REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 18 May 2000	3. REPORT TYPE AND DATES COVERED Final Report, 01 June 1997 - 31 May 2000

4. TITLE AND SUBTITLE Distributed Multisensor Fusion Algorithms for Tracking Applications (Final Report)	5. FUNDING NUMBERS Grant N00014-97-1-0642
6. AUTHOR(S) Lucy Y. Pao	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES) Electrical & Computer Engineering Department Campus Box 425 University of Colorado Boulder, CO 80309-0425	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAMES(S) AND ADDRESS(ES) Drs. Allen Moshfegh and William King Strike Technology Division, Office of Naval Research 800 North Quincy Street Arlington, Virginia 22217	10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
a. DISTRIBUTION / AVAILABILITY STATEMENT UNLIMITED	12. DISTRIBUTION CODE

20000530 062

13. ABSTRACT *(Maximum 200 words)*

The objective of the research under this ONR award is to develop multisensor fusion algorithms and sensor management techniques for tracking applications. Under this award, we have achieved a number of results: (1) We have developed a method of distributed fusion that is amenable to general distributed architectures; (2) We have developed non-simulation techniques for comparing multisensor fusion algorithms that are significantly more computationally efficient than performing Monte Carlo simulation evaluations; (3) We have investigated and compared the computational complexity and tracking performance of sequential and parallel implementations of multisensor fusion algorithms; (4) We have investigated the order of processing sensors of unequal qualities in sequential implementations of multisensor fusion algorithms; (5) We have developed several schemes for controlling sensor information and have evaluated the effects of sensor request delays; and (6) We have investigated the application of ordinal optimization and super-heuristic techniques for developing efficient implementations of our new sensor management methods. Our results have provided insight as to the relative performance of various multisensor fusion methods, and the results have also provided a basis for assessing the tradeoffs between performance and computational and communication requirements when planning new sensor network architectures or communication link protocols.

14. SUBJECT TERMS Target tracking, data association, sensor fusion, sensor management	15. NUMBER OF PAGES 6
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

DTIC QUALITY INSPECTED 3

Young Investigator Award: Distributed Multisensor Fusion Algorithms for Tracking Applications (ONR Grant N00014-97-1-0642)

Final Report, May 2000

Lucy Y. Pao
University of Colorado at Boulder

A review of the accomplishments achieved under this Young Investigator Award is given here. The following papers, funded fully or partially under this ONR award, were published or submitted and are attached.

[A] L. Y. Pao and M. K. Kalandros. "Algorithms for a Class of Distributed Architecture Tracking," *Proc. American Control Conf.*, Albuquerque, NM, pp. 1434–1438, June 1997.

[B] C. W. Frei and L. Y. Pao. "Alternatives to Monte Carlo Simulation Evaluations of Two Multisensor Fusion Algorithms," *Automatica*, 34(1): 103–110, Jan. 1998.

[C] M. K. Kalandros and L. Y. Pao. "Controlling Target Estimate Covariance in Centralized Multisensor Systems," *Proc. American Control Conf.*, Philadelphia, PA, pp. 2749–2753, June 1998.

[D] L. Y. Pao and M. K. Kalandros. "The Effects of Delayed Sensor Requests on Sensor Manager Systems," *Proc. AIAA Guidance, Navigation, and Control Conf.*, Boston, MA, pp. 1127–1135, Aug. 1998.

[E] N. T. Baltz. *Allocation of Sensing Resources for Distributed Multiprocessor Systems*, M.S. Thesis, University of Colorado at Boulder, May 1999.

[F] L. Y. Pao and N. T. Baltz. "Control of Sensor Information in Distributed Multisensor Systems," *Proc. American Control Conf.*, San Diego, CA, pp. 2397–2401, June 1999.

[G] L. Y. Pao and L. Trailovic. "On the Order of Processing Sensor Information in Sequential Implementations of Fusion Algorithms," *Proc. American Control Conf.*, San Diego, CA, pp. 2407–2411, June 1999.

[H] M. K. Kalandros, L. Y. Pao, and Y. C. Ho. "Randomization and Super-Heuristics in Choosing Sensor Sets in Target Tracking Applications," *Proc. IEEE Conf. Decision and Control,* Phoenix, AZ, pp. 1803–1808, Dec. 1999.

[I] L. Y. Pao and W. Khawsuk. "Determining Track Loss Without Truth Information for Distributed Target Tracking Applications," *Proc. American Control Conf.*, Chicago, IL, June 2000, in press.

[J] L. Y. Pao and L. Trailovic. "The Optimal Order of Processing Sensor Information in Sequential Multisensor Fusion Algorithms," *IEEE Trans. Automatic Control*, 45(8), Aug. 2000, in press.

[K] M. K. Kalandros and L. Y. Pao. "The Effects of Data Association on Sensor Manager Systems," *Proc. AIAA Guidance, Navigation, and Control Conf.*, Denver, CO, Aug. 2000, in press.

[L] M. K. Kalandros and L. Y. Pao. "Covariance Control for Multisensor Systems," submitted in Apr. 1999 for publication in the *IEEE Trans. Aerospace Electronic Systems.*

[M] L. Y. Pao and N. T. Baltz. "Varying Rate and Resolution to Control Estimation Error Covariance in Centralized Target Tracking Systems," submitted in May 2000 for publication in *Automatica.*

[N] L. Y. Pao and N. T. Baltz. "Controlling Target Estimate Covariance in Distributed Multisensor Systems," to be submitted in June 2000 for publication in the *IEEE Trans. Aerospace Electronic Systems.*

Other related papers that were submitted or published include:

[O] L. Y. Pao, T. N. Chang, and E. Hou. "Input Shaper Designs for Minimizing the Expected Level of Residual Vibration in Flexible Structures," *Proc. American Control Conf.*, Albuquerque, NM, pp. 3542–3546, June 1997.

[P] L. Y. Pao and W. E. Singhose. "Verifying Robust Time-Optimal Commands for Multi-Mode Flexible Spacecraft," *AIAA J. Guidance, Control, and Dynamics*, 20(4): 831–833, July-Aug. 1997.

[Q] L. Y. Pao. "An Analysis of the Frequency, Damping, and Total Insensitivities of Input Shaping Designs," *AIAA J. Guidance, Control, and Dynamics*, 20(5): 909–915, Sept.-Oct. 1997.

[R] L. Y. Pao and W. E. Singhose. "Robust Minimum Time Control of Flexible Structures," *Automatica*, 34(2): 229–236, Feb. 1998.

[S] L. Y. Pao and D. A. Lawrence. "Synergistic Visual/Haptic Computer Interfaces," *Proc. Japan/USA/Vietnam Workshop on Research and Education in Systems, Computation, and Control Engineering*, Hanoi, Vietnam, pp. 155–162, May 1998.

[T] L. Y. Pao and Craig F. Cutforth. "An Analysis and Comparison of Frequency-Domain and Time-Domain Input Shaping," *Proc. American Control Conf.*, Philadelphia, PA, pp. 3072–3074, June 1998.

[U] L. Y. Pao and M. A. Lau. "Input Shaping Designs to Account for Uncertainty in Both Frequency and Damping in Flexible Structures," *Proc. American Control Conf.*, Philadelphia, PA, pp. 3070–3071, June 1998.

[V] L. Y. Pao and M. A. Lau. "An Analysis of the Expected Residual Vibration of Input Shaping Designs," *Proc. AIAA Guidance, Navigation, and Control Conf.*, Boston, MA, pp. 354–364, Aug. 1998.

[W] D. A. Lawrence, L. Y. Pao, A. M. Dougherty, Y. Pavlou, S. W. Brown, and S. A. Wallace. "Human Perceptual Thresholds of Friction in Haptic Interfaces," *Proc. ASME Dynamic Systems and Control Division*, DSC-Vol. 64, pp. 287–294, *ASME Int. Mech. Engr. Cong. & Expo.*, Anaheim, CA, Nov. 1998.

[X] L. Y. Pao and M. Tomizuka. "Control Education and Research in Vietnam," *IEEE Control Systems Magazine*, 18(6): 94–97, Dec. 1998.

[Y] L. Y. Pao. "Multi-Input Shaping Design for Vibration Reduction," *Automatica*, 35(1): 81–89, Jan. 1999.

[Z] L. Y. Pao and M. A. Lau. "The Expected Residual Vibration of Traditional and Hybrid Input Shaping Designs," *AIAA J. Guidance, Control, and Dynamics*, 22(1): 162–165, Jan.–Feb. 1999.

[AA] C. D. Lee, D. A. Lawrence, and L. Y. Pao. "Guaranteed Convergence Rates for Five Degree of Freedom In-Parallel Haptic Interface Kinematics," *Proc. IEEE Int. Conf. Robotics and Automation*, Detroit, MI, pp. 3267–3274, May 1999.

[BB] C. F. Cutforth and L. Y. Pao. "An Analysis of Frequency-Domain Input Shaping Designs for Three-Mode Flexible Systems," *Proc. American Control Conf.*, San Diego, CA, pp. 4388–4392, June 1999.

[CC] C. F. Cutforth and L. Y. Pao. "A Modified Method for Multiple Actuator Input Shaping," *Proc. American Control Conf.*, San Diego, CA, pp. 66–70, June 1999.

[DD] F. Infed, S. W. Brown, C. D. Lee, D. A. Lawrence, A. M. Dougherty, and L. Y. Pao. "Combined Visual/Haptic Rendering Modes for Scientific Visualization," *Proc. ASME Dynamic Systems and Control Division*, DSC-Vol. 67, pp. 93–99, *ASME Int. Mech. Engr. Cong. & Expo.*, Nashville, TN, Nov. 1999.

[EE] L. Y. Pao and M. A. Lau. "Robust Input Shaper Control Design for Parameter Variations in Flexible Structures," *ASME J. Dynamic Systems, Measurement, and Control*, 122(1): 63–70, March 2000.

[FF] M. A. Lau and L. Y. Pao. "Time-Optimal Commands with Specified Fuel Usage for Controlling Flexible Structures," *Proc. American Control Conf.*, Chicago, IL, June 2000, in press.

[GG] D. A. Lawrence, L. Y. Pao, A. M. Dougherty, M. A. Salada, and Y. Pavlou. "Rate-Hardness: A New Performance Metric for Haptic Interfaces," *IEEE Trans. Robotics and Automation*, 16(4), Aug. 2000.

[HH] C. F. Cutforth and L. Y. Pao. "Command Shaping Control for Vibration Reduction without Lengthening the Command," *Proc. AIAA Guidance, Navigation, and Control Conf.*, Denver, CO, Aug. 2000, in press.

[II] L. Y. Pao and C. F. Cutforth. "On Frequency-Domain and Time-Domain Input Shaping for Multi-mode Flexible Structures," submitted in Apr. 1999 for publication in the *IEEE Trans. Automatic Control.*

[JJ] C. D. Lee, D. A. Lawrence, and L. Y. Pao. "A High-Bandwidth Force-Controlled Haptic Interface," submitted in Feb. 2000 for the *Proc. Annual Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, to be held at the *Int. Mech. Engr. Cong. & Expo.*, Orlando, FL, Nov. 2000.

[KK] R. Y. Novoselov, D. A. Lawrence, and L. Y. Pao. "Haptic Rendering of Data on Irregular Grids," submitted in Feb. 2000 for the *Proc. Annual Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, to be held at the *Int. Mech. Engr. Cong. & Expo.*, Orlando, FL, Nov. 2000.

[LL] D. A. Lawrence, C. D. Lee, L. Y. Pao, and R. Y. Novoselov. "Shock and Vortex Visualization Using a Combined Visual/Haptic Interface," submitted in Mar. 2000 for the *Proc. IEEE Visualization Conf.*, Salt Lake City, UT, Oct. 2000.

Research Activities

The objective of the research under this ONR award is to develop multisensor data fusion algorithms for tracking applications. We have addressed the estimation and data association process, where different measurement types must be integrated into one common estimation process, and consistent probability metrics must be established for all sensor types. We also investigated techniques to manage sensor information and sensor resources in multisensor systems. In developing various algorithms, we focused our work and results to be useful for Naval tracking and surveillance systems.

Under this project, we have achieved results in several different areas:

- Multisensor target tracking is often performed using a single processor to monitor several sensors (centralized fusion), but this method is demanding of both computational power and communication bandwidth. Distributed sensor fusion is a method of addressing these limitations. However, the distributed sensor fusion problem is more complex due to the correlation of separate track estimates. We previously developed a method known as measurement reconstruction and showed that it addresses this correlation problem in a specific class of distributed architectures [1]. We have extended the measurement reconstruction approach to a more generalized architecture using two new algorithms [A]. Computational and communication requirements have been compared with centralized sensor fusion, and Monte Carlo simulation studies have been used to compare the performance of these and other algorithms. We have also investigated the robustness of our algorithms to modeling errors that can yield errors in the measurement reconstruction process.

  In distributed tracking, it is also of importance to determine when track loss has occurred at each processor so that bad estimates are not transmitted to other processors for use in forming updated target state estimates. We have explored a number of methods of determining the lifetimes of tracks of targets without knowledge of the actual states (as is the case in practice) and have compared the performance of these methods with track lifetimes determined using truth information (which is available in simulations) [I].

- Because Monte Carlo simulation evaluations of multisensor multitarget tracking algorithms are time consuming and expensive, we have developed two non-simulation techniques for comparing multisensor probabilistic data association filters (MSPDAF) [B]. While requiring only a fraction of the time for Monte Carlo simulation evaluation, the non-simulation techniques have been shown to accurately predict the performance of the MSPDAF in terms of RMS position error and track lifetime which has been observed in simulations.

- We have investigated and compared the computational complexity and tracking performance of sequential and parallel implementations of the multisensor probabilistic data association algorithm [B]. Our studies indicate that the sequential implementation is better on the average than the parallel implementation, in terms of both RMS position error and track lifetime metrics. We have further developed analytical results that show that the sequential implementation is exponentially more computationally efficient as the clutter density and number of sensors increase. These studies assumed that all sensors were of equal quality.

We have also investigated how the order of processing sensors of unequal qualities in a sequential implementation affects tracking performance. Our results [G,J] indicate that when using two sensors of different qualities, processing the worse sensor first leads to lower RMS position errors. These results have been verified via Monte Carlo simulations as well as analytically using the multi-sensor extension of the Modified Riccati Equation which we developed in [B] to approximate steady-state estimate covariances, and hence RMS errors in the estimates.

Through my involvement in this ONR program, my group has begun a collaboration with Professor Larry Ho at Harvard University. We will be continuing this particular line of research beyond the end of this ONR award, where we shall explore ordinal optimization techniques developed by Professor Ho's group to allow us to efficiently evaluate the best order for processing sensors when there are a large number of sensors of varying qualities.

- Using multiple sensors in surveillance systems allows the strengths of one sensor type to compensate for the weaknesses of another and further provides redundance, therefore increasing system robustness. However, because of limited sensor resources and limited processing capabilities, only a subset of the sensors can be allocated to various targets at each time interval. We have developed several schemes for controlling sensor information. In order to keep the mathematics more tractable, we initially [C,L,M] assumed a centralized processing architecture, where the measurements from all sensors are sent to a global processor where the measurements are fused and used for estimating the states (position, velocity, etc.) of the objects in the surveillance region. We have developed three algorithms that maintain a target's state estimate covariance near a desired level without over-taxing the computational resources of a tracking system. We have also modeled and evaluated the effects that (inevitable) sensor request delays can have on performance [D,L]. We have further extended our sensor manager algorithms for decentralized multisensor systems, where we have developed two computationally efficient techniques that have low communication bandwidth requirements and allow sufficient nodal autonomy [E,F,N]. Finally, we have also developed sensor management techniques that incorporate models of the complex data association process so that more accurate predictions of the effects of the use of various sensors can be made [K].

  We will be proceeding with our work on sensor management techniques after this ONR award ends. On this sub-project, we shall also work with Professor Larry Ho at Harvard, where we will be investigating efficient methods for implementing the sensor management algorithms we have developed. Moreover, we will work with Data Fusion Corporation in Northglenn, CO on extending these sensor management methods for sensors used in tracking ground targets (as opposed to air targets as has been generally assumed in this ONR project).

Collaboration with Industry and Other Researchers

To ensure that our work is properly motivated and directed, we have interacted with industry as well as other research groups:

- We have worked with Data Fusion Corporation (DFC) (Northglenn, CO) on sensor management issues. DFC is a small company which receives much of their funding from the Air Force (Wright Patterson). Our discussions with DFC have helped to motivate much of our work in developing techniques to manage the large quantities of sensor information that must be processed in military surveillance systems.
- We have collaborated with Professor Larry Ho at Harvard University, investigating whether ordinal optimization and super-heuristic concepts can be incorporated in our development of multisensor fusion algorithms to make them or evaluations of them more computationally efficient. In particular, we have investigated the combination of ordinal optimization and super-heuristic techniques for developing efficient implementations of our new sensor management algorithms [H]. I visited Professor Ho at Harvard in August 1998 as well as May 2000, and my group plans to host Professor Ho for a visit at the University of Colorado at Boulder within the next year.
- We have also had a number of discussions with Professor Isaac Kaminer at the Naval Postgraduate School to collaborate in the development of sensor fusion algorithms when the measurements are nonlinear. We hosted a visit by Professor Kaminer to the University of Colorado at Boulder in December 1998, and we recently resolved some issues in the investigation of the nonlinear filtering problem that we are jointly tackling.
- We have also had a series of discussions with Professor Jason Speyer at the University of California at Los Angeles which have lead to a number of ideas for applying some of his work on fault detection to our efforts in developing methods of determining the track lifetimes of targets without truth information [I].
- Finally, we have also had interactions with Professor Stuart Russell of the University of California at Berkeley to compare his recent work on developing a data association algorithm based on probabilistic reasoning with methods of data association that my group has developed as well as other methods that have been documented in the literature. I visited with Professor Russell briefly during a recent trip to San Francisco in May 2000.

Summary

Our results have provided insight as to the relative performance of various multisensor fusion methods, and they have also provided a basis for assessing the tradeoffs between performance and computational and communication requirements when planning new sensor network architectures or communication link protocols.

[1] L. Y. Pao. "A Measurement Reconstruction Approach for Distributed Multisensor Fusion," *AIAA J. Guidance, Control, and Dynamics*, 19(4): 842–847, July-Aug. 1996.

Pergamon

PII: S0005-1098(97)00167-2

Automatica, Vol. 34, No. 1, pp. 103-110, 1998
© 1998 Elsevier Science Ltd. All rights reserved
Printed in Great Britain
0005-1098/98 $19.00 + 0.00

Brief Paper

Alternatives to Monte-Carlo Simulation Evaluations of Two Multisensor Fusion Algorithms*

CHRISTIAN W. FREI† and LUCY Y. PAO‡

Key Words—Target tracking; data association; sensor fusion; non-simulation performance evaluation.

Abstract—Parallel and sequential multisensor extensions of the probabilistic data association filter for multitarget tracking in clutter are presented, and a non-simulation technique is developed and used to compare tracking performance of the two algorithms in case of a single target. While requiring only a fraction of the time for Monte-Carlo simulation evaluation, the non-simulation technique is shown to accurately predict the average superior performance of the sequential implementation in terms of RMS position error and track lifetime which has been observed in simulations. © 1998 Elsevier Science Ltd. All rights reserved.

1. INTRODUCTION

The computational requirements and the performance of a parallel implementation of the Multisensor Joint Probabilistic Data Association (MSJPDA) Algorithm have been studied in Pao (1994). The fact that computational complexity for the parallel implementation grows exponentially with the number of sensors led to the search for other ways of implementing the MSJPDA algorithm that are less complex and still have comparable performance. A sequential implementation of the MSJPDA algorithm is presented in this paper and will be shown to only have linear growth in complexity with the number of sensors. While parallel and sequential implementations for pure Kalman filtering (i.e. when no data association is required) are equivalent in terms of performance (Willner *et al.*, 1976), this is not true for tracking in a cluttered environment.

Due to the complexity of multisensor multitarget tracking algorithms, the performance of these algorithms is generally compared by running Monte-Carlo simulations. The stochastic nature of tracking algorithms requires numerous simulations to give a reliable comparison. Running these simulations is time-consuming and expensive and furthermore only provides comparison information for parameter sets that are considered in the simulations. It is thus desirable to develop more efficient methods of predicting the performance of tracking algorithms. In this paper, we derive an extension of the hybrid approximation of the covariance propagation (Li and Bar-Shalom, 1991) to multiple sensors. Using this approximation to compare tracking performance requires a fraction of the time that would be needed for Monte-Carlo simulations and is thus much more efficient while at the same time predicting the correct performance difference. While the two multisensor fusion algorithms apply to multiple targets, because of the intractability of extension for multiple targets, the approximation method is only derived for multisensor single-target scenarios. Nevertheless, looking at single-target scenarios still gives a good indication on how different the two MSJPDA implementations perform.

The paper is organized as follows. The multisensor multitarget tracking problem is defined in Section 2, and the parallel and sequential implementations of the MSJPDA algorithm are presented in Section 3. In Section 4, the non-simulation technique for comparing multisensor tracking algorithms is developed. In Section 5, this technique is used to compare the tracking performance of the parallel and sequential implementations of the MSJPDA for an example system. The non-simulation evaluation results are compared with actual Monte-Carlo simulation results, and the computational complexity of the parallel and sequential implementations is also analyzed. Finally, concluding remarks are given in Section 6.

2. MULTISENSOR MULTITARGET TRACKING

The multisensor multitarget tracking problem is to track T targets in clutter with N_s sensors. Measurements (also called reports or returns) from

*Received 4 April 1996; revised 30 January 1997. Parts of this paper have been presented at the 1995 American Control Conference, June 1995, Seattle, USA and the 1996 IFAC World Congress, July 1996, San Francisco, U.S.A. This paper was recommended for publication in revised form by Editor H. Kwakernaak. *Corresponding author* C. W. Frei. Tel. 0041 1632 2846; Fax 0041 1632 1211; E-mail frei@aut.ee.ethz.ch.

†Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH), 8092 Zurich, Switzerland. The author was supported by grants form ABB Management AG, Baden, Switzerland and Landis & Gyr, Zug, Switzerland.

‡ECE Department, University of Colorado at Boulder, Boulder, CO 80309-0425, U.S.A.

the sensors are received by a central processor at discrete-time intervals. The data from the different sensors are assumed to be received synchronously; the algorithms to be discussed can be applied to asynchronous measurements by simply propagating the measurements to a common time. Each measurement can originate from at most one target. Some sensors may not provide measurements at every interval. Some of the measurements arise from targets, and some arise from clutter; some targets may not yield any measurements at all in a particular time interval or for a particular sensor. The probability of detection is assumed to be constant across targets for a given sensor i and will be denoted by P_D^i. Measurement errors due to measurements from one sensor are assumed to be independent of those from another sensor.

Let $\mathbf{x}^t(k)$ $(1 \le t \le T)$ denote the state vectors of each target t at the kth time interval. Suppose the target dynamics are determined by known matrices $\mathbf{F}^t(k)$ and $\mathbf{G}^t(k)$ and random noise vectors $\mathbf{w}^t(k)$ as follows:

$$\mathbf{x}^t(k+1) = \mathbf{F}^t(k)\,\mathbf{x}^t(k) + \mathbf{G}^t(k)\,\mathbf{w}^t(k),$$

where $t = 1, \ldots, T$. The noise vectors $\mathbf{w}^t(k)$ are stochastically independent Gaussian random variables with zero mean and known covariance matrices $\mathbf{Q}^t(k)$.

With N_s sensors, let M_k^i, $i = 1, 2, \ldots, N_s$, be the number of reports from each sensor i at the kth time interval. Assuming a pre-correlation gating process is used to eliminate some of the returns (Bar-Shalom and Fortmann, 1988), let m_k^i denote the number of validated returns from sensor i at time k. The volume of that gate at time k, i, is chosen such that with probability P_G^i the target originated measurements, if there are any, fall into the gate of sensor i. The target originated measurements are determined by

$$\mathbf{z}_{i,l_i}^t(k) = \mathbf{H}_i(k)\,\mathbf{x}^t(k) + \mathbf{v}_i^t(k),$$

where $t = 1, \ldots, T$, $i = 1, \ldots, N_s$, and $1 \le l_i \le M_k^i$. The $\mathbf{H}_i(k)$ matrices are known, each $\mathbf{v}_i^t(k)$ is a zero-mean Gaussian noise vector uncorrelated with all other noise vectors, and the covariance matrices $\mathbf{R}_i(k)$ of the noise vectors $\mathbf{v}_i^t(k)$ are known. For a given target t and sensor i, it is not known which measurement l_i $(1 \le l_i \le M_k^i)$ originates from the target. That is the problem of data association whereby it is necessary to determine which measurements originate from which targets [see Bar-Shalom and Fortmann (1988) for more background]. In any time interval, it is assumed that a target can give rise to at most one measurement from a particular sensor. Measurements not originating from targets are known as false measurements (i.e., clutter), and false measurements are assumed to be uniformly distributed throughout the surveillance region with a density of λ. The number of gated false measurements is therefore usually modeled by a Poisson distribution, that is, $\mu_F(m_k^i)$ is given by

$$\mu_F(m_k^i) = e^{-\lambda V_k^i} \frac{(\lambda V_k^i)^{m_k^i}}{m_k^i!}.$$

Let $\mathscr{Z}(k)$ denote the set of gated measurements at time k:

$$\mathscr{Z}(k) = (\mathbf{z}_{1,1}(k), \ldots, \mathbf{z}_{1,m_k^1}(k), \mathbf{z}_{2,1}(k), \ldots, \mathbf{z}_{2,m_k^2}(k), \ldots, \mathbf{z}_{N_s,1}(k), \ldots, \mathbf{z}_{N_s,m_k^{N_s}}(k)).$$

The t superscripts are not indicated, since it is not known which measurements originate from which target. Finally, let $\mathscr{Z}^k$ denote the sequence of the first k observations, i.e. $\mathscr{Z}^k = (\mathscr{Z}(1), \ldots, \mathscr{Z}(k))$.

3. MULTISENSOR JPDA (MSJPDA)

3.1. *Single sensor JPDA*

For single-sensor tracking, $N_s = 1$ and the goal is to associate the T targets with the m_k^1 measurements based on the current target state estimates and to update these estimates. The actual association being unknown, the conditional estimate is determined by taking a weighted average over all possible associations. For $1 \le t \le T$ and $0 \le l \le m_k^1$, let $\beta_l^t(k)$ denote the conditional probability that measurement l is the true measurement from target t given $\mathscr{Z}^k$. The conditional estimate $\hat{\mathbf{x}}^t(k|k)$ for $\mathbf{x}^t(k)$ given $\mathscr{Z}^k$ is (Bar-Shalom and Fortmann, 1988)

$$\hat{\mathbf{x}}^t(k|k) = \sum_{l=0}^{m_k^1} \beta_l^t(k)\,\hat{\mathbf{x}}_l^t(k|k),$$

where $\hat{\mathbf{x}}_l^t(k|k)$ is the estimate of $\hat{\mathbf{x}}^t(k|k)$ given by the Kalman filter on the basis of the previous estimate and the association of the tth target with the lth measurement.

3.2. *Parallel MSJPDA*

In the parallel implementation of the MSJPDA algorithm (Pao, 1994), all the measurements from all N_s sensors are taken into account in one pass through the multisensor data association and filtering routines (see Fig. 1). For multisensor tracking, the T targets now have to be associated with the m_k^i measurements for each of the N_s sensors. For $1 \le t \le T$ and $\mathscr{L} = (l_1, l_2, \ldots, l_{N_s})$ where $0 \le l_1 \le m_k^1, \ldots, 0 \le l_{N_s} \le m_k^{N_s}$, let $\beta_{\mathscr{L}}^t(k)$ denote the conditional probability of the event that $\mathscr{L}$ is the true set of measurements from the N_s sensors for the kth observation given $\mathscr{Z}^k$, which is computed as

$$\beta_{\mathscr{L}}^t(k) = \prod_{i=1}^{N_s} \beta_{l_i,i}^t(k),$$

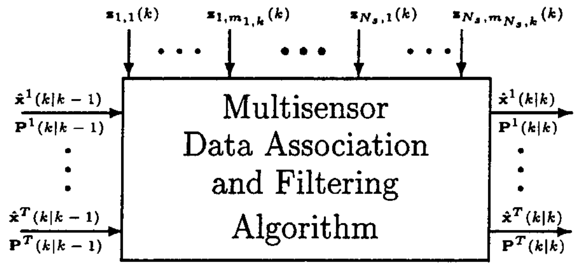

$\mathbf{z}_{1,1}(k)$
$\mathbf{z}_{1,m_{1,k}}(k)$
$\mathbf{z}_{N_s,1}(k)$
$\mathbf{z}_{N_s,m_{N_s,k}}(k)$
$\hat{\mathbf{x}}^1(k|k-1)$
$\mathbf{P}^1(k|k-1)$
$\hat{\mathbf{x}}^T(k|k-1)$
$\mathbf{P}^T(k|k-1)$
Multisensor Data Association and Filtering Algorithm
$\hat{\mathbf{x}}^1(k|k)$
$\mathbf{P}^1(k|k)$
$\hat{\mathbf{x}}^T(k|k)$
$\mathbf{P}^T(k|k)$

Fig. 1. Parallel implementation of multisensor tracking algorithm.

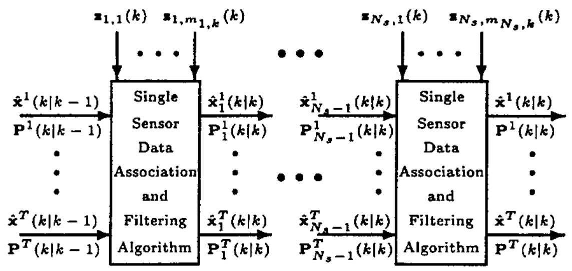

$\mathbf{z}_{1,1}(k)$
$\mathbf{z}_{1,m_{1,k}}(k)$
$\mathbf{z}_{N_s,1}(k)$
$\mathbf{z}_{N_s,m_{N_s,k}}(k)$
$\hat{\mathbf{x}}^1(k|k-1)$
$\mathbf{P}^1(k|k-1)$
$\hat{\mathbf{x}}^T(k|k-1)$
$\mathbf{P}^T(k|k-1)$
Single Sensor Data Association and Filtering Algorithm
$\hat{\mathbf{x}}_1^1(k|k)$
$\mathbf{P}_1^1(k|k)$
$\hat{\mathbf{x}}_1^T(k|k)$
$\mathbf{P}_1^T(k|k)$
$\hat{\mathbf{x}}_{N_s-1}^1(k|k)$
$\mathbf{P}_{N_s-1}^1(k|k)$
$\hat{\mathbf{x}}_{N_s-1}^T(k|k)$
$\mathbf{P}_{N_s-1}^T(k|k)$
Single Sensor Data Association and Filtering Algorithm
$\hat{\mathbf{x}}^1(k|k)$
$\mathbf{P}^1(k|k)$
$\hat{\mathbf{x}}^T(k|k)$
$\mathbf{P}^T(k|k)$

Fig. 2. Sequential implementation of multisensor tracking algorithm.

where the $\beta^t_{l_i,i}(k)$ are just the single-sensor event probabilities described above. The conditional estimates $\hat{\mathbf{x}}^t(k|k)$ for the MSJPDA algorithm are given by

$$\hat{\mathbf{x}}^t(k|k) = \sum_{\mathscr{L}} \beta^t_{\mathscr{L}}(k)\,\hat{\mathbf{x}}^t_{\mathscr{L}}(k|k),$$

where the sum is over all sets of associations $\mathscr{L}$ with target t. The estimate $\hat{\mathbf{x}}^t_{\mathscr{L}}(k|k)$ of $\mathbf{x}^t(k)$ is based on the prediction $\hat{\mathbf{x}}^t(k|k-1)$ and the association of the tth target with the set of $\mathscr{L}$ returns from the N_s sensors. The covariance update corresponding to $\hat{\mathbf{x}}^t_{\mathscr{L}}(k|k)$ is computed by

$$\mathbf{P}^t(k|k) = \sum_{\mathscr{L}} \beta^t_{\mathscr{L}}(k)[\mathbf{P}^t_{\mathscr{L}}(k|k) + \hat{\mathbf{x}}^t_{\mathscr{L}}(k|k)\,\hat{\mathbf{x}}^t_{\mathscr{L}}(k|k)^{\mathrm{T}}]$$
$$- \hat{\mathbf{x}}^t(k|k)\,\hat{\mathbf{x}}^t(k|k)^{\mathrm{T}},$$

where $\mathbf{P}^t_{\mathscr{L}}(k|k)$ are covariances corresponding to $\hat{\mathbf{x}}^t_{\mathscr{L}}(k|k)$.

3.3. *Sequential MSJPDA*

Another way of implementing the MSJPDA algorithm is to process the measurements from each sensor one sensor at the time, as shown in Fig. 2. The measurements of a first sensor are used to compute an intermediate state estimate $\hat{\mathbf{x}}^t_1(k|k)$ and the corresponding covariance $\mathbf{P}^t_1(k|k)$ for each target. The performed computation is equivalent to the one described for the single-sensor case (Section 3.1). The measurements of the next sensor are then used to further improve this intermediate state estimate, again using the single-sensor JPDA filter. With $\hat{\mathbf{x}}^t_i(k|k)$ and $\mathbf{P}^t_i(k|k)$ as the state estimate and covariance, respectively, after processing the data of the i^{th} sensor, the update equations are

$$\hat{\mathbf{x}}^t_i(k|k) = \hat{\mathbf{x}}^t_{i-1}(k|k) + \mathbf{K}^t_i(k) \sum_{l_i=0}^{m^i_k} \beta^t_{l_i,i}(k)$$
$$\times [\mathbf{z}^i_{l_i}(k) - \mathbf{H}_i(k)\,\hat{\mathbf{x}}^t_{i-1}(k|k)], \quad i = 1, \dots, N_s,$$

where $\hat{\mathbf{x}}^t_0(k|k) = \hat{\mathbf{x}}^t(k|k-1)$ and $\hat{\mathbf{x}}^t_{N_s}(k|k) = \hat{\mathbf{x}}^t(k|k)$. With $\mathbf{P}^t_0(k|k) = \mathbf{P}^t(k|k-1)$ and $\mathbf{P}^t_{N_s}(k|k) = \mathbf{P}^t(k|k)$, the update of the covariance matrices is

$$\mathbf{P}^t_i(k|k) = \beta^t_{0,i}(k)\mathbf{P}^t_{i-1}(k|k) + [1 - \beta^t_{0,i}(k)]$$
$$\times [\mathbf{I} - \mathbf{K}^t_i(k)\,\mathbf{H}_i(k)]\mathbf{P}^t_{i-1}(k|k)$$
$$+ \mathbf{K}^t_i(k)\Bigg[\sum_{l_i=0}^{m^i_k} \beta^t_{l_i,i}(k)\mathbf{z}^i_{l_i}(k)\mathbf{z}^i_{l_i}(k)^{\mathrm{T}}$$
$$- \sum_{l_i=0}^{m^i_k} \beta^t_{l_i,i}(k)\mathbf{z}^i_{l_i}(k) \sum_{l_i=0}^{m^i_k} \beta^t_{l_i,i}(k)\mathbf{z}^i_{l_i}(k)^{\mathrm{T}}\Bigg]\mathbf{K}^t_i(k)^{\mathrm{T}},$$
$$i = 1, \dots, N_s.$$

4. MULTISENSOR HYBRID COVARIANCE APPROXIMATION

By extending the hybrid approximation of the covariance propagation proposed in Li and Bar-Shalom (1991) to multiple sensors, a non-simulation comparison of the sequential and parallel implementations of the multisensor probabilistic data association filter (MSPDAF) in terms of the average RMS position (or any other) error and expected track lifetime can be made. Because of the difficulty of taking into account the probability of interfering targets, these hybrid approximations have not been extended to multiple targets. The essence of the extension to multiple sensors is that the approximation of the expected error covariance matrix $E\{\mathbf{P}(k|k)|\mathscr{Z}^k\}$ is conditioned on the number of validated measurements from all N_s sensors. The covariance approximation matrices $\bar{\mathbf{P}}(k|k, m^1_k, \dots, m^{N_s}_k)$ and joint probabilities $P\{m^1_k, \dots, m^{N_s}_k\}$ are computed at each time step. The expected RMS position error can be obtained by iterating the approximation to steady state. Extracting the conditioned position RMS $e(\infty, m^1_\infty, m^2_\infty, \dots, m^{N_s}_\infty)$ from the steady-state matrix $\bar{\mathbf{P}}(\infty, m^1_\infty, m^2_\infty, \dots, m^{N_s}_\infty)$ and averaging over all steady-state probabilities $P\{m^1_\infty, \dots, m^{N_s}_\infty\}$ gives the average RMS position error:

$$e(\infty) = \sum_{m^1_\infty=0}^{N_{\mathrm{T}}} \cdots \sum_{m^{N_s}_\infty=0}^{N_{\mathrm{T}}} e(\infty, m^1_\infty, m^2_\infty, \dots, m^{N_s}_\infty)$$
$$\times P\{m^1_\infty, \dots, m^{N_s}_\infty\}, \tag{1}$$

where the truncation at N_{T} of the otherwise infinite sums is chosen such the resulting approximation error is small. Information about the expected track lifetime can be obtained through the probability of track loss. Extending the definition given for the probability of track loss for single target tracking (Li and Bar-Shalom, 1991), the probability of track loss at time k when there are multiple sensors is defined as the probability that the number of measurements validated for any of the N_s sensors exceeds a certain threshold N_{tl} at time k given that the target was not lost at time $k-1$:

$$P_{tl}(k) \triangleq 1 - \sum_{m_k^1=0}^{N_{tl}-1} \cdots \sum_{m_k^{N_s}}^{N_{tl}-1} P^*\{m_k^1, \ldots, m_k^{N_s}\}, \tag{2}$$

where $P^*\{m_k^1, \ldots, m_k^{N_s}\}$ denotes the probability that a track is lost at time k given that the track was not lost at time $k-1$. The cumulative probability of track loss for evaluation of track lifetime is defined (Li and Bar-Shalom, 1991) as

$$\begin{aligned} P_{TL}(k) &\triangleq 1 - \prod_{i=1}^{k} \left(\sum_{m_i^1=0}^{N_{tl}-1} \cdots \sum_{m_i^{N_s}=0}^{N_{tl}-1} P^*\{m_i^1, \ldots, m_i^{N_s}\} \right) \\ &= P_{TL}(k-1) + [1 - P_{TL}(k-1)]P_{tl}(k) \end{aligned} \tag{3}$$

and a track is deemed lost if $P_{TL}(k)$ exceeds a certain limit P_{lost}. N_T, N_{tl}, and P_{lost} in equations (1)–(3) are tuning parameters, and some guidelines for choosing them are given in Frei (1995). In the next two sections, the hybrid approximation procedure is detailed for $N_s = 2$ only to simplify the illustration of the method. The derivation for these multisensor extensions can be found in Frei (1995).

4.1. *The hybrid approximation for the parallel MSPDAF*

Starting with the conditional covariance approximation $\bar{\mathbf{P}}(k-1|k-1, m_{k-1}^1, m_{k-1}^2)$ and the joint probabilities $P\{m_{k-1}^1, m_{k-1}^2\}$, the one-step prediction is

$$\begin{aligned} &\mathbf{P}(k|k-1, m_{k-1}^1, m_{k-1}^2) \\ &\quad = \mathbf{F}(k-1)\bar{\mathbf{P}}(k-1|k-1, m_{k-1}^1, m_{k-1}^2)\,\mathbf{F}(k-1)^T \\ &\qquad + \mathbf{G}(k-1)\,\mathbf{Q}(k-1)\,\mathbf{G}(k-1)^T, \end{aligned}$$

$$\begin{aligned} &\mathbf{S}_i(k, m_{k-1}^1, m_{k-1}^2) \\ &\quad = \mathbf{H}_i(k)\mathbf{P}(k|k-1, m_{k-1}^1, m_{k-1}^2)\mathbf{H}_i(k)^T \\ &\qquad + \mathbf{R}_i(k), \\ &\mathbf{K}_i(k, m_{k-1}^1, m_{k-1}^2) \\ &\quad = \left[\mathbf{P}(k|k-1, m_{k-1}^1, m_{k-1}^2)^{-1} \right. \\ &\qquad \left. + \sum_{i=1}^{2} \mathbf{H}_i^T(k)\mathbf{R}_i(k)^{-1}\mathbf{H}_i(k)\right]^{-1} \\ &\qquad \times \mathbf{H}_i^T(k)\,\mathbf{R}_i(k)^{-1}. \end{aligned} \tag{4}$$

The approximation of $E\{\mathbf{P}(k|k)\}$ conditioned on the number of validated measurements at the current time k and the previous time $k-1$ is then

$$\begin{aligned} &\bar{\mathbf{P}}(k|k, m_k^1, m_k^2, m_{k-1}^1, m_{k-1}^2) \\ &\quad = \bar{\beta}_{0,1}\bar{\beta}_{0,2}\mathbf{P}(k|k-1, m_{k-1}^1, m_{k-1}^2) \\ &\qquad + \bar{\beta}_{0,1}[1 - \bar{\beta}_{0,2}]\mathbf{P}_2(k|k, m_{k-1}^1, m_{k-1}^2) \\ &\qquad + \bar{\beta}_{0,2}[1 - \bar{\beta}_{0,1}]\mathbf{P}_1(k|k, m_{k-1}^1, m_{k-1}^2) \\ &\qquad + [1 - \bar{\beta}_{0,1}][1 - \bar{\beta}_{0,2}]\mathbf{P}_{2\text{sensors}}(k|k, m_{k-1}^1, m_{k-1}^2) \\ &\qquad + \sum_{i=1}^{2} [u_1(m_k^i, m_{k-1}^1, m_{k-1}^2) - u_2(m_k^i, m_{k-1}^1, m_{k-1}^2)] \\ &\qquad \times \mathbf{K}_i(k, m_{k-1}^1, m_{k-1}^2)\mathbf{S}_i(k, m_{k-1}^1, m_{k-1}^2) \\ &\qquad \times \mathbf{K}_i(k, m_{k-1}^1, m_{k-1}^2)^T, \end{aligned} \tag{5}$$

where

$$\begin{aligned} \bar{\beta}_{0,i} &= \bar{\beta}_{0,i}(k, m_k^i, m_{k-1}^1, m_{k-1}^2) \\ &= \frac{(1 - P_D^i P_G^i)\,\lambda V_i(k, m_{k-1}^1, m_{k-1}^2)}{(1 - P_D^i P_G^i)\,\lambda V_i(k, m_{k-1}^1, m_{k-1}^2) + P_D^i P_G^i m_k^i}, \end{aligned} \tag{6}$$

$$u_1(m_k^i, m_{k-1}^1, m_{k-1}^2) = \begin{cases} 0, & m_k^i = 0, \\ \dfrac{P_D^i}{(1 - P_D^i P_G^i)\dfrac{\lambda V_i(k, m_{k-1}^1, m_{k-1}^2)}{m_k^i} + P_D^i P_G^i} I_1, & m_k^i > 0. \end{cases} \tag{7}$$

$$u_2(m_k^i, m_{k-1}^1, m_{k-1}^2) = \begin{cases} 0, & m_k^i = 0, \\ \dfrac{P_D^i \dfrac{C_M}{(2\pi)^{M/2}} \left(\dfrac{M}{g^M}\right)^{m_k^i - 1} I_2(m_k^i)}{(1 - P_D^i P_G^i)\dfrac{\lambda V_i(k, m_{k-1}^1, m_{k-1}^2)}{m_k^i} + P_D^i P_G^i} I_1, & m_k^i > 0. \end{cases} \tag{8}$$

$V_i(k, m_{k-1}^1, m_{k-1}^2)$ denotes the volume of the validation gate of sensor i at time k given that there were m_{k-1}^1 and m_{k-1}^2 measurements validated at time

$k-1$:

$$V_i(\cdot) = c_M g^M |\mathbf{S}_i(\cdot)|^{1/2}, \quad i = 1, 2.$$

M denotes the dimension of the measurement space, c_M denotes the volume of the M-dimensional unit sphere, and g is a parameter determining the size of the validation gate. $\mathbf{P}_{2\text{sensors}}(\cdot)$ is the covariance for pure Kalman filtering with two sensors. $\mathbf{P}_1(\cdot)$ and $\mathbf{P}_2(\cdot)$ denote the covariance matrices obtained if for either sensor (2 and 1, respectively) no return is associated with the target. Finally, the integrals I_1 and I_2 are

$$I_1 = \int_0^g r^{M+1} e^{-r^2/2}\,dr,$$

$$I_2(m_k^i) = \int_0^g \cdots \int_0^g \frac{e^{-r_1^2} r_1^2 (r_1 r_2 \cdots r_{m_k^i})^{M-1}}{b + \sum_{j=1}^{m_k^i} e^{-r_j^2/2}}\,dr_1 \cdots dr_{m_k^i},$$

$$b = (2\pi)^{M/2} \left(\frac{\lambda V_i(k, m_{k-1}^1, m_{k-1}^2)}{c_M g^M} \right) \frac{(1 - P_D^i P_G^i)}{P_D^i}.$$

The integrals I_1 and $I_2(m_k^i)$ can be computed numerically. I_1 is usually constant for a particular application, assuming a constant validation probability and $I_2(m_k^i)$ can be interpolated from a table of $I_2(m_k^i)$ values that are computed off line.

