2004 Int'l Conference on Robotics and Autonmation
http://www.cds.caltech.edu/~murray/papers/2003w_chu+04-icra.html

Scheduling for Distributed Sensor Networks with Single Sensor Measurement per Time Step

Timothy H. Chung, Vijay Gupta, Babak Hassibi, Joel Burdick, and Richard M. Murray
Division of Engineering and Applied Science
California Institute of Technology
Pasadena, California 91125
Email: {timothyc,gupta,hassibi,jwb,murray}@caltech.edu

***Abstract*— We examine the problem of distributed estimation when only one sensor can take a measurement per time step. We solve for the optimal recursive estimation algorithm when the sensor switching schedule is given. We then consider the effect of noise in communication channels. We also investigate the problem of determining an optimal sensor switching strategy. We see that this problem involves searching a tree in general and propose two strategies for pruning the tree to minimize the computation. The first is a sliding window strategy motivated by the Viterbi algorithm, and the second one uses thresholding. The performance of the algorithms is illustrated using numerical examples.**

I. Introduction and Motivation

Recently there has been a lot of interest in networks of sensing agents which act cooperatively to obtain the best parameter estimates possible, (e.g. [1] and the references therein). Usually the estimate resulting from measurements from many sensors is better than the estimate of any individual sensor in the non-cooperative scenario. The improved performance comes at the cost of increased complexity. As pointed out in [1], the advantages of forming sensor networks are even greater if the sensors are heterogenous. The increased complexity arises from the needed communication infrastructure and the need to fuse the measurements to obtain a better estimate.

Because of the above-mentioned advantages, much attention has been focused on data fusion of heterogeneous sensor measurements, as in [2]. Works such as the EYES project [3], WINS [4], and Smart Dust [5], are examples of systems implementing such networks. The assumption usually made in the analysis of such systems is that all the sensors take measurements at the same time and the data is then fused to get a better estimate. One example of the multiple data fusion algorithms available in literature can be found in [6]. The sensor management issues, if present at all, are in the context of energy efficiency [7], [8], imperfect localization of sensor platforms [9], optimal coverage of a given region [9], [10], or efficient networking and communication protocols [11].

However, in some applications, the use of one sensor places restrictions on the use of other sensors. This situation exists whenever simultaneous use of sensors causes interference in measurements. This is a common problem in robotic systems, e.g. when acoustic sensors are used for ranging. When the individual sensor platforms are using sonar range-finding devices, only one sonar sensor may be active at any time, so as to isolate the reflected signal appropriately. In such a case, apart from the issue of optimal multi-sensor data fusion, there is the additional issue of optimally scheduling the sensor measurements so as to minimize the error covariance associated with state estimation.

In this paper, we study the problem of multi-sensor data fusion when only one sensor is allowed to take a measurement at every time step. Assuming that measurements are being exchanged between sensors, we also consider the case of the communication channels being noisy. We also investigate the issue of constructing the optimal sensor schedule. In the case of tracking an object moving amongst dispersed sensing agents, we seek a sequencing of sonar measurements among the sensors that best accomplishes this task. While optimization of sensor schedules have been examined using optimal or stochastic control theory techniques, as in [12], [13], solutions to Ricatti differential equations, and even information-theoretic methods, as in [14], we pursue two simpler methods, sliding window and thresholding, for determining an optimal sensing schedule. These methods trade computation/memory requirements for sub-optimality; however, they seem to work well on the simulation examples. A more detailed description of the optimizing algorithms can be found in [15]; in this paper we focus more on setting up the problem and solving for the optimal data fusing algorithm.

The paper is organized as follows. The next section, sets up the problem and describes the optimal data-fusion algorithm for a given sensor schedule. Section III considers the degradation in the performance when this scheme is used in the presence of communication noise. Section IV considers the question of choosing the optimal sensor schedule. We present some methods that obtain sub-optimal sensor schedules, but have the advantage of simplicity. We demonstrate these algorithms via examples and simulations.

