPIE* **5204**, pp. 258 – 269, August 3 - 8 2003.
15. P. Bidigare, C. Kreucher, and R. Conti, "A tracking approach to localization and synchronization in mobile ad-hoc sensor networks," in *Defense Transformation and Network-Centric Systems*, R. Suresh, ed., *Proc. SPIE* **6249**, pp. 146–157, April 17 - 21 2006.