The forward transition probabilities are

$$P\{m_k^i | m_{k-1}^1, m_{k-1}^2\} = \begin{cases} (1 - P_D^i P_G^i)\mu_F(m_k^i | m_{k-1}^1, m_{k-1}^2), & m_k^i = 0, \\ (1 - P_D^i P_G^i)\mu_F(m_k^i | m_{k-1}^1, m_{k-1}^2) + P_D^i P_G^i \mu_F(m_k^i - 1 | m_{k-1}^1, m_{k-1}^2), & m_k^i > 0, \end{cases} \quad (9)$$

where the distribution of false measurements μ_F is determined by a Poisson model assumption. The joint forward probabilities can be obtained from them as

$$P\{m_k^1, m_k^2 | m_{k-1}^1, m_{k-1}^2\} = P\{m_k^1 | m_{k-1}^1, m_{k-1}^2\} P\{m_k^2 | m_{k-1}^1, m_{k-1}^2\}.$$

Finally, the marginal probabilities are computed from the joint probabilities as

$$P\{m_k^1, m_k^2\} = \sum_{m_{k-1}^1=0}^{\infty} \sum_{m_{k-1}^2=0}^{\infty} P\{m_k^1, m_k^2 | m_{k-1}^1, m_{k-1}^2\} P\{m_{k-1}^1, m_{k-1}^2\}.$$

Using Bayes' rule to obtain the backward transition probabilities gives

$$P\{m_{k-1}^1, m_{k-1}^2 | m_k^1, m_k^2\} = \frac{P\{m_k^1, m_k^2 | m_{k-1}^1, m_{k-1}^2\} P\{m_{k-1}^1, m_{k-1}^2\}}{P\{m_k^1, m_k^2\}}. \quad (10)$$

With (5) and (10), the approximation $\bar{\mathbf{P}}(k|k, m_k^1, m_k^2)$ can be computed as

$$\bar{\mathbf{P}}(k|k, m_k^1, m_k^2) = \sum_{m_{k-1}^1=0}^{\infty} \sum_{m_{k-1}^2}^{\infty} \bar{\mathbf{P}}(k|k, m_k^1, m_k^2, m_{k-1}^1, m_{k-1}^2) \times P\{m_{k-1}^1, m_{k-1}^2 | m_k^1, m_k^2\}. \quad (11)$$

Iterating equations (4), (5), and (11) to steady-state yields $\bar{\mathbf{P}}(\infty, m_\infty^1, m_\infty^2)$.

4.2. *The hybrid approximation for the sequential MSPDAF*

Starting with the conditional covariance approximation $\bar{\mathbf{P}}(k-1|k-1, m_{k-1}^1, m_{k-1}^2)$ and the joint probabilities $P\{m_{k-1}^1, m_{k-1}^2\}$, $\mathbf{P}(k|k-1, m_{k-1}^1, m_{k-1}^2)$ and $\mathbf{S}_1(k, m_{k-1}^1, m_{k-1}^2)$ are computed according to equation (4) and the filter gain is

$$\mathbf{K}_1(k, m_{k-1}^1, m_{k-1}^2) = \mathbf{P}(k|k-1, m_{k-1}^1, m_{k-1}^2) \times \mathbf{H}_1^T(k)\, \mathbf{S}_1^{-1}(k, m_{k-1}^1, m_{k-1}^2).$$

The intermediate approximation of the covariance, conditioned on the measurements from both sensors of the previous time $k-1$ and the measurements of the first sensor at current time k, is a function of the number of gated measurements from the previous time (m_{k-1}^1 and m_{k-1}^2) and of the number of measurements gated by the first sensor at the current time (m_k^1). It is also the conditional covariance prediction for the second sensor:

$$\begin{aligned} \bar{\mathbf{P}}_1(k|k, m_k^1, m_{k-1}^1, m_{k-1}^2) &= \mathbf{P}(k|k-1, m_{k-1}^1, m_{k-1}^2) \\ &\quad - [1 - \bar{\beta}_{0,1} - u_1(m_k^1, m_{k-1}^1, m_{k-1}^2) \\ &\quad + u_2(m_k^1, m_{k-1}^1, m_{k-1}^2)]\mathbf{K}_1(k, m_{k-1}^1, m_{k-1}^2) \\ &\quad \times \mathbf{S}_1(k, m_{k-1}^1, m_{k-1}^2)\, \mathbf{K}_1(k, m_{k-1}^1, m_{k-1}^2)^T \end{aligned}$$

where $\bar{\beta}_{0,1}$, u_1, and u_2 are computed according to equations (6), (7), and (8), respectively. The innovations covariance and the filter gain for the second sensor are thus

$$\mathbf{S}_2(k, m_k^1, m_{k-1}^1, m_{k-1}^2) = \mathbf{H}_2(k)\, \bar{\mathbf{P}}_1(k|k, m_k^1, m_{k-1}^1, m_{k-1}^2)\, \mathbf{H}_2(k)^T + \mathbf{R}_2(k),$$

$$\mathbf{K}_2(k, m_k^1, m_{k-1}^1, m_{k-1}^2) = \bar{\mathbf{P}}_1(k|k, m_k^1, m_{k-1}^1, m_{k-1}^2) \times \mathbf{H}_2^T(k) \mathbf{S}_2^{-1}(k, m_k^1, m_{k-1}^1, m_{k-1}^2)$$

Table 1. Approximate operation counts for parallel and sequential MSJPDA

Procedure	Parallel MSJPDA	Sequential MSJPDA
Gating	$\mathcal{O}\left(Ts_{T^2}\sum_{i=1}^{N_s} M_k^i s_i\right)$	$\mathcal{O}\left(Ts_{\mathrm{T}^2}\sum_{i=1}^{N_s} M_k^i s_i\right)$
Likelihood	$\mathcal{O}\left(T\sum_{i=1}^{N_s} m_k^i s_i^3\right)$	$\mathcal{O}\left(T\sum_{i=1}^{N_s} m_k^i s_i^3\right)$
Association probabilities	$\mathcal{O}\left(T2^{T-1}\sum_{i=1}^{N_s} m_k^i(m_k^i - 1)\right)$	$\mathcal{O}\left(T2^{\mathrm{T}-2}\sum_{i=1}^{N_s} m_k^i(m_k^i - 1)\right)$
State estimates	$\mathcal{O}\left(T\sum_{i=1}^{N_s} m_k^i s_i\right)$	$\mathcal{O}\left(T\sum_{i=1}^{N_s} m_k^i s_i\right)$
State covariances	$\mathcal{O}\left(Ts_{\mathrm{T}^3 2^{N_s}} + T\sum_{i=1}^{N_s} m_k^i s_i^2\right)$	$\mathcal{O}\left(T\sum_{i=1}^{N_s} m_k^i s_i^2\right)$

and the approximation of $E\{\mathbf{P}(k|k)\}$ conditioned on the number of gated measurements in both sensors at time $k-1$ and time k is now a function of the number of all the validated measurements at the previous time (m_{k-1}^1 and m_{k-1}^2) and the current time (m_k^1 and m_k^2):

$$\begin{aligned}
\bar{\mathbf{P}}&(k|k, m_k^1, m_k^2, m_{k-1}^1, m_{k-1}^2)\\
&= \bar{\mathbf{P}}_1(k|k, m_k^1, m_{k-1}^1, m_{k-1}^2)\\
&\quad - [1 - \bar{\beta}_{0,2} - u_1(m_k^1, m_k^2, m_{k-1}^1, m_{k-1}^2)\\
&\quad + u_2(m_k^1, m_k^2, m_{k-1}^1, m_{k-1}^2)]\\
&\quad \times \mathbf{K}_2(k, m_k^1, m_{k-1}^1, m_{k-1}^2)\mathbf{S}_2(k, m_k^1, m_{k-1}^1, m_{k-1}^2)\\
&\quad \times \mathbf{K}_2(k, m_k^1, m_{k-1}^1, m_{k-1}^2)^{\mathrm{T}} \qquad (12)
\end{aligned}$$

where again equations (6), (7), and (8) apply for the computation of $\bar{\beta}_{0,2}$, u_1, and u_2 with the appropriate modifications.

The forward transition probabilities for the first sensor $P\{m_k^1|m_{k-1}^1, m_{k-1}^2\}$ and for the second sensor $P\{m_k^2|m_k^1, m_{k-1}^1, m_{k-1}^2\}$ are (with the appropriate conditioning for sensor two) both computed according to (9). The joint forward transition probabilities can then be computed as

$$\begin{aligned}
P&\{m_k^1, m_k^2|m_{k-1}^1, m_{k-1}^2\}\\
&= P\{m_k^1|m_{k-1}^1, m_{k-1}^2\}P\{m_k^2|m_k^1, m_{k-1}^1, m_{k-1}^2\}. \qquad (13)
\end{aligned}$$

Computing the remaining (backward transition and marginal) probabilities follows the same procedure as for the parallel hybrid approximation, and the approximation $\bar{\mathbf{P}}(k|k, m_k^1, m_k^2)$ can then also be computed according to (11).

5. COMPARISON

5.1. *Comparisons of computational complexity*

If tracking algorithms are supposed to track in real time, the computational requirements play an important role as one observation interval provides very limited time to perform the filtering computations. The computational complexity becomes especially important as the number of sensors and the clutter density grow.

Table 1 shows the order of operation counts for the various parts of the parallel and the sequential implementations of the MSJPDA algorithm. An addition/subtraction and multiplication/division are counted as one operation. The variable s_{T} denotes the number of elements in the target state vector and s_i denotes the size of the measurement vector of the ith sensor. Details illustrating how these approximate computational complexity expressions are computed are provided in Frei (1995).

We see that the computational complexity of both algorithms is equivalent except in the covariance update routines. It can be seen that the complexity of the parallel implementation grows exponentially with the number of sensors while the complexity of the sequential implementation only shows linear growth with the number of sensors.

5.2. *Comparison of tracking performance*

In this section, the multisensor hybrid approximation will be applied to a particular sample system defined in Pao (1994), and the results are compared with simulations for the same system. The simulations were run for tracking two targets moving in two dimensions with random acceleration and measurement noise. The clutter density λ was varied from 0.2 to 1.0. For the simulation parameters, the expected number of false measurements per validation region, using steady-state Kalman filter covariances, varies from 0.38 to 1.92.

Figure 3 shows the average track lifetimes for parallel and sequential implementations of the algorithm as the clutter density is varied. Both simulation and non-simulation results are shown in the figure. As expected, the average track lifetime decreases as the clutter density increases. The simulations show the same trend in the relative performance, but they do not yield the same values for the

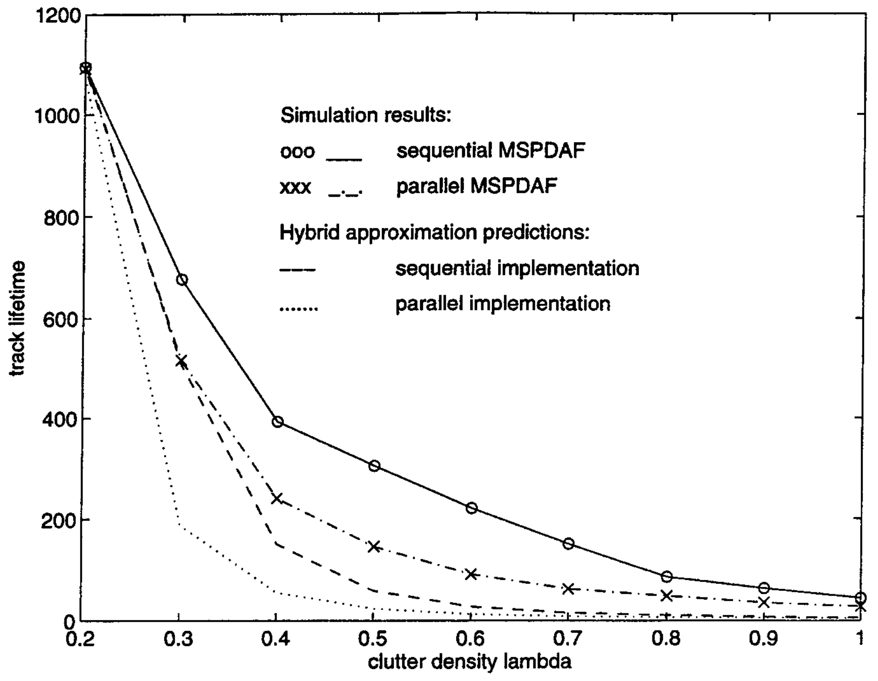


Fig. 3. Variation of track lifetime with clutter density λ. Parameters: $\mathbf{Q}^1 = \mathbf{Q}^2 = \mathbf{R}_1 = \mathbf{R}_2 = \mathrm{diag}(0.0144, 0.0144)$, $P_G^1 = P_G^2 = 0.999$, $N_T = 19$, $N_{tl} = 5$, and $P_{lost} = 0.9$.

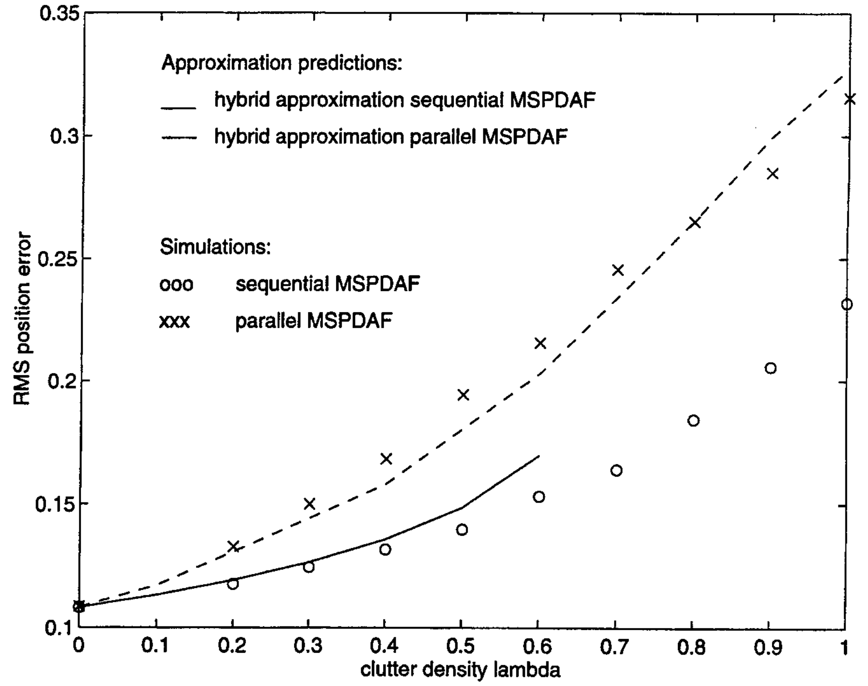


Fig. 4. Variation of RMS position error with clutter density λ. Parameters: $\mathbf{Q}^1 = \mathbf{Q}^2 = \mathbf{R}_1 = \mathbf{R}_2 = \mathrm{diag}(0.0144, 0.0144)$, $P_G^1 = P_G^2 = 0.999$, and $N_T = 19$.

average track lifetime because track lifetime was defined differently for the simulations and for the multisensor hybrid approximation. Clearly, the sequential implementation yields longer track lifetimes over the whole range of clutter density values.

Figure 4 presents similar results for the average RMS position errors. The values obtained through approximation methods are shown together with simulations results on the same plot. Multisensor extensions (Frei, 1995) of the modified Riccati equation (MRE) (Bar-Shalom and Fortmann, 1988) are also capable of providing predictions for average RMS errors but not for average track lifetimes and has therefore not been discussed in this paper. With the RMS position error being defined in the same manner for simulations and approximations, there is a good match of the RMS predictions with the simulation results. As expected, RMS position errors increase as the clutter density increases. Further, sequential filtering yields lower RMS position error, as predicted by the hybrid approximation.

6. DISCUSSION AND CONCLUSIONS

A non-simulation technique for comparing single-sensor tracking algorithms, the hybrid approximation of the covariance propagation, has been extended for comparing parallel and sequential implementations of the multisensor probabilistic data association filter. The method allows comparison in terms of both RMS position error and track lifetime. While this technique does not predict the performance of either implementation exactly in all cases, it has been shown to always correctly predict the qualitative performance differences between the two implementations observed in simulations. Even though the hybrid approximations gives the right predictions for the qualitative performance difference between the parallel and the sequential implementations for all parameter sets studied, a more systematic comparison of predictions and simulations should be done to validate the use of the developed approximations, with parameter sets chosen to represent all possible extreme situations.

Based on our approximations and simulations, a superior performance (in terms of RMS position error and track lifetime metrics) of the sequential implementation of the MSJPDA over the parallel implementation has been shown. The sequential implementation has also been shown to be less computationally complex than the parallel implementation.

For the sequential implementation, a question that arises is which sensor's data should be processed first if the sensors do not possess equal characteristics. Our studies with two unequal sensors show that processing measurements from the better sensor first leads to longer expected track lifetimes but larger expected RMS position errors; more details can be found in Frei (1995).

A heuristic explanation on why the sequential implementation yields superior tracking performance is as follows. The parallel implementation uses the predicted measurements and covariances based on the state estimates and covariances of the previous interval for data association and filtering of measurements from all sensors in the current interval, whereas the sequential implementation only uses this information for data association and filtering of measurements for the first sensor. After processing data from this first sensor, better estimates are available which are then used for data association and filtering of measurements from the next sensor. Thus, successively better estimates are used for data association and filtering for each subsequent sensor, in that (i) unlikely measurements which are considered for the parallel filtering might be rejected for the sequential filtering and (ii) different (probably better) association probabilities are obtained.

Acknowledgements—This work was supported in part by an Office of Naval Research Young Investigator Award, the Colorado Advanced Software Institute, Data Fusion Corporation, and a National Science Foundation CAREER Award under Grant CMS–9625086.

REFERENCES

Bar-Shalom, Y. and T. E. Fortmann (1988). *Tracking and Data Association.* Academic Press, San Diego.

Frei, C. W. (1995). Comparision of parallel and sequential implementations of a multisensor multitarget tracking algorithm. Master's thesis. Northwestern University.

Li, X. and Y. Bar-Shalom (1991). 'Stability evaluation and track-life of the PDAF for tracking in clutter. *IEEE Trans. Automat. Control*, **36**(5), 581–602.

Pao, L. (1994). 'Centralized multisensor fusion algorithms for tracking applications'. *Control Enging Practice* **2**(5), 875–887.

Willner, D., C. B. Chang and K. P. Dunn (1976). Kalman filter algorithms for a multi-sensor system. *Proc. IEEE Conf. Decision and Control*, Clearwater, FL, USA, December 1–3, 1976, pp. 570–574.

Proceedings of the American Control Conference
Albuquerque, New Mexico June 1997
0-7803-3832-4/97/$10.00 © 1997 AACC

Algorithms for a Class of Distributed Architecture Tracking

Lucy Y. Pao and Michael Kalandros

Electrical and Computer Engineering Department
University of Colorado
Boulder, CO 80309-0425
pao@colorado.edu, kalandro@colorado.edu

Abstract

Multi-sensor target tracking has traditionally been performed using a single processor to monitor several sensors (centralized fusion), but this method is demanding of both computational power and communication bandwidth. Distributed sensor fusion is a method of addressing these limitations. However, the distributed sensor fusion problem is more complex due to the correlation of separate track estimates. A method known as measurement reconstruction has recently been shown to address this problem in a specific architecture. This paper extends the measurement reconstruction approach to a more generalized architecture using two new algorithms. Computational and communication requirements are compared with centralized sensor fusion, and Monte Carlo simulation studies are used to compare the performance of these algorithms.

1. Introduction

Target tracking in various environments presents several challenges including noise, target clutter, and multiple, interacting targets. The Kalman filter is an optimal solution to dealing with noise both in the sensor as well as in the target (assuming linear motion in the target). The Joint Probabilistic Data Association (JPDA) filter has proven to be very effective in dealing with clutter and multiple targets by combining the measurements into a single weighted average for each of the targets [2]. The use of multiple sensors has improved the performance of these algorithms. However, the cost of this improvement is a dramatic increase in computational complexity. The number of computations increases exponentially with the number of sensors and targets and quadratically with the number of measurements [7, 9]. A sequential implementation of the JPDA/Kalman filter can yield superior performance over a parallel implementation while limiting the computational complexity due to the number of sensors to linear growth [6, 9]. Because of this, we will use only the sequential JPDA Kalman filter as a basis of our distributed fusion techniques.

Another method of reducing the computational load is to spread the work over several processors in a technique known as distributed sensor fusion. Distributed fusion has the advantage of not only decreasing the demand on the processor, but increasing the resistance of the tracking system to damage. The system will be able to lose a processor and continue to track targets. The disadvantages include the loss of information to the global processor and the cross-correlation of the track estimates [1, 3].

Previous studies in distributed tracking have considered specific local and global algorithms and specific processing architectures and communication schemes [4, 5]. In [8], a technique known as measurement reconstruction was presented that can be used with a variety of tracking algorithms and is an effective method of reducing computational complexity, especially in high clutter environments. This paper expands measurement reconstruction techniques for a more general class of distributed architectures. A brief review of both the sequential Kalman filter and JPDA algorithms is given in section 2. General information on distributed architectures and measurement reconstruction to date is presented in section 3, and algorithms for a more general architecture are presented in sections 3.1 and 3.2. The computational complexity of the algorithms is examined in section 4, and simulation results are presented in section 5. Finally, concluding remarks are given in section 6.

2. Sequential Kalman and JPDA Filtering

The sequential Kalman filter is an algorithm for estimating stochastic or slightly non-linear systems using a state space representation. The Kalman filter is based on the following assumptions about the target and measurement systems:

$$x^t(k) = F^t x^t(k-1) + G^t u^t(k-1) + q^t(k-1) \quad (1)$$
$$z_j^t(k) = H_j x^t(k) + w_j(k) \quad (2)$$

where $x^t(k)$ is the current state estimate of target t; F^t, G^t, H_j are known system matrices; $u^t(k)$ is the control signal; and $z_j^t(k)$ is a measurement of target t from sensor j. $q^t(k)$ is a variable representing process noise or higher-order motion, and $w_j(k)$ is a variable representing measurement noise or error. Both $q^t(k)$ and $w_j(k)$ are assumed to have zero-mean, white, Gaussian probability distributions.

the target states and measurements in the next time interval can be predicted by:

$$\hat{x}^t(k \mid k-1) = F^t \hat{x}^t(k-1 \mid k-1) + G^t u^t(k-1) \quad (3)$$
$$\hat{z}_j^t(k) = H_j \hat{x}^t(k \mid k-1) \quad (4)$$

The input $u(k)$ is considered known and will be omitted in future equations because it can be easily reinserted. The quantity $v_j^t(k) = \hat{z}_j^t(k) - z_j^t(k)$ is known as the innovation. The predicted covariance of the state and innovation can be found by:

$$P^t(k \mid k-1) = F^t P^t(k-1 \mid k-1)(F^t)' + Q^t(k-1) \quad (5)$$
$$S_j^t(k+1 \mid k) = H_j P(k+1 \mid k) H_j' + R_j(k) \quad (6)$$

where $Q^t(k)$ and $R_j(k)$ are the covariances of the noise in the plant and the sensor, respectively.

The sequential algorithm runs a separate Kalman filter for each sensor [10]:

$$\begin{aligned}\hat{x}_1^t(k \mid k) &= \hat{x}^t(k \mid k-1) + K_1^t(k)(z_1^t(k) \\ &\quad - H_1 \hat{x}^t(k \mid k-1)) \quad (7)\\ \hat{x}_j^t(k \mid k) &= \hat{x}_{j-1}^t(k \mid k) + K_j^t(k)(z_j^t(k) \\ &\quad - H_j \hat{x}_{j-1}^t(k \mid k)), \quad j = 2, \ldots, N_s \quad (8)\\ \hat{x}^t(k \mid k) &= \hat{x}_{N_s}^t(k \mid k) \quad (9)\end{aligned}$$

where

$$K_1^t(k) = P^t(k \mid k-1) H_1' R_1^{-1}(k) \quad (10)$$
$$K_j^t(k) = P_{j-1}^t(k \mid k) H_j' R_j^{-1}(k), \quad j = 2, \ldots, N_s \quad (11)$$

The state covariance is updated for each filter by:

$$\begin{aligned}P_1^t(k \mid k) &= (I - K_1^t(k) H_1) P^t(k \mid k-1) \quad (12)\\ P_j^t(k \mid k) &= (I - K_j^t(k) H_j) P_{j-1}^t(k \mid k), \\ &\quad j = 2, \ldots, N_s \quad (13)\\ P^t(k \mid k) &= P_{N_s}^t(k \mid k) \quad (14)\end{aligned}$$

Once the state and covariance estimates have been updated, they are fed back into the algorithm and the entire process is repeated for the new set of measurements at the next time step.

In the JPDA filter [2], the measurement in the Kalman filter is replaced by the combined measurement — a sum of the measurements, each weighted by the probability of that measurement being the actual return from the target:

$$z_j^t(k) = \sum_{m=0}^{m_k} z_{m,j}(k) \beta_{m,j}^t(k) \quad (15)$$

where $z_{m,j}(k)$ is measurement m of sensor j at time k, $\beta_{m,j}^t(k)$ is the probability that measurement $z_{m,j}(k)$ is the true measurement of target t, and m_k is the number of that the target was not detected during scan k, in which case $z_{m,j}(k) = \hat{z}_j^t(k)$. The combined measurement is then used in the Kalman Filter to compute the state covariance as

$$\begin{aligned}P^t(k \mid k) &= \beta_{0,j}^t(k) P^t(k \mid k-1) \\ &\quad + (1 - \beta_{0,j}^t(k)) P^C(k \mid k) + \tilde{P}^t(k) \quad (16)\\ P^C(k) &= (I - K_j^t(k) H_j) P^t(k \mid k-1) \\ \tilde{P}^t(k) &= K_j^t(k) \Big[\sum_{i=1}^{m_k} \beta_{i,j}(k) v_{i,j}(k) v_{i,j}'(k) \\ &\quad - v(k) v'(k) \Big] (K_j^t)'(k)\end{aligned}$$

All other quantities in the Kalman Filter are calculated as discussed above.

3. Distributed Fusion

In distributed sensor fusion, several microprocessors monitor the sensor outputs instead of only one. Usually the sensors themselves will be divided among several processors (known as local processors). Each local processor runs its own sensor fusion algorithm and passes its estimates of the targets along with any other necessary information to the global processor. The global processor takes the local estimates from each processor and computes its own target state estimates based on this information.

The estimates from different local processors for a given target are biased since each processor is affected by the same target process noise. Because of this, the state estimates are no longer normally distributed and most of the assumptions made by the Kalman and JPDA algorithms no longer hold. Calculations to take this cross-correlation into account are complicated enough to negate most of the gains in computational efficiency made by moving to a distributed architecture [1, 3, 4]. One method of dealing with this problem is known as pseudo-measurement reconstruction, where the original measurements or combined measurements are extracted from the state estimates of the local processors [8]. Since the measurement errors are independent across the sensors, the pseudo-measurements will also be uncorrelated.

There are various distributed fusion architectures. In [8], the measurement reconstruction approach was developed and demonstrated on an architecture where each local processor monitors one sensor and transmits information to a global processor. Each local processo may be tracking different targets or following false tracks Measurement reconstruction simply involves solving th Kalman filter equation for the measurement quantity

$$\begin{aligned}\bar{z}_i^t(k) &= (K_i^t(k))^\dagger (\hat{x}_i^t(k \mid k) - \hat{x}_i^t(k \mid k-1)) \\ &\quad + H_i \hat{x}_i^t(k \mid k-1) \quad (17\end{aligned}$$

where i is the processor number and $(K_i^t(k))^\dagger$ is the pseudo-inverse of $K_i^t(k)$. In the case of JPDA, $\tilde{z}_i^t(k)$ is the combined measurement used by processor i to arrive at the local estimate of target t. This reconstruction process is performed for each sensor and requires each of the N_s processors to transmit its set of state estimates $\hat{x}_i^t(k \mid k)$ and Kalman filter gains $K_i^t(k)$ to the global processor. The F^t and H_i matrices are assumed to be known by the global processor and $\hat{x}_i^t(k \mid k-1)$ can be calculated using F^t and the previous state estimate.

Once the pseudo-measurements are recovered, they can be fused at the central processor the same way as normal measurements would be, except that each pseudo-measurement represents a target — the clutter points have essentially been eliminated by the local processors. Note that since each processor is following its own targets, the central processor may apply the pseudo-measurement of the local processor's target to a completely different target.

Figure 1 shows a more general distributed architecture where each local processor monitors several sensors, and we shall extend the measurement reconstruction approach for this architecture. Measurement reconstruction for the architecture of Figure 1 is not as straightforward as in [8]. In particular, the state estimate from a local processor is now produced by measurements from more than one sensor, each one with a different Kalman filter gain matrix. Making the task more difficult is the fact that the sequential multi-sensor Kalman filter algorithm results in a combination of measurements that is much more complicated than simply creating a sum of pseudo-measurements weighted by multi-sensor Kalman filter gains (this would be the case in a parallel implementation) [9]. The following two subsections present two techniques for extending measurement reconstruction for the more general architecture of Figure 1.

3.1. Individual Estimate Reconstruction

One reconstruction method for the distributed architecture in Figure 1 is to use a simple modification of the reconstruction technique in [8]:

$$\tilde{z}_{i,1}^t(k) = (K_{i,1}^t(k))^{-1}(\hat{x}_{i,1}^t(k \mid k) - \hat{x}_i^t(k \mid k-1)) + H_1 x_i^t(k \mid k-1) \quad (18)$$

$$\tilde{z}_{i,j}^t(k) = (K_{i,j}^t(k))^{-1}(x_{i,j}^t(k \mid k) - \hat{x}_{i,j-1}^t(k \mid k)) + H_j \hat{x}_{i,j-1}^t(k \mid k), \quad j = 2, \ldots, N_s \quad (19)$$

where i is the number of the local processor, j is the number of the sensor of that processor and N_{s_i} is the total number of sensors for processor i. With this Individual Estimate Reconstruction, it is possible to reconstruct the combined measurement for each sensor using the state estimates and Kalman gain matrices generated for each sensor (see equations (7)-(11)).

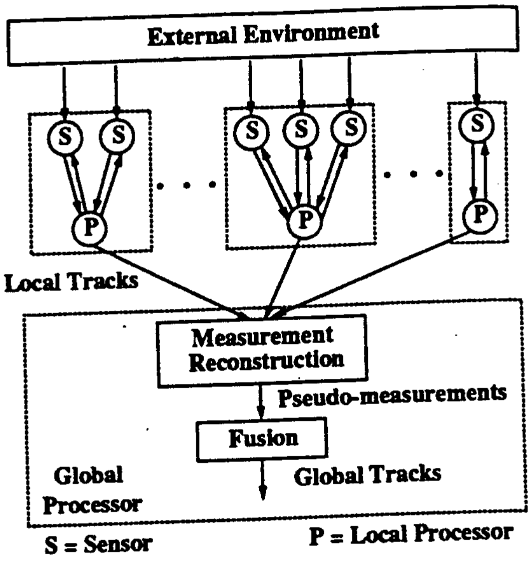


Figure 1: Distributed Sensor Fusion Architecture.

3.2. Combined Estimate Reconstruction

While Individual Estimate Reconstruction is an effective method of measurement reconstruction, it is possible to reduce the amount of data transmitted to the global processor, and hence the amount of bandwidth required by the system. The following Combined Estimate Reconstruction requires transmitting only the local state estimates formed by each processor and the Kalman filter gains from each sensor.

Since each of the local state estimates is generated using only the previous estimates and the new measurements, it should be possible to extract the individual components of the sequential Kalman algorithm. The state estimate at the end of the sequential Kalman filter algorithm can be expressed as

$$\begin{aligned}\hat{x}^t(k \mid k) &= \prod_{l=1}^{N_s}(I - K_l^t(k)H_l)\hat{x}^t(k \mid k-1) \\ &\quad + \sum_{q=1}^{N_s-1}\prod_{j=q+1}^{N_s}(I - K_j(k)^t H_j)K_q(k)^t z_q^t(k) \\ &\quad + K_{N_s}^t(k) z_{N_s}^t(k) \quad (20)\\ &= C_0^t(k)\hat{x}^t(k \mid k-1) \\ &\quad + \sum_{j=1}^{N_s} C_j^t(k)K_j^t(k)z_j^t(k) \quad (21)\end{aligned}$$

where $C_j^t(k)$ is defined as

$$C_{N_s}^t(k) = I \quad (22)$$

$$C_j^t(k) = C_{j+1}^t(k)(I - K_{j+1}^t(k)H_{j+1}), \quad j = 1, \ldots, N_s - 1 \quad (23)$$

$$C_0^t(k) = C_1^t(k)(I - K_1^t(k)H_1) \quad (24)$$

By moving the state estimates and measurements to opposite sides of the equation, we obtain

$$\sum_{j=1}^{N_s} C_j^t(k)K_j^t(k)z_j^t(k) = x^t(k \mid k) - C_0^t(k)\hat{x}^t(k \mid k-1) \quad (25)$$

Pre-multiplying both sides of the equation by $(\sum_{j=1}^{N_s} C_j^t(k)K_j^t(k))^\dagger$ creates, on the left-hand side, a sum of weighted measurements where the weights sum to one and that represent the effect each measurement has on the algorithm:

$$\begin{aligned} \tilde{z}_{t,i}(k) &= \sum_{j=1}^{N_{s_i}} \bar{K}_{i,j}^t(k) z_{i,j}^t(k) \\ &= \left(\sum_{j=1}^{N_{s_i}} C_{i,j}^t(k)K_{i,j}^t(k)\right)^\dagger (\hat{x}_i^t(k \mid k) \\ &\quad - C_{i,0}^t(k)\hat{x}_i^t(k \mid k-1)) \quad (26) \\ \bar{K}_{i,j}^t(k) &= \left(\sum_{q=1}^{N_{s_i}} C_{i,q}^t(k)K_{i,q}^t(k)\right)^\dagger C_{i,j}^t(k)K_{i,j}^t(k) \quad (27) \end{aligned}$$

where $\tilde{z}_t(k)$ represents the single pseudo-measurement of target t, created by a weighted average of combined measurements. Since the measurement reconstruction is performed on the estimate from each local processor, we add a subscript i to indicate processor i.

Since the noise from each sensor is independent and Gaussian, the pseudo-measurement noise covariance is

$$R_i^t(k) = \sum_{j=1}^{N_{s_i}} \bar{K}_{i,j}^t(k)R_{i,j}(k)(\bar{K}_{i,j}^t(k))' \quad (28)$$

This value is passed to the global processor for use in its Kalman filter algorithm in (6), where the predicted innovation covariance is now target dependent because of the application of $\bar{K}_{i,j}^t(k)$. Note that in the original Kalman filter algorithm, j corresponded to a sensor. In the global processor j corresponds to a local processor which essentially acts as a sensor for the global processor.

4. Complexity

The computational load corresponds to the limit of the number of operations required to complete the task, where an operation is defined as a combination of one addition and one multiplication. Once the order of the number of operation counts is calculated, the result is a complex equation based on the number of sensors N_s, the number of measurements from each sensor, the number of targets T, the length of the state vector n, and the length of the measurement vector. For distributed target tracking, we also include the number of local processors N_p and the number of sensors on each local processor. To simplify the equations for comparison of the relative computational complexities, we replace the dimension of the measurement vector with the quantity αn where α is the fraction of the state vector that is measured. We also assume the same number of measurements, m, is received by all of the sensors. In addition, the total number of sensors in the distributed architecture are assumed to be evenly distributed among the local processors. With these assumptions, the following table compares the computational complexity of the various algorithms:

Algorithm	Approx. Operation Count
Centralized	$O(N_sTm((1+\alpha)n^3 + m2^{T-1}))$
Individual Estimate	$O(N_sT^2((1+\alpha)n^3 + T2^{T-1}))$
Combined Estimate	$O(N_pT^2((1+\alpha)n^3 + T2^{T-1})$

The complexity of the distributed algorithms is listed for the global processor only, the demand at the local processors can be calculated by replacing N_s with N_{s_i} in the row for the centralized algorithm. Notice that while the operations count for the centralized algorithm is linear with respect to the number sensors, it shows quadratic growth with the number of measurements received by those sensors. The number of measurements on each sensor has been replaced with the number of targets in both of the distributed algorithms, lowering the computational demand. This advantage becomes more pronounced in high clutter environments.

Communication demands are also a concern. The centralized algorithm requires that all measurements be sent to the processor. The individual estimate algorithm requires that each local processor sends all target estimates and Kalman gain matrices for each sensor. The combined estimate algorithm requires that each local processor sends the Kalman gain matrix for each target from each sensor and a single state estimate for each target to the global processor. The actual communication demands are shown in the following table:

Algorithm	Values transmitted
Centralized	$N_sm\alpha n$
Individual Estimate	$N_sTn(\alpha n+1)$
Combined Estimate	$N_pn + N_sT\alpha n^2$

As the number of sensors grows, the distributed fusion algorithms have a distinct advantage over centralized systems in both computational complexity and communication requirements, especially when the clutter density is very high. However, in low-clutter environments or when using very few sensors, the centralized system has comparable or lower computational and bandwidth demands.

5. Simulation Results

In this section we present the results of several Monte Carlo simulation studies that were run using the sensor fusion techniques discussed in this paper. The tracking situation presented is two targets moving in 2D in nominally straight lines corrupted by acceleration or control noise. This noise is zero-mean, white, and Gaussian with covariance matrices of 0.0144 times the identity matrix.

The tracking systems consisted of a 3-sensor centralized platform running a centralized fusion algorithm and 4-sensor distributed architecture platforms running the Individual and Combined Estimate Reconstruction algorithms. Both distributed fusion systems had two local processors with 2 sensors each. Additionally, the local processor data was used to provide information on 2-sensor centralized fusion algorithms. Sensor noise is also zero-mean, white, and Gaussian with covariance matrices of 0.0144 times the identity matrix. Clutter density was varied between 0.6 and 0.9, resulting in an average of 1.5 to 2.25 clutter points per gate. Each simulation assumed perfect knowledge of both the initial target states as well as noise levels. One hundred Monte Carlo runs were performed in each tracking simulation.

Figure 2 shows the RMS position error versus clutter density for the two distributed fusion tracking algorithms as well as two- and three-sensor centralized fusion algorithms. The distributed fusion algorithms clearly outperform the 2-sensor centralized system, but not the 3 sensor system. However, the computational and communication requirements of the 3-sensor centralized system are generally larger than the requirements for either of the distributed systems.

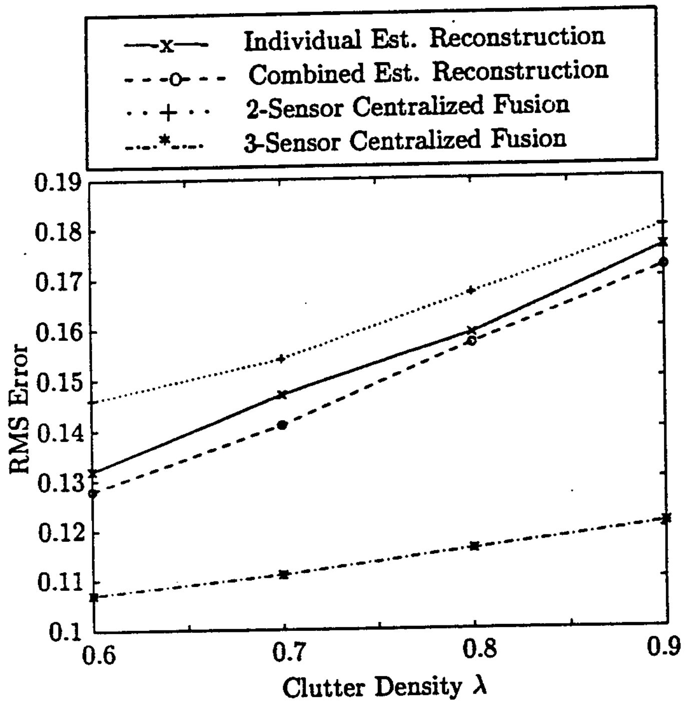


Figure 2: RMS error is shown vs clutter density in a series of two-target tracking simulations for the various tracking algorithms.