II. Modeling and Problem Formulation

A. Problem Set-up

Consider a system evolving as follows

$$x[k+1] = Ax[k] + Bw[k], \tag{1}$$

where $x[k] \in \mathbf{R}^n$ is the process state at time step k and $w[k]$ is the process noise. The process noise is assumed

white, Gaussian and zero mean with covariance matrix Q. The process state is observed by N sensors with the measurement equation for the i-th sensor being

$$y_i[k] = C_i x[k] + v_i[k], \tag{2}$$

where $y_i[k] \in \mathbf{R}^s$ is the measurement. The measurement noises $v_i[k]$'s for the sensors are assumed independent of each other and of the process noise. Further the noise $v_i[k]$ is assumed to be white, Gaussian and zero mean with covariance matrix R_i. For the example of tracking a moving target, (1) and (2) describe the linearization of the target's nonlinear dynamical model and the observers' sensing models, respectively. It is assumed that only one sensor can be used at any time. However, unless stated otherwise, we assume that the measurements are communicated to all the sensors in an error-free manner. The estimate of the i-th sensor given the measurements till time step $k-1$ is denoted by $\hat{x}_i[k|k-1]$, or in short as $\hat{x}_i[k]$. More generally, let the estimate of the i-th sensor for the variable $z[k]$, given the measurements till time step $k-1$, be given as $\hat{z}_i[k|k-1]$, or abbreviated as $\hat{z}_i[k]$. We first pose the question: Assuming that the sensor switching sequence is given, what is the optimal filtering algorithm for the i-th node?

B. Optimal Fusion Algorithm

Define the innovation (see, e.g., [16]) for the i-th node $e_i[k]$ as the difference between the actual measurement at time step k ($y_i[k]$) and the predicted measurement ($\hat{y}_i[k|k-1]$). Assuming that the j-th sensor takes the measurement at time step k, we obtain that

$$e_i[k] = y_j[k] - C_j \hat{x}_i[k|k-1], \tag{3}$$

Defining the inner product $\langle x, y\rangle$ as $E\left[xy^T\right]$, we have the form of the linear estimator as

$$\begin{aligned}\hat{x}_i[k+1|k] &= \sum_{n=0}^{k} \langle x_i[k+1], e_i[n]\rangle R^{-1}_{e_i[n]} e_i[n] \\ &= \hat{x}_i[k+1|k-1] + \langle x_i[k+1], e_i[k]\rangle R^{-1}_{e_i[k]} e_i[k],\end{aligned}$$

where $R_{e_i[k]} = \langle e_i[k], e_i[k]\rangle$. However, using (1) gives

$$\hat{x}_i[k+1|k-1] = A\hat{x}_i[k|k-1].$$

Now define the error by

$$\tilde{x}_i[k|k-1] = x[k] - \hat{x}[k|k-1]$$

and let $P_i[k|k-1]$ be the error covariance. Also define $K^i_k = \langle x_i[k+1], e_i[k]\rangle R^{-1}_{e_i[k]}$. Then we see that the error state equation is given by

$$\tilde{x}_i[k+1|k] = (A - K^i_k C_j)\tilde{x}_i[k|k-1] + Bw[k] - K^i_k v_j[k].$$

By definition, we immediately obtain that $P_i[k|k-1]$'s evolve as

$$\begin{aligned}P_i[k+1|k] &= (A - K^i_k C_j) P_i[k|k-1](A - K^i_k C_j)^T \\ &\quad + BQB^T + K^i_k R_j (K^i_k)^T.\end{aligned}$$

