Distributed Sensor Resource Management and Planning

Deepak Khosla, James Guillochon
HRL Laboratories LLC, Malibu, CA, USA

ABSTRACT

The goal of sensor resource management (SRM) is to allocate resources appropriately in order to gain as much information as possible about a system. In our previous paper, we introduced a centralized non-myopic planning algorithm, C-SPLAN, that uses sparse sampling to estimate the value of resource assignments. Sparse sampling is related to Monte Carlo simulation. In the SRM problem we consider, our network of sensors observes a set of tracks; each sensor can be set to operate in one of several modes and/or viewing geometries. Each mode incurs a different cost and provides different information about the tracks. Each track has a kinematic state and is of a certain class; the sensors can observe either or both of these, depending on their mode of operation. The goal is to maximize the overall rate of information gain, i.e. rate of improvement in kinematic tracking and classification accuracy of all tracks in the Area of Interest. We compared C-SPLAN's performance on several tracking and target identification problems to that of other algorithms. In this paper we extend our approach to a distributed framework and present the D-SPLAN algorithm. We compare the performance as well as computational and communications costs of C-SPLAN and D-SPLAN as well as near-term planners.

Keywords: Sensor management, resource management, sparse sampling, reinforcement learning, generative model, simulation, planning, distributed fusion, distributed management

1. INTRODUCTION

Sensor Resource Management (SRM) is "the control problem of allocating available sensor resources to obtain the best awareness of the situation."[3] SRM is important in terms of the benefits it provides over non-coordinated sensor operation. By automating the process, it reduces the operator workload. The operator defines the sensor tasking criteria instead of controlling multiple sensors individually by specifying each operation to be performed by each sensor. In an automated SRM system, the operator concentrates on the overall objective while the system works on the details of the sensor operations. Additionally, the feedback within the SRM system allows for faster adaptation to the changing environment. Problems that SRM has to deal with include insufficient sensor resources, highly dynamic environment, varied sensor capabilities/performance, failures and enemy interference, etc. Desired characteristics of a good sensor manager are that it should be goal oriented, adaptive, anticipatory, user friendly, require minimal interaction, account for sensor dissimilarities, perform in near real time, handle adaptive length planning horizons, etc.

Most SRM problems can be formulated as belonging to the class of Markov Decision Process (MDP) problems. In a MDP, future states are assumed to be the result of applying actions to the current state, ignoring the total history of the state space. In centralized methods, nodes with complete knowledge of the state space (Full-awareness nodes, or FANs) maintain the current state, and update the state with incoming measurements or the passage of time. This new state can now be used to determine future states. Similarly, in a decentralized system, each node's state is only dependent on that node's prior state, input from the environment, and process noise, just like any other Markovian process. Each node maintains its own individual picture of the state space, but since inter-node communication is imperfect, these state representations are often dramatically different from node to node.

Yet we must remain cautious: If communication times are non-zero, measurement information received from another node will have aged by an amount of time equal to the time required to send the measurement from one node to the other. This can be a problem if the state has already been advanced due to the passage of time.

For example: Take a state at t_0, and now advance this state to t_1. A measurement is now received from a sensor that was made at t_0. We would want to update the state at t_0 with the new measurement, and *then* update the state due to the passage of time. Since we do not know when packets will be received, this requires a set of states to be maintained in order to correctly predict future states. This violates our Markov conditions. One solution to this problem is to assume

Signal Processing, Sensor Fusion, and Target Recognition XVI, edited by Ivan Kadar,
Proc. of SPIE Vol. 6567, 65670B, (2007) · 0277-786X/07/$18 · doi: 10.1117/12.719929

that all sensor measurements are available cyclically in succession [30]. We avoid this problem entirely by assuming the communication time between sensors is negligible, and that the state space will not have changed dramatically in such a small time-frame. Therefore, only one state is maintained, and our MDP assumption holds.

In all sensor management systems, there will be a notion of system goal and system action. Both of these can be modeled and addressed using many methods, most of which fall into one of the following categories:

Control theory: one specifies the desired level of performance (or the reference trajectory) defining the management goal that the closed-loop system [38] tries to achieve. The error index is used as an action selection basis to reduce, and/or maintain as small as possible, any observable discrepancy.

Optimization: If sensor management is modeled as an optimization problem, rather than specifying a desired performance level, the user defines a cost function that, once optimized, leads to the most desirable outcome. This optimization would lead to the best trade-off between the sensing action payoff and the associated costs. Optimization-based algorithms are among the techniques that have been applied the most to the sensor management problem. Nash [39] uses linear programming to determine sensor-to-target assignment by using the trace of the Kalman filter error covariance matrices as objective function. Malhotra [40] uses Dynamic Programming for solving a Markov process that determines minimum costs based on the final state and works backwards. Washburn, *et al.* [1] present a sensor management approach based on dynamic programming to predict the effects of future sensor management decisions.

Decision theory: When a decision formulation is used, there is no clear notion of level of performance. As in the case of the optimization formulation, the objective is to choose the action that maximizes some expected utility function. Therefore, what is specified here is the utility of executing a given action in a given situation. The best solution is the one that offers the highest utility, *i.e.*, the best achievable performance. Solving such a decision problem can be done efficiently using graphical methods such as decision trees or influence diagrams. Performance objectives can be injected indirectly into decision trees (example Raytheon F-18 radar manager) as membership or belief functions of leaf nodes. Fung, *et al.*, [41] use a decision theoretic sensor management architecture based on Bayesian probability theory and influence diagrams. Manyika and Durrant-Whyte [42] use a decision theoretic approach to sensor management in decentralized data fusion while Gaskell and Probert [43] develop a sensor management framework for mobile robots. Molina López, *et al*., [44] present a sensor management scheme based on knowledge-based reasoning and fuzzy decision theory.