6. Conclusions

The measurement reconstruction approach has been extended to a more generalized class of distributed fusion systems. The evaluation of the computational and communication demands of the different algorithms were presented, and simulation results show promising improvements over local processor performance. While adding more sensors in a centralized scheme may sometimes provide better tracking performance, in applications where microprocessor capabilities and communication bandwidth are limited, distributed algorithms provide a reliable method of tracking.

Acknowledgments

This work was supported in part by an Office of Naval Research Young Investigator Award.

References

[1] Y. Bar-Shalom, "On the Track-to-Track Correlation Problem," *IEEE Trans. Automatic Control*, 26(2), 1981.

[2] Y. Bar-Shalom and T Fortmann. *Tracking and Data Association*, Academic Press, Inc., San Diego, 1988.

[3] C. B. Chang and L. C. Youens, "Measurement Correlation for Multiple Sensor Tracking in a Dense Target Environment," *IEEE Trans. Automatic Control*, 27(6), 1982.

[4] K. C. Chang, C. Y. Chong, and Y. Bar-Shalom, "Joint Probabilistic Data Association in Distributed Sensor Networks," *IEEE Trans. Automatic Control*, 31(10) 1986.

[5] K. C. Chang, C. Y. Chong, and S. Mori, "Distributed Tracking in Distributed Sensor Networks," *Proc. American Control Conference*, Seattle, WA, 1987.

[6] C. W. Frei and L. Y. Pao, "Non-Simulation Performance Prediction Methods for Different Implementations of a Multi-sensor Fusion Algorithm," *Proc. IFAC World Congress*, San Francisco, CA, Vol. J, 1996.

[7] L. Y. Pao. "Centralized Multi-sensor Fusion Algorithms for Tracking Applications," *IFAC J. Control Engineering Practice*, 2(5), 1994.

[8] L. Y. Pao. "Measurement Reconstruction Approach for Distributed Multi-sensor Fusion," *J. Guidance, Control, and Dynamics*, 19(4), 1996.

[9] L. Y. Pao and C. W. Frei, "A Comparison of Parallel and Sequential Implementations of a Multi-sensor Multitarget Tracking Algorithm," *Proc. American Control Conference*, Seattle, WA, 1995.

[10] D. Willner, C. B. Chang and K. P. Dunn, "Kalman Filter Algorithms for a Multi-Sensor System," *Proc. IEEE Conf. Decision and Control*, 1976.

Proceedings of the American Control Conference
Philadelphia, PA, June 1998, pp. 2749-2753.

Controlling Target Estimate Covariance in Centralized Multisensor Systems

MICHAEL KALANDROS[1] and LUCY Y. PAO[1]

Department of Electrical and Computer Engineering
University of Colorado
Boulder, CO 80309-0425
pao@colorado.edu and kalandro@colorado.edu

Abstract

Current multisensor fusion tracking systems can be easily overwhelmed by incoming data, especially as the number of targets and sensors increases. Sensor management schemes have been proposed to reduce the computational demand of these systems while minimizing the loss of tracking performance. This paper presents a system that will maintain a desired covariance level for each target while reducing the resource demands on the tracking system. Other functions performed by a sensor manager like prioritizing and scheduling are assumed to be done elsewhere, but result in delays in the execution of sensing requests made by the system. Three sensor selection algorithms are presented based on different resource and performance metrics and show a dramatic improvement over "dumb" sensing systems in simulation. Execution delay is shown to have a deleterious effect on the tracking performance of the system, but most of that performance can be restored when a prediction algorithm is used to model the delay.

1. Introduction

The application of multisensor fusion to surveillance systems has provided superior tracking performance at the cost of increased computational demand. As the number of targets and sensors increases, tracking systems can very quickly become overloaded by the incoming data. What is needed is a sensor management system that can balance tracking performance with system resources. Such a system also needs to be able to generate sensing actions, then prioritize and schedule those actions [3].

To date, most sensor management techniques have treated this as an optimization problem, where the goal is to apply combinations of sensors to each target to minimize a cost function generated using target priority, threat level, and the covariance of each target state estimate [4]. A variation of this is to maximize a cost functional based on the increase in state information from each sensor combination [5]. This method, however, does not address the problems of target priority and scheduling. Additionally, neural nets and decision theory have been applied to the sensor management problem [3].

A drawback of the cost functional approach is that it is difficult to specify a target-specific covariance goal, like reducing the covariance of a target estimate to accurately fire a weapon. A solution to this is to separate the system into a covariance controller and a sensor scheduler. The covariance controller can assign sensor combinations to each target to meet a desired covariance level. If the desired covariance changes for a target, then the sensor assignment changes for only one target. The scheduler prioritizes sensing actions and executes them as time allows. Low priority actions may be delayed until future scans or may be dropped altogether.

In this paper, the sensor scheduler is relegated to a "black box" without specifying its operations. However, as mentioned above, one of the expected effects of the separate sensor scheduler is the delay of the execution of sensing requests. This arises due to scheduling delays and the limited computational resources of the tracking system. Because of this, not all sensor requests can be executed in a single scan, causing sensor requests to accumulate in the command queue. This results in future requests being delayed as well.

The paper is organized as follows. In Section 2, we briefly review the Kalman filter equations used in the tracking algorithms. We develop the covariance control algorithms in Section 3. The effect of delay on the tracking performance of the algorithms and how to reduce that effect are discussed in Section 4. Section 5 presents preliminary simulation results demonstrating the performance of the developed algorithms. Finally, some conclusions and issues to consider in future work are given in Section 6.

[1]This work was supported in part by the Data Fusion Corporation, the Colorado Advanced Software Institute, an Office of Naval Research Young Investigator Award (Grant N00014-97-1-0642), and a National Science Foundation Early Faculty CAREER Award (Grant CMS-9625086). The authors would also especially like to thank Woody Kober and John K. Thomas of the Data Fusion Corporation for suggesting this area of research and for their feedback in the development of the sensor selection algorithms.

2. Mathematical Preliminaries

The sequential Kalman filter is an algorithm for combining multiple inputs from stochastic or slightly nonlinear systems to form an estimate in a state space representation. The Kalman filter is based on the following assumptions about the target and measurement systems [1, 2]:

$$
\begin{aligned}
x(k) &= Fx(k-1) + Gu(k-1) + q(k-1) && (1)\\
z_j(k) &= H_j x(k) + w_j(k), \quad j = 1, \ldots, N_s && (2)
\end{aligned}
$$

where $x(k)$ is the current state of the target; F, G, H_j are known system matrices; $u(k)$ is the control signal; $z_j(k)$ is a measurement of the target from sensor j; and there are N_s sensors. $q(k)$ is a variable representing process noise or higher-order motion not modeled by F, and $w_j(k)$ is a variable representing measurement noise in sensor j. Both $q(k)$ and $w_j(k)$ are assumed to have zero-mean, white, Gaussian probability distributions.

Since $q(k)$ and $w_j(k)$ are zero-mean noise processes, the target states and measurements in the next time interval can be predicted by:

$$
\begin{aligned}
\hat{x}(k \mid k-1) &= F\hat{x}(k-1 \mid k-1) + Gu(k-1) && (3)\\
\hat{z}_j(k) &= H_j \hat{x}(k \mid k-1) && (4)
\end{aligned}
$$

The input $u(k)$ is considered known and will be omitted in future equations because it can be easily reinserted. The quantity $v_j(k) = z_j(k) - \hat{z}_j(k)$ is known as the innovation. The predicted covariance of the state and innovation can be found by:

$$
\begin{aligned}
P(k \mid k-1) &= FP(k-1 \mid k-1)F' + Q(k-1) && (5)\\
S_j(k) &= H_j P(k \mid k-1) H_j' + R_j(k) && (6)
\end{aligned}
$$

where $Q(k)$ and $R_j(k)$ are the covariances of the noises in the plant and the sensor, respectively.

With N_s sensors, there are 2^{N_s} possible combinations or subsets of those sensors that can be used by the covariance controller. The ith possible subset is defined as Φ_i where N_{s_i} is the number of sensors in that combination. For a given i, the sequential algorithm runs a separate Kalman filter for each sensor, propagating its estimate to the next filter [6]:

$$
\begin{aligned}
&\hat{x}_1(k \mid k) = \hat{x}(k \mid k-1) + K_1(k)\big(z_1(k) - H_1\hat{x}(k \mid k-1)\big)\\
&\hat{x}_j(k \mid k) = \hat{x}_{j-1}(k \mid k) + K_j(k)\big(z_j(k) - H_j\hat{x}_{j-1}(k \mid k)\big), \quad j\epsilon\Phi_i\\
&\hat{x}(k \mid k) = \hat{x}_{N_{s_i}}(k \mid k) \qquad (7)
\end{aligned}
$$

where

$$
\begin{aligned}
K_1(k) &= P(k \mid k-1)H_1' S_1^{-1}(k)\\
K_j(k) &= P_{j-1}(k \mid k)H_j' S_j^{-1}(k), \quad j\epsilon\Phi_i && (8)
\end{aligned}
$$

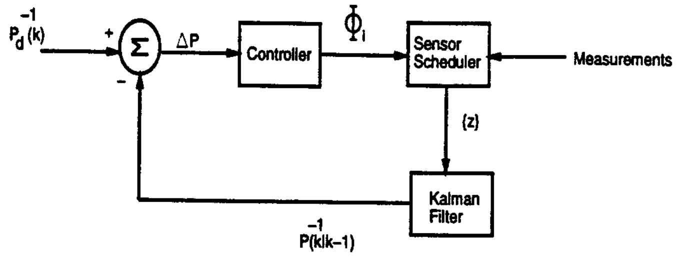


Figure 1: Block Diagram of a Tracking System with Covariance Controller

The state covariance is updated for each filter by:

$$
\begin{aligned}
P_1(k \mid k) &= (I - K_1(k)H_1)P(k \mid k-1)\\
P_j(k \mid k) &= (I - K_j(k)H_j)P_{j-1}(k \mid k), \quad j\epsilon\Phi_i\\
P(k \mid k) &= P_{N_{s_i}}(k \mid k) && (9)
\end{aligned}
$$

Once the state and covariance estimates have been updated, they are fed back into the algorithm and the entire process is repeated for the new set of measurements at the next time step. Alternatively, the covariance update can be calculated in a single step using the inverses of the covariance matrices [2]:

$$
P^{-1}(k \mid k) = P^{-1}(k \mid k-1) + \sum_{j\epsilon\Phi_i} H_j' R_j^{-1} H_j \quad (10)
$$

Since the sum $\sum_{j\epsilon\Phi_i} H_j' R_j^{-1} H_j$ is used frequently, we shall define it as the sensor information gain:

$$
J_i = \sum_{j\epsilon\Phi_i} H_j' R_j^{-1} H_j, \quad i = 1, \ldots, 2^{N_s} \quad (11)
$$

where J_i is the sensor information gain for the ith combination of sensors.

3. Covariance Control Algorithm

Figure 1 shows the block diagram of the tracking system. The Kalman filter can be thought of as the plant while the sensor scheduler acts as a system delay. Control of the covariance of the system is implemented via a sensor selection algorithm. The sensor selection is determined based on the difference between the inverses of the predicted covariance in Equation (5), and the desired covariance, $P_d(k)$. Replacing the updated covariance matrix in Equation (10) with the desired covariance and solving for the necessary sensor information gain, we see that we want J_i to equal ΔP, where

$$
\Delta P = P_d^{-1}(k) - P^{-1}(k \mid k-1) \quad (12)
$$

To achieve the desired covariance, the sensor information gain will ideally equal the difference between the inverses of the actual and desired covariance matrices. Generally, none of the sensor information gains will exactly equal this difference; thus $J_i - \Delta P$ will typically *not* be zero for any i. An algorithm is needed to select a set of sensors that will make $J_i - \Delta P$ as "small" as possible, allowing the desired covariance to be reasonably well approximated.

The use of $J_i - \Delta P$ to evaluate sensor combinations also reduces the computational complexity of sensor selection. To evaluate a sensor combination, that combination must be used to update the predicted covariance. Updating the covariance using Equations (6), (8), and (9) requires calcuating the matrix inverse of $S_j(k)$ for each sensor in a given combination. If multiple sensor combinations are evaluated, this must be repeated for each sensor combination in the search. On the other hand, using Equation (10) to update the covariance requires the calculation of the matrix inverse of $P^{-1}(k|k-1)$ and the calculation of J_i for each sensor combination. However, the J_i matrix for each combination can be precalculated and stored in a library. With this library, only one matrix inverse, $P^{-1}(k|k-1)$, needs to be calculated in each scan, regardless of the number of sensor combinations.

3.1. Sensor Selection Algorithms

One way to define the objective of the covariance control algorithm is to require that the sensors used produce an updated covariance that is within the desired covariance at all times. This will result in the difference, $P_d(k) - P_i(k|k)$, where $P_i(k|k)$ is the updated covariance using sensor combination i, having all positive eigenvalues (as well as the difference $J_i - \Delta P$). Since the goal is also to reduce the computational load on the tracking system, the sensor combination with the fewest number of sensors that produces all positive eigenvalues in the covariance error should be used at each scan. We shall call this the Eigenvalue/Minimum Sensors Algorithm.

Another method of rationing sensor resources is to view positive eigenvalues in the covariance error as excess resources applied to a target and negative eigenvalues as too little resources applied to that target. As such, the goal of the sensor selection algorithm should be to minimize the norm of the inverse covariance error, ΔP. This is the Matrix Norm Algorithm. One major drawback to this approach is that the inverse covariances used in Equation (12) are not always well-behaved, and the use of the difference $P_d(k) - P_i(k|k)$ instead of $J_i - \Delta P$ yields more reliable evaluations of sensor combinations. The library of pre-calculated J_i's can still be used, but will require an extra matrix inverse that converts $P_i(k|k)^{-1}$ in Equation (10) to $P_i(k|k)$ to be calculated for each sensor combination. This technique does not guarantee that the resulting covariance will be within the desired covariance limits since the algorithm does not take the sign of the eigenvalues of $P_d(k) - P_i(k|k)$ into account.

A third algorithm, Norm/Sensors, relaxes the requirements of the Matrix Norm technique, allowing the norm of the covariance difference to vary within a pre-

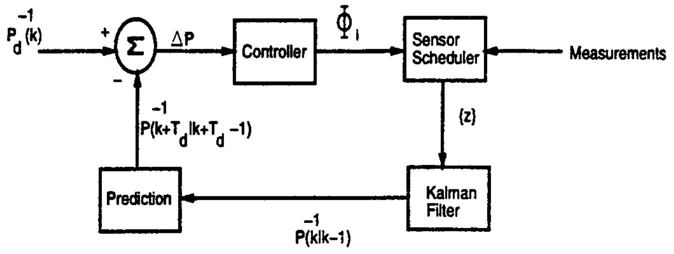


Figure 2: Block Diagram of a Tracking System with Covariance Controller and Prediction

defined boundary $\pm\delta$, selecting the sensor combination that uses the fewest sensors while keeping the covariance within that boundary.

4. The Effects of Delay

Similar to the effects of delay on dynamic control systems, the tracking performance of the sensor selection algorithms when there is delay becomes less stable. The effects of delay can be ameliorated by predicting the covariance estimate after the delay, allowing the sensor selection algorithm to make its decision based on what the predicted covariance will actually be when the delayed sensor selection is executed. To implement this prediction scheme, we simply run the Kalman filter for the projected length of the delay, using the prior sensor selections for that time period (see Figure 2). Correctly assuming the delay and sensor selections will restore most of the performance reduction caused by the delay.

5. Simulation Results

The following figures are the results of computer simulations of the three covariance control algorithms for multi-sensor tracking systems. A "dumb" system that simply always uses all its sensor resources is also included for a performance comparison. Of the several simulations that have been performed, we have chosen to present a few cases which best showcase the distinctions between the various systems. Each system uses three sensors that measure the position states x and y with different noise variance values in the x and y directions. Sensor 1 has a measurement noise variance of 1 and 0.05 in the x and y directions, respectively. Sensor 2 has noise variances of 0.05 and 1. Sensor 3 has a noise variance of $\sqrt{0.05}$ in both directions (all sensor noise covariance matrices are diagonal). Hence, Sensor 1 is very accurate in the y direction, Sensor 2 is very accurate in the x direction, and Sensor 3 is moderately accurate in both directions. However, overall, the sensors are approximately equally accurate in that the determinants of the noise covariance matrices for the sensors are about equal.

A single object nominally moving in the positive y direction of an $x-y$ space is tracked using a sequential Kalman filter. The target state consists of $[x, \dot{x}, y, \dot{y}]^T$. Its motion is corrupted by a zero-mean, white, Gaussian noise with a covariance of $0.12I$ (I = identity matrix). A desired estimate covariance, P_d, is defined and follows a step pattern, starting as a diagonal matrix with eigenvalues $[0.2, 0.3, 0.2, 0.3]$ at scan 0 and decreasing to a matrix with eigenvalues $[0.05, 0.25, 0.13, 0.22]$ at scan 25. The boundary size δ for the Norm/Sensors algorithm is 0.2.

Figure 3 shows the covariance error using two metrics: the smallest eigenvalue of the difference between the desired and actual covariances, and the 2-norm of that difference. Figure 4 shows the number of sensors used per scan – corresponding to the computational work load imposed on each tracking system.

Compare the performance of the "dumb" system to that of the Eigenvalue/Minimum Sensors algorithm. While both systems always meet the desired covariance goal, note that in the first half of the tracking task, the Eigenvalue/Minimum Sensors algorithm uses fewer sensors than the "dumb" system, yet suffers very little loss in covariance error performance. In the second half of the tracking task, both systems use all of the sensor resources to meet the desired covariance. While the "dumb" system wastes sensor resources by using all sensors for each scan, the sensor selection algorithm is able to balance tracking performance goals with system demands, allocating maximum resources only when necessary.

In the first half of the tracking task, the two norm-based algorithms choose the same sensors while in the second half, each algorithm chooses a different sensor combination. In each case, the Norm/Sensors algorithm uses the fewest sensors, while the Eigenvalue/Minimum Sensors algorithm requires the most of all of the sensor selection algorithms. However, except for the "dumb" system, the Eigenvalue/Minimum Sensors algorithm provides the best tracking performance, always selecting sensors so that the covariance is within the desired covariance, hence leading to the smallest RMS errors between the state estimate and the truth. The Norm/Sensors algorithm typically allows the largest covariance. The Matrix Norm algorithm's performance generally falls between the other two techniques.

These algorithms represent a continuum of trade-offs of computational demand versus tracking accuracy. Of the sensor selection algorithms, the Eigenvalue/Minimum Sensors algorithm will, in general, use the most sensor resources, since it has the strictest covariance requirement (the covariance must be less than the desired covariance in all directions). The Matrix Norm may choose fewer sensors, since it does not require the covariance to be within the desired covariance. Finally, the Norm/Sensors algorithm should choose the fewest sensors, but generally allows the largest covariance.

The effect of delay on the system is also simulated, using the Eigenvalue/Minimum Sensors Algorithm and a slightly different system model. The Sensor 3 now has variances of 0.01 in both directions. The desired covariance is initially $0.5I$, decreasing to $0.2I$ at scan 25. The simulations characterize a "dumb" system that uses all three sensors throughout the tracking task and has no delay; a "smart" system running the EV/Minimum Sensors Algorithm with no delay; a system running the EV/Minimum Sensors Algorithm, but whose choices are delayed by five scans; and a system with its sensor choices delayed by 5 scans, but that compensates by predicting the correct covariance at the end of that delay. In all but the "dumb" system, no sensors are selected initially – meaning the systems with delay will not make any target measurements for the first 5 scans.

Figure 5 shows a plot of the smallest eigenvalue of the difference between the desired and actual covariances (the various curves have been shifted slightly both vertically and horizontally to improve the readability of the figures). The plot shows the poor performance observed when a delay is added to the sensor selection system and not accounted for in the algorithm – the actual covariance both over- and under- shoots the desired covariance.

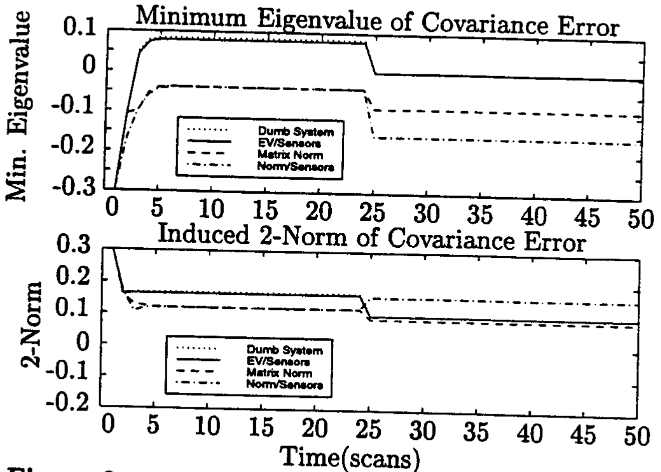


Figure 3: Comparison of the Covariance Tracking Accuracy of Sensor Selection Systems with no Delay

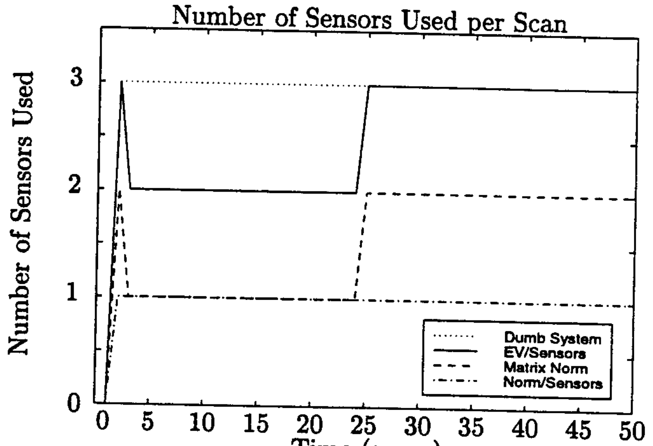


Figure 4: Comparison of the Efficiency of Sensor Selection Systems with no Delay

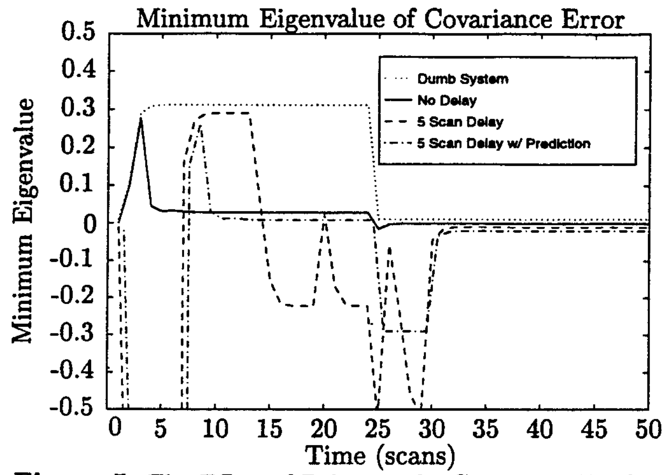


Figure 5: The Effect of Delay on the Covariance Tracking Accuracy of the EV/Minimum Sensors System

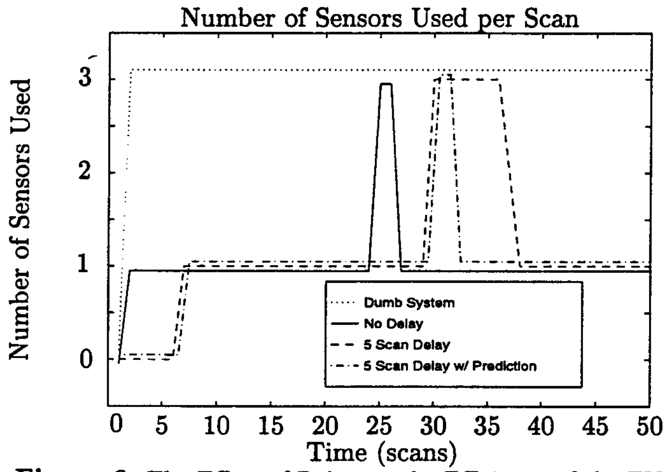


Figure 6: The Effect of Delay on the Efficiency of the EV/ Minimum Sensors System

Notice that the predictive system recovers quickly from the delay-induced errors. Once again, since the "dumb" system uses all three sensors each scan, its covariance is always contained within the desired covariance ellipsoid.

Figure 6 shows the number of sensors used in each scan – again corresponding to the computational work load imposed on each tracking system. The "dumb" system uses the most system resources since it uses all of the sensors throughout the tracking task. The "smart" system without delay and the predictive system use the fewest – increasing the number of sensors only briefly when the desired covariance is reduced. The delayed system without the predictive compensation uses less resources than the "dumb" system, but more than the undelayed or predictive systems. While uncompensated delays can severely degrade system performance, predictive compensation of those delays can restore most of that performance.

6. Conclusions and Future Work

Several sensor selection algorithms have been proposed for maintaining a target's state estimate covariance near a desired level without over-taxing the computational resources of a tracking system. The proposed algorithms maintain a specific desired covariance for each target while reducing the resource demand of current unmanaged or "dumb" systems. Simulation results indicate that the three sensor selection algorithms presented in this paper clearly outperform "dumb" systems in terms of resource efficiency. Other sensor manager functions including prioritizing and scheduling are assumed to be performed separately and will impact the covariance control algorithms in the form of request execution delays. As in dynamic systems, delay dramatically reduces the performance of the control algorithm, but if it can be accurately modeled, most of the performance can be restored.

There are many issues remaining for future study. First, a more rigorous exploration of the relationship between tracking performance and delay is needed, including an analysis of the effect of the size of the delays and errors in the algorithms' estimate of the actual delay. Second, in this paper the controller and sensor manager have been assumed to operate at the same rate as the Kalman filter, which is unlikely in practice. Future studies should allow different scan rates between the controller/manager and Kalman filter. Finally, while this work concentrates on control of the covariance estimate of a single target, these algorithms will eventually be applied in multi-target scenarios.

References

[1] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*, Academic Press, Inc., San Diego, 1988.

[2] Y. Bar-Shalom and X. Li. *Estimation and Tracking: Principles, Techniques, and Software*, Artech House, Boston, 1993.

[3] S. Musick and R. Malhotra, "Chasing the Elusive Sensor Manager," *Proceedings of the IEEE NAECON*, vol. 1, pp. 606-613, 1994.

[4] J. Nash, "Optimal Allocation of Tracking Resources," *Proceedings of the IEEE Conference on Decision and Control*, vol. 1, pp. 1177-1180, 1977.

[5] W. Schmaedeke, "Information-based Sensor Management," *SPIE Proceedings*, vol. 1955, pp. 156-164, 1993.

[6] D. Willner, C. B. Chang, and K. P. Dunn, "Kalman Filter Algorithms for a Multi-Sensor System," *Proceedings of the IEEE Conference on Decision and Control*, pp. 570-574, 1976.

Proc. AIAA Guidance, Navigation, and Control Conference
Boston, MA, Aug. 1998.
A98-37114

AIAA-98-4311

THE EFFECTS OF DELAYED SENSOR REQUESTS ON SENSOR MANAGER SYSTEMS

Lucy Y. Pao, Member AIAA
and Michael Kalandros

Department of Electrical and Computer Engineering
University of Colorado
Boulder, CO 80309-0425
pao@colorado.edu and kalandro@colorado.edu

keywords: target tracking, sensor management, sensor fusion

Abstract

Current multisensor tracking systems can be easily overwhelmed by incoming data, especially as the number of targets and sensors increases. A sensor management scheme has been proposed in previous work to reduce the computational demand of these systems while minimizing the loss of tracking performance by selecting only enough sensing resources to maintain a desired covariance level for each target, reducing the resource demands on the tracking system. However, the proposed system is sensitive to delays in the execution of sensor assignments. This paper analyzes the effect of that delay and examines methods of eliminating that effect. Because of the lack of a closed form solution for the covariance matrix of the discrete-time Kalman filter, the analysis centers on the performance of the continuous-time scalar Kalman-Bucy filter and then extends those results to the discrete-time case. The analysis shows that for all stable systems and unstable systems under certain conditions, the sensitivity of the covariance estimate to delays of sensing actions decreases steadily with time. Furthermore, when attempting to estimate unknown delays, overestimating the delay will produce smaller covariance prediction errors than underestimating the delay by a similar amount.

1. Introduction

The application of multisensor fusion to surveillance systems has provided superior tracking performance at the cost of increased computational demand. As the number of targets and sensors increases, tracking systems can very quickly become overloaded by the incoming data. What is needed is a sensor management system that can balance tracking performance with system resources. Such a system also needs to be able to generate sensing actions, then prioritize and schedule those actions.[4]

To date, most sensor management techniques have treated this as an optimization problem, where the goal is to apply combinations of sensors to each target to minimize a cost function generated using target priority, threat level, and the covariance of each target state estimate.[5] A variation of this is to maximize a cost functional based on the increase in state information from each sensor combination.[7] This method, however, does not address the problems of target priority and scheduling. Additionally, neural nets and decision theory have been applied to the sensor management problem.[4]

A drawback of the cost functional approach is that it is difficult to specify a target-specific covariance goal, like reducing the covariance of a target estimate to accurately fire a weapon. A solution to this is to separate the system into a covariance controller and a sensor scheduler.[3] The covariance controller can assign sensor combinations to each target to meet a desired covariance level. If the desired covariance changes for a target, then the sensor assignment changes for only one target. The scheduler prioritizes sensing actions and executes them as time allows. Low priority actions may be delayed until future scans or may be dropped altogether.

In this paper, the sensor scheduler is relegated to a "black box" without specifying its operations. However, as mentioned above, one of the expected effects of the separate sensor scheduler is the delay of the execution of sensing requests. It is important to note that delayed sensing requests do not represent measurement delays, where the output from the sensor is delayed before reaching the Kalman filter. Measurement delays have been studied and methods have been proposed to account for their effects.[2,6] A sensor request is the assignment of a sensor to or the removal from a tracking task. Once assigned, a sensor will provide tracking information at each scan for the assigned target until it is removed.

Delayed sensor requests arise due to scheduling delays and the limited computational resources of the tracking system. Because of this, not all sensor requests can be executed in a single sampling period, causing sensor requests to accumulate in the command queue. This results in future requests being delayed as well.

This paper is organized as follows. The covariance control architecture is described in Section 2. Since the equations that govern the behavior of the covariance of the discrete Kalman filter generally have no closed form solution, Section 3 analyzes the effect of delay on the scalar continuous Kalman filter, the covariance of which has a well-defined closed-form solution. The use of delay estimation to predict what the covariance will be when the sensing requests are executed is covered in Section 4. Section 5 covers the extension of these results to the discrete-time Kalman filter. General trends are observed that limit the results that can be extended to the discrete-time filter. Our initial work in expanding this analysis to more general vector spaces is included in Section 6. Future work in generalizing to higher-order models is also proposed in this section. Conclusions are included in Section 7.

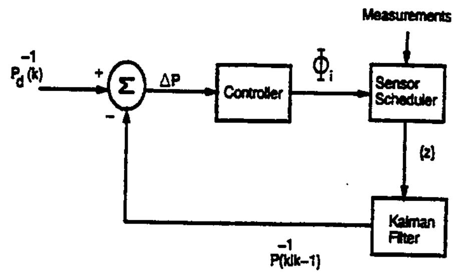


Figure 1: Block Diagram of a Tracking System with Covariance Controller (Sensor Selection Algorithm).

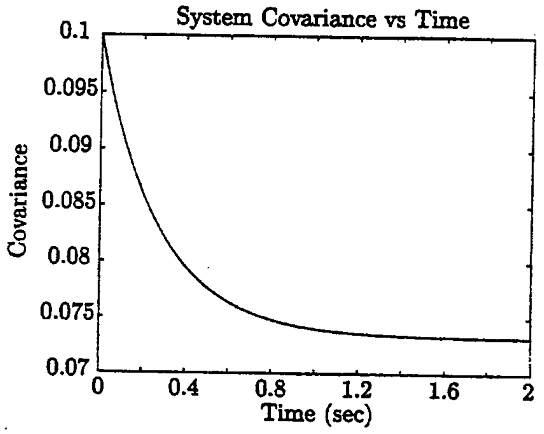


Figure 2: The covariance of a system converges quickly to its steady-state value.

2. Covariance Control

Figure 1 shows the block diagram of the tracking system. The Kalman filter can be thought of as the plant while the sensor scheduler acts as a system delay. Control of the covariance of the system is implemented via a sensor selection algorithm. The sensor selection is determined based on the difference between the predicted covariance for the next sampling period and the desired covariance. Algorithms for this task have been presented in Ref. 3. Note that the only input to the controller is the difference between the desired and actual target estimate covariances. Actual target tracking and estimation are performed by the Kalman filter. The controller's job is to regulate the sensing resources used by the Kalman filter to reduce the computational load on the tracking system.

Similar to the effects of delay on dynamic control systems, the tracking performance of the sensor selection algorithms when there is delay becomes less stable. The problems that delay causes are due to the fact that the covariance controller makes sensor selections based on the current covariance. For example, the controller decides on a sensor selection and executes it at time zero (see Figure 2). Then 0.2 seconds later, if the desired covariance P_d changes, the controller selects a different set of sensors based on the difference between P_d and the covariance at

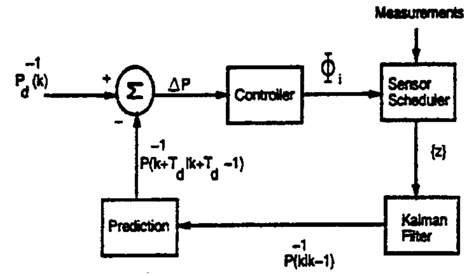


Figure 3: Block Diagram of a Tracking System with Covariance Controller and Prediction

that time. If the execution of this change is delayed for 0.2 seconds, then since the covariance difference $P_d - P_{k|k-1}$ at $t = 0.4$ is different from the covariance difference at $t = 0.2$, either excessive or insufficient sensor resources have probably been assigned to this tracking task. The effects of delay can be ameliorated by predicting the covariance estimate after the delay, allowing the sensor selection algorithm to make its decision based on what the predicted covariance will actually be when the delayed sensor selection is executed (see Figure 3). Correctly predicting the delay and sensor selections will restore most of the performance reduction caused by the delay.[3]

Because the controller is separate from the Kalman filter, it can run at a slower speed than the Kalman filter, allowing several iterations of the tracking algorithm to be performed before a new sensor combination is considered by the controller. This can be done by predicting the covariance at the next controller sampling period using the same architecture that is used to model a delay. In such a system, one sampling period of the controller will correspond to multiple sampling periods of the Kalman filter. One advantage of having a slower controller is that the target estimate covariance from the Kalman filter will be closer to steady-state, which will, in general, make the covariance prediction more robust to errors caused by delay.

3. Analysis of Delay in the Continuous Kalman Filter

We begin with the scalar multisensor version of the Kalman-Bucy filter:

$$\begin{aligned} \dot{x}(t) &= -ax(t) + w(t) && (1) \\ y_i(t) &= x(t) + v_i(t) \quad i = 1, \ldots, N_s && (2) \end{aligned}$$

where x is the target state variable to be tracked and y_i are the measurements of that state by a suite of N_s sensors. Note that when $a > 0$, the nominal system is stable. The state evolves according to a linear differential equation corrupted by white, zero mean noise $w(t)$ with variance q. The measurements are also corrupted by white, zero mean noise $v_i(t)$ with variance r_i, and $E[r_i r_j]^2 = 0$ when $i \neq j$. The measurements can also be represented as a vector:

$$\begin{aligned} Y(t) &= Hx(t) + V && (3) \\ &= [y_1(t), y_2(t), \ldots, y_{N_s}(t)]' \\ V(t) &= [v_1(t), v_2(t), \ldots, v_{N_s}(t)]' \\ H &= [1, 1, 1, \ldots, 1]' \end{aligned}$$

The variance of the measurements becomes a diagonal matrix R, with eigenvalues equal to r_i, $i = 1, \ldots, N_s$.

The state estimate is then

$$\dot{\hat{x}}(t) = -a\hat{x}(t) + p(t)H'R^{-1}[Y(t) - H\hat{x}(t)] \quad (4)$$

where $p(t)$ is the state estimate variance defined by the following differential equation:

$$\dot{p}(t) = -2ap(t) + q - p^2(t)H'R^{-1}H \quad (5)$$

which reduces to the scalar equation,

$$\begin{aligned} \dot{p}(t) &= -2ap(t) + q - \frac{p^2}{r} && (6) \\ \frac{1}{r} &= H'R^{-1}H \\ &= \sum_{i=1}^{N_s} \frac{1}{r_i} \end{aligned}$$

Equation (6) is identical to the differential equation describing the covariance of a single sensor Kalman-Bucy filter with measurement noise variance of r. Thus the covariance analysis of scalar, multisensor systems can be reduced to that of scalar, single sensor systems with no loss of generality.

Equation (6) has the following closed form solution:

$$\begin{aligned} p(t) &= p_1 + \frac{p_1 + p_2}{Ce^{2\alpha t} - 1} & (7) \\ \alpha &= \sqrt{a^2 + \frac{q}{r}} & (8) \\ p_0 &= p(0) \\ p_1 &= r(\alpha - a) \\ p_2 &= r(\alpha + a) \\ C &= \frac{p_0 + p_2}{p_0 - p_1} \end{aligned}$$

Notice that as t goes to infinity, the second term in Equation (7) goes to zero. Thus p_1 is the steady-state value of the state estimate variance. We now define the error in the estimate of the updated state variance due to a delay in sensor request execution as

$$\begin{aligned} \Delta p(t,d) &= p(t) - p(t+d) & (9) \\ &= \frac{2r\alpha Ce^{2\alpha t}(e^{2\alpha d} - 1)}{(Ce^{2\alpha t} - 1)(Ce^{2\alpha(t+d)} - 1)} \\ &= \frac{2r\alpha Ce^{2\alpha t}(e^{2\alpha d} - 1)}{C^2e^{4\alpha t}e^{2\alpha d} - Ce^{2\alpha t}(e^{2\alpha d} + 1) + 1} \end{aligned}$$

where $t, d \geq 0$. This is a valid assumption since the equation only describes the variance after $t = 0$ and a sensing request can never be executed before it is requested. Divide the numerator and denominator by $e^{4\alpha t}$ to get

$$\Delta p(t,d) = e^{-2\alpha t} \frac{2r\alpha C(e^{2\alpha d} - 1)}{C^2e^{2\alpha d} - Ce^{-2\alpha t}(e^{2\alpha d} + 1) + e^{-4\alpha t}} \quad (10)$$

It is now easy to see that as t increases, $\Delta p(t,d)$ goes to zero. If the convergence is monotonic, then the sensitivity of the covariance estimate to a sensor request delay will always decrease with time. If this is the case, then a lower controller scan rate (compared to the Kalman filter scan rate) will result in a more robust performance of the sensor selection algorithms.

If the convergence to zero is monotonic, the sign of the derivative should not change (if $\Delta p(t,d)$ is negative, it is always increasing to zero; if it is positive, it is always decreasing to zero).

$$\frac{\partial}{\partial t}\Delta p(t,d) = \frac{-4r\alpha^2e^{-2\alpha t}(e^{2\alpha d} - 1)(C^3e^{2\alpha d} - Ce^{-4\alpha t})}{(C^2e^{2\alpha d} - Ce^{-2\alpha t}(e^{2\alpha d} + 1) + e^{-4\alpha t})^2} \quad (11)$$

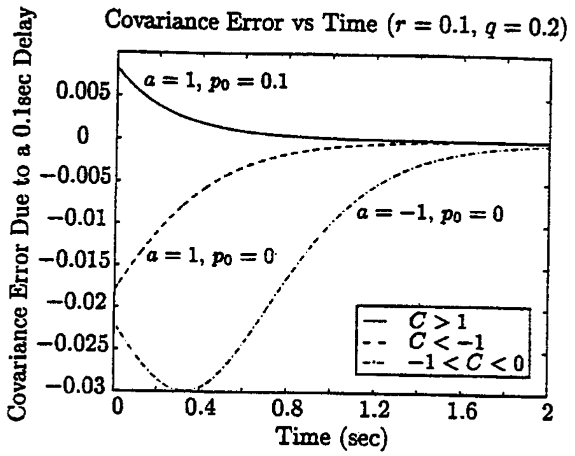


Figure 4: Examples of the qualitative performance of variance evolution with time for various C. For the case where $C > 1$, $p_0 > p_1$ and for both cases where $C < 0$, $p_0 < p_1$.

Since $e^{2\alpha d}$ is always greater than 1 and the denominator is squared, only the term $C^3e^{2\alpha d} - Ce^{-4\alpha t}$ will affect the sign of the derivative. The assumption that $t, d \geq 0$ leads to $|C| > 1$ as a sufficient but not necessary condition for the monotonicity of $\Delta p(t,d)$. This behavior can be seen in Figure 4. A stricter requirement for monotonicity is $C^2e^{2\alpha d} > 1$ since $e^{-4\alpha t}$ is never larger than 1.

3.1. Analysis of C

The next question becomes: When is $|C| < 1$?

$$\begin{aligned} C &= \frac{p_0 + p_2}{p_0 - p_1} & (12) \\ &= \frac{p_0 + r\alpha + ra}{p_0 - r\alpha + ra} \end{aligned}$$

Observe from Equation (8) that $\alpha \geq |a|$. Then, regardless of the sign of a, the numerator of C is always positive. This in turn means that the sign of C is solely related to the relationship between the initial and final variances of the state estimate. Thus if $p_0 > p_1$, then C is positive and if $p_0 < p_1$, then C is negative.

For the case of $0 < C < 1$, the following must hold

$$\begin{aligned} p_0 + r\alpha + ra &< p_0 - r\alpha + ra & (13) \\ r\alpha &< -r\alpha \end{aligned}$$

Since $r\alpha$ is always positive, this condition cannot exist. Therefore, $p_0 < p_1 \Rightarrow C > 0 \Rightarrow C > 1$. Thus, the covariance of all systems will converge monotonically when the initial covariance is larger than the steady-state covariance.

For the case of $-1 < C < 0$, the following must hold

$$\begin{aligned} p_0 + r\alpha + ra &< -(p_0 - r\alpha + ra) \quad (14) \\ p_0 &< -ra \end{aligned}$$

Since p_0 and r are always positive, this implies that a must be less than zero ($x(t)$ is unstable) for this to occur. Therefore, when $p_0 < p_1$, $a > 0 \Rightarrow C < -1$. This, combined with the monotonicity of all systems with $C > 0$ indicates that the covariance error due to delay of a stable target will always converge to zero monotonically with time. When $p_0 < p_1$ and $a < 0$, $p_0 > -ra \Rightarrow C < -1$. When $p_0 < -ra$ holds (meaning $(-1 < C < 0)$, then $C^2 e^{2ad} > 1$ is required for monotonicity.