Moreover, since

$$e_i[k] = C_j \tilde{x}_i[k|k-1] + v_j[k],$$

we see that

$$R_{e_i[k]} = C_j P_i[k|k-1] C_j^T + R_j.$$

Finally using the fact that

$$\begin{aligned}\langle x_i[k], \tilde{x}_i[k|k-1]\rangle &= \langle \hat{x}_i[k|k-1] + \tilde{x}_i[k|k-1], \tilde{x}_i[k|k-1]\rangle \\ &= 0 + P_i[k|k-1],\end{aligned}$$

we compute

$$K^i_k = AP_i[k|k-1]C_j^T R^{-1}_{e_j[k]}$$

Thus we see that the recursive optimal filtering equation is given by

$$\hat{x}_i[k+1|k] = A\hat{x}_i[k|k-1] + K^i_k e_i[k],$$

where

$$K^i_k = AP_i[k|k-1]C_j^T R^{-1}_{e_j[k]} \tag{4}$$

$$R_{e_i[k]} = C_j P_i[k|k-1] C_j^T + R_j \tag{5}$$

and $P_i[k|k-1]$'s evolve as

$$\begin{aligned}P_i[k+1|k] &= (A - K^i_k C_j) P_i[k|k-1](A - K^i_k C_j)^T \\ &\quad + BQB^T + K^i_k R_j (K^i_k)^T.\end{aligned} \tag{6}$$

Assuming the initial state $x[0]$ has zero mean and covariance Π_0, the initial covariance matrix for above recursions is also given by $P_i(0|-1) = \Pi_0$. Note that $P_i[k|k-1]$ is of independent interest as it is the error covariance for the i-th sensor at time step k when it has processed the measurements till time step $k-1$. We will refer to it as $P_i[k]$ in short. Since all the nodes have access to the same measurements, there is only one innovation and hence all the state estimates are the same. So the subscript i is unnecessary in this case and $P_i[k] = P[k]$ for all i.

C. Optimal Algorithm - Communication Noise Case

Let us assume now that any signal exchanged between sensor nodes i and j is corrupted by additive, zero-mean, Gaussian white noise, v_{ij}. We wish to see how the performance of the scheme of exchanging measurements between the sensors outlined above is affected. Going through a similar derivation as above, we find that (3) is modified to

$$e_i[k] = y_j[k] - C_j \hat{x}_i[k|k-1] + v_{ij}[k],$$

assuming that the j-th sensor has taken the measurement at time step k. Let us assume the noise vector $\zeta[k] = (w[k], v_i[k], v_{ij}[k])^T$ to be described by

$$E\left[\zeta[k]\zeta[l]^T\right] \triangleq \begin{pmatrix} Q & 0 & 0 \\ 0 & R_i & 0 \\ 0 & 0 & R_{ij} \end{pmatrix} \delta(k-l).$$

Then, we find that the Kalman filter form remains the same as before except that (4) becomes

$$R_{e_i[k]} = C_j P_i[k|k-1] C_j^T + R_j + R_{ij}. \tag{7}$$

and (6) changes to

$$P_i[k+1|k] = (A - K_k^i C_j) P_i[k|k-1](A - K_k^i C_j)^T + BQB^T + K_k^i R_j (K_k^i)^T + K_k^i R_{ij} (K_k^i)^T \quad (8)$$

We note that the only difference from the earlier case is that the effective measurement noise includes the actual sensor noise plus the communication noise. Observe, however, that sending only the measurement from one sensor to the other might not be the optimal thing to do in this case. Sending more information (e.g., the state estimates) might lead to better performance for all the sensors considered together.

III. Optimization Algorithms

A. Optimization of the Sensor Schedule

In the analysis presented so far, we have assumed that the sensor schedule was given. It is obvious that the minimum error covariance achievable is a function of the sensor schedule. Next, we wish to find the sensor schedule that minimizes the error covariance over a given time horizon. In this and subsequent sections, we consider this problem. For simplicity and without loss of generality, we consider only two sensors and define the cost function, J, to be the sum of the error covariance matrices for the two sensors over the running time of the system:

$$J = \sum_{k=0}^{N} \text{trace}\left(P_1[k] + P_2[k]\right),$$

where, as before, $P_1[k]$ and $P_2[k]$ are error covariances of the estimates at time step k. We have assumed that the system begins at time $k = 0$ and goes on till $k = N$. In a more general case, the covariances can be variously weighted to set up the cost function if getting a good estimate either at some time steps or for some sensors is more important than others.