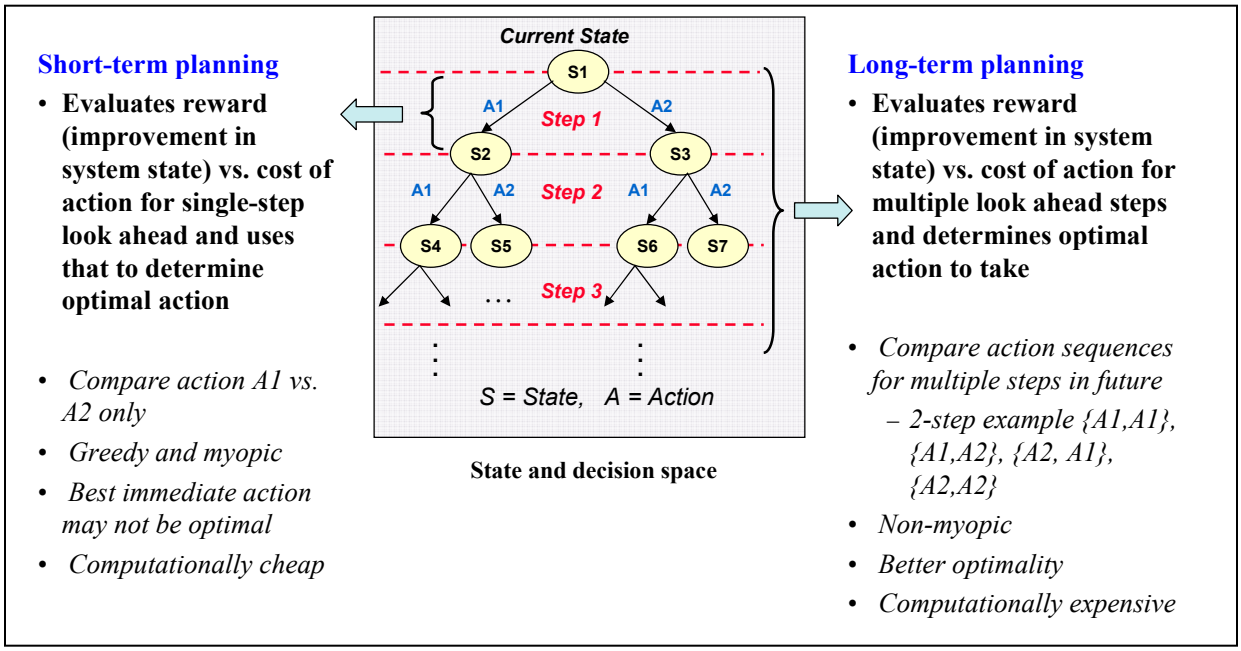
Use of information-theoretic measures such as entropy for sensor management has been around for many years now. Most of the literature is in the area of managing sensors to maximize kinematic information gain only [1-2]. Some prior art exists in managing sensors for maximizing ID and search as well [3-4]. One can use information theoretic criteria such as entropy, discrimination information, mutual information, etc. In these cases, the goal is to determine tasks that maximize the information gain at a system wide level and priorities are assigned based on this goal. Recent work on finite set statistics (FISST) and Joint Multitarget Probabilities (JMP - a subset of FISST) can also be applied in this context. The advantage of JMP is that there is a global notion of system tasks priority with all task priorities in a common space. Hintz and McVey [45] used entropy for search, track and ID tasks. Work in [46-53] use information measures such as entropy and discrimination gain for goals such as determining resolution level of a sensor, determining priority of search and track tasks, etc. Hintz *et al.* [9, 10] use the Shannon entropy, as we do in this paper, while Schmaedeke and Kastella [11] have chosen to use Kullback-Leibler (KL) divergence as measure of information gain. Mahler [55-57] proposed Finite set statistics (FISST) which reformulates the problem of multi-sensor, multi-target data fusion as if it were a single-sensor, single-target tracking problem. It is then posed as a problem in statistical optimal nonlinear control theory. Musick, et al. [58] applied Joint Multitarget Probabilities (JMP), a subset of FISST, to the problem of target detection and tracking.

We also must consider how to handle fusion and management for distributed systems. When dealing with substantially different information received from multiple sources at different times, an efficient and accurate fusion method is required. Zhang *et al*. [27] and Wang *et al.* [17] propose hierarchical decision fusion. As decisions are passed up a tree of intermediate levels, certain nodes are given higher weights than other nodes depending on the quality of the fused information. Stroupe *et al.* [16] apply distributed fusion to a network of independent robots, and explain how to properly fuse measurements received from different frames of reference. Thomopoulos *et al.* [18] provide a general proof that the optimal solution to distributed fusion amounts to a Neyman-Pearson test at the fusion center and a likelihood-ratio test at the individual sensors. A method proposed by Xiao *et al.* [19] involves each node updating its own data with a weighted average of its neighboring nodes. Qi *et al.* [20] detail a system of mobile agents which travel between local data sites by

transmitting themselves from one node to the next. This has the advantage of only having to transmit the agent itself, rather than the data, which could be several times larger in size. A discussion of the pros and cons of data and decision fusion is provided by Brooks *et al.,* D'Costa and Sayeed, and Clouqueuer *et al.* [21-23].

A variety of distributed management methods and issues have been explored as well. Durrant-Whyte *et al*. have primarily used information-theoretic approaches to address the problem [24-26, 42], using entropy measures to determine the relative values of possible actions. Xiong *et al*. [29] builds upon Durrant-Whyte's work and discuss a number of different approaches and issues when using information-centric methods in a distributed framework. An alternative is described by Ögren [28], who uses gradient methods to determine the best placement for a system of distributed sensors. This method involves setting up artificial potentials and letting these virtual force models converge to a global solution. Maholtra [8] addresses the impact of receiving measurements at different times, and how to properly model temporal effects.

When planning future actions, we can choose how far ahead in time to search when trying to pick the optimal action. Short-term (or *myopic*) approaches only look ahead one step; the goal of the planner is to pick the action that will yield the best result after the action is executed. Long-term methods explore many actions into the future. Their goal is to choose an action that will yield the best result after *d* actions have been performed. The differences between these two options is highlighted in the figure below.



Long-term approaches have the potential to produce the best results, but the computation time required for even simple environments is enormous when compared to near-term approaches. Several researchers have come up with solutions that provide approximate answers. An example is Krishnamurthy covering a technique involving a multi-arm bandit method that utilizes hidden Markov models. In [5] and [6], Krishnamurthy employs two approaches which limit the number of states accessible to each track to a finite number. While these approximations improve computation time, they are still very intensive. Bertsekas and Castanon [7] propose a method that uses heuristics to approximate a stochastic dynamic programming algorithm, while Malhotra [8] suggests using reinforcement learning.

The method we use in this paper to shrink the action space is known as *sparse planning*. Sparse planning considers a chain of actions up to a certain depth time when making a decision. The advantage it has over an exhaustive search is that it covers less and less of the action space as the algorithm looks farther ahead into the future. This makes sparse planning significantly faster than other long-term approaches that consider the tree in its entirety. In an exhaustive search, the belief state grows as $\text{classes}^{\text{decision points}}$. For example: If we have three possible classes, and we have five decisions to make before the depth time is reached, then the belief state will be $3^5 = 243$ entries long at the bottom of the action tree. In the context of sensor management, we can save processor time by making a finite number of measurements to determine a track's intermediate belief state as opposed to propagating the entire belief tree forward in

time. By updating the belief state at each intermediate state, we ensure that the belief state is a single entry long at all points on the tree. This produces a tree that is more top-heavy; a greater accuracy is provided at times closer to the decision point.

It is anticipated that the short-term algorithms should do better than long-term algorithms as the system of sensors begins to gather information, as they are designed to pick the action that yields the greatest gain at the present. However, we expect that the long-term planners should gain more information over the long-run and leave the system in a steady-state with a smaller entropy value. For small problem spaces, we expect that the differences will not be dramatic, but still measurable.

The distributed versions of the short-term and long-term algorithms are expected to perform slightly worse than their centralized counterparts, attributed to the fact that each sensor has an incomplete picture of the other sensors. This effect is explained in detail in section 5.