4. Prediction of Covariance Via Delay Estimation

It has been shown that the effects of delay can be almost completely eliminated by predicting the actual variance after the delay.[3] The sensor manager can then select sensor combinations based on $p(t+d)$ rather than $p(t)$. We now look at the effect of errors in the estimate of the delay. Define the covariance prediction error due an error, δ, in the delay estimate as

$$\begin{aligned} \Delta\hat{p}(t,d,\delta) &= p(t+d) - p(t+d+\delta) \quad (15) \\ &= p(t') - p(t'+\delta) \\ &= \Delta p(t',\delta) \end{aligned}$$

Thus a redefinition of variables allows us to use the variance error from before. Note, however, that δ can be positive or negative.

One aspect of covariance prediction that is useful is whether it is better to overestimate or underestimate the actual delay. To examine this, assume a delay estimate error of $\pm\delta$. If $\delta > 0$, the delay has been overestimated; if $\delta < 0$, the delay has been underestimated. Since we can expect the variance error to have opposite signs for the two delay estimate errors, the sum of the two will have the same sign as the larger of the two errors. Assume for the moment, that $\delta \geq 0$. Then look at the following:

$$\begin{aligned} &\Delta p(t',\delta) + \Delta p(t',-\delta) = \\ &\quad \frac{2r\alpha C e^{2\alpha t'}(e^{2\alpha\delta} - 1)}{C^2 e^{4\alpha t'} e^{2\alpha\delta} - C e^{2\alpha t'}(e^{2\alpha\delta} + 1) + 1} \\ &\quad + \frac{2r\alpha C e^{2\alpha t'}(e^{-2\alpha\delta} - 1)}{C^2 e^{4\alpha t'} e^{-2\alpha\delta} - C e^{2\alpha t'}(e^{-2\alpha\delta} + 1) + 1} \\ = &\quad \frac{2r\alpha C e^{6\alpha t'}(2 - e^{-2\alpha\delta} - e^{2\alpha\delta})}{(Ce^{2\alpha t'} - 1)(Ce^{2\alpha(t'+\delta)} - 1)(Ce^{2\alpha t'} - 1)} \\ &\quad \times \frac{(C^2 - e^{-4\alpha t'})}{(Ce^{2\alpha(t'-\delta)} - 1)} \end{aligned} \quad (16)$$

If we look at the term $(2 - e^{-2\alpha\delta} - e^{2\alpha\delta})$ we find the derivative with respect to δ is

$$\frac{\partial(2 - e^{-2\alpha\delta} - e^{2\alpha\delta})}{\partial\delta} = 2\alpha(e^{-2\alpha\delta} - e^{2\alpha\delta}) \quad (17)$$

For $\delta \geq 0$ this derivative is always negative except at $\delta = 0$, which represents the local maximum. The value of $(2 - e^{-2\alpha\delta} - e^{2\alpha\delta})$ at this point is zero. Thus, this term is always less than or equal to zero.

Looking at Equation (16), the denominator consists of four terms of the form $Ce^{(\cdot)} - 1$. Since the exponents are always positive, when $C > 1$, each term is always positive. When $C < 0$, each term is always negative. Thus the denominator is always positive and does not affect the sign of the sum of the two delay estimates.

The terms left to control the sign of the sum $\Delta p(t',\delta) + \Delta p(t',-\delta)$ are C and $C^2 - e^{-4\alpha t'}$. If $|C| > 1$ then $C^2 - e^{-4\alpha t'}$ is always positive. If $C > 0$ (and thus > 1), then the sum in Equation (16) is negative – indicating that the error due to $-\delta$ is greater than the error due to δ. When $C < -e^{2\alpha t'}$, the sum in Equation (16) is positive, indicating that once again, the error due to $-\delta$ is greater. Since $t' = t + d$, the relation $C < -e^{-2\alpha d}$ is a sufficient condition to ensure the superiority of overestimating the delay. Figure 5 shows the effect of different values of C on $\Delta p(t',\delta) + \Delta p(t',-\delta)$. In this case, when $C = -0.1$, the covariance prediction error is actually slightly larger when the delay is overestimated than when it is underestimated, but as $|C|$ increases, it becomes much more advantageous to overestimate the delay. Notice that when $C = -10$, the prediction error is very near zero regardless of how much the delay is overestimated.

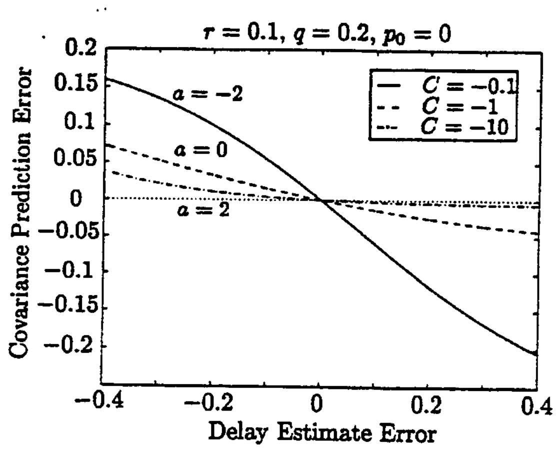


Figure 5: The effect of delay estimation error on covariance prediction shows that as $|C|$ increases, the benefits of overestimating the delay increase as well. Here, r, q, and $p0$ are fixed and a is varied to yield different values of C.

5. Extension to Discrete Kalman Filter

Since most tracking systems are discrete time systems, it is desirable to extend these results to the discrete Kalman filter. Because the discrete filter has no closed form solution, we will create a continuous-time model of the discrete system. From the continuous time covariance Equation (7), discrete equivalents of a, r, and q are needed to compute the discrete form of the covariance. Assume that the system represented in Equations (1) and (2) can be accurately modeled by the following discrete system:

$$\begin{aligned} x_{k+1} &= F x_k + w_k && (18) \\ y_k &= x_k + v_k && (19) \end{aligned}$$

The state estimate becomes

$$\begin{aligned} \hat{x}_{k+1|k} &= F\hat{x}_{k|k} && (20) \\ \hat{x}_{k+1|k+1} &= \hat{x}_{k+1|k} + K_{k+1}(y_{k+1} - x_{k+1|k}) \end{aligned}$$

The notation $\hat{x}_{k+1|k}$ means "the estimate of x at time $k+1$ given measurements through time k". The system variance can be calculated as follows:

$$\begin{aligned} p_{k+1|k} &= F^2 p_k + q_d && (21) \\ p_{k+1|k+1} &= \frac{1}{\frac{1}{p_{k+1|k}} + \frac{1}{r_d}} \\ K_{k+1} &= \frac{p_{k+1|k+1}}{r_d} && (22) \end{aligned}$$

where q_d and r_d are the target and measurement noise variances, respectively. The updated covariance $p_{k+1|k+1}$ will be abbreviated as p_{k+1} to keep the equations readable. To include the use of multiple tracking sensors in these equations, replace $y(k)$ with a vector containing the measurements from each sensor $Y(k) = [y_1(k), y_2(k), ..., y_{N_s}(k)]'$. The above equations become

$$\begin{aligned} \hat{x}_{k+1|k} &= F\hat{x}_{k|k} && (23) \\ \hat{x}_{k+1|k+1} &= \hat{x}_{k+1|k} + K_{k+1}(Y_{k+1} - Cx_{k+1|k}) \\ p_{k+1|k} &= F^2 p_k + q_d && (24) \\ p_{k+1|k+1} &= \frac{1}{\frac{1}{p_{k+1|k}} + \sum_{i=1}^{N_s} \frac{1}{r_{d_i}}} \\ K_{k+1} &= p_{k+1|k+1} H R^{-1} \\ H &= [1, 1, 1, ..., 1]' \end{aligned}$$

One method of converting the continuous-time Kalman filter to discrete time is to use the following conversions, which match the state estimates of the discrete model to those of the continuous at the sampling times:[1]

$$\begin{aligned} a &\approx \frac{1-F}{T_s} && (25) \\ q &= \frac{q_d}{T_s} \\ r &= T_s r_d \end{aligned}$$

However, the covariance of this discrete-time system does not converge to the continuous-time covariance. This should be expected since discrete- and continuous-time tracking are different processes. However, since the goal is to approximate the discrete-time covariance process with the continuous-time equation, a better method is to attempt to match the steady-state covariance values of the two models. The differential equation for the continuous-time covariance is

$$\begin{aligned} \dot{p}(t) &= -\frac{p^2(t)}{r} - 2ap(t) + q && (26) \\ &= 0 \quad \text{(in steady state)} \end{aligned}$$

This can be approximated by the discrete time system as

$$\begin{aligned}\dot{p}(t_k) &\approx \frac{p_{k+1} - p_k}{T_s} \qquad (27)\\ &\approx \frac{(1 - K_{k+1})(F^2 p_k + q_d) - p_k}{T_s}\\ &\approx \frac{(1 - \frac{F^2 p_k + q_d}{F^2 p_k + q_d + r_d})(F^2 p_k + q_d) - p_k}{T_s}\end{aligned}$$

Setting $\dot{p}(t_k) = 0$ and multiplying both sides of the equation by $(F^2 p_k + q_d + r_d)$ results in

$$0 = \frac{-F^2 p^2(t_k) - p(t_k)(r_d - F^2 r_d + q_d) + q_d r_d}{T_s} \qquad (28)$$

Dividing both sides by r_d gives

$$0 = \frac{F^2}{r_d T_s} p^2(t_k) - \frac{1 - F^2 + \frac{q_d}{r_d}}{T_s} p(t_k) + \frac{q_d}{T_s} \qquad (29)$$

Comparing the right side of this equation with the right side of Equation (26) provides continuous equivalents of the discrete Kalman filter:

$$\hat{a} = \frac{1 - F^2 + \frac{q_d}{r_d}}{2T_s} \qquad (30)$$

$$\hat{q} = \frac{q_d}{T_s} \qquad (31)$$

$$\hat{r} = \frac{r_d T_s}{F^2} \qquad (32)$$

The constants α, p_1, and p_2 are calculated as before using the above equivalent values of $\hat{a}$, $\hat{q}$, and $\hat{r}$.

Figure 6 shows the covariance of a discrete system and the equivalent continuous system (calculated as above). Note that although the two curves are similar and converge to the same point, they are not exactly the same. Figure 7 shows the covariance estimate error, $\Delta p(t, d)$, versus time for two different delays. Note that in the first plot, the period of time where the error due to delay is increasing with time is longer for the discrete system than for the continuous system. In the second plot, the continuous system is monotonically decreasing, while the discrete system is not. Obviously the strict bounds derived for the continuous case do not hold for the discrete case. However, simulations do seem to show that when $|C| > 1$, all of the limits for the continuous case hold for the discrete case as well.

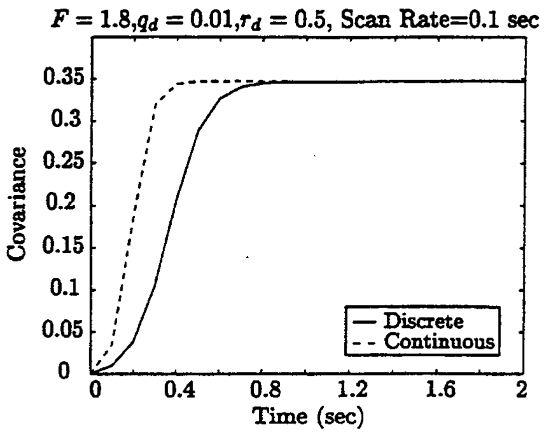


Figure 6: A comparison of the covariance response for a discrete system and its continuous equivalent for $C = -0.013$.

6. Generalization to Vector Spaces and Future Work

We are currently extending our analysis to higher-order vector spaces:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + w(t)\\ y(t) &= Hx(t) + v(t)\end{aligned} \qquad (33)$$

where x is a length n state vector and y is a length m measurement vector. A is an $n \times n$ matrix, and H is an $m \times n$ matrix. Finally, $w(t)$ and $v(t)$ are length n and m (respectively) independent white Gaussian noise vectors with zero mean and covariance matrices of Q and R. The state estimate of this system becomes

$$\dot{\hat{x}}(t) = A\hat{x}(t) + P(t)H'R^{-1}[y(t) - H\hat{x}(t)] \quad (34)$$

where $P(t)$ is the state estimate covariance defined by the differential equation

$$\dot{P}(t) = AP(t) + P(t)A' + Q - P(t)H'R^{-1}HP(t) \qquad (35)$$

To include a multisensor suite of N_s sensors in the tracking equations, concatenate the measurement vectors, creating

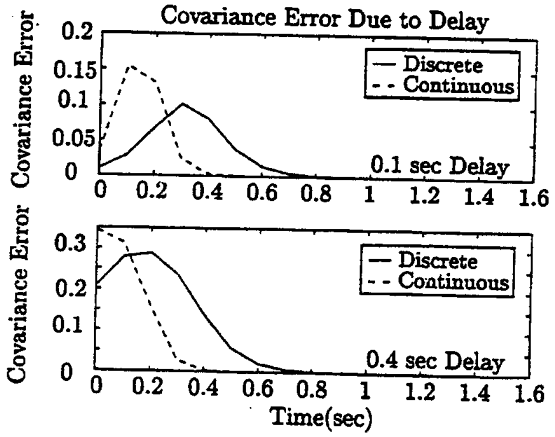


Figure 7: The non-decreasing region of the discrete system response is longer than that of its equivalent continuous system ($C = -0.013$).

$$
\begin{aligned}
Y(t) &= \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_{N_s}(t) \end{bmatrix} \qquad (36)\\
\bar{H} &= \begin{bmatrix} H_1(t) \\ H_2(t) \\ \vdots \\ H_{N_s}(t) \end{bmatrix} \\
\bar{R} &= \begin{bmatrix} [R_1] & & & \\ & [R_2] & & \\ & & \ddots & \\ & & & [R_{N_s}] \end{bmatrix}
\end{aligned}
$$

Applying these matrices to Equation (35) results in

$$
\begin{aligned}
\dot{P}(t) &= AP(t) + P(t)A' + Q - P(t)\bar{H}'\bar{R}^{-1}\bar{H}P(t) \\
&= AP(t) + P(t)A' + Q - \\
&\qquad P(t)\left(\sum_{i=1}^{N_s} \bar{H}_i'\bar{R}_i^{-1}\bar{H}_i\right)P(t) \qquad (37)
\end{aligned}
$$

Thus, the covariance of the multisensor Kalman-Bucy filter can be represented by an equivalent single-sensor Kalman-Bucy filter with measurement matrix $\bar{H}$ and a measurement noise covariance matrix $\bar{R}$.

The simplest higher-order system to analyze is the diagonal system, in which each state is orthogonal to the other. The system matrix is then

$$
A = \begin{bmatrix} -a_1 & & & \\ & -a_2 & & \\ & & \ddots & \\ & & & -a_n \end{bmatrix} \qquad (38)
$$

The assumption of whiteness of the noise will ensure that the noise covariance matrices are diagonal as well. The orthogonality includes the measurement matrices H_i, which must consist of subsets of the n orthogonal eigenvectors that define the state space. It is then easy to show that under these conditions the covariance matrix is diagonal, with the diagonal entries defined by

$$
\dot{P}_{ii}(t) = -2a_iP(t)_{ii} + Q_{ii} - \frac{(P_{ii}(t))^2}{(\bar{H}'\bar{R}^{-1}\bar{H})_{ii}} \qquad i = 1,\ldots,n \qquad (39)
$$

which, as shown in previous sections, has a closed form solution similar to that shown in Equation (7). Thus all of the conditions derived in this paper will hold for the eigenvalues of a orthogonal state space system. Diagonal discrete-time systems can be easily shown to decompose into a series of scalar equations as well.

Analysis of more general systems is much more complicated and is subject of future work. We are currently using a linear formulation of the continuous-time Ricatti equation to derive a closed form solution to a more realistic class of tracking models that are not diagonal in nature. We are currently working on a very simple block diagonal model, where each block is a second-order integrator observed by a scalar measurement. Although the results of our analysis will not be extendable to non-block-diagonal systems, a large number of tracking situations can be represented using this model. Furthermore, the techniques developed to analyze this model will be applicable to other systems.

7. Conclusion

The delay of the execution of sensor requests when tracking continuous-time scalar or diagonal target models can reduce the performance of the surveillance system. However, the effect of such a delay is reduced as time increases (and the covariance of the system converges to a steady-state value). Furthermore, when the actual delay is unknown or varies over time, overestimating the delay will produce smaller covariance prediction errors than underestimating the delay.

Based on these observations, strategies to reduce the effects of delay on covariance estimates could include reducing the scan rate of the controller, i.e. allowing the Kalman filter to run longer in between changing the sensor combination. However, this strategy is limited by the desired responsiveness of the system. If the scan rate of the controller is reduced, the time required to change the target covariance increases. Another strategy is to consistently overestimate the delay when attempting to predict the covariance. The drawback to this method is that it increases the computational demand on the controller.

The analysis of the effect of delay on the discrete-time Kalman filter is more complicated due to the lack of a closed form solution to the difference equation. The attempt to create a continuous-time system that is equivalent to the discrete-time system has been partially successful. While the trends of the discrete system match those of its continuous equivalent, time periods when the error due to delay does not decrease monotonically last longer for the discrete time system.

8. Acknowledgments

This work was supported in part by the Data Fusion Corporation, the Colorado Advanced Software Institute, an Office of Naval Research Young Investigator Award (Grant N00014-97-1-0642), and a National Science Foundation Early Faculty CAREER Award (Grant CMS-9625086). The authors would also especially like to thank Woody Kober and John K. Thomas of the Data Fusion Corporation for suggesting this area of research and for their feedback in the development of the sensor selection algorithms, and Nathan Baltz of the University of Colorado for his editorial comments.

References

[1] Y. Bar-Shalom and X. Li. *Estimation and Tracking: Principles, Techniques, and Software*, Artech House, Boston, 1993.

[2] S.B. Chang and M.H. Perng, "State Estimation from Incremental Sensor Data Corrupted by Track Miscounts and a Detection Delay," *IEEE Transactions on Control Systems Technology*, Vol 4. no. 1, pp. 65-67, January 1996.

[3] M. Kalandros and L. Y. Pao, "Controlling Target Estimate Covariance in Centralized Multisensor Systems," *Proc. of the 1998 American Control Conference*, Philadelphia, PA, June 1998.

[4] S. Musick and R. Malhotra, "Chasing the Elusive Sensor Manager," *Proceedings of the IEEE 1994 NAECON*, vol. 1, pp. 606-613, May 1994, Dayton, OH, IEEE: New York, NY.

[5] J. Nash, "Optimal Allocation of Tracking Resources," *Proceedings of the 1977 IEEE Conference on Decision and Control*, vol. 1, pp. 1177-1180, December 1977, New Orleans, LA, IEEE: New York, NY.

[6] A. Ray, L.W. Liou, J.H. Shen, "State Estimation Using Randomly Delayed measurements," *Journal of Dynamic Systems, Measurement, and Control*, vol. 115, pp. 19-26, March 1993.

[7] W. Schmaedeke, "Information-based Sensor Management," *SPIE Proceedings*, vol. 1955, April 1993.

Submitted in May 2000 to Automatica.

Varying Rate and Resolution to Control Estimation Error Covariance in Centralized Target Tracking Systems

L. Y. Pao and N. T. Baltz
Electrical and Computer Engineering Department
University of Colorado
Boulder, CO 80309-0425
Fax: 303-492-2758
Email: *pao@colorado.edu* & *baltzn@colorado.edu*

Abstract

This paper provides an analysis of error covariance control techniques for allocating sensing resources in multisensor target tracking systems. We focus on tracking one target using multiple sensors each having different rates and resolutions. The algorithm we develop for allocating sensing resources manages the rates and resolutions at which sensor information is processed. An elliptical annulus described by two covariance matrices is used to control the prediction and update covariances in the Kalman filter.

Keywords: Covariance Control, Rate, Resolution, Sensor Management, Tracking

1. Introduction

In many multisensor surveillance systems, sensor management techniques are needed to balance tracking performance with system resources (Popoli, 1992). Rate and resolution are two fundamental parameters that can be used to affect the tracking performance in such systems. While most sensor management techniques have considered rate and resolution separately (Singer 1970, Nash 1977, Van Keuk 1978, Schmaedeke 1993, Schmaedeke & Kastella 1998), we developed a sensor management scheme that varies both rate and resolution in the Kalman filter. We define the resolution as a matrix product involving the inverse of the measurement covariance matrix and the measurement matrix. The rate is simply the inverse of the sample period.

Sensors generally have a number of different parameters that affect rate and resolution. For instance, when managing a monopulse Electronically Steered Array (ESA) radar some parameters we may be concerned with are radar beam shape, electromagnetic emissions, average energy, average power, modulated waveform, modulating waveform, carrier frequency, pulse period, and sample period (Blair & Watson 1996). Although not all radars can change these parameters in real time, at some point in the design process these parameters must be chosen. Other noncontrollable parameters that affect detection and estimation performance are Radar Cross Section (RCS) and channel noise. These parameters together determine the detection and estimation performance in a tracking system.

An infrared CCD array is an example of a passive, narrow band or broadband, parallel sensor. Parameters associated with this sensor are its pixel frequency response, image resolution, and sample rate. After the CCD array is designed, the frequency response and image resolution are generally fixed, while the sample period or frame rate can be a variable parameter depending on the hardware. Quantization effects such as finite image resolution and measurement or channel noise affect subsequent estimates.

Several sensor management systems have been proposed for centralized systems (Nash 1977, Schmaedeke 1993) based on the optimization of a cost function generated using target priority, the covariance of each target state estimate, and the cost of using specific sensor combinations. Two problems associated with using these techniques are that 1) using target priority is a coarse adjustment for maintaining tracking performance, and 2) these methods do not consider using sample rate to maintain tracking performance.

The drawbacks of the above approaches in addressing tracking performance motivated the development of algorithms that can obtain specific bounds on estimation performance. These methods are based upon maintaining desired covariance goals (Kalandros & Pao 1998, Pao & Baltz 1999). Although using all the sensing resources will obtain the best state estimates this requires increased computational resources and does not allow for sensing resources to be applied to other targets or sensing tasks. Using the desired covariance goals we are able to develop algorithms that use both sensor rate and sensor resolution to jointly optimize the target error covariance. In this approach, the algorithms are implemented using two functional blocks that separate the sensor manager into a covariance controller, which selects the sensor combinations and sample rate based on their ability to achieve the covariance goals, and a sensor scheduler that prioritizes sensing actions and executes them as time allows. The sensor scheduler can cause low priority actions to be delayed until future scans or may be dropped altogether. The covariance controller maintains the covariance level of each target estimate to within some desired level while reducing system resource demands.

A block diagram of a centralized tracking system is shown in Figure 1.1. This model shows the different components of the system including sensors along with signal and information processors and a

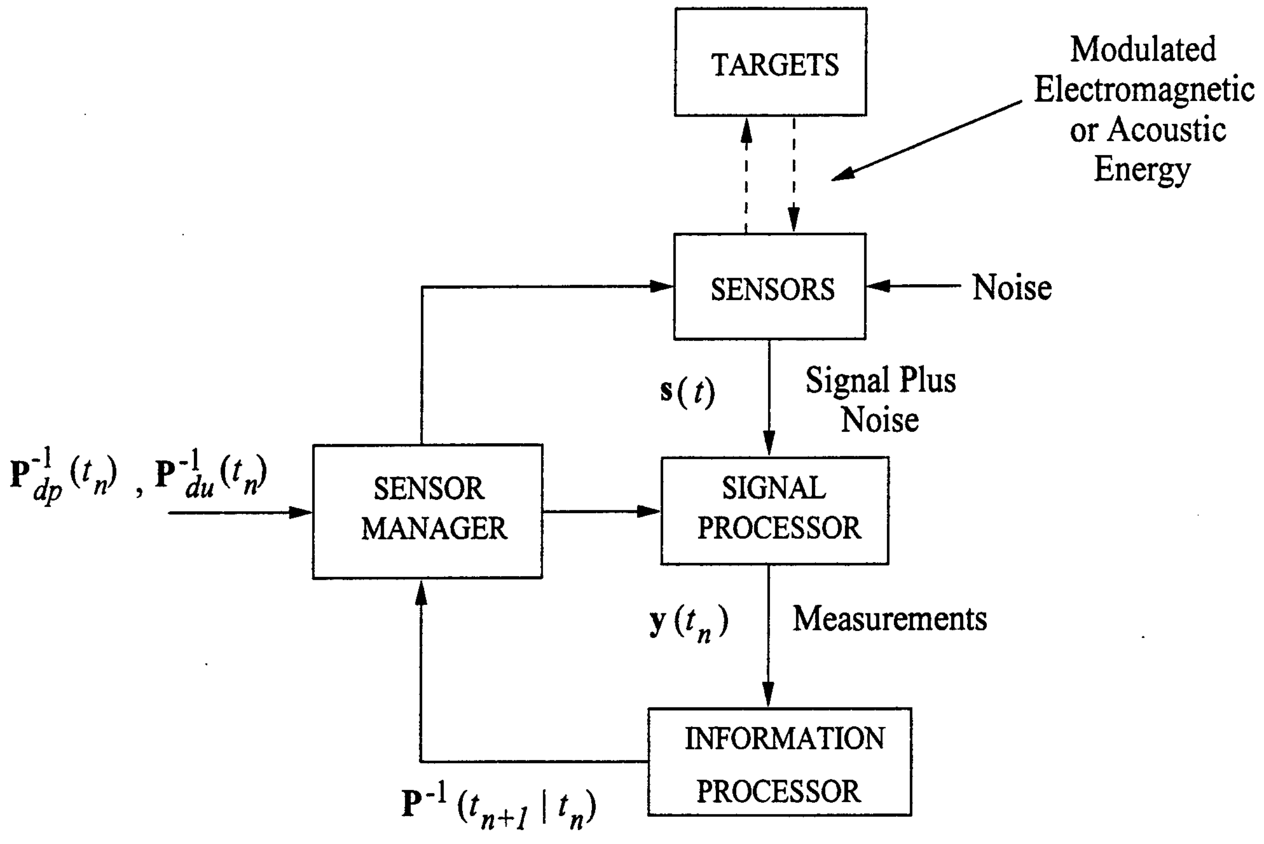

TARGETS
Modulated Electromagnetic or Acoustic Energy
SENSORS
Noise
$\mathbf{s}(t)$
Signal Plus Noise
$\mathbf{P}^{-1}_{dp}(t_n)$, $\mathbf{P}^{-1}_{du}(t_n)$
SENSOR MANAGER
SIGNAL PROCESSOR
$\mathbf{y}(t_n)$
Measurements
INFORMATION PROCESSOR
$\mathbf{P}^{-1}(t_{n+1} \mid t_n)$

Figure 1.1 Components of a Centralized Tracking System

sensor manager. This block diagram illustrates the sensor control problem, where the sensor manager uses rate and resolution to maintain the information matrices $\mathbf{P}^{-1}(t_{n+1}|t_n)$ within an elliptic annulus described by a desired update information matrix and a desired prediction information matrix given by $\mathbf{P}_{dp}^{-1}(t_n)$ and $\mathbf{P}_{du}^{-1}(t_n)$, respectively. The desired matrices for each target are chosen so that they are related as

$$\mathbf{P}_{du}^{-1} \geq \mathbf{P}_{dp}^{-1} \Leftrightarrow \mathbf{P}_{dp} \geq \mathbf{P}_{du} \tag{1.1}$$

Both information and covariance matrices can be illustrated using ellipses or ellipsoids. We define the ellipse or ellipsoid as the set of points that satisfy $ellipse(\mathbf{P}) \equiv \{x; x^T\mathbf{P}^{-1}x = 1\}$. Some typical desired information and covariance goals in $\Re^2$ are shown in Figure 1.2a and Figure 1.2b respectively. The desired update information and update covariance are shown using solid thick lines. The desired prediction information and prediction covariance are shown using thin solid lines.

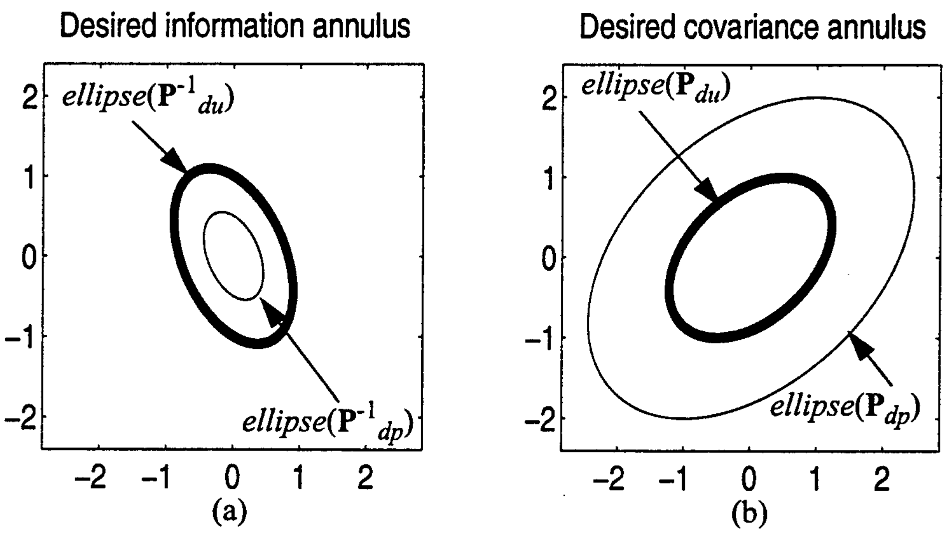


Figure 1.2 Elliptical annulus for specifying desired tracking performance

Figure 1.3 is a block diagram of the sensor manager. The inputs are the desired update and prediction information matrices $\mathbf{P}_{du}^{-1}(t_n)$ and $\mathbf{P}_{dp}^{-1}(t_n)$ respectively, as well as the predicted information matrix $\mathbf{P}^{-1}(t_{n+1}|t_n)$ for each target. These matrices are used to compute rate $T(t_n)$ and resolution $\Gamma(t_n)$ for each target, where $\Gamma(t_n)$ is a set of sensors with associated measurement covariances and measurement matrices.

The sensor scheduler is responsible for applying the rate and resolution signals to the sensor time lines. When the sample periods for each target are not commensurate then sensors can not be scheduled at the same time causing disturbances in our ability to control the covariance. Using this methodology for a Sensor Manager allows us to develop the Covariance control techniques independent of the scheduler. We therefore focus our efforts on tracking a single target and treat the scheduler as a black box which causes

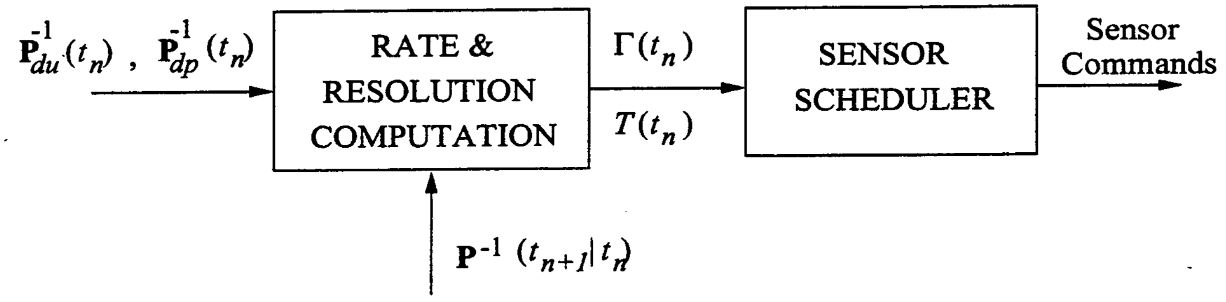

$\mathbf{P}_{du}^{-1}(t_n)$, $\mathbf{P}_{dp}^{-1}(t_n)$
RATE & RESOLUTION COMPUTATION
$\Gamma(t_n)$
$T(t_n)$
SENSOR SCHEDULER
Sensor Commands
$\mathbf{P}^{-1}(t_{n+1}|t_n)$

Figure 1.3 Sensor Manager Block Diagram

delay and drop out.

This paper is organized as follows. Section 2 reviews the equations for the Kalman Filter. Section 3 develops open loop and closed loop sensor allocation strategies. The sensor allocation strategies are used to develop specific algorithms. Issues of sensor delay and sensor drop out are discussed in Section 4. In Section 5, the results of simulations are used to illustrate how the algorithms are applied to a tracking system. Finally, Section 6 gives some conclusions and discusses issues for further investigation.

2. Kalman Filter

There are M sensors each capable of taking measurements of length $k_i < N$, $\forall i \in \{1, \ldots, M\}$, with N the length of the state vector. Assuming a single target and simultaneous reception of all measurements, the measurement vector, the measurement matrix, and the measurement noise components are

$$y(t_n) = \left[y_1^{\mathrm{T}}(t_n) \ \ldots \ y_M^{\mathrm{T}}(t_n)\right]^{\mathrm{T}} \tag{2.1}$$

$$\mathbf{C}(t_n) = \left[\mathbf{C}_1^{\mathrm{T}}(t_n) \ \ldots \ \mathbf{C}_M^{\mathrm{T}}(t_n)\right]^{\mathrm{T}} \tag{2.2}$$

$$v(t_n) = \left[v_1^{\mathrm{T}}(t_n) \ \ldots \ v_M^{\mathrm{T}}(t_n)\right]^{\mathrm{T}} \tag{2.3}$$

respectively. When the measurement noise terms across sensors are uncorrelated, the covariance has a block diagonal structure, $\mathbf{R}(t_n) = blockdiag\left[\mathbf{R}_1(t_n) \ \ldots \ \mathbf{R}_M(t_n)\right]$.

The state and measurement equations are

$$x(t_{n+1}) = \mathbf{A}(t_n)x(t_n) + \mathbf{B}(t_n)w(t_n) \tag{2.4}$$

$$y(t_n) = \mathbf{C}(t_n)x(t_n) + \mathbf{D}(t_n)v(t_n) \tag{2.5}$$

respectively, where $\mathbf{A}(t_n), \mathbf{B}(t_n) \in \Re^{N \times N}$, $\mathbf{C}(t_n) \in \Re^{L \times N}$, and $L = \sum_{i=1}^{M} k_i$. The noise terms $\mathbf{B}(t_n)w(t_n)$ and $\mathbf{D}(t_n)v(t_n)$ are zero mean, with covariances $\mathbf{B}(t_m)E[w(t_m)w^{\mathrm{T}}(t_n)]\mathbf{B}^{\mathrm{T}}(t_n) = \mathbf{Q}(t_n)\delta_{m-n}$ and $\mathbf{D}(t_m)E[v(t_m)v^{\mathrm{T}}(t_n)]\mathbf{D}^{\mathrm{T}}(t_n) = \mathbf{R}(t_n)\delta_{m-n}$, respectively. With these definitions the prediction and update equations of the Kalman filter are (Bar-Shalom & Li, 1993)

Prediction Equations:

$$\hat{x}(t_n|t_{n-1}) = \mathbf{A}(t_n)\hat{x}(t_{n-1}|t_{n-1}) \tag{2.6}$$

$$\hat{y}(t_n|t_{n-1}) = \mathbf{C}(t_n)\hat{x}(t_n|t_{n-1}) \tag{2.7}$$

$$\mathbf{P}(t_n|t_{n-1}) = \mathbf{A}(t_n)\mathbf{P}(t_{n-1}|t_{n-1})\mathbf{A}^{\mathrm{T}}(t_n) + \mathbf{Q}(t_n) \tag{2.8}$$

Update Equations:

$$\mathbf{P}^{-1}(t_n|t_n) = \mathbf{P}^{-1}(t_n|t_{n-1}) + \mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n)\mathbf{C}(t_n) \tag{2.9}$$

$$\mathbf{K}(t_n) = \mathbf{P}(t_n|t_n)\mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n) \tag{2.10}$$

$$\hat{x}(t_n|t_n) = \hat{x}(t_n|t_{n-1}) + \mathbf{K}(t_n)(y(t_n) - \hat{y}(t_n|t_{n-1})) \tag{2.11}$$

where the carets (^) denote state and measurement estimates. The sensor information matrix $\mathbf{R}^{-1}(t_n)$ and the measurement matrix $\mathbf{C}(t_n)$ together define the sensor resolution $\mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n)\mathbf{C}(t_n)$. When only a subset of sensors take mesurements we can model this using a sumation as given in (2.12)

$$\mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n)\mathbf{C}(t_n) = \begin{bmatrix} \mathbf{C}_{\Gamma_1} \\ \\ \mathbf{C}_{\Gamma_K} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{R}_{\Gamma_1}^{-1} & & \\ & \cdots & \\ & & \mathbf{R}_{\Gamma_K}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{C}_{\Gamma_1} \\ \\ \mathbf{C}_{\Gamma_K} \end{bmatrix} = \sum_{j \in \Gamma} \mathbf{C}_j^{\mathrm{T}}\mathbf{R}_j^{-1}\mathbf{C}_j \tag{2.12}$$

where $\Gamma = \{\Gamma_1, \ldots, \Gamma_K\}$ is a subset of sensors of size $|\Gamma| = K \le M$. The prediction and update equations from (2.6) to (2.11) define what is called the information matrix filter (Bar-Shalom & Li, 1993) because it uses the Fisher information in (2.9) to compute the Kalman gain in (2.10). This recursion is one form of the Kalman filter. The state update in (2.11) can also be expressed as

$$\hat{x}(t_n|t_n) = (\mathbf{I} - \mathbf{P}(t_n|t_n)\mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n)\mathbf{C}(t_n))\mathbf{A}(t_n)\hat{x}(t_{n-1}|t_{n-1}) + \mathbf{P}(t_n|t_n)\mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n)y(t_n) \tag{2.13}$$

which after simplification takes the form

$$\hat{x}(t_n|t_n) = \mathbf{E}(t_n)\hat{x}(t_{n-1}|t_{n-1}) + \mathbf{K}(t_n)y(t_n) \tag{2.14}$$

where $\mathbf{E}(t_n) = (\mathbf{I} - \mathbf{P}(t_n|t_n)\mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n)\mathbf{C}(t_n))\mathbf{A}(t_n)$ and $\mathbf{K}(t_n) = \mathbf{P}(t_n|t_n)\mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n)$. The process noise covariance $\mathbf{Q}(t_n)$ and gain $\mathbf{B}(t_n)$ do not appear directly in (2.13). The matrices $\mathbf{Q}(t_n)$ and $\mathbf{B}(t_n)$ are subsumed into the state error covariance. This is a "coefficient" adaptive filter because the gains between the previous state estimate $\hat{x}(t_{n-1}|t_{n-1})$ and the new measurement $y(t_n)$ change based upon the state error covariance $\mathbf{P}(t_n|t_n)$. A "model" adaptive filter would change any of the matrix coefficients $\mathbf{A}(t_n)$, $\mathbf{B}(t_n)$, $\mathbf{C}(t_n)$, or $\mathbf{D}(t_n)$.

3. Sensor Allocation

The covariance control methods developed here build upon some of the techniques developed in (Kalandros & Pao 1998, Pao & Baltz 1999). The desired matrices have several interpretations. The *first* interpretation is that we try to maintain the state error covariance for each target inside the elliptical annulus. This can be expressed as

$$\mathbf{P}_{dp}(t_n) \geq \mathbf{P}(t_{n+1}|t_n) \geq \mathbf{P}_{du}(t_n) \tag{3.1}$$

These matrix bounds restrict the prediction and update covariance to be inside an elliptical annulus. This requires computation of eigenvalues of the matrix differences so that the matrix inequalities can be tested.

The *second* interpretation treats each desired covariance as an upper bound on the prediction and update covariances. This interpretation is desirable because it insures that the error covariances are less than the desired, i.e. more target information than desired. These conditions on the prediction and update covariance are

$$\begin{aligned} \mathbf{P}_{dp}(t_n) &\geq \mathbf{P}(t_{n+1}|t_n) \\ \mathbf{P}_{du}(t_n) &\geq \mathbf{P}(t_n|t_n) \end{aligned} \tag{3.2}$$

respectively. The desired update and prediction covariance (information) matrices are treated as upper bounds (lower bounds) on the error covariance (information update) matrices. The desired information update matrix is used with (2.9) and (2.12) to compute the optimal sensor resolution (set of sensors), and the desired prediction covariance matrix is used with (2.8) to compute the optimal sampling rate. All sensors have a minimum sample period determined by sensor limitations and possibly communication and processing delays. We therefore specify a minimum sample period T_{min} for each sensor.

A fundamental question is whether the sensing resources can achieve the desired covariance goals. Given system coefficients $\mathbf{A}(T)$, $\mathbf{Q}(T)$, $\mathbf{C}_i$, $\mathbf{R}_i^{-1}$, $i = 1, \ldots, M$, and constant desired information matrices $\mathbf{P}_{du}^{-1}(t_n)$ and $\mathbf{P}_{dp}^{-1}(t_n)$ with $\mathbf{P}(t_n|t_n) \leq \mathbf{P}_{du}(t_n)$ and $\mathbf{J}^{-1} = \sum_{i \in \Gamma} \mathbf{C}_i^{\mathrm{T}} \mathbf{R}_i^{-1} \mathbf{C}_i$, then if the nominal sample period and nominal sensor resolution are given by

$$T_{nom} = \arg \min_{T > 0} \; det[\mathbf{P}_{dp}(t_n) - \mathbf{A}^{\mathrm{T}}(T)\mathbf{P}_{du}(t_n)\mathbf{A}(T) - \mathbf{Q}(T)] = 0 \tag{3.3}$$

$$\mathbf{J}_{nom}^{-1} = \mathbf{P}_{du}^{-1}(t_n) - \mathbf{P}_{dp}^{-1}(t_n) \tag{3.4}$$

respectively, then $T \leq T_{nom}$ and $\mathbf{J}^{-1} \geq \mathbf{J}_{nom}^{-1}$ will insure that $\mathbf{P}_{dp}(t_n) > \mathbf{P}(t_n + T|t_n)$ and $\mathbf{P}_{du}(t_n) > \mathbf{P}(t_n|t_n)$ for all t_n, which are conditions imposed by (3.2). This means that when the sample period is *less* than the nominal sample period and the sensor resolution is *greater* than the nominal resolution the covariance goals will be met for all $t_{n+1} > t_n$.

The positive definite matrix difference in (3.4) is referred to as the nominal sensor resolution because from (2.9) we can form the matrix difference $\mathbf{P}^{-1}(t_n|t_n) - \mathbf{P}^{-1}(t_n|t_{n-1})$ which equals the sensor resolution. Another positive definite matrix difference can be defined using covariance matrices rather than information matrices. This matrix is defined as $\mathbf{K}_{nom} = \mathbf{P}_{dp}(t_n) - \mathbf{P}_{du}(t_n)$. This matrix along with the nominal sensor resolution will be used in the following section.

3.1 Rate and Resolution

The ellipsoids of the information matrix $\mathbf{P}^{-1}(t_n|t_n)$ in (2.9) increase monotonically with measurements from additional sensors. Figure 3.1a illustrates a set of monotonically increasing ellipses for information matrices in $\Re^{2\times 2}$, where the sets S_i have the properties $S_1 \subset S_2 \subset \ldots \subset S_i$ and $|S_i| = i$ ($|S_i|$ denotes number of sensors in the ith set). The innermost ellipse in Figure 3.1a corresponds to the ellipse of the information update using only one sensor. Each successive ellipse proceeding outwards is a result of using one additional sensor, with the outermost ellipse using six sensors.