We can represent all the possible sensor schedule choices by a tree structure, as shown in Fig. 1 for the case of two sensors. Each node on the tree represents the active (ie. measurement-taking) sensor at its particular time step, with the root defined to be time zero. The branches from each node correspond to choosing a particular sensor to be active at the next time instant. Thus, the path from the root to any node at depth d represents a particular sensor schedule choice for time steps 0 to d. We can associate with each node the cost function evaluated using the sensor schedule corresponding to the path from the root to that node. Obviously, finding *the* optimal sequence requires traversing all the paths from the root to the leaves in a binary tree (for the case of two sensors). If the leaves are at a depth N, a total of 2^N schedules need to be compared. This procedure might place too high a demand on the computational and memory resources of the system. Moreover, in practical applications N might not be fixed a-priori. Hence we need some sort of on-line optimization procedure. We present some approximations which address these difficulties.

The basic idea behind the two approximations is to prune the tree to a manageable size. However, the pruning should ensure with a high probability that the optimal sequence is not lost. The algorithms presented involve choosing some arbitrary parameters which depend on the problem and the computation/memory resources available. Choosing these parameters conservatively will ensure that the sub-optimal solution will be closer to the optimal solution but it might mean maintaining a large part of the tree intact. Therein lies the trade-off involved. However, in the numerical examples studied, relatively liberal choices keep the tree size fairly small.

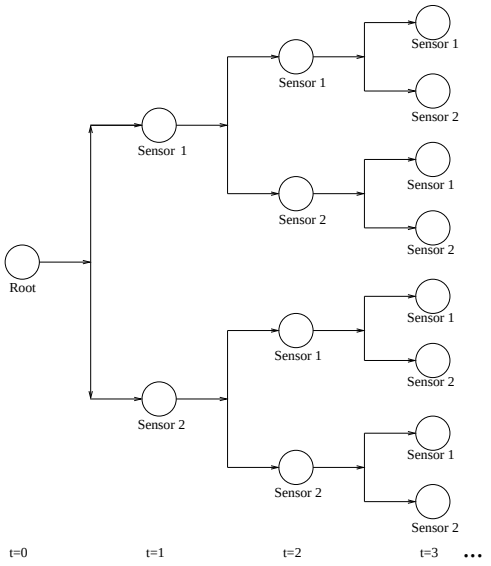

Fig. 1. The tree structure defined by the various possible choices of sensor schedules illustrated for the case of 2 sensors.

B. Sliding Window Algorithm

This algorithm is similar to a pseudo real time version of the Viterbi algorithm ([17]). We define a window size d where $d < N$. The algorithm proceeds as follows:

1) Initialization: Start from root node with time $k = 0$.
2) Traversal:
   a) Traverse all the possible paths in the tree for the next d levels from the present node.
   b) Identify the sensor sequence S_k,S_{k+1},S_{k+2},..., S_{k+d-1} that yields the minimum cost at the end of this window of size d.
   c) Choose the first sensor S_k from the sequence.
3) Sliding the Window:
   a) If $k = N$ then quit, else go to the next step.
   b) Designate the sensor S_k as the root.
   c) Update time $k = k + 1$.
   d) Repeat the traversal step.

The arbitrary parameter for this algorithm, mentioned earlier, is the window size d. If the window size is large enough, the sequence yielding the lowest cost will resemble the optimal sequence for the entire time horizon. Also note that when we slide the window, we already have the error covariances for the first $d - 1$ time steps stored; hence they do not need to be recalculated. Consequently, the method is not very

computationally intensive. In essence, this sliding window approach employs a less computationally-intensive variation of the A^* search algorithm [18] by determining the minimum cost path over each window rather than the entire tree.