2. SENSOR RESOURCE MANAGEMENT

In the SRM problem we consider, our network of sensors observes a set of tracks; each sensor can be set to operate in one of several modes and/or viewing geometries. Each mode incurs a different cost and provides different information about the tracks. Each track has a kinematic state (that tracks, for example, position and velocity in two dimensions) and a discrete type; the sensors can observe either or both of these, depending on their mode of operation. The goal in this problem is to gain as much information as possible while incurring as little cost as possible. These goals are of course contradictory, and so a more precise wording is necessary: The goal in this problem is to maximize the average rate of information gain (i.e. total information gain divided by total cost).

2.1 Formal problem specification

An instance of the SRM consists of a set of S sensors, each of which has some number of actions (mode/geometry combinations) available to it, and a set of N tracks, each of which has an initial probability distribution over the C types and the K-dimensional kinematic states.

2.1.1 Track state (kinematic)

In the examples we consider, the kinematic state of each track is modeled by a linear dynamical system and tracked with a Kalman filter; however, in principle it could be modeled by any generative model whose state can be estimated from observational data. The dynamics of the linear dynamical system are governed by the following equations.

$$X_t = \Phi X_{t-1} + w_t \tag{1}$$

$$w_t \sim \mathrm{N}(0, Q) \tag{2}$$

Here, X_t[†] is the state of one of the tracks at time t. (If it is necessary to refer to the state of a particular track, i, a superscript will be added: X_t^i; the tracks are independent of each other) Φ and Q are parameters of the system, and N(m, Σ) denotes the multivariate normal distribution with mean vector m and covariance matrix Σ. If the track is observable at time t by sensor j (which depends on the state of the track and the action selected for the sensor), then a kinematic observation ($z_{t,j}$) will be generated according to

$$z_{t,j} = H_{t,j} X_t + v_{t,j} \tag{3}$$

$$v_{t,j} \sim N(0, R_{t,j}) \tag{4}$$

Here, $H_{t,j}$ determines what is measured by the sensor and $R_{t,j}$ is a measure of the accuracy of the measurement. We take Z_t to be the set of all the kinematic observations of a track at time t.

[†] Like all vectors in this report, this is a column vector.

Since X_t is unobservable, it must be estimated through the use of a Kalman filter [13]. The Kalman filter maintains a least-squares estimate $x(t|t) = \mathrm{E}[X_t \mid Z_1, \ldots, Z_t]$ and a covariance matrix $P(t|t) = \mathrm{E}[x(t|t)x^{\mathrm{T}}(t|t) \mid Z_1, \ldots, Z_t]$ of the error. This is recursively maintained through the following sets of equations:

$$x(t \mid t-1) = \Phi x(t-1 \mid t-1) \tag{5}$$

$$P(t \mid t-1) = \Phi P(t-1 \mid t-1)\Phi^{\mathrm{T}} + Q \tag{6}$$

$$P^{-1}(t \mid t) = P^{-1}(t \mid t-1) + \sum_{j=1}^{S} \chi_{t,j} H_{t,j}^{\mathrm{T}} R_{t,j}^{-1} H_{t,j} \tag{7}$$

$$x(t \mid t) = P(t \mid t)\left(P^{-1}(t \mid t-1)x(t \mid t-1) + \sum_{j=1}^{S} \chi_{t,j} H_{t,j}^{\mathrm{T}} R_{t,j}^{-1} z_{t,j} \right) \tag{8}$$

where $\chi_{t,j}$ is an indicator variable that is 1 when sensor j produces a kinematic observation of the track at time t and 0 otherwise.

2.1.2 Track state (identification)

As with the kinematic state, the identification of the track can be reasoned about in a number of ways; we apply Bayesian reasoning to the problem. We model the sensors using confusion matrices; the klth element of $\Theta_{t,j}$ gives the probability at time t that sensor j reports the track as type k when it is type l. The uncertainty is modeled as a multinomial distribution; the kth element of the belief state $b(t)$ is the belief (i.e. probability) at time t that the track is type k, given all the observations that have come up to (and including) time t. If the track is observable at time t by sensor j, then an identification observation ($o_{t,j}$) will be produced. We take O_t to be the set of all the identification observations of a track at time t.

Let $\Theta(o,t,j)$ be the diagonal matrix whose kth element is the probability that sensor j would produce observation o at time t given that the track is of type k (i.e., the diagonal of this matrix is the oth row of $\Theta_{t,j}$). Then the belief state can be updated with the following equation:

$$b(t+1) = \left(\prod_{j=1}^{S} \Theta(o,t,j)^{\kappa_{t,j}} \right) \frac{b(t)}{\Gamma} \tag{9}$$

where $\kappa_{t,j}$ is an indicator variable that is 1 when sensor j produces an identification observation of the track at time t and 0 otherwise, and Γ is a normalizing constant (the elements of $b(t+1)$ must add to 1).

2.1.3 Information measure

The measure defined here is used to judge how much information is gained when transitioning from one state to another. To do this, we must measure the information gained about the (discrete) type as well as the (continuous) kinematic state, and weigh them against each other. To measure information gained about the type, we turn to the Shannon entropy:

$$h_S(b(t)) = -\sum_{k=1}^{C} b_k(t) \lg b_k(t) \tag{10}$$

The identification information gain is then $h_S(b(t)) - h_S(b(t+1))$. Similarly, the entropy in the kinematic state can be measured by the differential entropy of a normal distribution:

$$h_D(P(t \mid t)) = -\frac{1}{2} \ln\left((2\pi e)^K \det(P(t \mid t)) \right) \tag{11}$$

Here, $\det(P)$ is the determinant of P; recall also that K is the dimension of the kinematic state of a track. As before, the kinematic information gain is given by $h_D(P(t|t)) - h_D(P(t+1|t+1))$. In both the discrete case and the continuous case, it can be shown that the combined entropy of multiple tracks' state estimates is the sum of their individual entropies (assuming,

as we do, that the estimates for each track are independent of each other). In order to get an overall measure of information gain, we combine the information gains, kinematic and identification, of all the tracks as follows:

$$\Delta h(t) = \sum_{i=1}^{N} \left(h_S(b^i(t)) - h_S(b^i(t+1)) \right) + \alpha \sum_{i=1}^{N} \left(h_D(P^i(t \mid t)) - h_D(P^i(t+1 \mid t+1)) \right) \tag{12}$$

The parameter α can be used to trade off the importance of kinematic information gain and identification information gain. In all of our experiments, we take α=1.