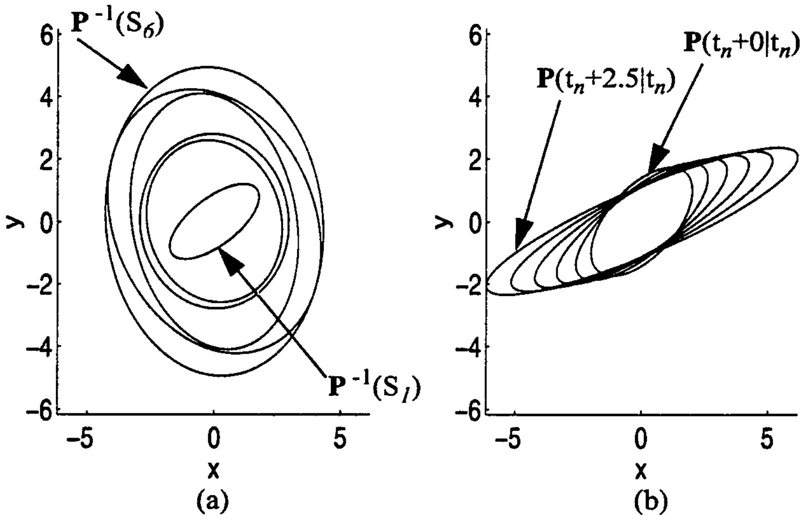


Figure 3.1 (a) Ellipses representing matrices achieving higher resolution (smaller covariance) as the number of sensors increases, (b) prediction error covariance ellipses as a function of the sample period.

The ellipses of the prediction covariance in (2.8) may or may not increase monotonically with the sample period *T*. Figure 3.1b shows non-monotonically "growing" ellipses where $\mathbf{A}(T)$, $\mathbf{Q}(T)$, and $\mathbf{P}(t_n|t_n)$ are given by

$$\mathbf{A}(T) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \mathbf{Q}(T) = \sigma^2 \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix}, \mathbf{P}(t_n|t_n) = \begin{bmatrix} 4 & 2 \\ 2 & 3 \end{bmatrix} \tag{3.5}$$

As illustrated in Figure 3.1b, the prediction covariance may decrease in some directions as the sampling period is increased. The innermost ellipse shows the prediction covariance when the sample period is 0*s*. The sample period increases by 0.5*s* for each successive ellipse proceeding outwards, with the outermost ellipse having a sample period of 2.5*s*.

3.2 Adjusting Covariance Goals

There are three different paradigms we have used for changing the information and covariance goals. Through choice of the update and prediction matrices we can influence how sensing resources are allocated in terms of rate and resolution. These paradigms for choosing the information goals provide a framework for changing the rate and resolution. However, because the prediction covariances and information updates are not independent processes, choice of one parameter will have consequences on the choice of the other parameter.

When we want a *fixed resolution* but *variable rate* then we keep the nominal sensor resolution, $\mathbf{J}_{nom}^{-1}$ constant. Given a desired update and prediction information matrix we can keep the desired resolution constant and increase the rate by adding a positive definite matrix to both desired information matrices. For example, if we let the prediction information matrix and update information matrix equal $\mathbf{P}_{dp}^{-1}(t_{n+1}) = \mathbf{P}_{dp}^{-1}(t_n) + \Delta$ and $\mathbf{P}_{du}^{-1}(t_{n+1}) = \mathbf{P}_{du}^{-1}(t_n) + \Delta$ respectively, with $\Delta > 0$, then this will achieve obtaining better estimates while keeping the nominal sensor resolution constant.

When we want a *fixed rate* but *variable resolution* then one method is to keep the matrix difference $\mathbf{K}_{nom}$ constant while increasing or decreasing each desired covariance matrix by the same positive definite matrix. For example, to increase the sensor resolution while maintaining the same desired rate the new desired prediction and update covariances are chosen to be $\mathbf{P}_{dp}(t_{n+1}) = \mathbf{P}_{dp}(t_n) - \Delta$ and $\mathbf{P}_{du}(t_{n+1}) = \mathbf{P}_{du}(t_n) - \Delta$, respectively where $\Delta > 0$. This reduces each desired covariance matrix by the same amount so that the annulus defined by the covariance difference is constant.

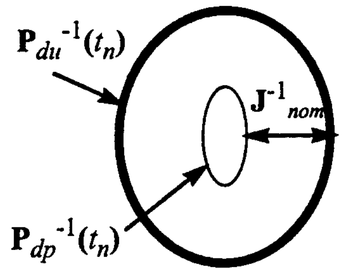
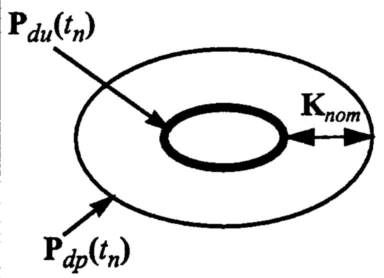
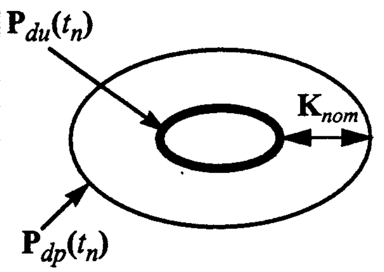
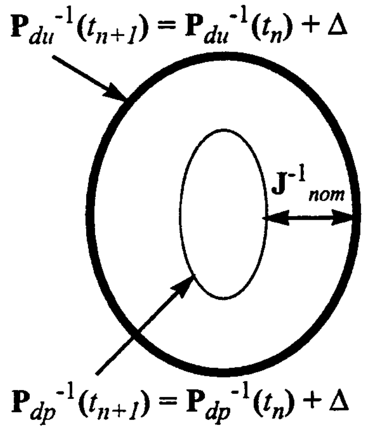
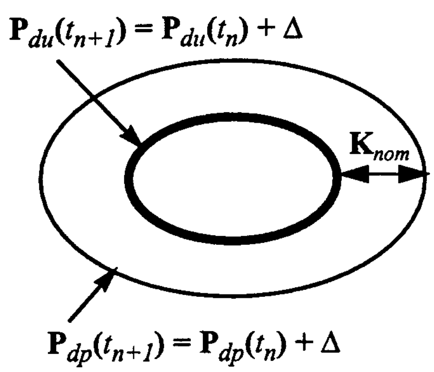
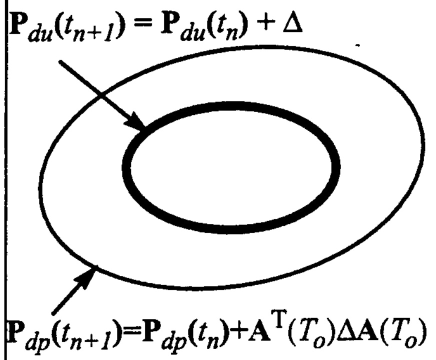
Although the matrix difference $\mathbf{K}_{nom}$ remains constant a better method for choosing new covariance goals for maintaining a fixed rate is to let $\mathbf{P}_{du}(t_{n+1}) = \mathbf{P}_{du}(t_n) - \Delta$ and $\mathbf{P}_{dp}(t_{n+1}) = \mathbf{P}_{dp}(t_n) - \mathbf{A}^{\mathrm{T}}(T_{nom})\Delta\mathbf{A}(T_{nom})$ where T_{nom} is the solution to (3.3) at time t_n. Since $\mathbf{A}^{\mathrm{T}}(T_{nom})\Delta\mathbf{A}(T_{nom})$ may or may not be positive definite we must always check the condition in (1.1). Substituting the new desired covariance goals into (3.3) we get

$$\begin{aligned} f(T) &= det[\mathbf{P}_{dp}(t_{n+1}) - \mathbf{A}^{\mathrm{T}}(T)\mathbf{P}_{du}(t_{n+1})\mathbf{A}(T) - \mathbf{Q}(T)] \\ &= det[\mathbf{P}_{dp}(t_n) - \mathbf{A}^{\mathrm{T}}(T)\mathbf{P}_{du}(t_n)\mathbf{A}(T) - \mathbf{Q}(T) + \mathbf{A}^{\mathrm{T}}(T)\Delta\mathbf{A}(T) - \mathbf{A}^{\mathrm{T}}(T_{nom})\Delta\mathbf{A}(T_{nom})] \end{aligned} \tag{3.6}$$

Since the last two matricies become the zero matrix at $T = T_{nom}$, T_{nom} is forced to be a root of $f(T)$. This method for choosing the desired covariance goals will be shown to maintain a more constant sample rate compared with the first method "constant rate" method which fixes $\mathbf{K}_{nom}$.

Computation of the nominal rate and resolution are illustrated using the ellipses shown in Table 1. Using the nominal rate and resolution to select sensors and choose a sample rate is a one time computation and is an open loop covariance control scheme. This technique is good when we have limited computational resources and insures that in steady-state that the covariance goals will be achieved. When there are multiple targets the problem is more complicated because sensing resources must then be allocated among different targets.

Table 1: Methods for Choosing Desired Matrix Goals for Maintaining a Nominal Rate or Resolution

Desired Information Matrices	Desired Covariance Matrices	Desired Covariance Matricies with Modified $\mathbf{P}_{dp}$
 $\mathbf{P}_{du}^{-1}(t_n)$ $\mathbf{J}^{-1}_{nom}$ $\mathbf{P}_{dp}^{-1}(t_n)$	 $\mathbf{P}_{du}(t_n)$ $\mathbf{K}_{nom}$ $\mathbf{P}_{dp}(t_n)$	 $\mathbf{P}_{du}(t_n)$ $\mathbf{K}_{nom}$ $\mathbf{P}_{dp}(t_n)$
 $\mathbf{P}_{du}^{-1}(t_{n+1}) = \mathbf{P}_{du}^{-1}(t_n) + \Delta$ $\mathbf{J}^{-1}_{nom}$ $\mathbf{P}_{dp}^{-1}(t_{n+1}) = \mathbf{P}_{dp}^{-1}(t_n) + \Delta$	 $\mathbf{P}_{du}(t_{n+1}) = \mathbf{P}_{du}(t_n) + \Delta$ $\mathbf{K}_{nom}$ $\mathbf{P}_{dp}(t_{n+1}) = \mathbf{P}_{dp}(t_n) + \Delta$	 $\mathbf{P}_{du}(t_{n+1}) = \mathbf{P}_{du}(t_n) + \Delta$ $\mathbf{P}_{dp}(t_{n+1})=\mathbf{P}_{dp}(t_n)+\mathbf{A}^{\mathrm{T}}(T_o)\Delta\mathbf{A}(T_o)$

3.3 Open Loop Covariance Control Strategies

Control over both the *sampling rate* (T^{-1}) and *sensor resolution* ($\mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C}$) provides two methods for maintaining the desired covariances. One method fixes T and then solves for an optimal $\mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C}$ which represents the best subset of sensors to use from the available suite of sensors. The other method is to fix the sensor resolution $\mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C}$ and then solves for an optimal T. These covariance control techniques are outlined in the following two subsections.

3.3.1 Steady State Covariance Control Using Sensor Resolution

Substituting the inverse of the information update from (2.9) into (2.8) and suppressing the time dependence on the matrix coefficients we get the following matrix Riccati equation (Bittanti, Laub, & Willems 1991) for the propagation of the prediction covariance:

$$\mathbf{P}(t_{n+1}|t_n) = \mathbf{Q} + \mathbf{A}\mathbf{P}(t_n|t_{n-1})\mathbf{A}^{\mathrm{T}} - \mathbf{A}\mathbf{P}(t_n|t_{n-1})\mathbf{C}^{\mathrm{T}}(\mathbf{R} + \mathbf{C}\mathbf{P}(t_n|t_{n-1})\mathbf{C}^{\mathrm{T}})^{-1}\mathbf{C}\mathbf{P}(t_n|t_{n-1})\mathbf{A}^{\mathrm{T}} . \quad (3.7)$$

Using this formulation we can find the theoretically optimal sensor set for maintaining the desired covariance given a fixed sampling rate. We first choose a nominal sample period *T*, such that the state transition matrix $\mathbf{A}(T)$ and process noise covariance matrix $\mathbf{Q}(T)$ are held constant. If the full state vector is measured such that $\mathbf{C} = \mathbf{I}$, we can solve for an optimal measurement covariance $\mathbf{R}$ or sensor resolution $\mathbf{R}^{-1}$ corresponding to the desired covariance. Now let the steady state prediction error covariance equal the desired prediction error covariance $\mathbf{P}(t_{n+1}|t_n) = \mathbf{P}(t_n|t_{n-1}) = \mathbf{P}_{dp}$.

$$\begin{aligned} \mathbf{P}_{dp} &= \mathbf{Q} + \mathbf{A}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} - \mathbf{A}\mathbf{P}_{dp}\mathbf{C}^{\mathrm{T}}(\mathbf{R} + \mathbf{C}\mathbf{P}_{dp}\mathbf{C}^{\mathrm{T}})^{-1}\mathbf{C}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} \\ \mathbf{A}\mathbf{P}_{dp}(\mathbf{R} + \mathbf{P}_{dp})^{-1}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} &= \mathbf{Q} + \mathbf{A}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} - \mathbf{P}_{dp} \\ (\mathbf{R} + \mathbf{P}_{dp})^{-1} &= (\mathbf{A}\mathbf{P}_{dp})^{-1}(\mathbf{Q} + \mathbf{A}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} - \mathbf{P}_{dp})(\mathbf{A}\mathbf{P}_{dp})^{-\mathrm{T}} \end{aligned} \quad (3.8)$$

Computing the inverse of the last equation in (3.8) we have the optimal covariance denoted by $\mathbf{R}_o$

$$\mathbf{R}_o = \mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}}(\mathbf{Q} + \mathbf{A}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} - \mathbf{P}_{dp})^{-1}\mathbf{A}\mathbf{P}_{dp} - \mathbf{P}_{dp} \quad (3.9)$$

Given a limited set of sensors, the desired prediction covariance $\mathbf{P}_{dp}$ must be chosen such that $\mathbf{R}_o$ is positive definite. If the desired covariance results in an $\mathbf{R}_o$ that is not positive definite, then the sampling rate is not adequate for maintaining the desired prediction covariance matrix. When faster convergence of the covariance to steady-state is desired, then the sample rate may be increased so that faster convergence is achieved. Once the optimal measurement covariance is found, a search over all possible sensor combinations is performed to determine which set of sensors yields a combined $\mathbf{R}$ or $\mathbf{R}^{-1}$ that is "closest" (according to some metric) to $\mathbf{R}_o$ or $\mathbf{R}_o^{-1}$, respectively.

When $\mathbf{C} \in \Re^{p \times N}, p < N$, the full state vector is not measured and we can not compute the required inverses to solve for $\mathbf{R}_o$. We can, however, use the pseudoinverse to get an estimate of the optimal measurement covariance matrix. Let $\mathbf{M} = \mathbf{A}\mathbf{P}_{dp}\mathbf{C}^{\mathrm{T}}$ and its pseudoinverse be $\mathbf{M}^{\dagger} = (\mathbf{M}^{\mathrm{T}}\mathbf{M})^{-1}\mathbf{M}^{\mathrm{T}}$. Using the pseudoinverse, the optimal measurement covariance that gets us closest to the desired prediction covariance is

$$\mathbf{R}_o = (\mathbf{M}^{\dagger}(\mathbf{Q} + \mathbf{A}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} - \mathbf{P}_{dp})\mathbf{M}^{\dagger\mathrm{T}})^{-1} - \mathbf{C}\mathbf{P}_{dp}\mathbf{C}^{\mathrm{T}} \tag{3.10}$$

The mapping in (3.10) is from $\mathbf{P}_{dp} \in \Re^{N \times N}$ to $\mathbf{R}_o \in \Re^{p \times p}$. This allows us to solve for $p(p+1)/2$ parameters in $\mathbf{R}_o$ to maintain a desired covariance matrix $\mathbf{P}_{dp}$ with $N(N+1)/2$ parameters.

Given a fixed sample rate, the measurement resolution can be used to control the covariance. The above technique is an open loop covariance control scheme where an optimal set of sensors is computed a priori that drives the steady state prediction covariance close to the desired prediction covariance. Figure 3.2 shows the convergence of the error variance for a single state system. The optimal sensor resolution is used so that we obtain perfect convergence to the desired prediction variance.

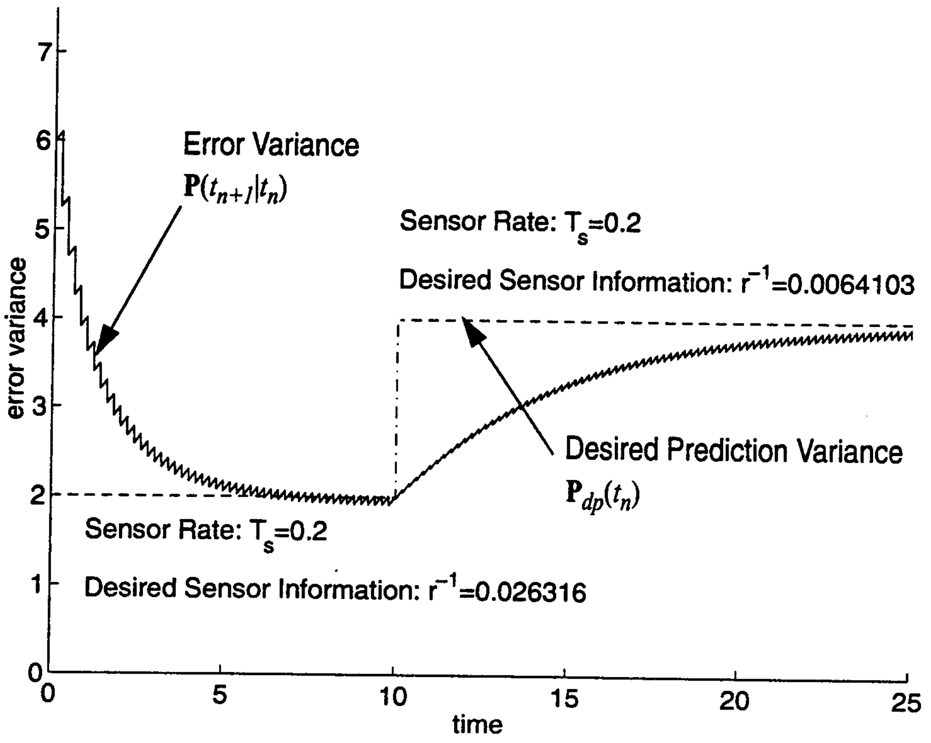


Figure 3.2 Variance convergence using variable resolution

Notice how the peaks of the sawtooth waveform converge to the desired prediction variance. At $T = 10s$ the desired prediction variance is increased to *4*. The sample rate stays constant throughout the entire simulation. When the desired prediction variance is increased, the optimal solution for the resolution decreases. The new optimal sensor resolution is used and causes the prediction variance to converge to the desired prediction variance at steady state.

3.3.2 Steady State Covariance Control Using Sensor Rate

Given a fixed sensor resolution (subset of sensors), the sample rate can be used to control the covariance. We use the same equation as in (3.7), however we try to solve for the optimal sample period T that drives the steady state prediction covariance to the desired prediction covariance. Using the same scalar system as in the previous example except with a fixed set of sensors we obtain the Riccati equation given by

$$p(t_{n+1}|t_n) = \frac{a^2}{p^{-1}(t_n|t_{n-1}) + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}} + q(T) \tag{3.11}$$

For a scalar system the optimal sample period can be found directly without the use of the determinant for the minimization in (3.3). In this example we have

$$\begin{aligned} a &= 1, q(T) = \sigma^2 T, \sigma = 2 \\ r_e^{-1} &= \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C} = 0.03125 \\ \mathbf{P}_{dp}(t_{n+1}|t_n) &= 2, 0 < t < 10 \\ \mathbf{P}_{dp}(t_{n+1}|t_n) &= 4,\ 10 < t < 25 \end{aligned} \tag{3.12}$$

Figure 3.3 shows the convergence of the error variance for this scalar system. The same desired prediction variances are used in this simulation as the one shown in Figure 3.2.

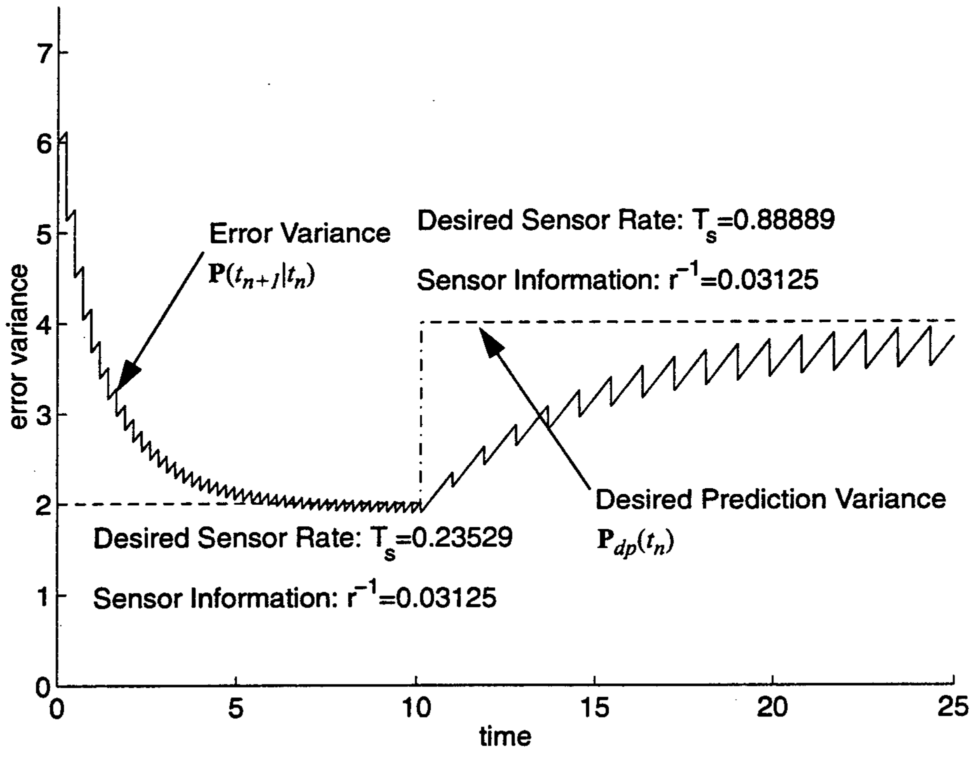


Figure 3.3 Variance convergence using variable rate

At $T = 10s$, the desired variance is increased to *4*. The sensor resolution stays constant throughout the entire simulation. When the desired variance is increased, the optimal solution for the sample rate decreases. The new sample rate is used and causes the prediction variance to converge to the desired prediction variance at steady state.

3.4 Closed Loop Covariance Control Strategies

The open loop covariance control techniques have low computational complexity and allocate sensing resources so that the steady-state desired covariance goals are achieved. However, faster convergence of the error covariance to the desired covariance goals can be achieved if feedback from the Kalman Filter is used to allocate sensing resources. In the following development of closed loop covariance control techniques, the rate and resolution are assumed to both be variable parameters.

3.4.1 Resolution Computation

We considered a number of functions and metrics (Baltz 1999) and found that the trace of the difference between actual and desired information matrices (3.13) to yield good results in the sensor selection process.

$$g(\Gamma) = trace(\mathbf{P}^{-1}(t_n|t_n) - \mathbf{P}_{du}^{-1}(t_n)) \tag{3.13}$$

The resolution (set of sensors) is computed by minimizing the function $g(\Gamma)$

$$\Gamma_o(t_n) = \underset{\Gamma}{\text{argmin}}\ g(\Gamma) \tag{3.14}$$

where Γ is a subset of the available sensors and $\mathbf{P}^{-1}(t_n|t_n)$ is given in (2.9) with the summation in (2.12) over the sensors in Γ. Since we treat $\mathbf{P}_{du}^{-1}(t_n)$ as a lower bound, we also impose the condition that the diagonal values of the matrix difference in the argument of the *trace* be greater than or equal to zero, which is a neccasary but not sufficient condition for positive semidefiniteness. Define the argument of the trace to be

$$\mathbf{H}(\Gamma) = \mathbf{P}^{-1}(t_n|t_n) - \mathbf{P}_{du}^{-1}(t_n) = \mathbf{P}^{-1}(t_n|t_{n-1}) + \sum_{j \in \Gamma} \mathbf{C}_j^T \mathbf{R}_j^{-1} \mathbf{C}_j - \mathbf{P}_{du}^{-1}(t_n) \tag{3.15}$$

With the trace function, the sets of sensors that yield negatives on the diagonal of $\mathbf{H}(\Gamma)$ can immediately be removed from the search set because a negative diagonal element guarantees at least one negative eigenvalue. The number of possible sensor sets Γ is 2^M, where M is the number of sensors available. The upper bound on the number of sensor combinations grows exponentially with additional sensors. Because of this exponential growth it is important to find computationally efficient functions and efficient search strategies for finding the optimal set of sensors (Baltz, 1999).

While eliminating these sets reduces the search size, having positive diagonals does not guarantee that the difference $\mathbf{H}(\Gamma)$ will be positive definite. To provide better covariance control, with additional computation, we can also check the condition

$$\lambda_{min}(\mathbf{P}^{-1}(t_n|t_n) - \mathbf{P}_{du}^{-1}(t_n)) \geq 0 \tag{3.16}$$

which will insure that the information update is greater than the desired information update.

3.4.2 Rate Computation

The covariance matrix of the prediction state estimate may also be controlled through choice of the sensor sample period. The function we minimize to solve for the optimal sample period is

$$f(T) = det(\mathbf{P}_{dp}(t_n) - \mathbf{P}(t_n + T|t_n)) \tag{3.17}$$

$$T_o(t_n) = \min_{T > 0} \arg f(T) = 0 \tag{3.18}$$

where $\mathbf{P}(t_n + T|t_n)$ is given in (2.8) and T_o is the optimal sample period. The function $f(T)$ will always have at least N real roots because the matrix difference in the determinant is symmetric. Denoting the roots of $f(T)$ as z_j, we let the optimal sample period be $T_o = min(z_j) \geq T_{min}$. When $min(z_j) < T_{min}$, then we let $T_o = T_{min}$. In this method, after the state estimate covariance has been updated, the optimal sample period is then determined by analyzing the difference between the desired and prediction covariance as in (3.15). This optimization can be performed on a per-sample basis to obtain the best covariance control or performed at a lower rate to reduce computational complexity, but this would also reduce the sensor manager's ability to control the covariance.

The prediction covariance is $\mathbf{P}(t_{n+1}|t_n) = \mathbf{A}(T)\mathbf{P}(t_n|t_n)\mathbf{A}^{\mathrm{T}}(T) + \mathbf{Q}(T)$ where matrices $\mathbf{A}(T)$ and $\mathbf{Q}(T)$ are functions of the sample period $T = t_{n+1} - t_n$. Define the argument of the determinant to be

$$\mathbf{M}_N(T) = \mathbf{P}_{dp}(t_n) - \mathbf{P}(t_n + T|t_n) = \mathbf{P}_{dp}(t_n) - \mathbf{A}(T)\mathbf{P}(t_n|t_n)\mathbf{A}^{\mathrm{T}}(T) - \mathbf{Q}(T) \tag{3.19}$$

where $\mathbf{M}_N(T) \in \Re^{N \times N}$ and the goal is to determine T such that $\mathbf{M}_N(T) = \mathbf{0}$ or is as "close" to the zero matrix as possible. The characteristic equation of $\mathbf{M}_N(T)$ is

$$f(s) = det(s\mathbf{I} - \mathbf{M}_N(T)) = \sum_{k=0}^{N} b_k(T)s^k = \prod_{k=1}^{N}(s - \lambda_k(T)) \tag{3.20}$$

where the eigenvalues are all real because $\mathbf{M}_N(T)$ is symmetric. Letting $s = 0$ we have $b_0(T) = \prod_{k=1}^{N} -\lambda_k(T)$. The roots of the polynomial $b_0(T)$ correspond to the values of T that make the product of the eigenvalues equal to zero. The $b_0(T)$ polynomial for a discretized-continuous model (Bar-Shalom & Li, 1993) has degree N^2.

When the update covariance has the property $\mathbf{P}_{dp}(t_n) > \mathbf{P}(t_n|t_n)$, we can *always* find a positive value for the sample period. Figure 3.4 provides a graphical argument of why there must exist a positive value of *T*. The innermost ellipse represents the update covariance and the dark solid ellipse represents the desired prediction covariance. Given $\mathbf{P}_{dp}(t_n) > \mathbf{P}(t_n|t_n)$, as $T \to \infty$ the ellipse of $\mathbf{P}(t_n + T|t_n)$ can either

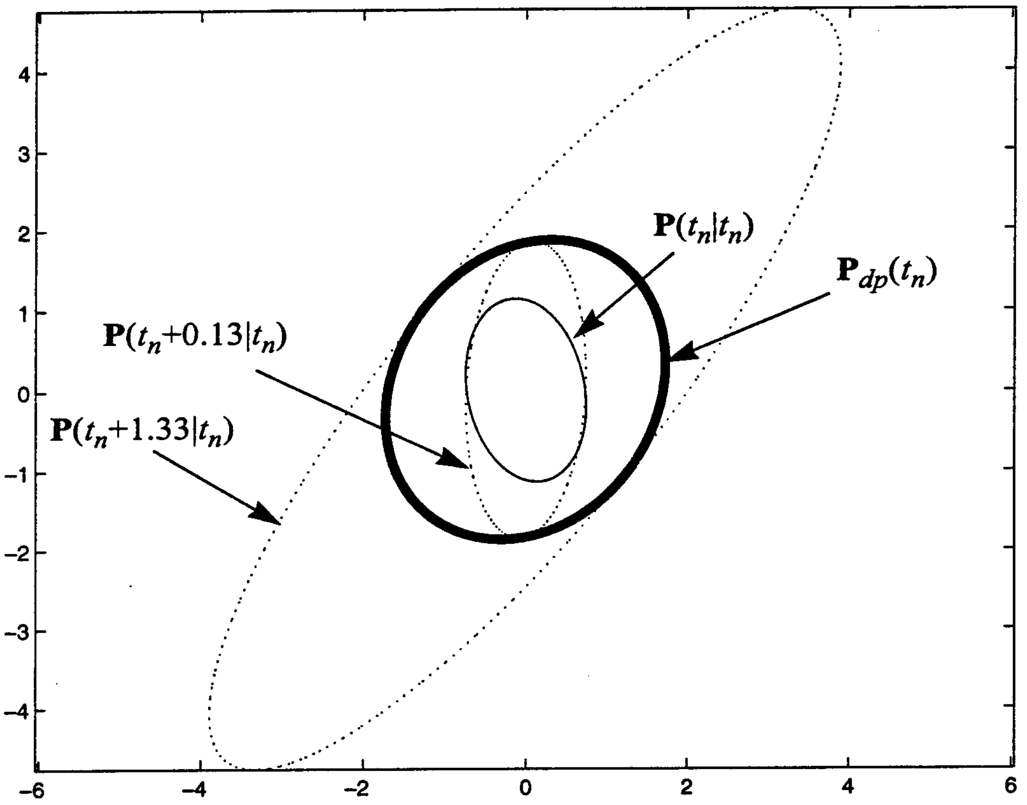


Figure 3.4 Two solutions for the sample period when the desired prediction covariance is greater than the update covariance

always stay inside of $\mathbf{P}_{dp}(t_n)$ or eventually equal or exceed $\mathbf{P}_{dp}(t_n)$. Since $\mathbf{P}(t_n + T|t_n)$ increases without bound as $T \to \infty$, then it must exceed $\mathbf{P}_{dp}(t_n)$ for some positive *T*. At *T*=0.13 seconds and *T*=1.33 seconds, the prediction covariance ellipse is tangent to the desired ellipse. These values of *T* are thus two positive roots of the polynomial $b_o(T)$. The minimum positive root of $b_o(T)$ is the largest sample period *T* for which $\mathbf{P}(t_n + T|t_n)$ remains completely within the ellipse of $\mathbf{P}_{dp}(t_n)$, in this case 0.13 seconds.

Depending on whether the discretized-continuous model or the direct-discretized model (Bar-Shalom & Li, 1993) are used will determine the coefficients of the polynomial. These models differ in the assumptions made on the continuous white noise input to the target dynamics. The discretized-continuous model assumes the white noise to be continuous and the direct-discretized model assumes the white noise input to be piece-wise constant during each sample interval. Using these two models, let us examine the equations for first and second order systems. Let each element of the desired prediction covariance be $[\mathbf{P}_{dp}(t_n)]_{ij} = d_{ij}$ and each element of the update covariance be $[\mathbf{P}(t_n|t_n)]_{ij} = p_{ij}$.

For a first order system using the discretized-continuous model to express the process noise covariance we have

$$\mathbf{M}_1(T) = d_{11} - a^2 p_{11} - T\sigma^2 \tag{3.21}$$

where $\mathbf{A}(T) = a$ and $\mathbf{Q}(T) = T\sigma^2$. Setting $\mathbf{M}_1(T)$ equal to zero and solving for the sample period we get

$$T_o = \frac{d_{11} - a^2 p_{11}}{\sigma^2} \tag{3.22}$$

Using the direct-discrete model to express the process noise covariance we have

$$\mathbf{M}_1(T) = d_{11} - a^2 p_{11} - T^2\sigma^2 \tag{3.23}$$

where $\mathbf{A}(T) = a$ and $\mathbf{Q}(T) = T^2\sigma^2$. Setting $\mathbf{M}_1(T)$ equal to zero and solving for the sample period we get

$$T_o = \left(\frac{d_{11} - a^2 p_{11}}{\sigma}\right)^{1/2} \tag{3.24}$$

The solutions (3.20) and (3.22) for the optimal sample periods in the two noise models are related by the square root.

Using a discretized-continuous model with a second order system tracking a target's position and velocity in one coordinate, the state transition matrix and process noise covariance matrix are

$$\mathbf{A}(T) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \mathbf{Q}(T) = \sigma^2 \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix} \tag{3.25}$$

respectively. This model describes a white noise acceleration model also called a Wiener process velocity model (Bar-Shalom & Li 1993). The matrix difference is then

$$\mathbf{M}_2(T) = \begin{bmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{bmatrix} - \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ T & 1 \end{bmatrix} - \sigma^2 \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix} \tag{3.26}$$

The determinant is $f(T) = det(\mathbf{M}_2(T)) = c_4T^4 + c_3T^3 + c_2T^2 + c_1T^1 + c_0 = 0$ where the coefficients are

$$\begin{aligned} c_4 &= \frac{1}{12}\sigma^4 \\ c_3 &= \frac{1}{3}p_{22}\sigma^2 - \frac{1}{3}d_{22}\sigma^2 \\ c_2 &= d_{12}\sigma^2 + p_{12}\sigma^2 - p_{22}d_{22} \\ c_1 &= 2d_{12}p_{22} - 2p_{12}d_{22} + p_{11}\sigma^2 - d_{11}\sigma^2 \\ c_0 &= d_{11}d_{22} - p_{12}^2 - d_{12}^2 + 2d_{12}p_{12} - p_{11}d_{22} - d_{11}p_{22} + p_{11}p_{22} \end{aligned} \tag{3.27}$$

Since the matrix $\mathbf{M}_2(T)$ is symmetric it has at least two real eigenvalues which implies that $f(T)$ must have at least two real roots. The smallest real root of $f(T)$ is the optimal sample period. The formulas for the roots of the quartic polynomial can be found in (Weisstein 1999) or computed numerically.

Using a direct-discrete model with a second order system tracking target position and velocity in one coordinate, the state transition matrix and process noise covariance matrix are

$$\mathbf{A}(T) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \mathbf{Q}(T) = \sigma^2 \begin{bmatrix} T^4/4 & T^3/2 \\ T^3/2 & T^2 \end{bmatrix} \tag{3.28}$$

With these coefficients the matrix difference is

$$\mathbf{M}_2(T) = \begin{bmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{bmatrix} - \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ T & 1 \end{bmatrix} - \sigma^2 \begin{bmatrix} T^4/4 & T^3/2 \\ T^3/2 & T^2 \end{bmatrix} \tag{3.29}$$

The determinant is $f(T) = det(\mathbf{M}_2(T)) = c_4T^4 + c_3T^3 + c_2T^2 + c_1T^1 + c_0$ where the coefficients are

$$\begin{aligned} c_4 &= \frac{1}{4}\sigma^2(p_{22}-d_{22}) \\ c_3 &= \sigma^2(d_{12}+p_{12}) \\ c_2 &= \sigma^2(p_{11}-d_{11}) - p_{22}d_{22} \\ c_1 &= 2d_{12}p_{22} - 2p_{12}d_{22} \\ c_0 &= d_{11}d_{22} - p_{12}^2 - d_{12}^2 + 2d_{12}p_{12} - p_{11}d_{22} - d_{11}p_{22} + p_{11}p_{22} \end{aligned} \tag{3.30}$$

Again, there are at least two real eigenvalues of $\mathbf{M}_2(T)$ that can be computed analytically or numerically. These same techniques can be extended to higher order models by using the appropriate matrices (Baltz 1999).

Once the nominal sample period T_{nom} has been calculated for any of the previous models.

4. Sensor Delay and Sensor Dropout

The sensor scheduler and specific sensors can cause adverse effects on the ability to control the covariance. In this section, we briefly discuss two such effects.

4.1 Sensor Delay

The first effect is *sensor delay.* There are two types of sensor delay. The first type of delay is measurement delay. This happens when a sensor either executes or obtains a measurement but the down stream processors do not receive the measured data until some later time. This causes the measurement to be delayed in time. The other type of delay happens when a sensor scheduler obtains a sensor request at time t_o but does not execute this request until a later time $t_1 > t_o$.

Figure 4.1 shows four sensor time lines each having different delay properties. Each sensor has the same sample rate and each sensor receives measurements at the same times. The pluses (+) on each sensor time line indicate when the sensor manager requests a measurement, the dots (•) indicate when the measurement execution starts, and the circles (o) indicate when the measurements are received.

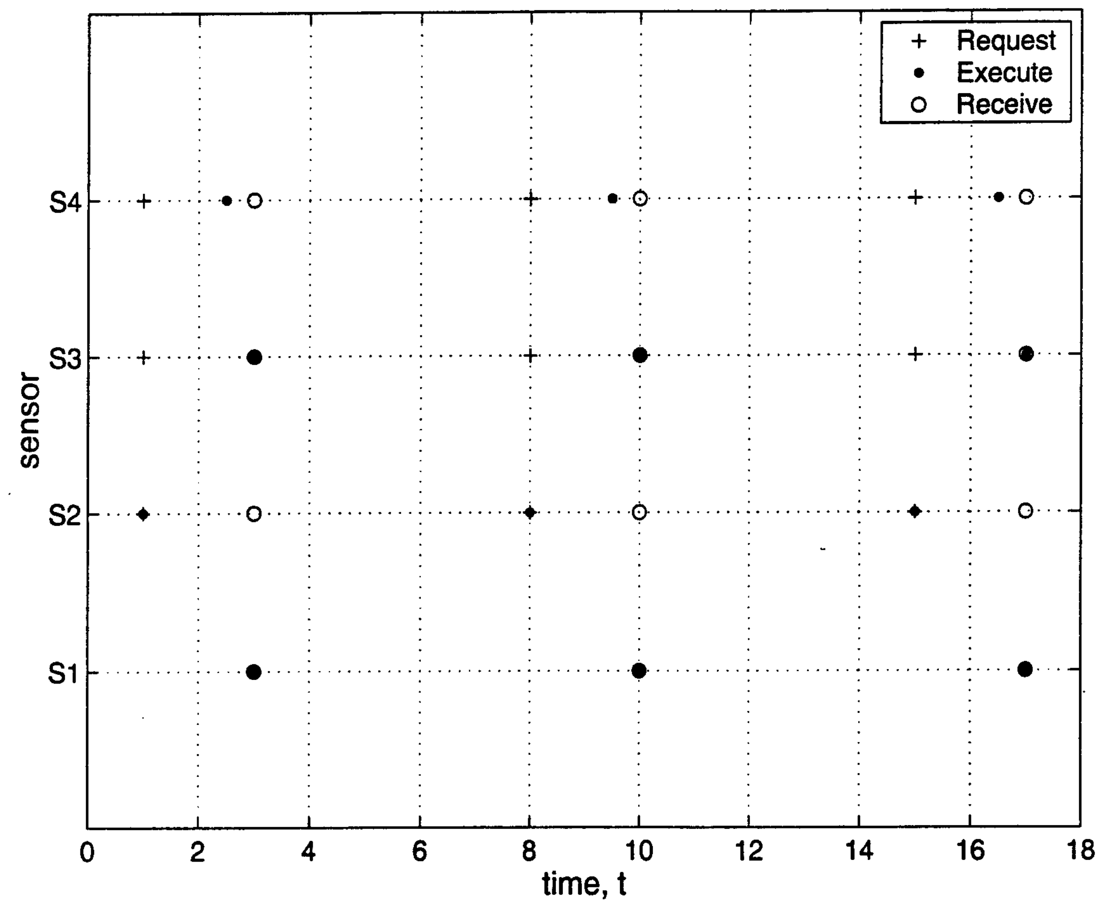


Figure 4.1 Illustration of measurement delay and sensor request delay

The first sensor S1 has neither measurement nor sensor request delay. The second sensor S2 has measurement delay but no sensor request delay. The third sensor S3 has no measurement delay but has a sensor request delay. The fourth sensor S4 has both measurement delay and sensor request delay. These two types of delay must be considered when designing a sensor manager. The effects of sensor request delays have been analyzed in (Pao & Kalandros, 1998)

4.2 Sensor Dropout

The second type of adverse effects is *sensor dropout.* Sensor dropout is described by an extended loss of sensing resources due to sensor reallocation or sensor failure. This can cause degradation in the ability to maintain a desired estimation performance. Changing the rate however can often allow the tracking performance to recover to desired levels. The general idea is that when we lose the ability to use a sensor, the rate of the remaining sensors must increase in order to compensate for the reduction in sensor information. A simple illustration can be made using a scalar state equation and vector measurements.