C. Thresholding

This algorithm is similar to that presented in [19], in the context of choosing the optimal controller from a set of many possible choices. We define a factor f where $f \geq 1$. The algorithm proceeds as follows:

1) Initialization : Start from root node with cost $J = 0$.
2) Pruning:
   a) Extend the tree by one level (ie. time step) through all possible paths from the current node.
   b) Calculate the minimum cost up to that time step.
   c) Prune away any branch that yields the cost greater than f times the minimum.
   d) For the remaining branches, denote the cost of the nodes as the cost achieved by moving down the tree till the node.
3) Update: Consider each node in the next time step as the root node and repeat the pruning step.
4) After N time steps or a sufficiently large time interval, declare the optimal sequence to be the one yielding the minimum cost till that time step.

The intuition behind the method is that any sequence which yields too high a cost at any intermediate time step would probably not be the one that yields the minimum cost over-all. By playing with the factor f, we obtain a trade-off between the certainty that we would not prune away the optimal sequence and the number of branches in the tree that need to be traversed.

IV. Simulation Results

A. Example model and cost function

In this section, we walk through an example demonstrating the application of algorithms developed above. We assume two sensing vehicles trying to locate a non-cooperating target. We model the target vehicle with the standard constant acceleration model [20]. This model assumes that the vehicle has constant acceleration equal to zero except for a small random perturbation. We assume that the vehicle moves in two dimensions. Denoting the position of the vehicle in the two dimensions by p_x and p_y, and the velocities by v_x and v_y, we can model the state of the system by the vector

$$X = \begin{bmatrix} p_x & p_y & v_x & v_y \end{bmatrix}^T.$$

With a discretization step size of h, the dynamics of the vehicle can be modeled as

$$X[k+1] = AX[k] + Bw[k], \tag{9}$$

where

$$A = \begin{bmatrix} 1 & 0 & h & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} h^2/2 & 0 \\ 0 & h^2/2 \\ h & 0 \\ 0 & h \end{bmatrix}.$$

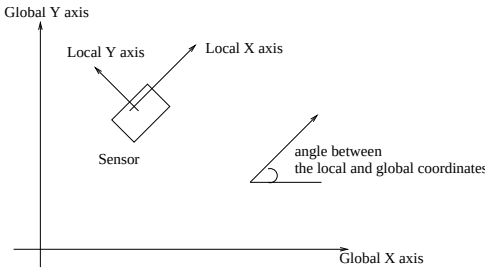

Fig. 2. If the sensor is oriented at an angle to X-axis, measurements need to be rotated to get their value in the global coordinates.

The term w_k represents the noise that models the perturbations to accelerations. The sensor model is the usual sonar model [21]. Being an echo-based device, it senses only the range to the target and not the relative velocities. If the sensor is oriented at an angle θ to the global x-axis (see Fig. 2), it can be shown ([21]) that the vehicle's measurement in the global frame is given by

$$y_{\text{global}}[k] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} X[k] + R(\theta)v[k], \tag{10}$$

where $R(\theta)$ is the rotation matrix between the local and the global coordinate systems given by

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

The term $v[k]$ in (10) represents the sensor noise. It has two components, the noise present in the range measurement, and the effective bearing noise arising from the modeling of the sonar beam as a sensing cone. The range noise is usually assumed smaller than the bearing noise. The range noise increases with the distance of the sensor from the target and the bearing noise variance can usually be modelled as a fixed multiple of the range noise variance for a given sensor. For simplicity, the two noises can be assumed independent. Thus the covariance matrix of $v(k)$ is typically given by

$$R = \begin{bmatrix} \sigma^2_{\text{range}} & 0 \\ 0 & r^2\sigma^2_{\text{bearing}} \end{bmatrix},$$

where σ^2_{range} is the range noise variance that increases with the distance to target, r. The bearing noise variance $\sigma^2_{\text{bearing}}$ can be modelled to be related to the range noise variance for the particular sensor.