Each action (i.e. assignment of modes and viewing geometries to settings) has a deterministic cost that is known *a priori*; the cost at time t is written as c_t. The rate of information gain at time t is thus given by

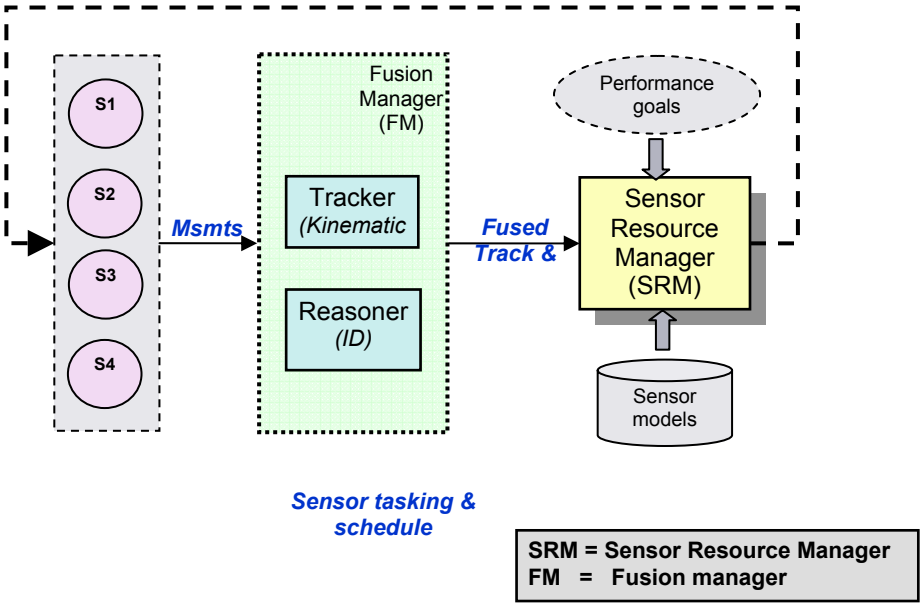
$$RIG(t) = \frac{\Delta h(t)}{c_t} \tag{13}$$

3. ARCHITECTURES FOR SENSOR MANAGEMENT

There are several choices to be made in regards to overall network architecture. There are two types of nodes in a network: *Sources*, which supply new information, and *sinks*, which collect information and process it. In centralized architectures, sensors primarily function as sources, while FANs act as sinks. As we move toward distributing the workload, more and more of the FAN functionality is off-loaded to the individual sensors. For fully-distributed architectures, no FANs are used. Each sensor is equally a source and a sink.

All of the architectures we explore in this paper utilize centralized fusion, so we shall only be examining the difference between centralized and distributed management. While the effects of having multiple fusion centers are interesting and warrant investigation, we will examine the issue in a future paper.

3.1 Centralized Resource Management with Centralized Fusion

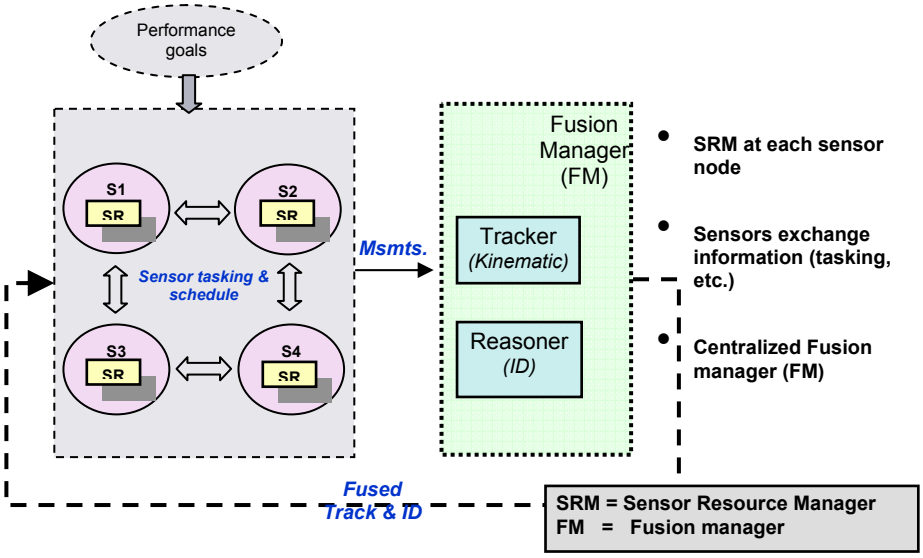


This is usually described as a fully-centralized sensor network. Sensors are completely "dumb," their sole task is the produce measurements, which they transmit to the FAN. The FAN processes the measurements received from each

sensor, refines the current state with those measurements, and then determines the best course of action. Commands are then relayed to the sensors, and the process repeats itself.

This method is guaranteed to produce the best results given any set of measurements among all of the available architectures. However, since all of the functionality is handled by a single node, it is extremely vulnerable to catastrophic failure. It allows suffers from computational burdens, detailed in section 6.

3.2 Distributed Resource Management with Centralized Fusion



In this architecture, the sensors are responsible for determining their next actions, while measurements are still fused together by a FAN. This has computational benefits over a completely centralized architecture. In centralized resource management, the full-awareness node has to consider the joint-action space of all sensors at each decision point. Since this action space grows as N^2, this process becomes computationally impossible for large N. In distributed resource management, sensors can utilize a myriad of decision schema when determining the best course of action. Voting schemes are popular; we use a fairly simple voting mechanism in D-SPLAN.

However, we still use a centralized fusion method in this particular architecture. The advantage of this is that all sensors will be privy to the same global state, and decisions made by the sensors will all be based on the same base of information. However, since this architecture is still reliant on a FAN, it suffers from the same weakness as a fully-centralized network: A failure of the FAN results in a failure of the entire network.

4. CENTRALIZED PLANNERS

4.1 Centralized Short-Term Planner (ING)

One simple and often effective planning algorithm that emerges from the goal statement given above ("maximize the average rate of information gain") is the information needs gain (ING) algorithm. The ING planner is myopic: It maximizes instantaneous expected information gain. To do this, it evaluates each assignment of actions to sensors (such an assignment is called a joint action) by computing the expected identification information gain and kinematic information gain for each track given the joint action.

4.2 Centralized Long-Term Planners (C-SPLAN)

ING addresses short-term planning, i.e., sensor tasking to maximize desired performance goals over short look-ahead horizon [12]. Thus it is greedy or myopic as it aims at optimizing immediate future performance. In many cases, such myopic planning can lead to sub-optimal performance over the long term. For example, if a certain target will be obscured in the near-future, taking that into consideration now during sensor tasking may result in overall better sensor tasking policy and tracking accuracy. We briefly describe our centralized sparse planner, C-SPLAN, below.

4.3 C-SPLAN

As described in our previous paper, we approximate the E[R_t] with the expectation:

$$V_t = \frac{E[r_t] + \gamma E[r_{t+1} + \gamma r_{t+2} + \ldots]}{E[c_t] + \gamma \frac{E[r_{t+1} + \gamma r_{t+2} + \ldots]}{V_{t+1}}} \tag{14}$$

This approximation is exact when the sequence of actions is deterministic and independent of intervening observations. Of course, our algorithm (described in pseudocode in Figure 1) further approximates by substituting sample averages for the expectations. Other than the new reward concept, the other difference between our algorithm and the basic sparse sampling algorithm is that, instead of searching a fixed number of decisions into the future, C-SPLAN searches until the total cost of actions reaches a given cost horizon. If the final action in the search would take the total cost beyond the desired horizon, its reward and cost are scaled.