The state and measurement equations are as in (2.4) and (2.5) with $\mathbf{A}(t_n) = a$, $\mathbf{B}(t_n) = b = 1$, and with $\mathbf{C} = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^{\mathrm{T}}$, and $\mathbf{D} = diag\left(\begin{bmatrix} d_1^{1/2} & \dots & d_n^{1/2} \end{bmatrix}\right)$. This is a single state estimation problem using multiple sensors, where the measurements are contained in the vector $y(t_n)$. The noise terms, $bw(t_n)$ and $\mathbf{D}v(t_n)$ are zero mean with covariances $\mathbf{Q} = q = T\sigma^2$ and $\mathbf{R} = \mathbf{D}\mathbf{D}^{\mathrm{T}}\rho^2\delta(t_n - t_m)$. Using (2.8) and (2.9) with the above coefficients we have

$$p(t_{n+1}|t_n) = \frac{a^2}{p^{-1}(t_n|t_{n-1}) + \mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C}} + q \tag{4.1}$$

Let the quadratic form in the denominator of (4.1) be $\frac{1}{r_e} \equiv \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C} = \frac{1}{d_1} + \ldots + \frac{1}{d_n}$. This equation is similar to the "resistors in parallel" equation. The total information $r_e^{-1}(t_n)$ decreases as sensors dropout. Letting $p(t_{n+1}|t_n) = p(t_n|t_{n-1}) = p_{dp}$, and solving for a desired q in (4.1) we get

$$q = p_{dp} - \frac{a^2}{p_{dp}^{-1} + r_e^{-1}(t_n)} \tag{4.2}$$

With a given sensor combination, we can compute a sensor sample rate that will achieve the desired variance goal. Using $q = T\sigma^2$ in (4.2) the sample rate is

$$T^{-1}(t_n) = \frac{\sigma^2(p_{dp}^{-1} + r_e^{-1}(t_n))}{1 + p_{dp}r_e^{-1}(t_n) - a^2} \tag{4.3}$$

Notice that the process noise variance is linearly related to the sample rate. We now give an example showing how increasing the measurement rate can compensate for sensor dropout to maintain a desired variance goal.

Example: The system coefficients, the noise variances, and desired prediction variance is

$$a = 1, b = 1, \mathbf{C} = \begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}^T, d_i = 0.1, \sigma^2 = 4, \rho^2 = 1, p_{dp} = 5 \tag{4.4}$$

where $i \in \{1, \ldots, 7\}$ indexes seven different sensors having equal measurement variances. Using these coefficients the simulation removes one sensor at each 20*th* sample. Figure 4.2a and Figure 4.2b show what happens to the resolution and rate, respectively, when each sensor drops out. The effective sensor resolution decreases linearly because each sensor has the same variance.

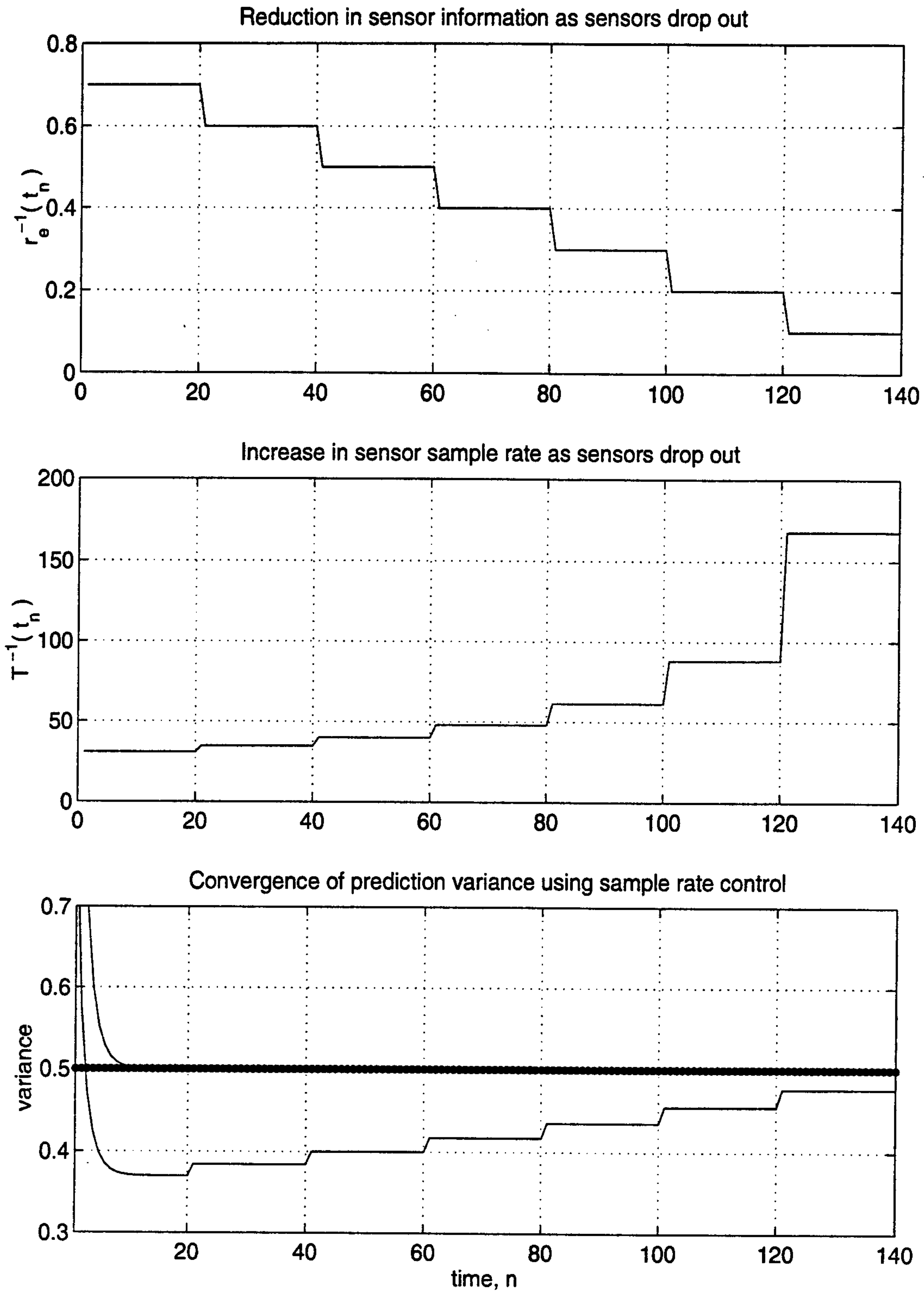


Figure 4.2 Compensation of sensor drop out using sensor rate

Figure 4.2c shows the desired prediction variance, the prediction variance, and the update variance. Once the prediction variance has converged to the desired prediction variance it is maintained there by adjusting the sensor rate of the remaining set of sensors. This gives a technique for controlling the variance when sensors dropout.

5. Simulation Results

In this section, we provide a sampling of our numerous simulation results to illustrate some of the effects of choosing different covariance goals on the rate and resolution. The first simulation fixes the difference in desired covariances $\mathbf{K}_{nom} = \mathbf{P}_{dp} - \mathbf{P}_{du}$, the second simulation fixes the desired sensor resolution $\mathbf{J}_{nom}^{-1} = \mathbf{P}_{du}^{-1} - \mathbf{P}_{dp}^{-1}$, and the third simulation uses desired covariances that cause both the rate and resolution to increase.

The three simulations use a suite of the same eight sensors, each having a minimum sample period of 0.2 seconds. Figure 5.1 shows the level curves of the sensor covariance matrices. Notice that each of the sensors have different qualities and provide more or less information in different directions. These ellipses were generated using the measurement covariance matrices given in (5.1). Matrices $\mathbf{R}_7$ and $\mathbf{R}_8$ have larger

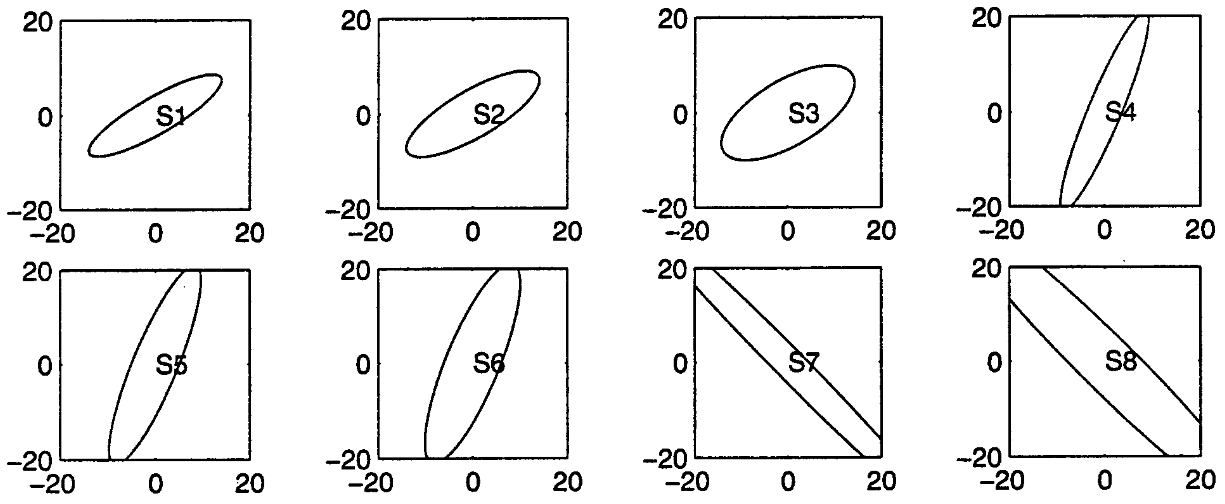


Figure 5.1 Suite of eight sensors having different covariances

condition numbers than the first six matrices, however the sensors associated with $\mathbf{R}_7$ and $\mathbf{R}_8$ still provide information about the random process in a direction that the other sensors provide very little information.

$$\mathbf{R}_1 = \begin{bmatrix} 195.75 & 104.3561 \\ 104.3561 & 75.25 \end{bmatrix}, \mathbf{R}_2 = \begin{bmatrix} 198.25 & 100.0259 \\ 100.0259 & 82.75 \end{bmatrix}, \mathbf{R}_3 = \begin{bmatrix} 204.00 & 90.0666 \\ 90.0666 & 100 \end{bmatrix}$$
$$\mathbf{R}_4 = \begin{bmatrix} 85.2233 & 176.7767 \\ 176.7767 & 438.7767 \end{bmatrix}, \mathbf{R}_5 = \begin{bmatrix} 93.7588 & 173.2412 \\ 173.2412 & 440.2412 \end{bmatrix}, \mathbf{R}_6 = \begin{bmatrix} 102.2944 & 169.7056 \\ 169.7056 & 441.7056 \end{bmatrix} \quad (5.1)$$
$$\mathbf{R}_7 = \begin{bmatrix} 1029.0 & -1019.0 \\ -1019.0 & 1029.0 \end{bmatrix}, \mathbf{R}_8 = \begin{bmatrix} 1040.0 & -1008.0 \\ -1008.0 & 1004.0 \end{bmatrix}$$

The eight sensors are used to track one target in the x-y plane. The system coefficients are

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{C}_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{Q} = \sigma^2 \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix}, i = 1, \ldots, 8 \quad (5.2)$$

The simulation stops after 60 samples have been received. After 30 samples, the user input to the sensor manager changes so that better estimates are requested by specifying different covariance goals. The new covariance goals cause different rates and resolutions to be used for tracking the target. Each simulation uses one figure to show the desired covariance goals with the update and prediction covariances; and a second figure is used to show the sensor usage, the sample rate, and the sensor resolution.

Simulation 1: The desired covariance goals are chosen to be

$$\mathbf{P}_{dp}(t_n) = \begin{bmatrix} 64 & 0 \\ 0 & 64 \end{bmatrix}, \mathbf{P}_{du}(t_n) = \begin{bmatrix} 32 & 0 \\ 0 & 32 \end{bmatrix}, n = 1, \dots, 30$$
$$\mathbf{P}_{dp}(t_n) = \begin{bmatrix} 44 & 0 \\ 0 & 44 \end{bmatrix}, \mathbf{P}_{du}(t_n) = \begin{bmatrix} 12 & 0 \\ 0 & 12 \end{bmatrix}, n = 31, \dots, 60 \tag{5.3}$$

so that $\mathbf{K}_{nom}(t_n) = \mathbf{P}_{dp}(t_n) - \mathbf{P}_{du}(t_n) = \begin{bmatrix} 32 & 0 \\ 0 & 32 \end{bmatrix}, \forall n$. The covariance ellipses are shown in Figure 5.2

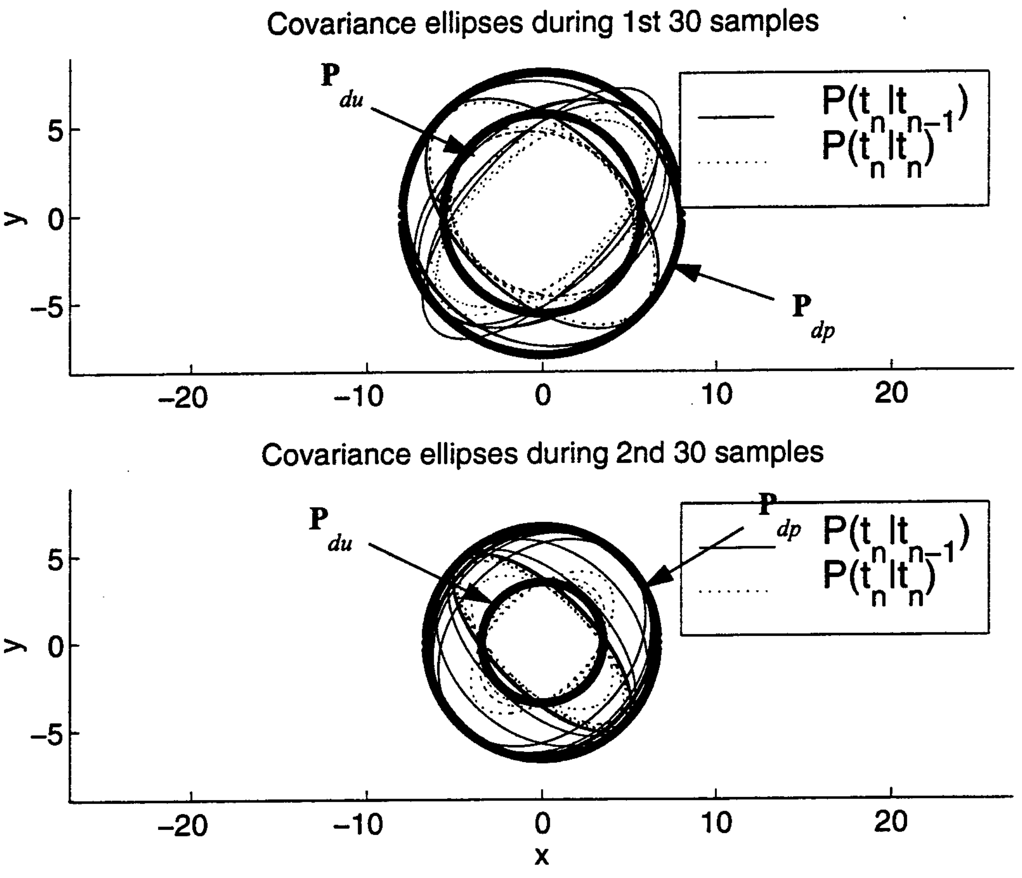


Figure 5.2 Covariance control results using constant $\mathbf{K}_{nom}(t_n)$

The sensor usage is shown in Figure 5.3a. Notice that the sensor usage is periodic after an initial transient period. When the covariance goals are changed, the rate and resolution both change. Choosing the matrix

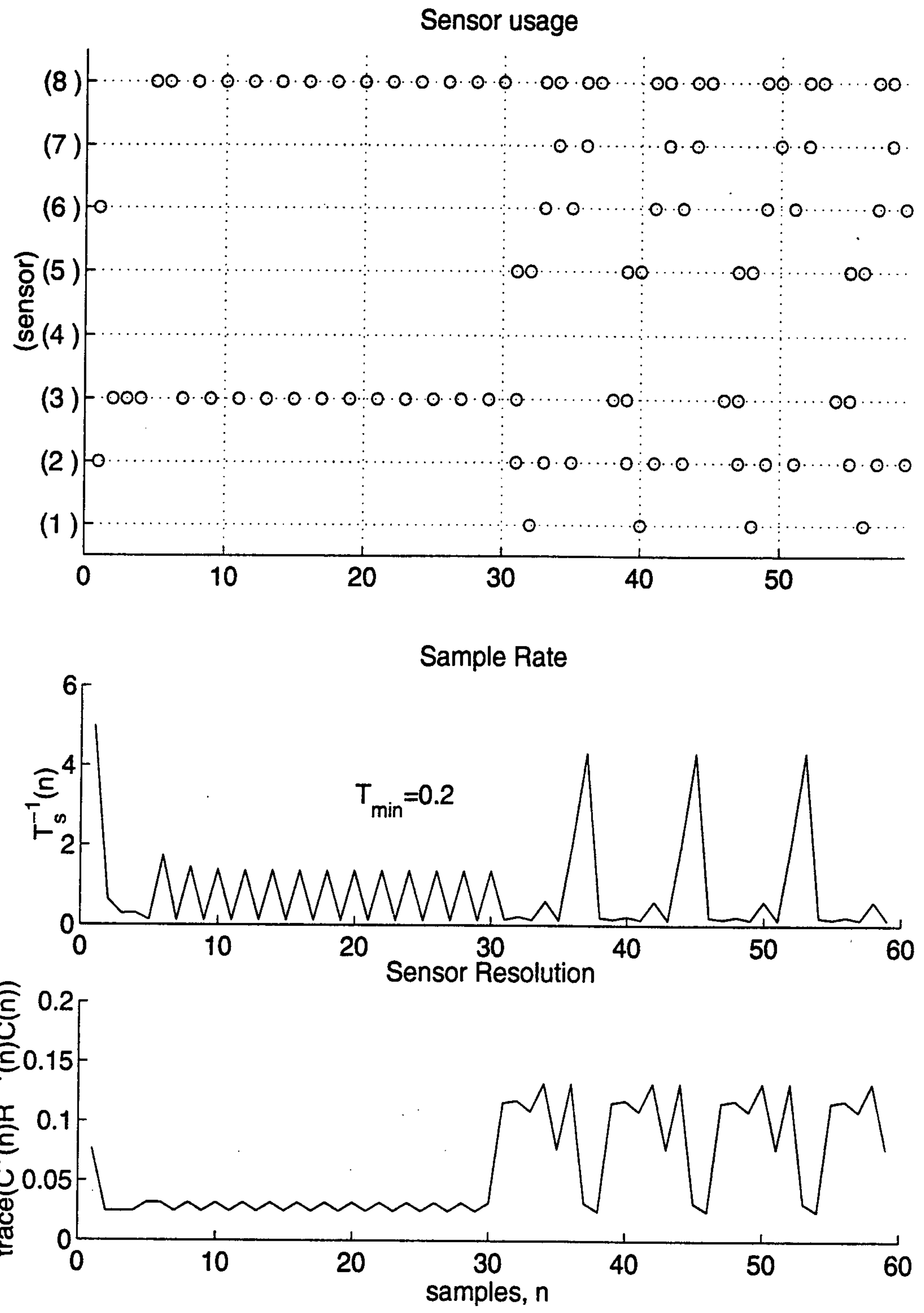


Figure 5.3 Sensor usage for constant $\mathbf{K}_{nom}(t_n)$ covariance control simulation.

$\mathbf{K}_{nom}(t_n)$ so that it is constant for the duration of the simulation fixes this annular region. One might expect that this would cause the rate to remain constant during the simulation, however because of feedback between the sensor selection and rate selection both signals become periodic.

Simulation 2: Now we will show a simulation where we hold the nominal sensor resolution constant. The desired covariance goals are chosen to be

$$\begin{aligned} \mathbf{P}_{dp}(t_n) &= \begin{bmatrix} 64 & 0 \\ 0 & 64 \end{bmatrix}, \mathbf{P}_{du}(t_n) = \begin{bmatrix} 32 & 0 \\ 0 & 32 \end{bmatrix}, n = 1, \ldots, 30 \\ \mathbf{P}_{dp}(t_n) &= \begin{bmatrix} 44 & 0 \\ 0 & 44 \end{bmatrix}, \mathbf{P}_{du}(t_n) = \begin{bmatrix} 26.1097 & 0 \\ 0 & 26.1097 \end{bmatrix}, n = 31, \ldots, 60 \end{aligned} \tag{5.4}$$

so that $\mathbf{J}_{nom}^{-1}(t_n) = \mathbf{P}_{du}^{-1}(t_n) - \mathbf{P}_{dp}^{-1}(t_n) = \begin{bmatrix} 0.0156 & 0 \\ 0 & 0.0156 \end{bmatrix}, \forall n$. For $n > 31$, the matrix $\mathbf{P}_{dp}(t_n)$ was set to be $diag(\begin{bmatrix} 44 & 44 \end{bmatrix})$ and $\mathbf{P}_{du}(t_n)$ was chosen so that the nominal sensor resolution $\mathbf{J}_{nom}^{-1}(t_n)$ is constant.

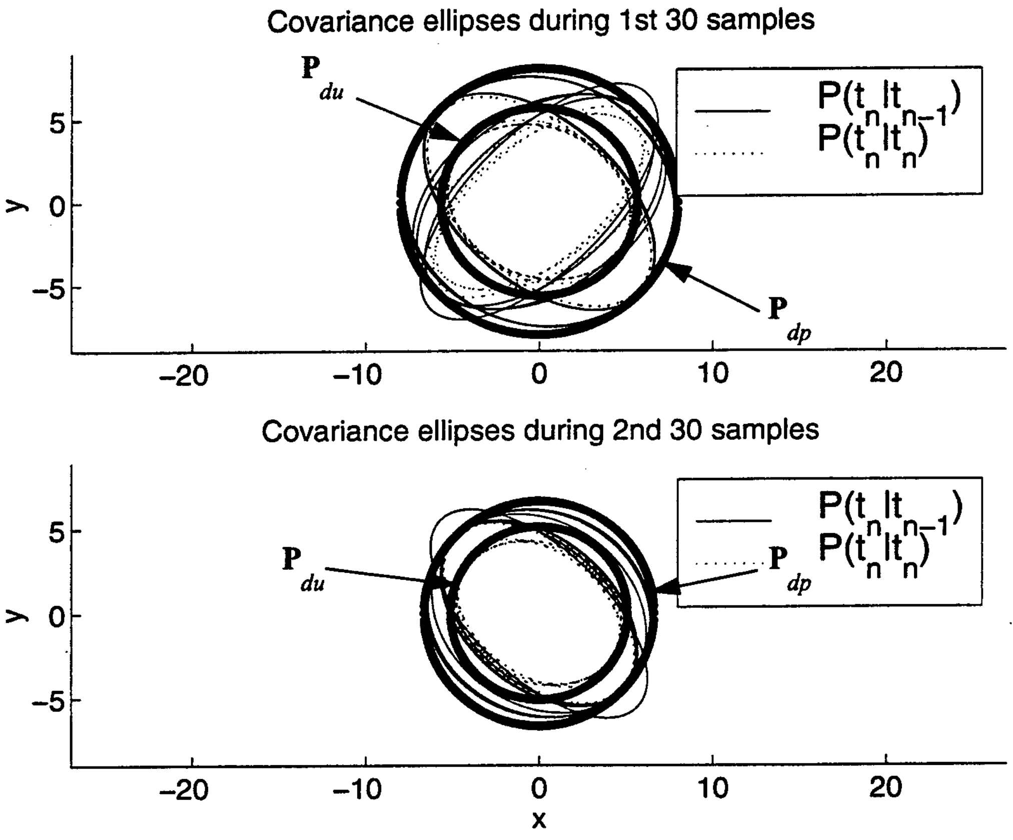


Figure 5.4 Covariance control results using constant $\mathbf{J}_{nom}^{-1}(t_n)$

The covariance ellipses are shown in Figure 5.4. The sensor usage is shown in Figure 5.5a. Notice that after the first sample the same sensors are used for the remainder of the simulation. Although the

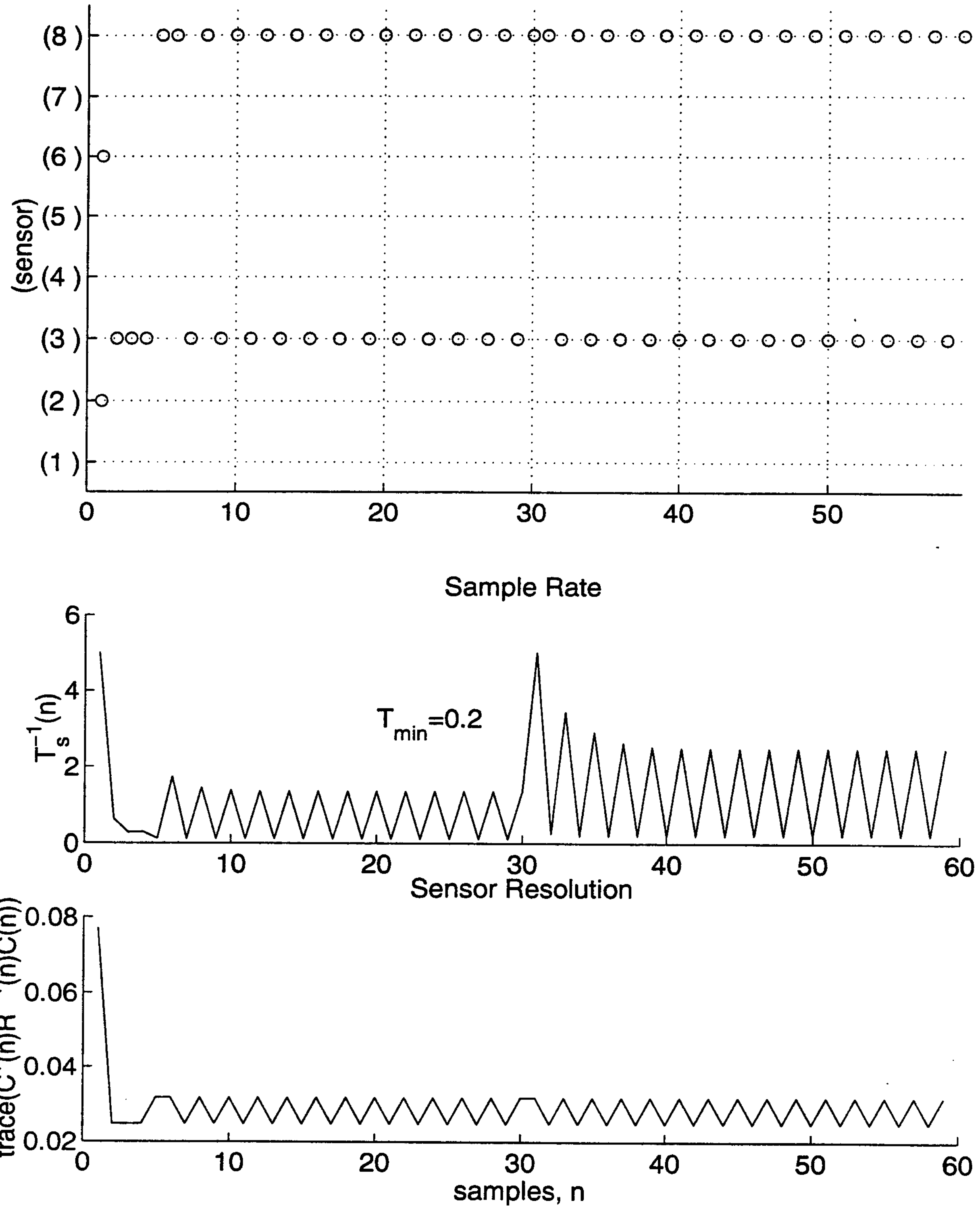


Figure 5.5 Sensor usage for constant resolution $\mathbf{J}_{nom}^{-1}$ covariance control simulation

resolution remains constant, we obtain better estimates because higher sample rates are used during the second half of the simulation.

Simulation 3: In this simulation, for $n > 30$, we switch to desired covariance goals that are correlated (i.e. non-diagonal). The desired covariance goals are chosen to be

$$\begin{aligned} \mathbf{P}_{dp}(t_n) &= \begin{bmatrix} 44 & 0 \\ 0 & 44 \end{bmatrix}, \mathbf{P}_{du}(t_n) = \begin{bmatrix} 12 & 0 \\ 0 & 12 \end{bmatrix}, n = 1, \ldots, 30 \\ \mathbf{P}_{dp}(t_n) &= \begin{bmatrix} 23.0294 & 16.9706 \\ 16.9706 & 23.0294 \end{bmatrix}, \mathbf{P}_{du}(t_n) = \begin{bmatrix} 5.7574 & 4.2426 \\ 4.2426 & 14.2426 \end{bmatrix}, n = 31, \ldots, 60 \end{aligned} \tag{5.5}$$

where the eigenvalues of the last two matrices are $\lambda_1 = 64, \lambda_2 = 16$ for $\mathbf{P}_{dp}(t_n), n > 30$ and $\lambda_1 = 16, \lambda_2 = 4$ for $\mathbf{P}_{du}(t_n), n > 30$. The desired covariances matrices for $n > 30$ are chosen so that more information is desired in a particular direction. This is shown in Figure 5.6.

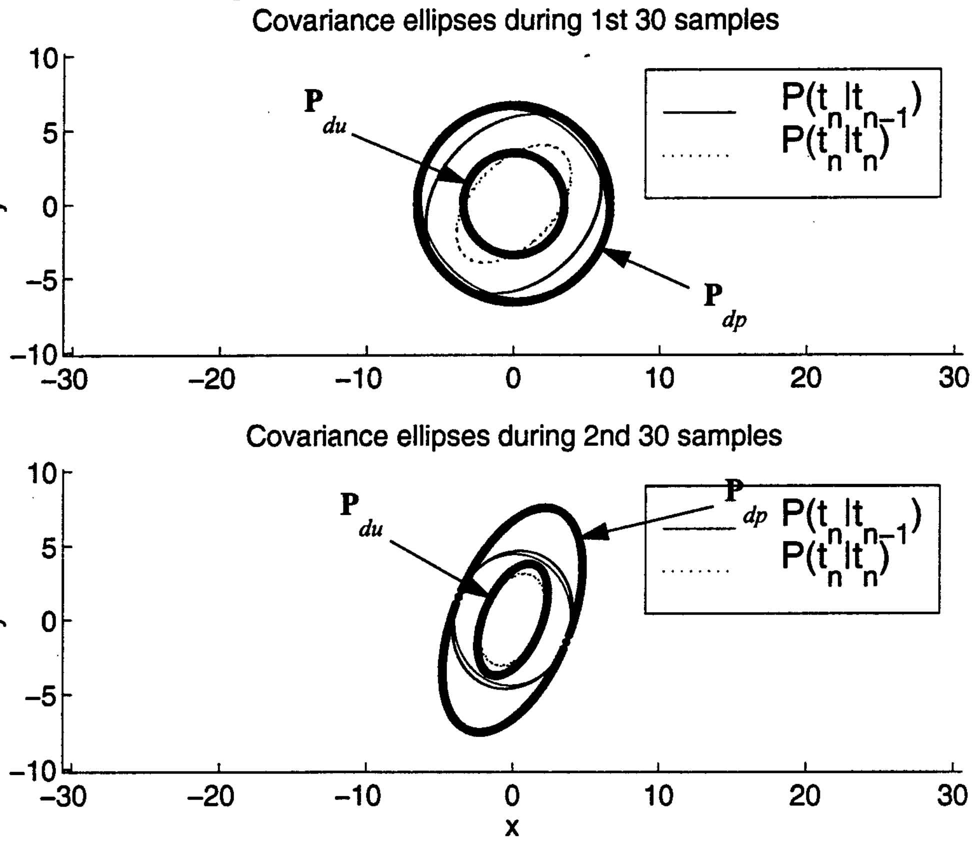


Figure 5.6 Covariance control results when switching to desired covariances that are non-diagonal

The sensor usage is shown in Figure 5.7a. Notice that after initial transient periods at $n = 1$ and $n = 31$ constant sensor sets are used.

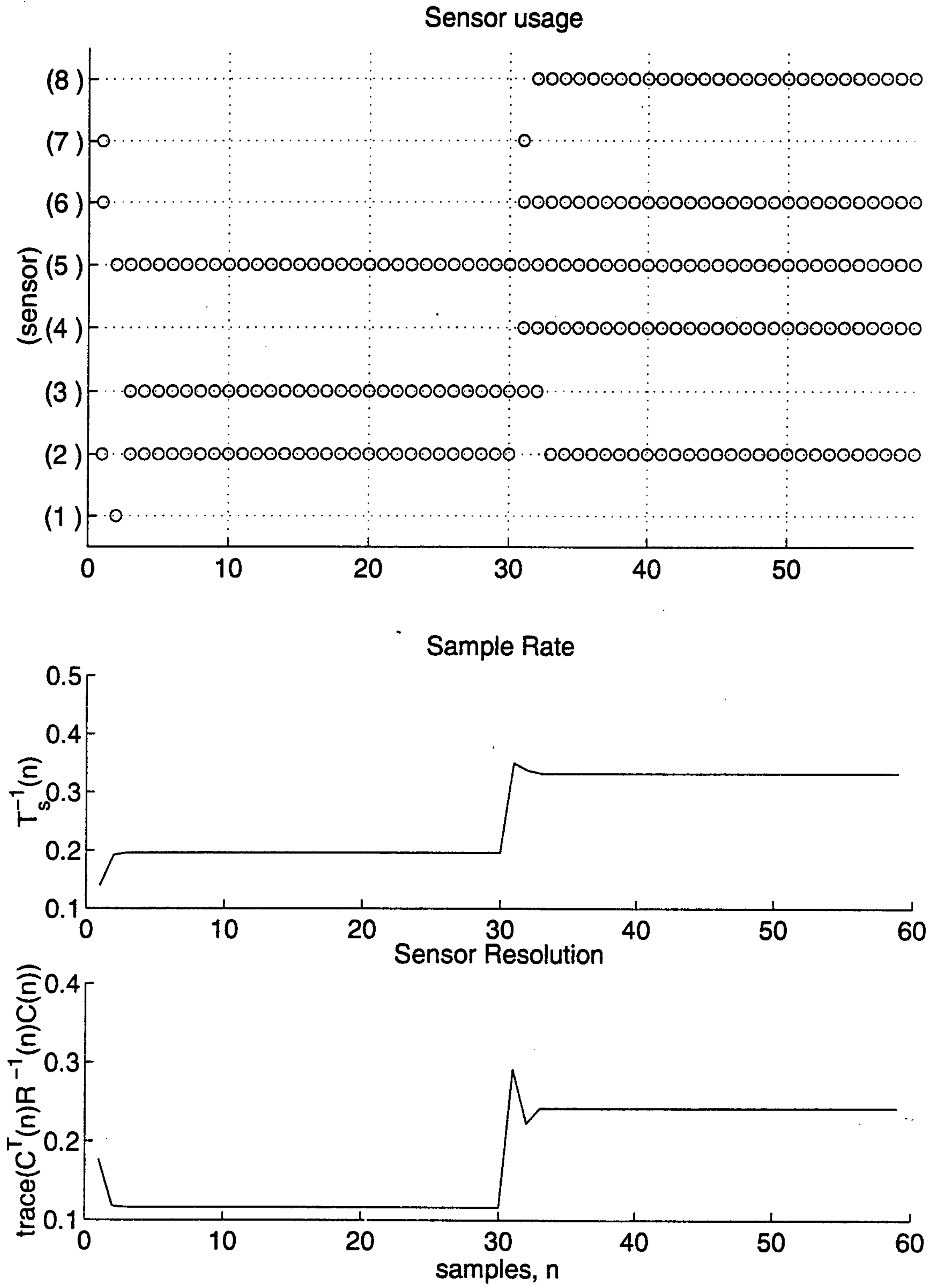


Figure 5.7 Sensor usage when switching to desired covariance goals that are non-diagonal

6. Conclusion

We have developed sensor management algorithms for allocating sensing resources in multisensor systems. The approach we have taken is that of covariance control using rate and resolution as the controllable parameters. For illustration purposes, we have kept the analysis to one and two dimensions, however the techniques introduced here for covariance control are applicable to systems with higher dimensional state vectors.

Using a covariance control approach we developed a novel sensor management scheme based upon the choice of two desired covariances. The desired prediction covariance is used to control the prediction covariance through the choice of sample rate, and the desired update covariance is used for controlling the update covariance through the choice of sensor combinations. Simulation results demonstrated the performance of the covariance control approach.

Future work consists of applying the above sensor management techniques to multitarget tracking scenarios. This may require development of new functions or metrics to optimize over, due to the additional issues involved with tracking multiple targets. Some of these issues are 1) better description of sensors in terms of agile and non-agile sensing resources and sensor capabilities, 2) crossing or interacting targets and a desire to resolve them, and 3) addressing cluttered measurements in the development of better filtering algorithms and its consequent effects on the sensor manager.

References

Baltz, N. T. (1999). *Allocation of Sensing Resources in Distributed Multiprocessor Systems*, M.S. Thesis, University of Colorado, Boulder, CO.

Bar-Shalom, Y. & Li, X. (1993). *Estimation and Tracking: Principles, Techniques, and Software,* Boston: Artech House.

Bittanti, S., Laub, A. J. & Willems, J.C. (1991). *The Riccati Equation,* Berlin: Springer-Verlag.

Blair, W. D. & Watson G. A. (1996) *Benchmark Problem for RADAR resource allocation and tracking maneuvering targets in the presence of ECM.* Dahlgren Division, Naval Surface Warfare Center, Systems Research and Technology Department.

Kalandros, M. & Pao, L. Y. (1998). "Controlling Target Estimate Covariance in Centralized Multisensor Systems," *Proceedings of the American Control Conference*, Philadelphia, PA, pp. 2749-2753.

Nash, J. (1977). "Optimal Allocation of Tracking Resources," *Proceedings of the 1977 IEEE Conference on Decision and Control*, Vol. 1, pp 1177-1180, New Orleans, LA, IEEE: New York, NY.

Pao, L. Y. & Baltz, N. T. (1999). "Control of Sensor Information in Distributed Multisensor Systems," *Proceedings of the American Control Conference*, San Diego, CA, pp. 2397-2401.

Pao, L. Y. & Kalandros, M. (1998). "The Effects of Delayed Sensor Requests on Sensor Manager Systems," *AIAA Guidance Navigation and Control Conference*, Boston, MA.

Popoli, R. (1992). "The Sensor Management Imperative," *Multitarget-Multisensor Tracking: Applications and Advances*, Bar-Shalom, ed. Vol. 2, pp. 325-392, Boston, MA: Artech House.

Schmaedeke, W. (1993). "Information-based Sensor Management," *SPIE Proceedings*, Vol. 1955.

Schmaedeke, W. & Kastella K. (1998). "Information Based Sensor Management and IMMKF," *SPIE Proceedings*, Vol. 3373, pp. 390-401.

Singer, R. A. (1970). "Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets," *IEEE Trans. on Aerospace Electronic Systems*, Vol. AES-5, pp. 473-483.

Van Keuk, G. (1978). "Software Structure and Sampling Strategy for Automatic Target Tracking with a Phased Array Radar," *AGARD Conf. Proc. No. 252, Strategies for Automatic Track Initiation*, Monterey, CA, pp. 11-1 to 11-13.

Weisstein, E. W. (1999). *CRC Concise Encyclopedia of Mathematics*, New York: CRC Press.

Proceedings of the American Control Conference
San Diego, California • June 1999

Control of Sensor Information in Distributed Multisensor Systems

L. Y. Pao and N. T. Baltz
Electrical and Computer Engineering Department
University of Colorado
Boulder, CO 80309-0425

Abstract

This paper provides an analysis of error covariance control techniques for allocating sensing resources in distributed, multiprocessor, multisensor systems. We present two algorithms for allocating sensing resources that manage the rates and resolutions at which sensor information from various nodes is processed. An elliptical annulus described by two covariance matrices is used to control the prediction and update covariances in the decentralized Kalman filter (DKF). These algorithms allow for nodal autonomy by letting each node control the usage of its own suite of sensors. With a single state filter, these sensor management techniques are shown to result in a discrete periodic Riccati equation (DPRE).

1. Introduction

In many multisensor surveillance systems, sensor management techniques are needed to balance tracking performance with system resources. Sensor management is concerned with improving or optimizing the measurement process [4]. Most sensor management techniques have considered rate and resolution separately [6,7,8]. We define the resolution as the inverse of an error covariance matrix, the so-called Fisher information. A block diagram of a decentralized tracking system is shown in Figure 1.1. This model shows the different components of the system including sensing and communications facilities along with signal and information processors and a sensor manager. This block diagram illustrates the decentralized sensor control problem, where the sensor manager uses rate and resolution to maintain the information matrix $\mathbf{P}^{-1}(t_{n+1}|t_n)$ within an elliptic annulus described by a desired update information matrix and a desired prediction information matrix given by $\mathbf{P}_{du}^{-1}(t_n)$ and $\mathbf{P}_{dp}^{-1}(t_n)$, respectively. In contrast to a centralized sensor manager, the decentralized sensor management problem is further complicated due to the presence of feedback of information from other nodes' sensors.

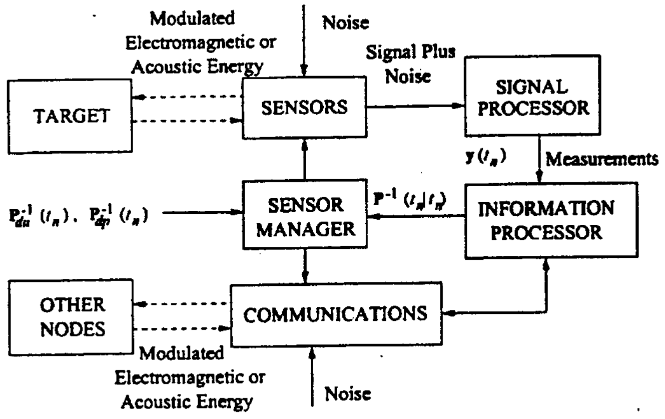

Modulated Electromagnetic or Acoustic Energy
Noise
Signal Plus Noise
TARGET
SENSORS
SIGNAL PROCESSOR
$y(t_n)$
Measurements
$\mathbf{P}_{du}^{-1}(t_n)$, $\mathbf{P}_{dp}^{-1}(t_n)$
SENSOR MANAGER
$\mathbf{P}^{-1}(t_n|t_n)$
INFORMATION PROCESSOR
OTHER NODES
COMMUNICATIONS
Modulated Electromagnetic or Acoustic Energy
Noise

Figure 1.1 Components of a Decentralized Tracking System

This paper is organized as follows. Section 2 develops the DKF. Section 3 develops the two distributed multiprocessor sensor control algorithms. The results of simulations are used to illustrate how the algorithms are applied to a tracking system. Finally, Section 4 gives some conclusions and discusses issues for further investigation.