In the numerical example, we consider the value $h = 0.2$. The process noise is considered to have covariance matrix Q given by

$$Q = \begin{bmatrix} 0.0100 & 0 \\ 0 & 0.0262 \end{bmatrix}.$$

We consider two sensors. The first sensor is placed at position corresponding to $\theta = 0^\circ$ (see Fig. 3.) It is closer to the target and accordingly the range noise is comparatively smaller. The second sensor is given to be at a position corresponding to $\theta = -90^\circ$. Specifically the numerical values of the sensor

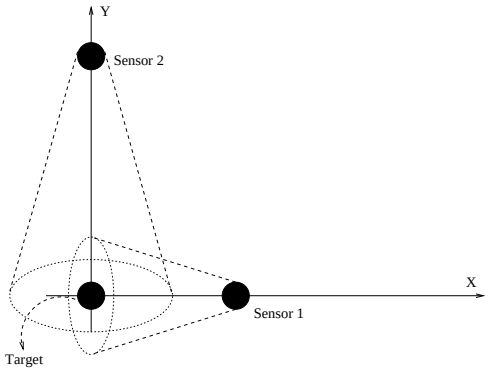

Fig. 3. Sensor orientation for the simulation examples.

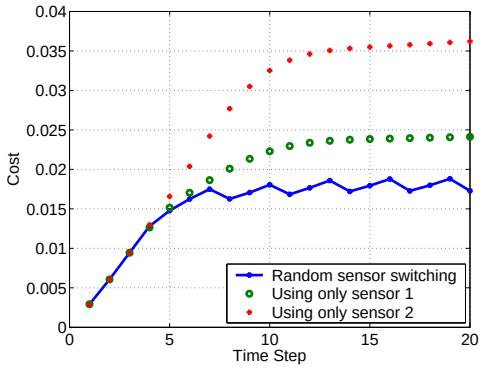

Fig. 4. Sensor switching helps to bring the cost down.

noise covariances considered are

$$R_1 = \begin{bmatrix} 0.0003 & 0 \\ 0 & 0.0273 \end{bmatrix} \qquad R_2 = \begin{bmatrix} 0.0018 & 0 \\ 0 & 0.0110 \end{bmatrix}.$$

Thus after rotation, R_1 remains the same while R_2 is transformed to

$$R_2 = \begin{bmatrix} 0.0110 & 0 \\ 0 & 0.0018 \end{bmatrix}.$$

We compare the algorithm performances over a time horizon of 20 steps. The cost function is simply the sum of the trace of the error covariance matrices of the two sensors from time $k = 0$ to time $k = 20$.

B. Choosing any one sensor always is not optimal

Note that the simple strategy of always choosing the closer sensor (sensor 1) is not optimal. We compare the strategy of choosing only sensor 1 or only sensor 2 with a randomly generated strategy that uses both the sensors with the sensor schedule [1,1,1,2,1,1,1,2,1,1,1,2,1,1,1,2,1,1,1,2] over the 20 time steps. The sum of traces of the error covariances of the two sensors for the three strategies as a function of time is shown in Fig. 4.

We see that even a random sensor switching strategy can help to bring down the cost. At any time step, the errors are much more if any single sensor is being used. In fact summed over the entire time horizon, we see that the switching strategy helps to bring down the cost by about 24% over any of the single sensor strategies.

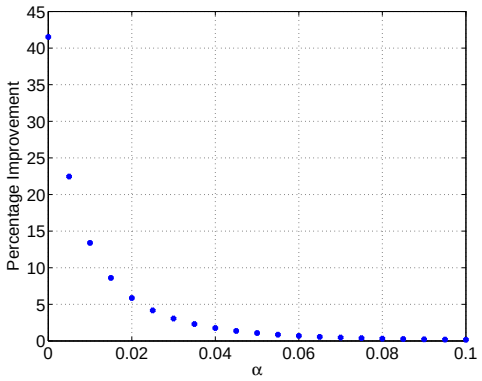

Fig. 5. Percent improvement in cost due to sensor switching as communication noise is increased.