Heuristic Action Restriction Unfortunately, this algorithm has an exponential running time; the generative model is called $O((wA)^d)$ times by the basic algorithm (letting A be the number of actions and ignoring the effect of reducing w by γ^2 at each level). We can thus reduce the running time significantly by restricting the size of the action space. We use two simple thresholding methods to heuristically reduce the action set. Both methods require that the one-step expected rate-of-gain be computed for each action, as in the near-term planner. In *n-best thresholding*, only the n actions with the highest one-step expected rates-of-gain are considered. This reduces the number of calls to the generative model to $O((wn)^d)$. If n is 1, this is nearly equivalent to the near-term planner (if there is a tie at the top-level, it is broken by the sparse sampling estimate of the Q value rather than arbitrarily as in the near-term planner). In α *thresholding*, the α parameter controls which actions are considered. Let *max* be the maximum over all the actions of the expected one-step value, i.e. the expected rate-of-gain of the near-term planner. If *max* > 0, those actions with expected one-step values at least $\alpha \times max$ are considered; otherwise, the actions with expected one-step values of at least max/α are considered. The effect of α upon the running time is highly problem-dependent.

```
C-SPLAN (γ, w, d, b)
inputs: discount factor γ, sample width w, cost horizon d, initial state s
outputs: the action to be taken (best) and the estimated discounted rate of that action (V), and the estimated
discounted reward over the cost horizon of that action (R)
        if d ≤ 0 then return [nil, 0]

        for each action a:
                let avg[a] = 0
                let ravg[a] = 0
                repeat w times:
                        let [o, r, c] = gen_model(s, a)
                        if (d-c) > 0 then
                                let s′ = state_update(s, a, o)
                                let [a′, v′, r′] = sparse_sampling_planner(γ, ⌈γ²w⌉, d-c, s′)
                                let c′ = r′ / v′
                                let avg[a] = avg[a] + (r + γr′) / ((c+γc′)w)
                                let ravg[a] = ravg[a] + (r + γr′) / w
                        else
                                let avg[a] = avg[a] + r / (cw)
                                let avg[a] = avg[a] + r / w

        let best = argmax_a avg[a]
        return [best, avg[best], ravg[best]]
```

Figure 1. Pseudocode for the sparse planner.

5. DISTRIBUTED PLANNERS

The planning algorithms presented so far have been centralized planners; all of them assume a single decision-maker that is privy to all of the information in the sensor network. However, decentralized planning algorithms have much to recommend them. Though they usually cannot perform as well as their centralized counterparts, they usually require less communication overhead and can be more resilient to the failure of a sensor in the network. We consider three distributed planning algorithms.

5.1. Short-term distributed planner

In the coordinated planner, the joint action is chosen over the course of S rounds. Before the first round, each sensor is marked "undecided." In each round, the undecided sensors compute the expected rate of information gain for each of its actions, *combined with the actions of the decided sensors*, given the current state estimate, ignoring the effects of the other undecided sensors. Each sensor communicates the value of its best action to the other undecided sensors; the sensor with the highest value is marked as decided and must then communicate information about its selected action to the undecided sensors before the next round may start. This is a direct implementation of the distributed resource management with centralized fusion architecture, as described in section 3.2. See the pseudocode in Figure 2. While this requires more computation than the distributed near-term planner with no coordination, it is still more considerably efficient than the centralized near-term planner, even before considering the gain from parallelizing the computation.

```
DISTRIBUTED SHORT-TERM PLANNER (state)
inputs: initial state state
outputs: an assignment of an action to each sensor action
        let undecided = nil
        for each sensor s
                let undecided = undecided ∪ {s}
                let action[s] = nil

        while | undecided | > 0
                for each s in undecided
                        for each a in get_available_actions(s, state, action)
                                let action[s] = a
                                let g[s, a] = expected_rate_of_information_gain(state, action)
                        let action[s] = nil
                let [s′, a′] = argmax_{s,a} g[s, a]
                let action[s′] = a′
                let undecided = undecided – {s′}
return action
```

Figure 2. Pseudocode of the distributed near-term planner. The function `get_available_actions`(*s*, *state*, *action*) returns a set of actions available for sensor *s* from the state *state* and given the action assignments of *action*.

Unfortunately, this planner requires considerable communication between the sensors. Additionally, it is myopic in two different ways. First, it is myopic in the intra-action scope; by picking the sensor giving the highest value first (and the action that maximizes the value for each sensor), the planner may be overlooking a better joint action. Second, it is myopic in the *inter*-action scope; it doesn't consider future actions at all. This last problem is partially addressed by the coordinated sparse planner.

5.2. Long-term distributed sparse planner (D-SPLAN)

The coordinated sparse planner combines one round of independent sparse planning with the coordinated near-term planner. That is, the first round of the coordinated near-term planner is replaced by a non-myopic procedure. Each sensor applies the sparse planning algorithm as if it were the only sensor; the sparse planner's value estimate is sent to the other sensors, and the one whose value is highest is marked as decided. It communicates information about its selected action

to the undecided sensors, whose actions are then selected by the coordinated near-term planner. The first round is more computationally complex than the first round of the coordinated near-term planner, but the two planners have the same communication requirements.

A potential issue with this approach is the uncertainty of the future state. Long-term strategies already suffer from unforeseen events such as the discovery of a new track, the loss of a known track, or drastic changes in the environment. In a perfect world, the parameters would remain unchanged, and the best possible action that is predicted for t_1 at t_0 would be identical to the action predicted for t_1 at t_1. However, due to the often dynamic nature of the problem, this is rarely the case. Thus we expect that planners with less-than-perfect scenario prediction qualities will perform weakly in highly dynamic environments. Now, if we couple that weakness with imperfect information in the *intra*-action scope as well, we are left with a strategy that could potentially perform quite poorly.

The scenarios we have devised are all highly *predictable*; there is no track association, blind spots are known to the planners in advance, and the environment is static. Therefore, we expect that these effects will not be present in the results presented later in this document.

6. COMPUTATIONAL AND COMMUNICATION ANALYSIS

6.1 Computational complexity

6.1.1 Centralized Fusion

The computational cost of centralized fusion is minimal. The sensors only need to produce the measurements and send those values to the FAN, which requires minimal processing power. The FAN has the task of fusing the new measurements with the previous state estimate. This can become intensive if the number of sensors is great or if the number of measurements returned per second is large. We want to estimate the number of floating point operations required for the equations listed above. K represents the number of vectors in the Kalman state, and M represents the number of variables in the measurement vector (M must be $\leq K$, since we cannot measure more state variables than there are). For the prediction step, eqs. (1) and (2), the FAN must use

$$3K^3 \tag{15}$$

flops. The update step requires

$$2K^3 + 6MK^2 + 4KM^2 + M^3 - (K^2 + 3MK) + (3C - 1) \tag{16}$$

flops per measurement, for a grand total of

$$3K^3 + N(2K^3 + 6MK^2 + 4KM^2 + M^3 + 3C - (K^2 + 3MK + 1)) \tag{17}$$

flops per cycle per sensor, where N is the number of sensors.

6.1.2 Centralized Management

In contrast, centralized management can get very computationally costly. Short-term methods have a computational complexity proportional to

$$A^N, \tag{18}$$

where A is the number of actions at each sensor. C-SPLAN has a computational complexity proportional to

$$A^{ND}\gamma^{D(D-1)}, \tag{19}$$

where D is the search tree depth and γ is the discount factor.