2. Decentralized Kalman Filter

We assume there are M fully connected nodes, each having m_i sensors capable of taking measurements of length k_{ij} where $i \in \{1, \ldots, M\}$ and $j \in \{1, \ldots, m_i\}$. The first step is to partition the centralized equations. Assuming simultaneous reception of all measurements, the measurement vector, the measurement matrix, and the noise components are

$$y(t_n) = \left[y_1^{\mathrm{T}}(t_n) \ \ldots \ y_M^{\mathrm{T}}(t_n)\right]^{\mathrm{T}}$$

$$\mathbf{C}(t_n) = \left[\mathbf{C}_1^{\mathrm{T}}(t_n) \ \ldots \ \mathbf{C}_M^{\mathrm{T}}(t_n)\right]^{\mathrm{T}}$$

$$v(t_n) = \left[v_1^{\mathrm{T}}(t_n) \ \ldots \ v_M^{\mathrm{T}}(t_n)\right]^{\mathrm{T}}$$

respectively. When the measurement noise terms across each processor are uncorrelated, the covariance has a block diagonal structure, $\mathbf{R}(t_n) = blockdiag\left[\mathbf{R}_1(t_n) \ \ldots \ \mathbf{R}_M(t_n)\right]$. These steps simply partition the centralized measurement equation so that each node receives the respective measurements. The ith node in the network has the following state and measurement equations

$$x(t_{n+1}) = \mathbf{A}_i(t_n)x(t_n) + \mathbf{B}_i(t_n)w_i(t_n) \tag{2.1}$$

$$y_i(t_n) = \mathbf{C}_i(t_n)x(t_n) + \mathbf{D}_i(t_n)v_i(t_n) \tag{2.2}$$

respectively, where $\mathbf{A}_i(t_n), \mathbf{B}_i(t_n) \in \Re^{N \times N}$, $\mathbf{C}_i(t_n) \in \Re^{L_i \times N}$, and $L_i = \sum_{j=1}^{m_i} k_{ij}$. The noise terms $\mathbf{B}_i(t_n)w_i(t_n)$ and $\mathbf{D}_i(t_n)v_i(t_n)$ are zero mean, with covariances $\mathbf{B}_i(t_n)\mathrm{E}[w_i(t_n)w_i^{\mathrm{T}}(t_m)]\mathbf{B}_i^{\mathrm{T}}(t_n) = \mathbf{Q}\delta_{m-n}$ and $\mathbf{D}_i(t_n)\mathrm{E}[v_i(t_n)v_i^{\mathrm{T}}(t_m)]\mathbf{D}_i^{\mathrm{T}}(t_n) = \mathbf{R}\delta_{m-n}$, respectively. Although the state equation coefficients have been indexed, they are assumed to be the same across all processors. With these definitions, each node starts by computing estimates based strictly on its own observations. The prediction and update equations at each node are the same as the standard Kalman filter equations [1].

This work has been supported in part by the Colorado Advanced Software Institute (CASI), the Data Fusion Corporation (DFC), and an office of Naval Research (ONR) Young Investigator Award (Grant N00014-97-1-0642).

Prediction Equations:

$$\hat{x}_i(t_n|t_{n-1}) = \mathbf{A}_i(t_n)\hat{x}_i(t_{n-1}|t_{n-1}) \tag{2.3}$$

$$\hat{y}_i(t_n|t_{n-1}) = \mathbf{C}_i(t_n)\hat{x}_i(t_n|t_{n-1}) \tag{2.4}$$

$$\mathbf{P}_i(t_n|t_{n-1}) = \mathbf{A}_i(t_n)\mathbf{P}_i(t_{n-1}|t_{n-1})\mathbf{A}_i^T(t_n) + \mathbf{Q}_i(t_n) \tag{2.5}$$

Update Equations:

$$\tilde{\mathbf{P}}_i^{-1}(t_n|t_n) = \mathbf{P}_i^{-1}(t_n|t_{n-1}) + \mathbf{C}_i^T(t_n)\mathbf{R}_i^{-1}(t_n)\mathbf{C}_i(t_n) \tag{2.6}$$

$$\mathbf{K}_i(t_n) = \tilde{\mathbf{P}}_i(t_n|t_n)\mathbf{C}_i^T(t_n)\mathbf{R}_i^{-1}(t_n) \tag{2.7}$$

$$\tilde{x}_i(t_n|t_n) = \hat{x}_i(t_n|t_{n-1}) + \mathbf{K}_i(t_n)(y_i(t_n) - \hat{y}_i(t_n|t_{n-1})) \tag{2.8}$$

where the carets (^) denote globally optimal estimates obtained from the previous measurement cycle and the tildes (~) denote partial estimates based strictly on new local observations. Once each processor's sensors go through a measurement cycle, each node communicates measurement (state) vectors and covariance matrices to every other node.

2.1 Assimilation of State and Variance

Using equations (2.3) - (2.8) along with the associated centralized Kalman filter equations we can derive two forms for the assimilation equations for combining state estimates [5]. The first method does not directly use measurements or measurement covariances, but rather the local state estimates and state covariances:

$$\begin{aligned}\hat{x}_i(t_n|t_n) &= \mathbf{P}_i(t_n|t_n)[\mathbf{P}_i^{-1}(t_n|t_{n-1})\hat{x}_i(t_n|t_{n-1}) \\ &+ \textstyle\sum_{j=1}^{M} \tilde{\mathbf{P}}_j^{-1}(t_n|t_n)\tilde{x}_j(t_n|t_n) - \mathbf{P}_j^{-1}(t_n|t_{n-1})\hat{x}_j(t_n|t_{n-1})]\end{aligned} \tag{2.9a}$$

$$\mathbf{P}_i^{-1}(t_n|t_n) = \mathbf{P}_i^{-1}(t_n|t_{n-1}) + \sum_{j=1}^{M} \tilde{\mathbf{P}}_j^{-1}(t_n|t_n) - \mathbf{P}_j^{-1}(t_n|t_{n-1}) \tag{2.9b}$$

The second method updates the global state and covariance using measurements and measurement covariances:

$$\begin{aligned}\hat{x}_i(t_n|t_n) &= \mathbf{P}_i(t_n|t_n)[\mathbf{P}_i^{-1}(t_n|t_{n-1})\hat{x}_i(t_n|t_n) \\ &+ \textstyle\sum_{j=1}^{M} \mathbf{C}_j^T(t_n)\mathbf{R}_j^{-1}(t_n)y_j(t_n)]\end{aligned} \tag{2.10a}$$

$$\mathbf{P}_i^{-1}(t_n|t_n) = \mathbf{P}_i^{-1}(t_n|t_{n-1}) + \textstyle\sum_{j=1}^{M} \mathbf{C}_j^T(t_n)\mathbf{R}_j^{-1}(t_n)\mathbf{C}_j(t_n) \tag{2.10b}$$

These two forms of the information update in the DKF are mathematically equivalent to the associated centralized Kalman filter information update [5]. We use the second form because it shows explicitly how the measurement information adds to the prediction information matrix. While these equations assume that every node produces a measurement, this assumption is relaxed in the algorithms developed.

With the above equations, the motivation is to develop computationally efficient ways of selecting the sample period and subsets of sensors so that a desired estimation performance is achieved.

3. Distributed Multiprocessor Sensor Control

3.1 Rate and Resolution

The level curves (ellipses) of the information matrices in (2.6) and (2.10b) increase monotonically with additional measurements from the sensors. Plot (a) in Figure 3.1 illustrates a set of monotonically increasing ellipses for information matrices in $\Re^{2\times2}$, where the sets S_i have the properties $S_1 \subset S_2 \subset \ldots \subset S_i$ and $|S_i| = i$ ($|S_i|$ denotes number of sensors in the *i*th set). The innermost ellipse in plot (a) of Figure 3.1 corresponds to the ellipse of the information update using only one sensor. Each successive ellipse proceeding outwards is a result of using one additional sensor, with the outermost ellipse using six sensors.

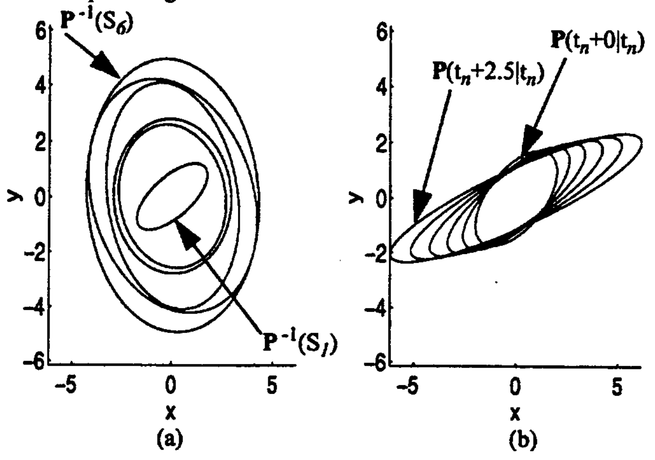


Figure 3.1 (a) Ellipses representing matrices achieving higher resolution (smaller covariance) as the number of sensors increases, (b) prediction error covariance ellipses as a function of the sample period.

The level curves of the prediction covariance in (2.5) may or may not increase monotonically with the sample period *T*. Plot (b) of Figure 3.1 shows non-monotonically "growing" ellipses where $\mathbf{A}_i(T)$, $\mathbf{Q}_i(T)$, and $\mathbf{P}_i(t_n|t_n)$ are given by

$$\mathbf{A}_i(T) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \mathbf{Q}_i(T) = \sigma_c^2 \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix}, \mathbf{P}_i(t_n|t_n) = \begin{bmatrix} 4 & 2 \\ 2 & 3 \end{bmatrix} \tag{3.1}$$

As illustrated in the plot (b), the prediction covariance may decrease in some directions as the sampling period is increased. The inner most ellipse shows the prediction covariance when the sample period is 0s. The sample period increases by 0.5s for each successive ellipse proceeding outwards, with the outermost ellipse having a sample period of 2.5s.

3.2 Covariance Control

The covariance control technique used here builds upon some of the techniques developed in [3]. The desired update and prediction covariance (information) matrices are treated as upper bounds (lower bounds) on the error covariance (information update) matrices. Since the information update is always greater than or equal to the inverse of the prediction error covariance, the desired matrices should be related as $\mathbf{P}_{du}^{-1} \geq \mathbf{P}_{dp}^{-1} \Leftrightarrow \mathbf{P}_{dp} \geq \mathbf{P}_{du}$. The desired information update matrix is used with (2.10b) to compute the optimal sensor resolution (set of sensors), and the desired prediction covariance matrix is used with (2.5) to compute the optimal internodal sampling rate. All practical systems have a minimum sample period determined by sensor limitations and possibly communication and processing delays. We therefore specify a minimum internodal sample period T_{min}.

3.2.1 Rate Computation

The function we minimize to solve for the optimal sample period is

$$T_o(t_n) = \arg \underset{\forall T > 0}{min}\ f(T) = det(\mathbf{P}_{dp}(t_n) - \mathbf{P}_i(t_n + T|t_n)) \tag{3.2}$$

where $\mathbf{P}_i(t_n + T|t_n)$ is given in (2.5) and T_o is the optimal sample period computed by the ith node. The roots of $f(T)$ are always real because the matrix difference in the determinant is symmetric. Denoting the roots of $f(T)$ as z_j, we let the optimal sample period be $T_o = min(z_j) \geq T_{min}$. When $min(z_j) < T_{min}$, then we let $T_o = T_{min}$ which will insure fast convergence of the covariance.

3.2.2 Resolution Computation

The resolution (set of sensors) is computed by minimizing

$$\Gamma_o(t_n) = \arg \underset{\Gamma}{min}\ g(\Gamma) = \|\mathbf{P}_i^{-1}(t_n|t_n) - \mathbf{P}_{du}^{-1}(t_n)\|_F \tag{3.3}$$

where Γ is a collection of subsets of the available sensors and $\mathbf{P}_i^{-1}(t_n|t_n)$ is given in (2.10b) with the summation over one of the subsets in Γ. With Φ_i the set of sensors at the ith node and letting $K = \sum_{i=1}^{M} |\Phi_i|$ and $\Phi = \bigcup_{i=1}^{M} \Phi_i$, the optimal solution for the minimization in (3.3) would be to let Γ equal the entire set of sensor combinations of Φ. The number of combinations in this set is 2^K. For a large number of nodes and/or sensors, the search for this optimal solution would be too computationally demanding for distributed real-time processing. In order to reduce the size of the search, we considered nodal ordering schemes where the minimization is done over one node's sensor(s) or groups of nodal sensor(s), between each sampling interval.

We used the Frobenius norm of the difference between actual and desired information matrices as given in (3.3). When the desired information update is required to be a lower bound, we also check the condition

$$\lambda_{min}(\mathbf{P}_i^{-1}(t_{n+1}|t_{n+1}) - \mathbf{P}_{du}^{-1}(t_n)) > 0 \tag{3.4}$$

which will insure that the information update is greater than the desired information update.

3.3 Ordered Nodes Algorithm

The *Ordered Nodes Algorithm* imposes a random ordering on the M nodes. Each node sequentially uses its own sensors and then passes the sensing task to the next node's sensor(s). When the desired update information is sufficiently "small", this algorithm is appropriate because each node's sensors can individually maintain the desired update information. This ordered sampling scheme utilizes all nodes in such a way that a single sample rate is used for each sensor while the sample rate between sensors from different nodes is multirate and periodic. The sample rate for one node's sensors will be referred to as the *intranodal rate* and the combined sample rate or communication rate between any two nodes will be referred to as the *internodal rate.*

Example:

Three nodes track one target in a single coordinate. Plot (a) of Figure 3.2 shows the sensor usage for the three nodes where nodes 1, 2, and 3 have 2, 3, and 4 sensors, respectively. This simulation used the following coefficients for the three nodes

$$\mathbf{A}_i = 1, \mathbf{Q}_i(t_n) = \sigma^2 T(t_n), i \in \{1, 2, 3\}$$
$$\mathbf{C}_1(t_n) \in \Re^2, \mathbf{C}_2(t_n) \in \Re^3, \mathbf{C}_3(t_n) \in \Re^4 \tag{3.5}$$
$$\mathbf{R}_1 = diag[13, 23], \mathbf{R}_2 = diag[6, 22, 42], \mathbf{R}_3 = diag[10, 15, 18, 35]$$

The sensors at each node are ordered from smallest to largest variance. The minimum internodal period was set to 1s. The internodal rate and sensor resolutions for all nodes are shown in plots (b) and (c), respectively. At sample $n = 21$, the desired update and prediction variances are increased which cause the sensor rate and resolution to both decrease. The peaks in the sensor resolution are due to nodes using sensor combinations that achieve the highest

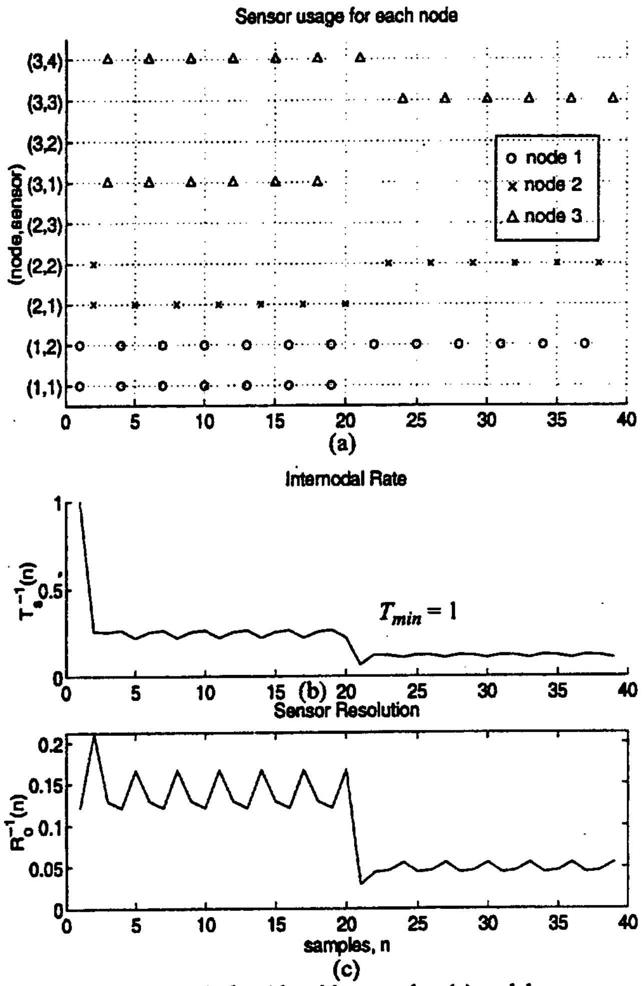


Figure 3.2 Ordered Nodes Algorithm results: (a) nodal sensor usage, (b) internodal rate, and (c) sensor resolution.

resolution during the respective time periods. Each of these peaks in the resolution have a corresponding reduction in the computed internodal rate. For instance, when node 2 uses its 1st sensor at $n = 5$, the resolution peaks and the rate (computed by node 3) decreases. These two plots illustrate the interplay between rate and resolution in maintaining a desired variance.

Figure 3.3 plots the error variance and the desired update and prediction variances. The *peaks* and *troughs* of the sawtooth waveform correspond to the prediction and update variances, respectively. Between samples the error variance increases linearly because a Wiener process is used to model the target motion. The slope of the line is determined by the white noise variance, σ^2. The error variance is plotted versus time, and illustrates how the sample period increases during the second part of the simulation.

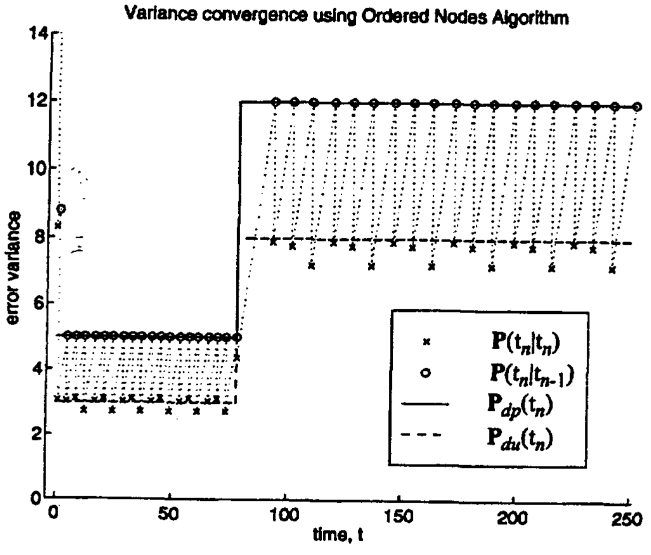


Figure 3.3 Variance convergence in Ordered Nodes Algorithm

When each node sequentially uses the same sensor(s) and the same internodal rate, the steady-state error covariance can be modeled with a discrete *periodic* Riccati equation (DPRE) [2]. The properties of the coefficients of a DPRE are

$$\begin{aligned} A(t_n) &= A(t_n+T), Q(t_n) = Q(t_n+T) \\ C(t_n) &= C(t_n+T), R(t_n) = R(t_n+T) \end{aligned} \quad (3.6)$$

where $T = \sum_{i=1}^{M} T_i(t_n)$ is the intranodal sample period. With a scalar state, the Ordered Nodes Algorithm will always result in a DPRE because the prediction variance is always the same for computing the resolution in (3.3).

The requirement of only using one node per sample period can limit the effectiveness of the Ordered Nodes Algorithm. In the Ordered Nodes algorithm, the covariance is controlled primarily through each node's choice of sample period and selection of its sensors.

3.4 Extended Ordered Nodes Algorithm

While the *Extended Ordered Nodes Algorithm* also imposes a random nodal ordering, when the desired update information can not be achieved, then the optimal sensor set at one node is passed as a group to the next node in order to perform a joint optimization over this group of sensors and its own sensors. If both nodes can not achieve the goal, a third node is used, and so on. After the state estimate is updated using this sensor combination, the process is repeated beginning with the next node. When condition (3.4) is met, then one node's sensors can achieve the bound. When condition (3.4) can not be met, the optimal set of sensors at the current node based upon the minimization of (3.3) is used along with the optimal set computed by the next node.

Example:
Five nodes track one target in the x-y plane. The state transition matrices and process noise covariances for each node are

$$A_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, Q_i(T) = \sigma^2 \begin{bmatrix} T(t_n) & 0 \\ 0 & T(t_n) \end{bmatrix}, i \in \{1, ..., 5\} \quad (3.7)$$

The five nodes each have 3 sensors that measure the state vector. Plots (a), (b), and (c) of Figure 3.4 show the sensor usage for each node, the internodal rate, and the sensor resolution, respectively. Plot (a) explicitly shows which nodes and sensors are used during each sample instance. After the first half of the simulation, the desired prediction and update covariance matrices are decreased,

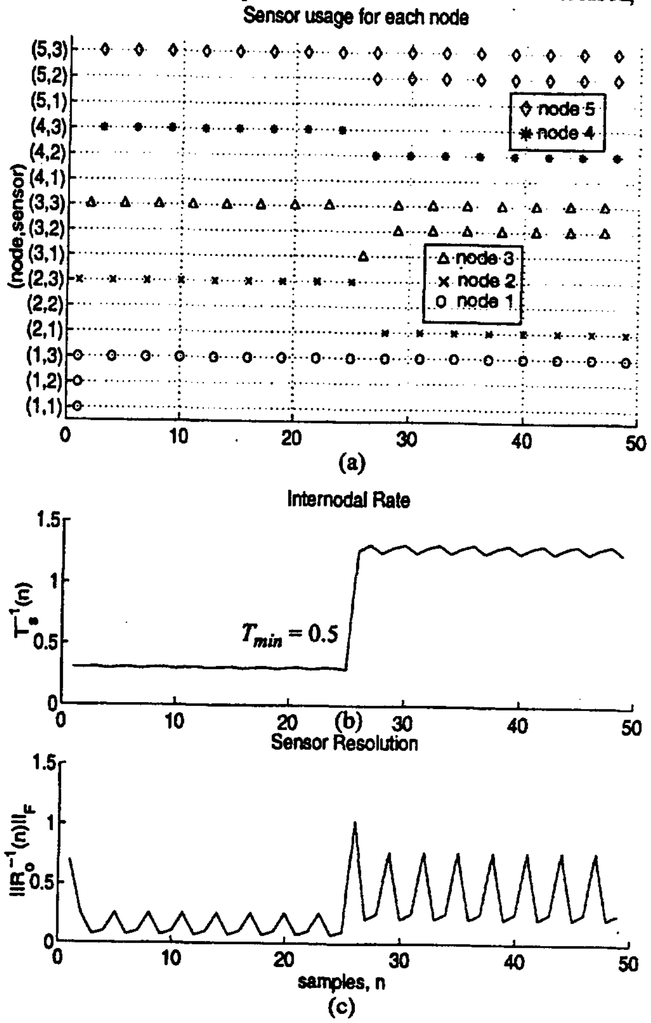


Figure 3.4 Extended Ordered Nodes Algorithm results: (a) nodal sensor usage, (b) internodal rate, and (c) Frobenius norm of combined sensor resolution.

resulting in an increase in both internodal rate and sensor resolution, i.e. more sensing resources are required to track the target. Further note how the choice of sensors is periodic. The level curves of the measurement covariance matrices are shown in Figure 3.5. The sensors for each node are ordered in increasing size, i.e. the best sensors to worst sensors. Nodes 1, 2, and 4 have ill-conditioned measurement covariances matrices.

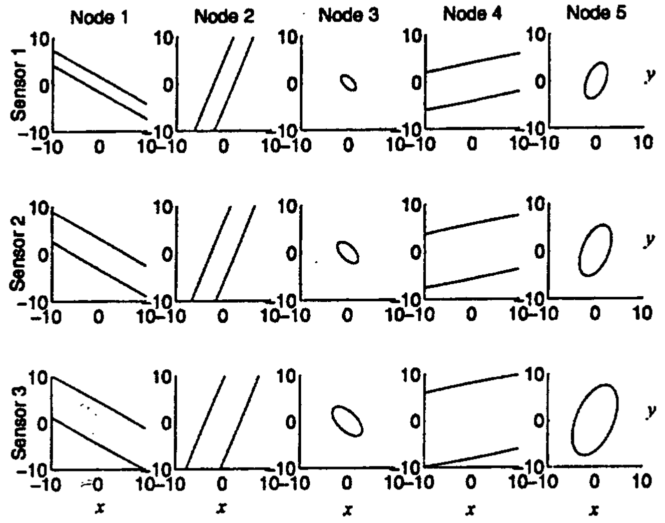


Figure 3.5 Level curves of sensor measurement covariances for each node.

As mentioned above, the elliptic annulus is decreased at $n = 26$. Plots (a) and (b) of Figure 3.6 show the desired covariances as thick solid lines and the prediction and update covariances as solid and dotted lines, respectively. The desired covariances during the 1st and 2nd halves of the simulation are related by $\mathbf{P}_{dp}(t_n) = c\mathbf{P}_{du}(t_n)$ where $c > 1$. Furthermore, the desired covariances are chosen so that the estimates are more accurate in one

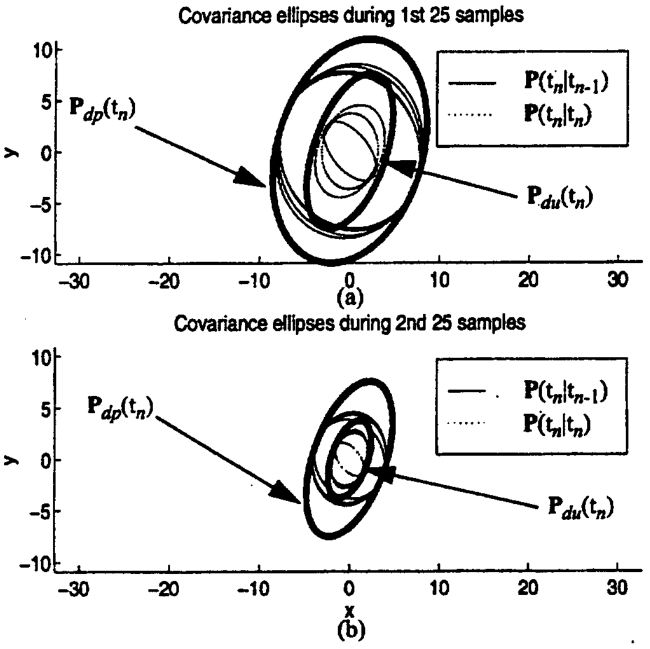


Figure 3.6 Covariance control using the Extended Ordered Nodes Algorithm showing ellipses for prediction and update covariance matrices: (a) covariance control during 1st 25 samples (b) covariance control during 2nd 25 samples.

direction than in another. The metric(s) used in this simulation were the Frobenius norm and the condition $\lambda_{min}(\mathbf{P}_{du}(t_n) - \mathbf{P}(t_n|t_n)) > 0$. As shown in Figure 3.6 this condition is met, because the level curves of $\mathbf{P}(t_n|t_n)$ are completely contained within the level curve of $\mathbf{P}_{du}(t_n)$. The update covariances then grow until the prediction covariance becomes tangent to the desired prediction covariance.

The Extended Ordered Nodes algorithm reduces to the Ordered Nodes algorithm when each node can individually satisfy the condition in (3.4).

4. Summary

We have developed two algorithms for allocating sensing resources in a distributed multisensor network. The approach we have taken is that of covariance control using rate and resolution as the controllable parameters. For illustration purposes, we have kept the analysis to one and two dimensions, however the techniques introduced here for covariance control are applicable to systems with larger state vectors. The Extended Ordered Nodes Algorithm is more flexible than the Ordered Nodes Algorithm because it allows for simultaneous sensing across nodes so that increased sensor resolution may be achieved. Issues for further study include determining the optimal nodal order that achieves the best covariance control and analyzing how the choice of the elliptic annulus affects the rate and resolution.

References

[1] Y. Bar-Shalom and X. Li, *Estimation and Tracking: Principles, Techniques, and Software*, Artech House, Boston, 1993.

[2] S. Bittanti, A. J. Laub, and J. C. Willems, *The Riccati Equation*, Springer-Verlag, Berlin, 1991.

[3] M. K. Kalandros and L. Y. Pao, "Controlling Target Estimate Covariance in Centralized Multisensor Systems", *Proceedings of the American Control Conference*, Philadelphia, PA, pp. 2749-2753, June 1998.

[4] Robert Popoli, "The Sensor Management Imperative," *Multi-target-Multisensor Tracking: Applications and Advances*, Vol. 2, pp. 325-392, Artech House, Boston, 1992.

[5] B. S. Y. Rao, H. F. Durrant-Whyte, and J. A. Sheen, "A Fully Decentralized Multi-Sensor System For Tracking and Surveillance", *The International Journal of Robotics Research*, Vol. 12, No. 1, February 1993.

[6] W. Schmaedeke, "Information-based Sensor Management," *SPIE Proceedings*, Vol. 1955, April 1993.

[7] R. A. Singer, "Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets," *IEEE Trans. on Aerospace Electronic Systems*, Vol. AES-5, pp. 473-483, July 1970.

[8] G. Van Keuk, "Software Structure and Sampling Strategy for Automatic Target Tracking with a Phased Array Radar," *AGARD Conf. Proc. No. 252, Strategies for Automatic Track Initiation*, Monterey, CA, pp. 11-1 to 11-13, October 1978.

Proceedings of the American Control Conference
San Diego, California • June 1999

On the Order of Processing Sensors in Sequential Implementations of Fusion Algorithms [1]

Lucy Y. Pao and Lidija Trailović
Department of Electrical and Computer Engineering
University of Colorado at Boulder

Abstract

We examine the order of sensor processing in the sequential Multisensor Probabilistic Data Association (MSPDA) filter for target tracking applications. If two sensors of different qualities are used, simulations and analyses show that the root mean square position error is smaller when the worse sensor is processed first.

1 Introduction

Tracking problems involve processing measurements from a target of interest, and producing, at each time step, an estimate of the target's current position and velocity vectors. Uncertainties in the target motion and in the measured values, usually modeled as additive random noise, lead to corresponding uncertainties in the target state. Additional uncertainty regarding the origin of the received data, which may or may not include measurement(s) from the targets or random clutter (false alarms), leads to the problem of data association [1].

In this paper, we analyze the sequential implementation of the Multisensor Probabilistic Data Association (MSPDA) filtering algorithm [3]. It was shown in [3, 4] that sequential processing of information from sensors of equal quality is superior to parallel processing of the sensor information, in terms of computational efficiency and two performance metrics. In tracking applications, however, sensors are usually of unequal qualities, and tracking performance may be affected by the order of processing sensor information. Thus, we investigate here the optimal order of processing sensor information (in terms of minimizing the root-mean-square position error) in sequential implementations of the MSPDA algorithm when two sensors of different qualities are used.

This paper is organized as follows. In Section 2, we review the sequential implementation of the Multisensor Joint Probabilistic Data Association (MSJPDA) algorithm. Simulation results are then presented in Section 3, followed by more analytical results in Section 4 where we consider the Modified Riccati Equation and present solutions for first through sixth-order target process models. Finally, conclusions are presented in Section 5.

2 Sequential MSJPDA Filtering

The multisensor multitarget tracking problem is to track T targets using N_s sensors in a cluttered environment. Some of the measurements arise from targets, and some from clutter; some targets may not yield any measurements in a particular time interval or for a particular sensor. The probability of detection P_D^i is assumed to be constant across targets for a given sensor i.

The dynamics of the target state $\mathbf{x}^t(k)$ are assumed to be determined by known matrices $\mathbf{F}^t(k)$ and $\mathbf{G}^t(k)$, and random vectors $\mathbf{w}^t(k)$ as follows

$$\mathbf{x}^t(k+1) = \mathbf{F}^t(k)\,\mathbf{x}^t(k) + \mathbf{G}^t(k)\,\mathbf{w}^t(k) \tag{1}$$

where $t = 1,\ldots,T$. The noise vectors $\mathbf{w}^t(k)$ are independent Gaussian random variables with zero mean and known covariance matrices $\mathbf{Q}^t(k)$.

With N_s sensors, let $M_k^i, i = 1, 2, \ldots, N_s$, be the number of measurements from each sensor i at the kth time interval. Assuming a pre-correlation gating process is used to eliminate some of the measurements [1], let m_k^i denote the number of validated measurements from sensor i at time k. The volume of a gate at time k is chosen such that with probability P_G^i the target originated measurements, if there are any, fall into the gate of sensor i. The target originated measurements are determined by

$$\mathbf{z}_{i,l_i}^t(k) = \mathbf{H}_i(k)\,\mathbf{x}^t(k) + \mathbf{v}_i^t(k), \tag{2}$$

where $t = 1,\ldots,T$, $i = 1,\ldots,N_s$, and $1 \le l_i \le M_k^i$. Matrices $\mathbf{H}_i(k)$ are known, each $\mathbf{v}_i^t(k)$ is a zero mean Gaussian noise vector uncorrelated with all other noise vectors, and the covariance matrices $\mathbf{R}_i(k)$ of the noise vectors $\mathbf{v}_i^t(k)$ are known. For a given target t and sensor i, it is not known which measurement l_i originates from the target. That is the problem of data association whereby it is necessary to determine which measurements originate from which targets [1]. Measurements not originating from targets are false measurements (or clutter), and

[1]This work was supported in part by an Office of Naval Research Young Investigator Award (Grant N00014-97-1-0642), and a National Science Foundation Early Faculty CAREER Award (Grant CMS-9625086). The authors thank Christian Frei of the Automatic Control Laboratory, Swiss Federal Institute of Technology, Zurich, Switzerland for his help in debugging the simulator.

they are assumed to be uniformly distributed throughout the surveillance region with a density λ.

A sequential implementation of the MSJPDA algorithm processes the measurements from each sensor one sensor at a time [3, 4]. The measurements of the first sensor are used to compute the intermediate state estimate $\hat{\mathbf{x}}_1^t(k|k)$ and the corresponding covariance $\mathbf{P}_1^t(k|k)$ for each target. The measurements of the next sensor are then used to further improve this intermediate state estimate. In processing each sensor's measurements, the actual association being unknown, the conditional estimate is determined by taking a weighted average over all possible associations. For $1 \le t \le T$, $1 \le i \le N_s$, and $0 \le l_i \le m_k^i$, let $\beta_{i,l_i}^t(k)$ denote the conditional probability that measurement l_i from sensor i is the true measurement from target t given all measurements received up to time k. $l_i = 0$ denotes the event that the target was not detected at time k. With $\hat{\mathbf{x}}_i^t(k|k)$ and $\mathbf{P}_i^t(k|k)$ as the state estimate and covariance, respectively, after processing the data of the ith sensor, the update equations are

$$\hat{\mathbf{x}}_i^t(k|k) = \hat{\mathbf{x}}_{i-1}^t(k|k) + \mathbf{K}_i^t(k) \sum_{l_i=0}^{m_k^i} \beta_{i,l_i}^t(k) \times \left[\mathbf{z}_{i,l_i}(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}_{i-1}^t(k|k)\right], \quad i = 1, \ldots, N_s, \tag{3}$$

where $\hat{\mathbf{x}}_0^t(k|k) = \hat{\mathbf{x}}^t(k|k-1)$ and $\hat{\mathbf{x}}_{N_s}^t(k|k) = \hat{\mathbf{x}}^t(k|k)$. With $\mathbf{P}_0^t(k|k) = \mathbf{P}^t(k|k-1)$ and $\mathbf{P}_{N_s}^t(k|k) = \mathbf{P}^t(k|k)$, the update of the covariance matrices is

$$\begin{aligned}\mathbf{P}_i^t(k|k) &= \beta_{i,0}^t(k)\mathbf{P}_{i-1}^t(k|k) \\ &+ \left[1 - \beta_{i,0}^t(k)\right]\left[\mathbf{I} - \mathbf{K}_i^t(k)\mathbf{H}_i(k)\right]\mathbf{P}_{i-1}^t(k|k) \\ &+ \mathbf{K}_i^t(k)\left[\sum_{l_i=0}^{m_k^i} \beta_{i,l_i}^t(k)\mathbf{z}_{i,l_i}(k)\mathbf{z}_{i,l_i}(k)^T \right. \\ &\left. - \sum_{l_i=0}^{m_k^i} \beta_{i,l_i}^t(k)\mathbf{z}_{i,l_i}(k) \sum_{l_i=0}^{m_k^i} \beta_{i,l_i}^t(k)\mathbf{z}_{i,l_i}(k)^T\right]\mathbf{K}_i^t(k)^T, \\ & i = 1, \ldots, N_s \end{aligned} \tag{4}$$

A superior performance (in terms of RMS position error, track lifetime, and computational efficiency metrics) of the sequential implementation of the MSJPDA over the parallel implementation was shown [3, 4] when multiple sensors of the same quality were used. If the sensors are not of equal qualities, however, a question that arises is what is the best order to process the sensor data in the sequential implementation.

3 Simulation Results

For initially comparing sequential implementations of the MSJPDA algorithm using different processing orders of sensors with different qualities, we ran Monte Carlo simulations for two sensors tracking two targets. We considered the dynamic target model (1) for $t = 1, 2$, with time-invariant matrices $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$, $\mathbf{Q}$, and $\mathbf{R}_i$. A typical state vector would include position and velocity variables. Hence, typical $\mathbf{F}$ and $\mathbf{G}$ are

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \Delta^2/2 & 0 \\ 1 & 0 \\ 0 & \Delta^2/2 \\ 0 & 1 \end{bmatrix}, \tag{5}$$

for the state vectors $\mathbf{x}^t(k) = [x \ \dot{x} \ y \ \dot{y}]^T(k)$ representing the positions and velocities of the targets at time $k\Delta$, where Δ is the time step between measurements. The two targets are initially 10 units apart and initially move in parallel directions with the same speed, but due to process and acceleration noise, directions and speed vary in time. There are two sensors whose measurements are governed by (2) with

$$\mathbf{H}_1 = \mathbf{H}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{6}$$

The process and measurement noise covariances are

$$\mathbf{Q} = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix}, \quad \mathbf{R}_1 = \begin{bmatrix} r_1 & 0 \\ 0 & r_1 \end{bmatrix}, \quad \mathbf{R}_2 = \begin{bmatrix} r_2 & 0 \\ 0 & r_2 \end{bmatrix}. \tag{7}$$

The measurements corresponding to the sensor with covariance $\mathbf{R}_1$ are processed first, and measurements from the sensor with covariance $\mathbf{R}_2$ are processed second in the sequential MSJPDA. The initial states of the targets are perfectly known, and each target is always well inside the surveillance region.

To evaluate tracking performance, one hundred Monte Carlo runs were performed for various values of clutter density λ and the average RMS position error over all runs was computed. Figure 1 shows a sampling of our results, where the following parameter values were used: $\Delta = 1$, $P_G = 0.999$, and $P_D = 1.0$. The clutter density λ was varied from 0.1 to 1.0. The system noise was varied ($q = 0.0144$ and $q = 0.0256$), and three pairs of curves were produced to compare sequential algorithm performance when the different sensors of different qualities were applied ($r_1, r_2 = 0.0064, 0.0256, 0.1024$). The two sensors used are of different qualities; the better sensor is the one with the "smaller" noise covariance matrix $\mathbf{R}_i$. With these parameter values, the expected number of false measurements per gate, using the steady-state Kalman filter covariances, varies from 0.085 to 4.672. From Figure 1, comparing the trends of the RMS position error when the simulations are run with different system noise covariance parameters q, and with different ratios of r_1 and r_2 in (7), we see that processing the worse sensor first yields smaller RMS position error.

4 Analyses

Since the Modified Riccati Equation (MRE) can be used to predict the RMS position (or other) errors of the

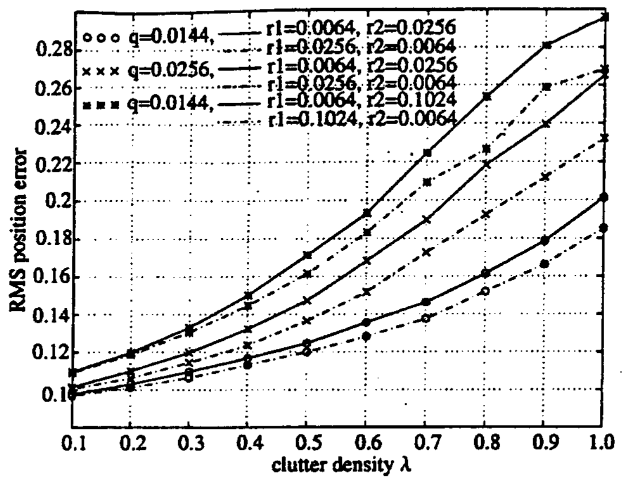


Figure 1: Average RMS position error from Monte Carlo simulations as a function of clutter density.

system [1, 2], we also applied the MRE to more efficiently evaluate whether processing the worse sensor first leads to smaller RMS errors for a wider range of system and scenario parameters. Multisensor extensions of the MRE [3] were used to predict the RMS tracking performance of the dynamic target model of (1)–(2) with $t = 1$ (single target). The time step $\Delta = 1$, $P_D = 1.0$, and $P_G = 0.999$, as before. The state vector was chosen to consist of position and velocity (as was in Section 3), and hence the position error covariance $\mathbf{P}$ is a block diagonal matrix.

For the appropriate time-invariant matrices $\mathbf{F}, \mathbf{G}, \mathbf{H}$ and $\mathbf{Q}, \mathbf{R_1}, \mathbf{R_2}$, the MRE iteration for the sequential MSPDA filter for two sensors tracking one target is

$$\mathbf{P}(k|k-1) = \mathbf{F}\,\bar{\mathbf{P}}(k-1|k-1)\,\mathbf{F}^T + \mathbf{G}\,\mathbf{Q}\,\mathbf{G}^T \qquad (8)$$

$$\mathbf{S_1}(k) = \mathbf{H}\,\mathbf{P}(k|k-1)\,\mathbf{H}^T + \mathbf{R_1} \qquad (9)$$

$$\mathbf{K_1}(k) = \mathbf{P}(k|k-1)\,\mathbf{H}^T\,\mathbf{S_1}^{-1}(k) \qquad (10)$$

$$\mathbf{P_1}(k|k) = \mathbf{P}(k|k-1) - C_k^1\,\mathbf{K_1}(k)\,\mathbf{S_1}(k)\,\mathbf{K_1}(k)^T \qquad (11)$$

$$\mathbf{S_2}(k) = \mathbf{H}\,\mathbf{P_1}(k|k)\,\mathbf{H}^T + \mathbf{R_2} \qquad (12)$$

$$\mathbf{K_2}(k) = \mathbf{P_1}(k|k)\,\mathbf{H}^T\,\mathbf{S_2}^{-1}(k) \qquad (13)$$

$$\bar{\mathbf{P}}(k|k) = \mathbf{P_1}(k|k) - C_k^2\,\mathbf{K_2}(k)\,\mathbf{S_2}(k)\,\mathbf{K_2}(k)^T \qquad (14)$$

where

$$C_k^1 = P_D P_G - q_1 + q_2(\lambda V_k^1) \qquad (15)$$

$$C_k^2 = P_D P_G - q_1 + q_2(\lambda V_k^2) \qquad (16)$$

The q_1 and q_2 functions are defined in [1, 2] and depend on the dimension of the system, P_D, λ, and V_k, the volume of the validation region (gate) at time k.