C. Effect of communication noise

In this section, we consider the same example but add communication noise in the channel between the two sensors. The noise covariance is given by

$$R_{12} = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}.$$

We consider the cost function as the sum of the traces of the error covariances of the two sensors over the time horizon $[0, 20]$. Fig. 5 shows the improvement in cost by the sensor switching strategy given above over always using sensor 2 as the parameter α is varied over small values. As α increases, we see the communication noise rapidly deteriorates the efficiency obtained by sensor switching since it deteriorates the estimates of both the sensors.

As noted earlier, in the presence of communication noise, sending measurements might not be the optimal thing to do.

D. Performance of the sliding window algorithm

In this section we study the performance of the sliding window algorithm described earlier. We consider the same example and cost function as before. Fig. 6 shows the improvement in the cost due to the predicted (sub)-optimal sensor sequence over using only sensor 2 as a function of varying window sizes.

It can be seen from the figure that even a window size of $k = 1$ leads to more than 42% improvement in the cost by predicting a good sensor switching strategy.

E. Performance of the thresholding algorithm

We now consider the thresholding algorithm presented earlier. The example and cost function considered are the same. Fig. 7 shows the improvement in cost due to the optimal sensor sequence predicted by the thresholding protocol as the cut-off factor f is varied.

A large improvement can be obtained by using a fairly small thresholding factor. For $f = 1$, the improvement is over 42%.

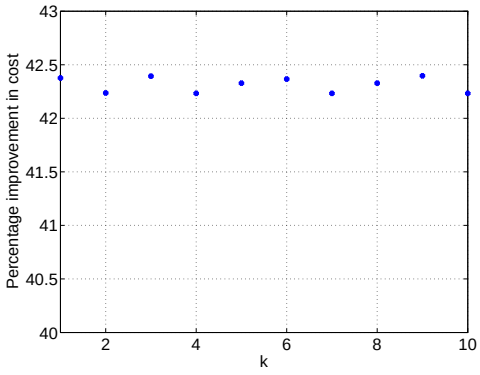

Fig. 6. Percent improvement in cost due to the optimal sensor switching strategy as predicted by the sliding window scheme.

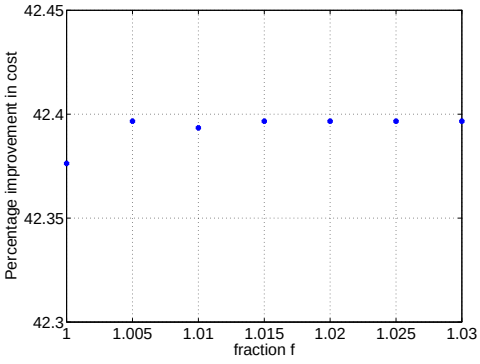

Fig. 7. Percent improvement in cost due to the optimal sensor switching strategy as predicted by the thresholding scheme.

V. Conclusions and Future Work

In this paper, we looked at the problem of distributed estimation when only one sensor is allowed to take a measurement per time step. We saw that exchanging measurements between sensors is sufficient if the communication channel is noiseless and solved for the optimal recursive estimation algorithm. We looked at performance degradation when communication noise is present. Then we investigated the problem of determining an optimal sensor switching strategy. We saw that this problem involves searching a tree in general and proposed two strategies for pruning the tree to keep the computation tractable. Some examples demonstrating these algorithms were presented.

The work can potentially be extended in many ways. Examining better strategies for addressing communication noise and types of channels are of interest. Additionally, this work hints at possibilities for maneuvering mobile sensor platforms to further improve the estimate.