6.1.3 Distributed Management

The complexity of short-term distributed management is proportional to

$$N!A, \tag{20}$$

while D-SPLAN has a computational complexity proportional to

$$N!A^D\gamma^{D(D-1)}. \tag{21}$$

The difference between these algorithms and centralized management is the dependence on N. If A^D is greater than or equal to the number of sensors, it is clear that $N^N > N!$ for all N. Thus, we expect D-SPLAN to be less computationally intensive than C-SPLAN in most scenarios.

6.2 Communication Requirements

6.2.1 Centralized Fusion

Power requirements for communications in a centralized architecture can be low by design. Since the sensors only have to communicate with one other node (the FAN), their transmissions can be directionalized, requiring less power than a isotropic broadcast to reach the same distance. Bandwidth requirements can be rather steep since constant communication is a must in a centralized architecture. The z_k vector is dimension M, while the R_k matrix is MxM in size, so sending a measurement to the FAN requires

$$4(M^2 + M + C) \tag{22}$$

bytes of information (C is the length of the belief state), assuming 32-bit precision. For example, a measurement of a track's position and velocity in three dimensions with three possible classifications requires a measurement size of 6 (M = 6, C = 3), giving a packet size of 180 bytes. If a sensor makes a measurement every 100 ms, and there are 10 sensors, each sensor must be capable of sending 1800 bytes/sec (14.4 kbps), and the FAN must be able to handle 18 kilobytes/sec (144 kbps) of incoming data transfer. After the data has been fused, the track states are sent back to the sensors, which requires

$$4(K^2 + K + C) \tag{23}$$

bytes per transmission. Assuming K = 6, this requires another 450 bytes/sec. This means that in total,

$$4(M^2 + M + K^2 + K + 2C) \tag{24}$$

bytes of information is either sent or received by each sensor. Given these parameters, each sensor has to be able to handle 3600 bytes/sec (28.8 kbps) of incoming/outgoing traffic, with the FAN handling 36 kilobytes/sec (288 kbps).

6.2.2. Centralized Management

In all architectures, we assume that each node has apriori knowledge of the workings of all its neighboring nodes. Therefore, no sensor model information is exchanged at runtime. In centralized management, the only management packets being sent originate from the FAN, which simply supplies each sensor with an action at each time-step. Therefore, the communication cost is simply

$$4N \tag{25}$$

bytes.

6.2.3 Distributed Management

In D-SPLAN, the sensors come to a consensus as to which sensor has the best individual action, and label that sensor as "decided." This process requires some additional communication exchange, requiring

$$4(N!+ N - 1) \tag{26}$$

bytes to be sent at each decision point.

7. EXPERIMENTAL RESULTS

Our experimental setup is a bit more involved than in our previous paper, where we only compared the centralized algorithms. Previously, we performed experiments using two simple test environments which we refer to as "all visible" and "blind spot." In both environments, each object (track) has a four-dimensional kinematic state (comprising the position and velocity of the track along two dimensions in space). In addition, each track can be one of three classes, with a uniform prior distribution over these types. Sensors in both environments pick a track to observe at each decision point, and each sensor picks one of several modes available to it. In the "all visible" environment, all tracks are visible at all times, while the "blind spot" has one or more of the tracks pass through an area where they are not visible to any of the sensors.

The difficulty in using this same setup when comparing the centralized and distributed algorithms is that the performance differences are subtle in simple environments. To better highlight the difference in performance, the scenarios now include 10 tracks, and the "blind spot" scenario has been altered to include several areas that the sensors cannot observe.

As we mentioned, the scenarios consist of 10 moving objects, moving with constant velocity in a random direction in the XY plane. Each object is assigned one of three identification classes. The class of each object is modeled as a 3-class vector {C1, C2, C3}. The position and class of these objects can be measured by two sensors. Each sensor can either look "left" or "right." In "left" mode, the sensors can only view tracks whose x-coordinate is less than 0, and in "right" mode, only tracks with x-coordinates greater than 0 are visible. Each sensor can operate in a kinematic measurement mode (Mode 1) and a class measurement mode (Mode 2). We realize that this a simplistic sensor model, but is chosen to demonstrate the sensor management algorithms for maximizing both kinematic and classification accuracies simultaneously. The measurement accuracies and cost of these modes is as follows:

For sensor 1, mode 1 provides position measurement with a variance of 0.01m^2 and a cost of 2 units. Mode 2 provides class information with a cost of 1 unit. This mode is modeled as a confusion matrix with P_c (Probability of correct class declaration) = 70% and P_f (Probability of incorrect class declaration) = 30% (with equal errors on the each of the 2 incorrect classes, i.e., 15%) For sensor 2, the modes have the same measurement accuracies as sensor 1. However the costs are reversed, i.e., mode 1 has a cost of 1 unit while mode 2 has a cost of 2 units.

Belief state updates are randomly chosen on the fly, we're *not* using a pre-generated, finite-length random table. For the sparse parameters, we use a discount factor of 0.5, and a depth of 2 for both the centralized and distributed sparse planners. Results are averaged over 1000 Monte Carlo runs and presented in Figure 7 below.

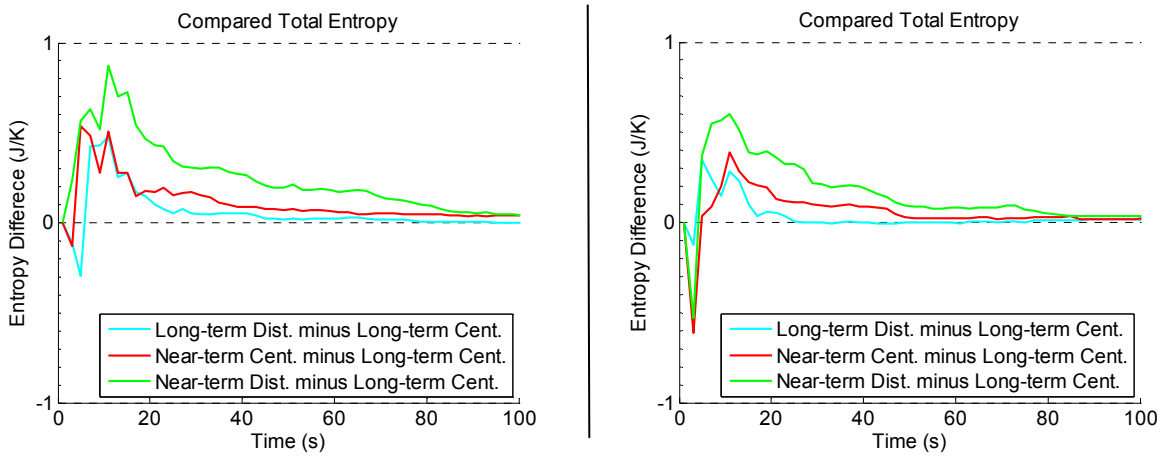


Figure 7. Total entropy values compared between planners. The left side shows "all visible" and the right side shows "blind spot" results.

The sparse planners perform notably better than their near-term counterparts in both scenarios, as we found in our previous paper. Both scenarios show that the long-term planners perform poorly in the first couple seconds. This is expected; the long-term sensors are not attempting to maximize their productivity in the long-term. It is not necessarily true that the action that leads to the lowest entropy value at t = 1 will optimally minimize the entropy when t = 2. After the planners have had a chance to settle, the long term planners perform better than their near-term counterparts. The

magnitude of this difference is small due to limited depth (we only looked ahead 2 actions), especially when compared to difference between picking actions randomly and running any of the planners.

On the average, near-term centralized performs slightly better than long-term distributed in both scenarios. This shows that the increase in search depth gives an advantage over a near-sighted methods, even when the search space does not consider all joint actions. This is a welcome result, since we showed in section 6 that the computational burden of D-SPLAN is far smaller than the burden imposed by C-SPLAN. We also can safely conclude that the performance degradation postulated in section 5.3 is not observed in these results; which is as expected.

The long-term sparse planner has the best overall performance after the first couple seconds. A potentially surprising result is that the two scenarios have almost identical outcomes, with some slight variations that might be attributable to stochastic effects. It is possible that a more pronounced difference would be apparent if the tree-depth or the number of runs was increased. Unfortunately, computation times for C-SPLAN become prohibitive at depth values much larger than 2, so a direct comparison is impractical.

8. CONCLUSION

This paper has investigated the benefits of distributed coordinated long-term sensor management and scheduling. The results presented in the previous section demonstrate that this method performs similarly to the centralized long-term method which considers the entire action space, including all joint actions. Since our coordinated architecture is markedly less costly, enjoys the resiliency of a distributed network, and yields similar performance to competing long-term planners, it stands out among its peers.

REFERENCES

1. W. Schmaedeke, "Information Based Sensor Management," *Signal Processing, Sensor Fusion, and Target Recognition II. Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 1955, Orlando, FL, April 12-14 1993, pp. 156-64.
2. W. Schmaedeke and K. Kastella, "Information Based Sensor Management and IMMKF," *Signal and Data Processing of Small Targets 1998: Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 3373, Orlando, FL, April 1998, pp. 390-401.
3. K. Kastella, "Discrimination Gain to Optimize Detection and Classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 27, no. 1, pp. 112-116, January 1997.
4. G.A. McIntyre and K.J. Hintz, "An Information Theoretic Approach to Sensor Scheduling," *Signal Processing, Sensor Fusion, and Target Recognition V. Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 2755, Orlando, FL, April 8-10 1996, pp. 304-312.
5. V. Krishnamurthy, "Algorithms for optimal scheduling and management of hidden Markov model sensors," *IEEE Trans. Signal Process.* 50 (6) (2002) 1382-1397.
6. V. Krishnamurthy, D. Evans, "Hidden Markov model multiarm bandits: a methodology for beam scheduling in multitarget tracking," *IEEE Trans. Signal Process*. 49 (12) (2001) 2893-2908.
7. D.P. Bertsekas, D. Castanon, "Rollout algorithms for stochastic scheduling problems," *J. Heuristics*, 5 (1) (1999) 89-108.
8. R. Malhotra, "Temporal considerations in sensors management," *Proceedings of the IEEE 1995 National Aerospace and Electronics Conference, NAECON, vol. 1,* Dayton, OH, 22-26 May 1995, pp. 86-93.
9. K.J. Hintz, "A measure of the information gain attributable to cueing," *IEEE Signal Process. Mag. (Special Issue on Math. Imaging)* 19 (5) (2002) 85-95.
10. K.J. Hintz, E. S. McVey, "Multi-process constrained estimation," *IEEE Trans. Man Systems Cybernet.* 21 (1991) 237-244.
11. W. Schmaedeke, K. Kastella, "Event-averaged maximum likelihood estimation and information-based sensor management," *Proceedings of SPIE, vol. 2232,* Orlando, FL, 1994, pp. 91-96.

12. M.K. Schneider., G.L. Mealy, and F.M. Pait, "Closing the Loop in Sensor Fusion Systems: Stochastic Dynamic Programming Approaches," *Proceedings of the 2004 American Control Conference Volume 5*, pp. 4752–4757, 2004.
13. R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME—Journal of Basic Engineering*, 82(D), 35–45, 1960.
14. M.J. Kearns, Y. Mansour, and A.Y. Ng, "A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes," *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, T. Dean (Ed.), pp. 1324–1331, Morgan Kaufmann, 1999.
15. M.J. Kearns, Y. Mansour, and A.Y. Ng, "A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes," *Machine Learning*, 49(2–3), pp. 193–208, 2002.
16. A.W. Stroupe, M.C. Martin, T. Balch, "Distributed sensor fusion for object position estimation by multi-robot systems," *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on* , vol.2, no.pp. 1092- 1098 vol.2, 2001
17. X. Wang, G. Foliente, Z. Su, L. Ye, "Multilevel Decision Fusion in a Distributed Active Sensor Network for Structural Damage Detection," *Structural Health Monitoring* 2006 5: 45-58
18. S. Thomopoulos, R. Viswanathan, D. Bougoulias, L. Zhang, "Optimal And Suboptimal Distributed Decision Fusion," *1988 American Control Conference*, 7th, Atlanta, Ga, United States, 15-17 June 1988. Pp. 414-418. 1988
19. L. Xiao, S. Boyd, S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on* , vol., pp. 63-70, 15 April 2005
20. H. Qi, X. Wang, S.S. Iyengar, K. Chakrabarty, "Multisensor Data Fusion in Distributed Sensor Networks Using Mobile Agents," *Proceedings of 5th International Conference on Information Fusion*. Annapolis, MD, 2005
21. R.R. Brooks, P. Ramanthan, A.M. Sayeed, "Distributed Target Classification and Tracking in Sensor Networks," *Signal Processing Magazine, IEEE*, vol. 19, no. 2, pp. 17-29, 2002.
22. A. D'Costa, A.M. Sayeed, "Data Versus Decision Fusion for Distributed Classification in Sensor Networks," *Military Communications Conference, 2003. MILCOM 2003. IEEE,* vol. 1, pp. 585-590, 2003.
23. T. Clouqueur, P. Ramanthan, K.K. Saluja, K. Wang, "Value-Fusion versus Decision-Fusion for Fault-tolerance in Collaborative Target Detection in Sensor Networks," *Proceedings of Fourth International Conference on Information Fusion, Aug,* 2001.
24. A. Makarenko, H.F. Durrant-Whyte, "Decentralized Data Fusion and Control in Active Sensor Networks," *7th Int. Conf. on Info. Fusion (Fusion'04)*
25. J.M. Manyika, H.F. Durrant-Whyte, "An Information-theoretic Approach to Management in Decentralized Data Fusion," *Proceedings of SPIE,* vol. 202, 1992.
26. B. Grocholsky, H.F. Durrant-Whyte, P. Gibbens, "An Information-Theoretic Approach to Decentralized Control of Multiple Autonomous Flight Vehicles," *Proceedings of SPIE,* vol. 4196, pp. 348-359, 2000.
27. J. Zhang, E.C. Kulasekere, K. Premaratne, "Resource Management of Task Oriented Distributed Sensor Networks," *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on,* vol. 3, pp. 513-516, 2001.
28. P. Ögren, E. Fiorelli, N.E. Leonard, "Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment," *Automatic Control, IEEE Transactions on,* vol. 49, no. 8, pp. 1292-1302, 2004.
29. N. Xiong, P. Svensson, "Multi-sensor Management for Information Fusion: Issues and Approaches," *Information Fusion,* vol. 3, no. 2, pp. 163-186, 2002.
30. H.R. Hashemipour, S. Roy, A.J. Laub, "Decentralized Structures for Parallel Kalman Filtering," *Automatic Control, IEEE Transactions on*, vol. 33, no. 1, pp. 88-94, 1988.
31. L.Y. Pao, N.T. Baltz, "Control of Sensor information in Distributed Multisensor Systems," *American Control Conference,* vol. 4, pp. 2397-2401, 1999.
32. H. Wang, K. Yao, D. Estrin, "Information-theoretic Approaches for Sensor Selection and Placement in Sensor Networks for Target Localization and Tracking," *Journal of Communications and Networks,* vol. 7, no. 4, pp. 438-449, December 2005.
33. M. Chu, H. Haussecker, F. Zhao, "Scalable Information-Driven Sensor Querying and Routing for ad hoc Heterogeneous Sensor Networks," *International Journal of High Performance Computing Applications*, vol. 16, no. 3, pp. 293-313, 2002.
34. F. Zhao, J. Shin, J. Reich, "Information-Driven Dynamic Sensor Collaboration for Tracking Applications," *IEEE Signal Processing Magazine,* vol. 19, no. 2, pp. 61-72, 2002.