Acknowledgements

Research supported in part by the AFOSR grant F49620-01-1-0460 for the first author and by the Engineering Research Centers Program of the National Science Foundation under Award Number EEC-9402726 and also a grant from NASA for the second author.

References

[1] S. Roumeliotis and G. Bekey, "Distributed multi-robot localization," *IEEE Trans. on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, Oct 2002.

[2] J. Nash, "Optimal Allocation of Tracking Resource," in *Proc. of the 1977 IEEE Conf. on Decision and Control*, vol. 1, New Orleans, LA, December 1977, pp. 1177–1180.

[3] H. Karl, "Making sensor networks useful: Distributed services - the eyes project," ESF Workshop, La Spezia, Italy, 2002.

[4] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," in *Proc. of the Fifth Annual Intl. Conf. on Mobile Computing and Networks*, 1999.

[5] J. Kahn, R. Katz, and K. Pister, "Next Century Challenges: Mobile Networking for "Smart Dust"," in *Proc. of ACM MobiCom Conf.* Seattle, WA: MobiCom, August 1999.

[6] B. Rao, H. Durrant-Whyte, and J. Sheen, "A fully decentralized multi-sensor system for tracking and surveillance," *Intl. Journal of Robotics Research*, vol. 12, pp. 20–44, 1993.

[7] R. Mina and M. Bhardwaj and S.H. Cho and A. Sinha and E. Shih and A. Wang and A. Chandrakasan, "Low-Power Wireless Sensor Networks," *VLSI Design 2001*, January 2001.

[8] A. Wang and A. Chandrakasan, "Energy Efficient System Partitioning for Distributed Wireless Sensor Networks," in *Proc. ICASSP 2001*, May 2001.

[9] S. Dhillon and K. Chakrabarty and S. Iyengar, "Sensor Placement for Grid Coverage under Imprecise Detections," in *Proc. Intl. Conf. on Information Fusion*, 2002, pp. 1571–1587.

[10] S. Meguerdichian and F. Koushanfar and M. Potkonjak and M. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Networks," in *Proc. of IEEE InfoCom 2001*, April 2001.

[11] C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor information networking architecture and applications," *IEEE Personel Communication Magazine*, vol. 8, no. 4, pp. 52–59, August 2001.

[12] A. Savkin, R. Evans and E. Skafidas, "The Problem of Optimal Robust Sensor Scheduling," in *Proc. of the 39th Conf. on Decision and Control*, no. 1, Sydney, Australia, December 2000.

[13] E. Skafidas and A. Nerode, "Optimal measurement scheduling in linear quadratic gaussian control problems," in *Proc. Of the 1998 IEEE Intl. Conf. On Control Applications*, Trieste, Italy, 1998, pp. 1225–1229.

[14] G. McIntyre and K. Hintz, "An Information Theoretic Approach to Sensor Scheduling," in *Proc. of the SPIE*, vol. 2755, Orlando, FL, Apr 1996, pp. 304–312.

[15] V. Gupta, T. Chung, B. Hassibi, and R. M. Murray, "Scheduling for Distributed Sensor Networks," in *Information Processing in Sensor Networks*, Berkeley, CA, 2004, submitted.

[16] T. Kailath, A. Sayed, and B. Hassibi, *Linear Estimation*. Prentice-Hall, 2000, ch. 9.

[17] J. G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 6, pp. 268–278, January 1973.

[18] Y. Hwang and N. Ahuja, "Gross motion planning – a survey," *ACM Computing Surveys*, vol. 24, no. 3, pp. 219–291, September 1992.

[19] B. Lincoln and B. Bernhardsson, "LQR optimization of linear system switching," *IEEE Tran. on Automatic Control*, vol. 47, pp. 1701–1705, Oct. 2002.

[20] Y. Bar-Shalom and X. R. Li, *Estimation and Tracking: Principles, Techniques and software*. Artech House, 1993.

[21] K. Ramachandra, *Kalman filtering techniques for radar tracking*. New York: Marcel Dekker Inc., 2000.