35. B. Horling, R. Mailler, M. Sims, V. Lesser, "Using and Maintaining Organization in a Large-Scale Sensor Network," *Proc. of Workshop on Autonomy, Delegation, and Control,* 2003.
36. C. Intanagonwiwat, R. Govindan, D. Estrin, "Direction Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking,* pp. 56-67, 2000.
37. M. Rabbat, R. Nowak, "Distributed Optimization in Sensor Networks," *Proceedings of the third international symposium on Information processing in sensor networks,* pp. 20-27, 2004.
38. A Savkin, R. Evans, E. Skafidas, "The Problem of Optimal Robust Sensor Scheduling" vol. 1, 3791 Paper number 1250 , *CDC2000*
39. J.M. Nash, "Optimal Allocation of Tracking Resources," *Proceedings of the 1977 IEEE Conference on Decision and Control*, vol. 1, New Orleans, LA, December 7-9 1977, pp. 1177-1180
40. R. Malhotra, "Temporal Considerations in Sensor Management," *Proceedings of the IEEE 1995 National Aerospace and Electronics Conference, NAECON 1995*, vol. 1, Dayton, OH, May 22-26 1995, pp. 86-93.
41. R. Fung, E. Horvitz, and P. Rothman, "Decision Theoretic Approaches to Sensor Management," DTIC# AB B172227, Wright-Patterson AFB, OH, Feb 1993.
42. J.M. Manyika and H. Durrant-Whyte, "On Sensor Management in Decentralized Data Fusion," *Proceedings of the 31st Conference on Decision and Control*, vol. 4, Tucson, AZ, December 16-18 1992, pp. 3506-3507.
43. A. Gaskell and P. Probert, "Sensor Models and a Framework for Sensor Management," *Sensor Fusion VI. Proceedings of the SPIE - The International Society for Optical Engineering,* vol. 2059, Boston, MA, September 7-8 1993, pp. 2-13.
44. J.M. Molina López, F.J. Jiménez Rodríguez, and J. R. Casar Corredera, "Fuzzy Reasoning For Multisensor Management," *1995 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 2, Vancouver, British Columbia, Canada, October 22-25 1995, pp. 1398-1403.
45. K.J. Hintz and E.S. McVey, "Multi-Process Constrained Estimation," *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 21, no. 1, pp. 434-442, Jan/Feb 1991.
46. W. Schmaedeke, "Information Based Sensor Management," *Signal Processing, Sensor Fusion, and Target Recognition II. Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 1955, Orlando, FL, April 12-14 1993, pp. 156-64.
47. K.J. Hintz, "A Measure of the Information Gain Attributable to Cueing," *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 21, no. 2, pp. 237-244, March/April 1991.
48. G.A. McIntyre and K.J. Hintz, "An Information Theoretic Approach to Sensor Scheduling," *Signal Processing, Sensor Fusion, and Target Recognition V. Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 2755, Orlando, FL, April 8-10 1996, pp. 304-312.
49. S. Kullback and R.A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, pp. 79-86.
50. W. Schmaedeke and K. Kastella, "Information Based Sensor Management and IMMKF," *Signal and Data Processing of Small Targets 1998: Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 3373, Orlando, FL, April 1998, pp. 390-401.
51. W. Schmaedeke and K. Kastella, "Event-averaged Maximum Likelihood Estimation and Information Based Sensor Management," *Signal Processing, Sensor Fusion, and Target Recognition III. Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 2232, Orlando, FL, April 4-6 1994, 91-96.
52. K. Kastella, "Discrimination Gain to Optimize Detection and Classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 27, no. 1, pp. 112-116, January 1997. K. Kastella, "Discrimination Gain to Optimize Detection and Classification," *Signal and Data Processing, of Small Targets 1995: Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 2561, San Diego, CA, July 11-13 1995, pp. 66-70.
53. K. Kastella, "Discrimination Gain for Sensor Management in Multitarget Detection and Tracking," *IEEE-SMC and IMACS Multiconference CESA '96*, vol. 1, Lille France, July 9-12 1996, pp. 167-172.
54. K. Kastella and S. Musick, "Comparison of Sensor Management Strategies for Detection and Classification," *9th National Symposium on Sensor Fusion*, Monterey, CA, March 1996.
55. R. Mahler, "Global Optimal Sensor Allocation," *Proceedings of the Ninth National Symposium on Sensor Fusion*, vol. I (unclassified), Naval Postgraduate School, Nonterey CA, Mar 12-14, 1996, pp. 347-366.

56. R.P.S. Mahler, “Global Posterior densities for Sensor Management,” *Acquisition, Tracking, and Pointing XII.. Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 3365, Orlando, FL, April 1998, pp. 252-263.
57. R.P.S. Mahler, “Multisource, Multitarget Filtering: A Unified Approach,” *Signal and Data Processing, of Small Targets 1998. Proceedings of the SPIE - The International Society for Optical Engineering,* vol. 3373, Orlando, FL, April 1998, pp. 296-307.
58. S. Musick, K. Kastella, R. Mahler, “A Practical Implementation of Joint, Multitarget Probabilities,” Signal Processing, Sensor Fusion, and Target Recognition VII: Proceedings of the SPIE - The International Society for Optical Engineering, vol. 2232, Orlando, FL, April 1998, pp. 26-37.