In the time-invariant case considered here, it was found [2] that for most values of P_D and λ, the equations (8)–(14) can be iterated until the covariance $\bar{\mathbf{P}}(k|k)$ converges to a steady-state covariance matrix $\bar{\mathbf{P}}$. No general stability results are known for the MRE, but numerical convergence and divergence have been observed. In order to obtain a scalar tracking performance metric, the steady-state RMS position error can be extracted [2] from the sum of the diagonal elements of the $\bar{\mathbf{P}}$ matrix corresponding to target position

$$RMS = e(P_D, \lambda) \stackrel{\text{def}}{=} \sqrt{\sum_{position} \operatorname{diag}(\bar{\mathbf{P}})} \qquad (17)$$

Using the MRE for a one target-two sensors scenario, we computed the steady state error covariance

$$\bar{\mathbf{P}}(k|k) = \bar{\mathbf{P}}(k-1|k-1) \qquad (18)$$

which is the same definition of RMS position error used in the simulations of Section 3.

4.1 One, Two, and Three-Dimensional MRE

We considered tracking systems in one, two, and three-dimensions, with appropriate system matrices (5), (6), and (7). In addition to numerically iterating the MRE in equations (8)–(14) in order to obtain the RMS position error in steady-state, we also analytically solved equation (18). The parameters used were the same as in the simulations discussed in Section 3. The state vector consisted of position, or position and velocity, for the one, two, and three-dimensional tracking scenarios.

With the state vector consisting of position and velocity components (second, fourth, and sixth order system models), from equations (8)–(14), the steady-state position error covariance matrix $\mathbf{P}$ is a block diagonal matrix with one block being

$$\mathbf{P_{block}} = \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}, \qquad (19)$$

entries of which can be computed to be the following:

$$p_{11} = \pi_{11} - C_2 \frac{\pi_{11}^2}{\pi_{11} + r_2} \qquad (20)$$

$$p_{12} = \pi_{12} - C_2 \frac{\pi_{11}\pi_{12}}{\pi_{11} + r_2} \qquad (21)$$

$$p_{22} = \pi_{22} - C_2 \frac{\pi_{12}^2}{\pi_{11} + r_2} \qquad (22)$$

where

$$\begin{aligned} \pi_{11} &= p_{11} + 2p_{12} + p_{22} + 0.25q \\ &\quad - C_1 \frac{(p_{11} + 2p_{12} + p_{22} + 0.25q)^2}{p_{11} + 2p_{12} + p_{22} + 0.25q + r_1} \qquad (23) \\ \pi_{12} &= p_{12} + p_{22} + 0.5q \\ &\quad - C_1 \frac{(p_{11} + 2p_{12} + p_{22} + 0.25q)(p_{12} + p_{22} + 0.5q)}{p_{11} + 2p_{12} + p_{22} + 0.25q + r_1} \qquad (24) \\ \pi_{22} &= p_{22} + q - C_1 \frac{(p_{12} + p_{22} + 0.5q)^2}{p_{11} + 2p_{12} + p_{22} + 0.25q + r_1}, \qquad (25) \end{aligned}$$

and C_1 and C_2 stand for C_k^1 and C_k^2, from equations (15) and (16), respectively.

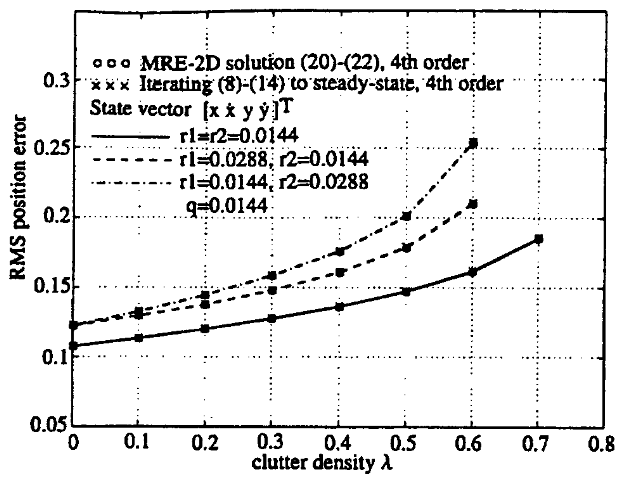


Figure 2: RMS position error in a two-dimensional model obtained by solving (20)–(22) and by MRE iterations (8)–(14) for a fourth-order system.

The RMS position error (17) can be obtained by solving the fourth degree equation system (20)–(22) numerically. The values of C_1 and C_2 are substituted from steady-state iteration of the appropriate MRE. The steady-state solution results for the fourth order system model are shown in Figure 2 for tracking in two dimensions, and compared with the results obtained by MRE iterations (8)–(14). The RMS position error is shown as a function of the clutter density λ for three sets of sensor parameters: $r_1 = r_2$, $r_1 > r_2$, and $r_1 < r_2$. The RMS position error obtained by solving the system (20)–(22) matches the results obtained by MRE iterations. When the two sensors are of different qualities ($r_1 \neq r_2$), the computed RMS position error depends on the order of sensor processing, giving the same trends as observed in the simulations: processing the worse sensor first ($r_1 > r_2$) results in smaller RMS position error (better performance). Similar results were obtained for second and sixth order target process models.

We also evaluated the MRE (8)–(14) with the state vector consisting of position only. The one-dimensional model reduces to a scalar system discussed here in detail, while the two and three-dimensional models become diagonal systems, with each diagonal entry being similar to the scalar system. For the scalar system,

$$\mathbf{F}=\mathbf{G}=\mathbf{H}=1,\ \mathbf{R}_1=r_1, \mathbf{R}_2=r_2, \mathbf{Q}=q,\ \ r_1, r_2, q>0. \quad (26)$$

The MRE iterations (8)–(14) become

$$P(k|k-1) = P(k-1|k-1)+q \quad (27)$$

$$P_1(k|k) = P(k|k-1) - C_k^1 \frac{P(k|k-1)^2}{P(k|k-1)+r_1} \quad (28)$$

$$P(k|k) = P_1(k|k) - C_k^2 \frac{P_1(k|k)^2}{P_1(k|k)+r_2} \quad (29)$$

Combining (27), (28), and (29), we have

$$\begin{aligned} P(k+1|k) &= P(k|k)+q \\ &= P(k|k-1)+q-C_k^1\frac{P(k|k-1)^2}{P(k|k-1)+r_1} \\ &\quad -C_k^2\frac{\left(P(k|k-1)-C_k^1\frac{P(k|k-1)^2}{P(k|k-1)+r_1}\right)^2}{P(k|k-1)-C_k^1\frac{P(k|k-1)^2}{P(k|k-1)+r_1}+r_2}. \end{aligned} \quad (30)$$

In steady-state, covariances are constant $P(k|k) = P(k-1|k-1) = P$, expectations are constant $P(k+1|k) = P(k|k-1) = P_1 = P+q$, C_k^1 and C_k^2 become C_1 and C_2, and expanding (30) yields a fourth degree equation

$$\begin{aligned} &(P+q)^4(1-C_1)\left[C_1+C_2(1-C_1)\right] \\ &+(P+q)^3\left[C_1(r_1+r_2)+C_2(1-C_1)2r_1-(1-C_1)q\right] \\ &+(P+q)^2\left[C_1r_1r_2+C_2r_1^2-q(r_1+r_2)-(1-C_1)qr_1\right] \\ &-(P+q)\left[qr_1r_2+qr_1(r_1+r_2)\right]-qr_1^2r_2=0. \end{aligned} \quad (31)$$

The analytical solution is [5]

$$(P+q)^4+a_3(P+q)^3+a_2(P+q)^2+a_1(P+q)+a_0=0 \quad (32)$$

$$\begin{aligned} P_{1,2,3,4} &= -q-\frac{a_3}{4}\pm\frac{1}{2}\sqrt{B} \\ &\mp\frac{1}{2}\left(-\frac{4a_2}{3}+\frac{a_3^2}{2}-\frac{\sqrt[3]{2}(12a_0+a_2^2-3a_1a_3)}{3\sqrt[3]{A}}\right. \\ &\left.-\frac{\sqrt[3]{A}}{3\sqrt[3]{2}}\pm\frac{-8a_1+4a_2a_3-a_3^3}{4\sqrt{B}}\right)^{1/2} \end{aligned} \quad (33)$$

where

$$\begin{aligned} A &= 27a_1^2-72a_0a_2+a_2^3-9a_1a_2a_3+27a_0a_3^2 \\ &+\left((27a_1^2-72a_0a_2+2a_2^3-9a_1a_2a_3+27a_0a_3^2)^3\right. \\ &\left.-4(12a_0+a_2^2-3a_1a_3)^3\right)^{1/2} \end{aligned} \quad (34)$$

$$B = -\frac{2a_2}{3}+\frac{a_3^2}{4}+\frac{\sqrt[3]{2}(12a_0+a_2^2-3a_1a_2)}{3\sqrt[3]{A}}+\frac{\sqrt[3]{A}}{3\sqrt[3]{2}} \quad (35)$$

and

$$a_0 = -\frac{qr_1^2r_2}{(1-C_1)[C_1+C_2(1-C_1)]} \quad (36)$$

$$a_1 = -\frac{qr_1^2+2qr_1r_2}{(1-C_1)[C_1+C_2(1-C_1)]} \quad (37)$$

$$a_2 = \frac{C_1r_1r_2+C_2r_1^2-(1-C_1)qr_1-q(r_1+r_2)}{(1-C_1)[C_1+C_2(1-C_1)]} \quad (38)$$

$$a_3 = \frac{C_1(r_1+r_2)+2C_2(1-C_1)r_1-(1-C_1)q}{(1-C_1)[C_1+C_2(1-C_1)]}. \quad (39)$$

The closed-form solution (33)–(39) for the steady-state scalar approximation shows how the system noise q and sensor parameters r_1 and r_2 affect the covariance P, and therefore the RMS position error. The RMS position error as a function of clutter density λ for first, second, and third order system models, obtained from equation (33) match the results obtained by MRE iterations (8)–(14) of the appropriate system models, and observed trends are similar to those of Figure 2.

4.2 Results over Larger Parameter Ranges

So far, we have shown that for several sets of sensor parameters, the system tracking performance in terms of the RMS position error improves if the worse sensor is processed first. It is of interest, however, to investigate how the order of sensor processing affects the RMS position error over a wider range of system parameters: sensor parameters r_1 and r_2, system noise q, and clutter density λ. Hence, we used MRE numerical iterations to efficiently evaluate over large parameter ranges.

We varied the ratio of the sensor parameters r_1/r_2 between 0.01 and 100, while keeping the parameter r of the equivalent sensor $\left(\frac{1}{r} = \frac{1}{r_1} + \frac{1}{r_2}\right)$ at $r = 0.01$. Figure 3 shows for a second-order, two-dimension system, how the RMS position error, normalized to the RMS position error when both sensors are equal ($r_1/r_2 = 1$), varies with r_1/r_2 for $q = 0.01$ and 0.1 and $\lambda = 0.001$, 0.2, 0.5, and 1.0. The results are given only for the parameter values where the MRE iterations converge. Results for other first through sixth order process models are similar. The range where $r_1/r_2 > 1$ corresponds to the case when the worse sensor is processed first. As expected, the RMS position error increases with increased system noise q and with increased clutter density λ.

Qualitative behavior over a large range of r_1/r_2 is mostly affected by the system order and noise q. If the noise q is low, no significant differences in the RMS position error can be observed for different orders of processing sensor information – the curves are almost symmetrical around the center point $r_1/r_2 = 1$. For larger noise q, the RMS position error exhibits a local minimum at or slightly to the right of the $r_1/r_2 = 1$ point. Around this point, there is a range of r_1/r_2 values where the RMS position error is smaller if the worse sensor is processed first. For example, in the two-dimensional, second order case, shown in Figure 3, the error for $r_1/r_2 = 10$ is smaller than the error for $r_1/r_2 = 0.1$.

5 Conclusions

We have analyzed the order of sensor processing in the sequential implementation of the MSJPDA filter. Our results indicate that processing the worse sensor first generally yields smaller RMS position error. Though counter intuitive at first, this trend was confirmed in one, two, and three-dimensional models (first through sixth-order systems) of the MSPDA filter using both the MRE numerical algorithm and its steady-state scalar approximation. A full explanation of this trend is difficult, because of the complexity of the data association process built in the MSPDA algorithm. Analyses over ranges of sensor parameters show that tracking system performance of the sequential MSJPDA filter, in terms of the RMS position error, favors using sensors of comparable qualities, and that processing the worse sensor first gives better results if the sensor qualities do not differ by a large amount.

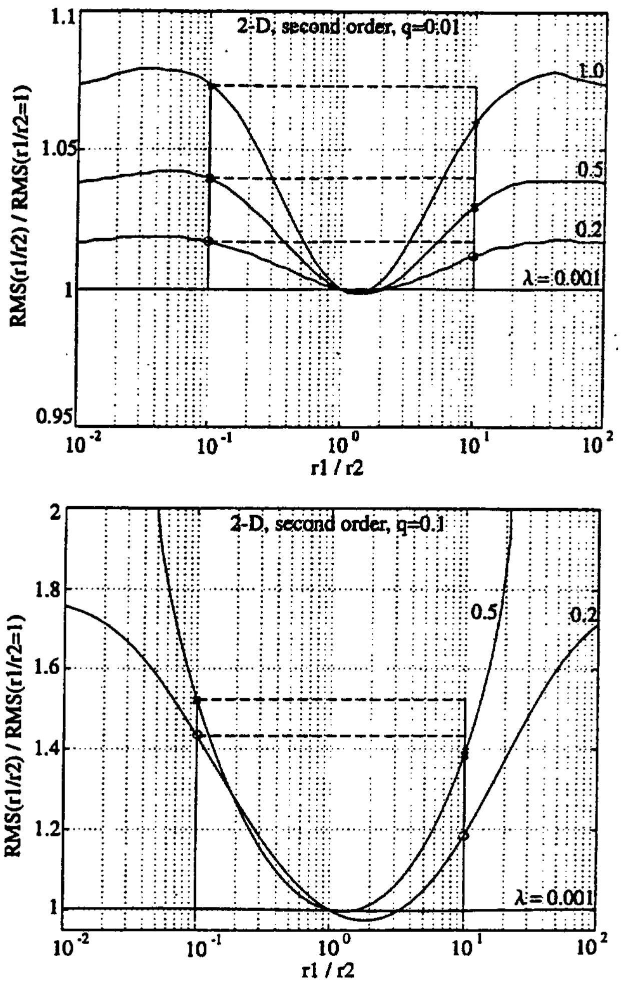


Figure 3: RMS position error from two-dimensional (state vector: $[x\ y]^T$) MRE iterations, for different q, r_1/r_2, and λ, with equivalent $r = 0.01$.

References

[1] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, Academic Press Inc., 1988.

[2] T. E. Fortmann, Y. Bar-Shalom, M. Scheffe, and S. Gelfand, "Detection Thresholds for Tracking in Clutter – A Connection Between Estimation and Signal Processing," *IEEE Trans. Aut. Control*, 30(3): 221–229, Mar. 1985.

[3] C. W. Frei, *A Comparison of Parallel and Sequential Implementations of a Multisensor Multitarget Tracking Algorithm*, M. S. thesis, Northwestern University, May 1995.

[4] C. W. Frei and L. Y. Pao, "Alternatives to Monte-Carlo Simulation Evaluations of Two Multisensor Fusion Algorithms," *Automatica*, 34(1): 103–110, Jan. 1998.

[5] S. Wolfram, *The Mathematic Book*, 3rd edition, Wolfram Media/Cambridge University Press, 1996.

Proceedings of the IEEE Conference on Decision and Control, Phoenix, AZ, December 1999, pp. 1803-1808.

Randomization and Super-Heuristics in Choosing Sensor Sets for Target Tracking Applications

MICHAEL KALANDROS and LUCY Y. PAO
Department of Electrical and Computer Engineering
University of Colorado
Boulder, CO 80309-0425
kalandro@colorado.edu and pao@colorado.edu

YU-CHI HO
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
ho@arcadia.harvard.edu

Abstract
Surveillance systems tracking multiple targets often do not have the sensing or computational resources to apply all sensors to all targets in the allocated time intervals. Hence, sensor management schemes have recently been proposed to reduce the tracking demands on these systems while minimizing the loss of tracking performance by selecting only enough sensing resources to maintain a desired covariance level for each target. The sensor manager algorithm itself, however, incurs a computational burden and needs to be implemented efficiently. This paper explores the use of randomization and super-heuristics to develop computationally efficient methods for implementing sensor manager algorithms.

1 Introduction

The application of multisensor fusion to surveillance systems has provided superior tracking performance at the cost of increased sensing and computational demands. Ideally, all available sensors can be applied to all targets to achieve the most accurate state estimate of each one. However, most sensors can track a finite number of targets in a single sampling period. Because of this, not all targets can be tracked with all sensors and improving tracking accuracy for one target may result in the degradation of track accuracy for a different target. Additionally, each measurement imposes a computational cost on a tracking system with a finite amount of processing capacity. What is needed is a sensor management technique that can balance tracking performance with available resources [4, 5, 6].

In [2], a system is proposed that separates sensor management into a control problem and a sensor scheduling problem. The scheduler prioritizes sensing actions and executes them as time allows. Low priority actions may be delayed until future scans or may be dropped altogether. The covariance controller maintains the covariance level of each target estimate to within a desired limit while reducing system resource demands. However, the sensor management algorithm itself imposes a computational burden on the system and hence must be implemented in a computationally efficient manner.

A search over all possible sensor combinations grows exponentially with the number of sensors. This clearly is unacceptable, and more efficient methods are needed. Super-heuristic methods [3] can be effectively used in this application to achieve near-optimal sensor selection performance while significantly reducing the computational burden imposed by the sensor manager.

This paper is organized as follows. A review of a covariance control-based sensor management method of [2] is given in Section 2. Section 3 discusses super-heuristic concepts, which are applied to sensor manager algorithms in Section 4. Simulation results of the various techniques are evaluated in Section 5, and Section 6 compares the computational demand of target tracking with and without sensor management. Conclusions and future research directions are discussed in Section 7.

2 Covariance Control

Figure 1 shows the block diagram of the tracking system. Control of the covariance of the system is implemented via a sensor selection algorithm. The sensor selection is determined based on the difference between the inverse of the desired covariance and that of the prediction covariance. Note that the only input to the controller is this difference.

Track estimation is performed by the Kalman filter. In actual target tracking applications a number of other

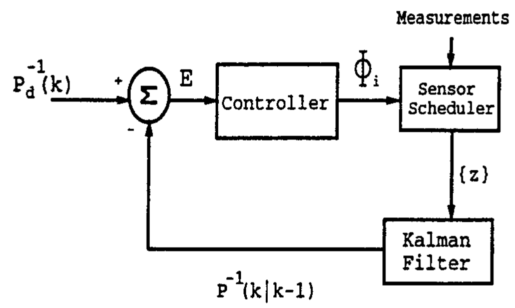


Figure 1: Block Diagram of a Tracking System with Covariance Controller (Sensor Selection Algorithm).

tasks are also performed, including data association (when clutter measurements or closely-spaced targets are present), track initiation and deletion, and registration [1]. These tasks compete for sensor and processor time as well, and would also fall under the control of the sensor manager. In the initial development of sensor managers in [2], only the state estimation task is considered. With this assumption, the controller's job is to regulate the sensing resources used by the Kalman filter to reduce the demands on the tracking system due to state estimation.

The Kalman filter combines multiple inputs from stochastic or linearized systems to form an estimate in a state space representation. Assume that the tracking system has N_s sensors. There are 2^{N_s} possible combinations or subsets of those sensors that can be selected by the multisensor manager. The ith possible subset is defined as Φ_i, and N_{s_i} is the number of sensors in that combination. The input from each selected sensor is used to update the state estimate of the target.

The Kalman filter is based on the following assumptions about the target and measurement systems [1]:

$$\begin{aligned} x(k) &= Fx(k-1) + Gu(k-1) + w(k-1) && (1) \\ z_j(k) &= H_j x(k) + v_j(k), \quad j = 1,\dots,N_{s_i} && (2) \end{aligned}$$

where $x(k)$ is the current state of the target; F, G, H_j are known system matrices; $u(k)$ is a control signal affecting the target dynamics; and $z_j(k)$ is a measurement of the target from the jth sensor in Φ_i. $w(k)$ represents process noise or higher-order motion not modeled by F, and $v_j(k)$ represents measurement noise in sensor j. Both $w(k)$ and $v_j(k)$ are assumed to have zero-mean, white, Gaussian probability distributions.

Since $w(k)$ and $v_j(k)$ are zero-mean noise processes, the target states and measurements in the next time interval can be predicted by

$$\begin{aligned} \hat{x}(k \mid k-1) &= F\hat{x}(k-1 \mid k-1) + Gu(k-1) && (3) \\ \hat{z}_j(k) &= H_j \hat{x}(k \mid k-1) && (4) \end{aligned}$$

The input $u(k)$ is considered known and will be omitted in future equations because it can be easily reinserted. The quantity $\nu_j(k) = \hat{z}_j(k) - z_j(k)$ is known as the innovation. The covariances of the state and the innovation predictions are

$$\begin{aligned} P(k \mid k-1) &= FP(k-1 \mid k-1)F' + Q(k-1) && (5) \\ S_1(k) &= H_1 P(k \mid k-1) H_1' + R_1(k) && (6) \\ S_j(k) &= H_j P_{j-1}(k \mid k) H_j' + R_j(k), \\ & \qquad j = 2,\dots,N_{s_i} && (7) \end{aligned}$$

respectively, where $Q(k)$ is the process noise covariance and $R_j(k)$ is the measurement noise covariance for the jth sensor. $P_j(k \mid k)$ is the updated covariance resulting from sensor j as defined in (10).

The sequential Kalman filter runs a separate filter for each sensor in the combination, propagating its estimate to the next filter [7]:

$$\begin{aligned} \hat{x}_1(k \mid k) &= \hat{x}(k \mid k-1) + K_1(k)\big(z_1(k) - H_1\hat{x}(k \mid k-1)\big) \\ \hat{x}_j(k \mid k) &= \hat{x}_{j-1}(k \mid k) + K_j(k)\big(z_j(k) - H_j\hat{x}_{j-1}(k \mid k)\big), \\ & \qquad j = 2,\dots,N_{s_i} \\ \hat{x}(k \mid k) &= \hat{x}_{N_{s_i}}(k \mid k) && (8) \end{aligned}$$

where

$$\begin{aligned} K_1(k) &= P(k \mid k-1) H_1' S_1^{-1}(k) \\ K_j(k) &= P_{j-1}(k \mid k) H_j' S_j^{-1}(k), \quad j = 2,\dots,N_{s_i} && (9) \end{aligned}$$

The state covariance is updated for each filter by

$$\begin{aligned} P_1(k \mid k) &= (I - K_1(k)H_1)P(k \mid k-1) \\ P_j(k \mid k) &= \big(I - K_j(k)H_j\big)P_{j-1}(k \mid k), \quad j = 2,\dots,N_{s_i} \\ P(k \mid k) &= P_{N_{s_i}}(k \mid k) && (10) \end{aligned}$$

Once the state and covariance estimates have been updated, they are fed back into the algorithm and the entire process is repeated for the new set of measurements at the next time interval. Alternatively, the covariance update can be calculated in a single step using the inverses of the covariance matrices [1] as follows:

$$P^{-1}(k \mid k) = P^{-1}(k \mid k-1) + J_i \qquad (11)$$

where

$$J_i = \sum_{j=1}^{N_{s_i}} H_j' R_j^{-1} H_j, \quad i = 1,\dots,2^{N_s} \qquad (12)$$

is the sensor information gain for the ith combination of sensors.

The sensor selection can be determined based on the difference between the inverses of the predicted covariance in (5) and the desired covariance $P_d(k)$. Replacing the updated covariance matrix in (11) with the desired covariance and solving for the necessary sensor information gain, we see that we want J_i to equal E, where

$$E = P_d^{-1}(k) - P^{-1}(k \mid k-1) \tag{13}$$

One sensor manager algorithm presented in [2] (and the one studied in this paper) is the Eigenvalue/Minimum Sensors Algorithm. It requires that the sensors used produce an updated covariance that is within the desired covariance at all times. This will result in the difference, $P_d - P_i$, where P_i is the updated covariance using sensor combination i, having all positive eigenvalues (as well as the difference $J_i - E$). While adding sensors will eventually achieve this goal, the computational demand on the Kalman filtering algorithm will increase linearly with the number of sensors. Since the goal is also to reduce the computational load on the tracking system, the sensor combination with the fewest number of sensors that produces all positive eigenvalues in the covariance error should be used at each scan.

The algorithm can be divided into on-line and off-line components. The off-line component precalculates J_i for each sensor combination. Use of (11) reduces the on-line computational demand by eliminating the calculation of matrix inverses during the Kalman gain calculation in (9) for each sensor. Instead, only the inverse of the predicted covariance must be computed each scan. The on-line component calculates $J_i - E$ for each i and selects those that are positive definite. This ensures that the updated covariance matrix will be within the desired covariance limits. Of those combinations that meet this criteria, the one with the fewest sensors is selected. A global search examines each of the 2^{N_s} possible sensor combinations before selecting the best combination. Unfortunately, the overall computational demand of the sensor manager is $\mathcal{O}(2^{N_s}n^3)$, where n is the size of the state vector. This demand increases far more rapidly than simply using all the sensors to track the target, where the computational complexity of the Kalman filter is $\mathcal{O}(2n^3 + 7N_s n^3)$ (ignoring any signal processing requirements in generating the measurements z_j and assuming that the entire state vector is measured). Note that this complexity can increase dramatically when other tracking tasks such as data association or track initiation/deletion are also included.

Obviously, the computational burden of a global search is prohibitively high. A heuristic search can alleviate some of the computational burden at the cost of possibly choosing a non-optimal sensor combination. One such algorithm is the "greedy" search algorithm. This algorithm chooses the sensors one at a time by picking the "best" sensor at each iteration. In our case, the best sensor is the one that maximizes the smallest negative eigenvalue of the difference between J_i and E. When there are no negative eigenvalues in the difference, the sensor combination is complete. The computational demand of this algorithm is at most $\mathcal{O}(\frac{n^3}{2}(N_s^2 + N_s))$.

3 Randomization and Super-Heuristics

Another approach for reducing the computational complexity of sensor manager algorithms is to use super-heuristics, a tool for improving any given solution to a problem via random perturbation [3]. It begins with a base solution (usually found through traditional heuristic methods) and changes it a little bit to see if the solution improves. In cases where there is little structure and the cost of evaluating a solution is not too high, this tool can be useful. The goal of super heuristics is to arrive at a near-optimal solution with much less computational complexity than is required to find the optimal solution [3].

In a sensor manager with N_s sensors, the probability of randomly picking (with uniform distribution) a "good enough" sensor combination is $\frac{N_g}{2^{N_s}}$, where N_g is the number of sensor combinations leading to acceptable estimation accuracy. We can increase our chance of success by selecting N_c random combinations and then choosing the best performer of the group. Out of N_c trials, the probability of finding an acceptable combination is $1 - \left(1 - \frac{N_g}{2^{N_s}}\right)^{N_c}$. Obviously, as the number of trials increases, so does the probability of finding acceptable sensor combinations, but the computational complexity increases linearly with the number of trials as well ($\mathcal{O}(N_c n^3)$). Heuristic information can be used to improve the performance of the sensor manager algorithm by increasing the chance that those combinations that are more likely to be in the "good enough" set are indeed chosen, creating a non-uniform distribution.

4 Heuristic Development

In the simulations of the next section, the heuristic information is generated using two different methods: frequency-of-selection and the Greedy algorithm. Frequency-of-selection is determined by running Monte Carlo simulations off-line using the global search strategy and recording the number of times each combination is picked. The resulting histogram data is normalized to create a probability distribution function that is used on-line to randomly select sensor combinations for evaluation. This greatly increases the chance of finding a near-optimal combination in a few trials.

The Greedy algorithm can be used to generate an initial solution about which a set of random solutions are generated. Once a greedy solution is derived, a random sensor combination with the number of sensors less than those in the greedy solution is generated in what is known as a probabilistic assignment rule (PAR) [3]. The probability of choosing each sensor in the combination can be uniformly distributed, in which case the only information derived from the initial solution is the number of sensors. While this may not seem useful, knowledge about the rough number of sensors required greatly reduces the search space, increasing the probability of finding a near-optimal solution in relatively few trials. The computational complexity of this algorithm is $\mathcal{O}(\frac{n^3}{2}(N_s^2+N_s)+n^3N_c)$.

Further information can be extracted from the Greedy algorithm. The result of the Greedy algorithm is an ordered list of the sensors. If the order of the sensors implies a value (where the first sensor is the most valuable, followed by the second, etc.), then this information can be used to guide a random selection of additional sensor combinations. Begin by using the Greedy algorithm to produce an ordered set $\{s_q\}_{q=1}^{N_s}$ of all N_s sensors (s_q is the qth sensor selected by the algorithm) by choosing the one that produces the largest minimum eigenvalue at each iteration. The actual greedy solution will be the first N_{s_g} sensors, but all sensors are evaluated to produce a complete evaluation of the sensor set. Define the following PAR for each sensor:

$$P(s_q) \;=\; \begin{cases} 1/b & 1 \le q \le b \\ 0 & q > b \end{cases} \qquad (14)$$

Thus only the first b sensors in the ordered set can be selected. The initial chosen combination is the greedy solution. To create an alternate sensor combination, randomly select one of the first b sensors using a uniform probability distribution. Remove the selected sensor from the set. Then randomly select one of the first b of the remaining sensors. Repeat until the desired number of sensors is reached (one less than the chosen combination). If the resulting combination achieves a positive definite covariance error $P_d - P_i$, it becomes the chosen combination. N_c random combinations are created and evaluated in this manner.

5 Simulation Results

To evaluate the effectiveness of the super-heuristic algorithms, a series of Monte Carlo simulations were performed on sensor managers using the Eigenvalue/Minimum Sensors Algorithm with various search methods. In each simulation, a multi-sensor system is tracking a target modeled by two states, both of which are measured by each of seven sensors (producing $2^7 = 128$ possible sensor combinations). The determinants of all the noise covariances R_j are equal, making all seven sensors of the same overall quality, though some sensors are more accurate in particular directions than others. To eliminate biases due to specific target dynamics, both the predicted covariance and the desired covariance were generated randomly at each iteration. The average number of sensors used as well as the number of computations performed in executing the search algorithms were recorded. The simulations compared the following algorithms:

- Optimal or Global Search: the "brute force" method that evaluates all combinations and guarantees finding the optimal sensor combination;

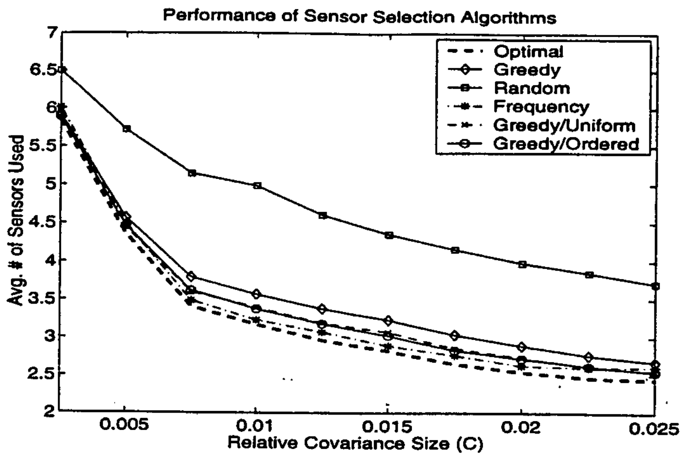


Figure 2: Average Number of Sensors Used

- Greedy Algorithm: builds a sensor combination by choosing the best sensor one sensor at a time;
- Random Subset Algorithm: the global search algorithm is applied to a random subset of $N_c = 8$ of the 128 possible sensor combinations;
- Frequency-of-Selection Super-heuristic Algorithm: same as the Random Subset algorithm, except that it randomly chooses a subset of $N_c = 8$ sensor combinations based on the a-priori probability density function (heuristic information) obtained from off-line simulations rather than a uniform density function;
- Greedy/Uniform Super-heuristic Algorithm: uses the Greedy algorithm to determine the maximum number of sensors needed, then randomly selects $N_c = 8$ sensor combinations (with fewer sensors) based on a uniform probability density;
- Greedy/Ordered Super-heuristic Algorithm: uses the Greedy algorithm and the PAR defined in (14) to select $N_c = 2$ sensor combinations (with fewer sensors) in addition to the greedy solution. In these simulations, $b = 4$ was determined experimentally to produce the best overall results.

Figure 2 shows the average number of sensors selected by each search algorithm as the relative size of the desired covariance compared to the predicted covariance is increased (described by the term C). A small relative size will, on average, require more sensors than a large relative size. In all cases the average size of the desired covariance is smaller than that of the predicted covariance, or else no sensors would be chosen. Figure 3 records the average number of floating point operations (flops) required by each search algorithm as the relative size of the desired covariance increases.

The global search algorithm requests the fewest sensing resources; however, it has by far the highest computational demand. On the other hand, the Random Subset

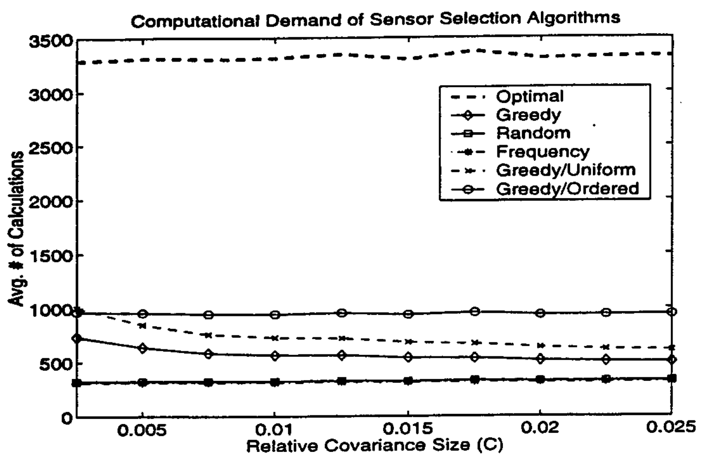


Figure 3: Number of Floating Point Operations

algorithm has the lowest computational demand, but is by far the poorest performer in terms of the number of sensors it selects. The Frequency algorithm performs almost as well as the global search in terms of number of sensors selected and has the same on-line computational demand as the Random algorithm.

The Greedy algorithm decisively outperforms the random subset algorithm in terms of sensors used for only a slight increase in computational demand. Both super-heuristic algorithms based on the Greedy algorithm improve performance by again increasing computational demand. The performance of the Greedy/Uniform algorithm matches that of the more computationally demanding Greedy/Ordered algorithm.

The strongest overall performer is the Frequency-of-Selection Super-heuristic algorithm. While it can produce near-optimal results for the least computational demand, the heuristic is developed through Monte-Carlo simulations, which can be time-consuming and computationally demanding, even off-line. Furthermore, it is not clear how the precalculated heuristic changes with specific target models, or how robust the performance

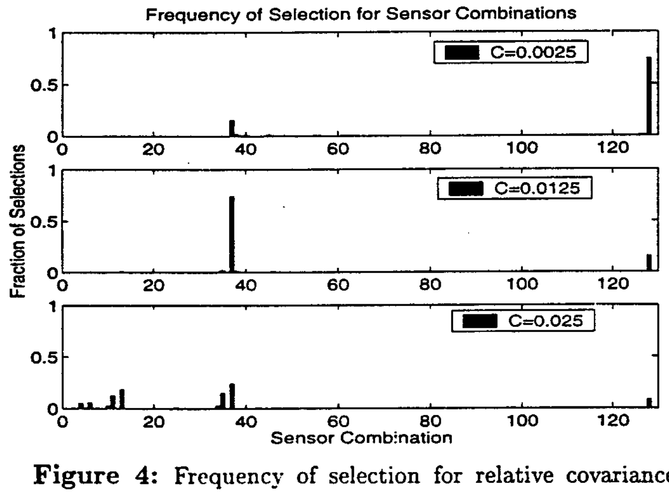


Figure 4: Frequency of selection for relative covariance sizes of $C = 0.0025$, 0.0125, and 0.025.

is to modeling errors. Figure 4 shows the frequency of selection of the 128 sensor combinations for the above simulations. Notice that while the distribution changes as the relative size of the desired covariance is changed, the set of most popular sensor combinations is not dramatically affected. More study is needed to determine the performance and robustness of the Frequency algorithm in actual tracking scenarios.

The greedy algorithms, while being more computationally demanding, are more flexible, since they can respond to new dynamic models or noise levels on-line. In these simulations they are outperformed by the Frequency algorithm, but increasing the number of random selections should improve the performance to near-optimal. This comes at the cost of increased computational demand, of course. The Greedy/Ordered algorithm generates extra information that lets it mimic the performance of the Greedy/Uniform algorithm with fewer trials, however the cost of generating that information negates this advantage.

The biggest drawback of the greedy algorithms is their computational demand. The best solution to this is to replace the calculation of the eigenvalues of the inverse covariance difference at each step with a lower complexity metric. However, while several of these have been tried, including the trace, the comparison of the direction of the largest eigenvalues, etc., none have provided the performance of the original Greedy algorithm.

6 Overall Computational Demand

Because tracking systems have limited computational as well as sensing capabilities, the ideal sensor manager will reduce the overall computational demand of the tracking task as well as the sensing demands. In Figure 5, the total computational demand of a "dumb" target tracking system that uses all of its sensors on the target with no sensor management is compared to the demand of systems with sensor managers (due to sensor management *and* the filtering of measurements from the chosen sensors) using either the Greedy algorithm or the Frequency-of-Selection algorithm (with $N_c = N_s$). The three algorithms are tested on tracking systems with 4, 8, and 12 sensors and the performance is averaged over 1000 runs. The optimal algorithm was eliminated because it is never less computationally demanding than using all of the sensors. The Greedy/Uniform and Greedy/Ordered are not included because they are always more demanding than the Greedy algorithm.

In the system with only 4 sensors, the dumb algorithm requires fewer computations than both the managed systems, partly because the managed systems always used most of their sensors (the Greedy system never uses less than an average of 68% of the available sensors and the Frequency never uses less than 77%). The "break-even" point of the Frequency system, where the computational

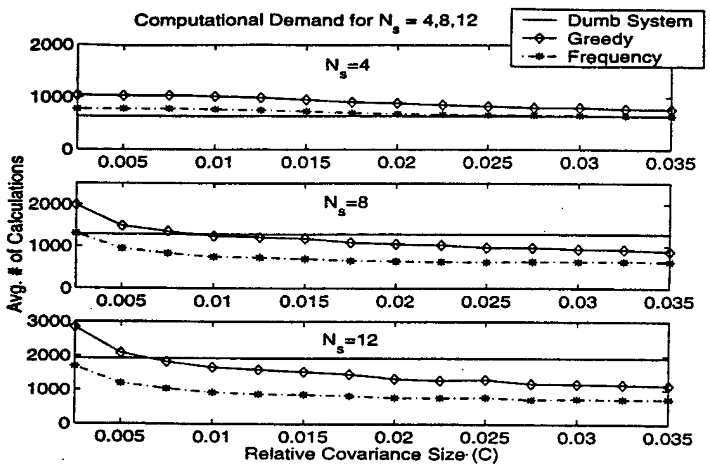


Figure 5: A comparison of the computational demand of multisensor tracking systems with and without sensor selection algorithms.

demand of the managed system is equal to that of the dumb system, is $C = 0.03$, which corresponds to the average use of about 80% of the sensors in the system. The number of sensors used at the break-even point defines an operating region where sensor management results in a reduction of total computational demand (in this case, it is an average of 3.2 sensors or less). The lower the number of sensors, the smaller this efficient operating region is.

When equipped with 8 sensors, the managed systems used fewer calculations than the unmanaged one for most of the relative covariance range tested ($C \geq 0.01$). For the Greedy system, the break-even point is around $C = 0.01$, which corresponds to the use of about 47% of the sensors. The Frequency system is as good as or better than the unmanaged system for the entire range of desired covariances tested, using 85% of the available sensors at the break-even point of $C = 0.0025$.

In the 12-sensor system, the sensor managers again outperform the dumb system for most of the range tested, but the number of sensors used at the break-even point for the Greedy system is 42%, lower than that of the 8-sensor system. As the number of available sensors increases, the number of sensors used at the break-even point for the Greedy system should drop dramatically, since its computational complexity increases faster than that of the dumb system. Even though the Frequency system outperforms the dumb system over the entire range of C tested, the average number of sensors used at the break-even point is probably less than 80%, also lower than that of the 8-sensor system. As more sensors are added, it is unclear how fast N_c must increase to maintain an acceptable level of performance, making predictions about the Frequency system's behavior as the number of sensors increases difficult. However, if additional tracking tasks (data association, etc.) are included in the calculations, the overall computational demand of unmanaged dumb systems is expected to become overwhelmingly larger than that of systems with these sensor managers as the number of sensors increases. Furthermore, while the reductions in sensor demand of the Greedy/Uniform and Greedy/Ordered algorithms do not outweigh their extra computational demand in these simulations, the increased cost of sensor use should eventually offset that burden as well.

7 Conclusion

Standard sensor manager algorithms are too computationally demanding to be implemented in many systems. The use of super-heuristic search techniques can greatly reduce the computational complexity of existing sensor managers. Several such approaches were explored and evaluated in this paper. The Frequency algorithm should be used when computational resources are scarce and the tracking models are known beforehand. When computational resources are not as limited, or if the tracking models vary greatly or are not known beforehand, the greedy super-heuristic algorithms can produce near-optimal results for a little extra computational demand. The use of these techniques will allow sensor managers to be implemented on a variety of platforms. Evaluating the robustness of performance when using a-priori heuristics, developing a less computationally complex heuristic for the greedy algorithms, and accounting for additional tasks such as data association are areas of future work.

Acknowledgments

The work of the first two authors is supported in part by an Office of Naval Research Young Investigator Award (Grant N00014-97-1-0642), the Data Fusion Corporation, and the Colorado Advanced Software Institute. That of the third author is supported in part by NSF grant EEC 95-27422, Army contract DAAH04-94-0148, AFOSR contract 49620-98-1-0387, ONR contract N00014-98-1-0720, and EPRI -W-08333-03.

References

[1] Y. Bar-Shalom and X. Li. *Estimation and Tracking: Principles, Techniques, and Software*, Artech House, Boston, 1993.

[2] M. Kalandros and L. Y. Pao. "Controlling Target Estimate Covariance in Centralized Multisensor Systems," *Proc. American Control Conf.*, Philadelphia, PA, pp. 2749–2753, June 1998.

[3] T. W. E. Lau and Y. C. Ho. "Super-Heuristics and Their Applications to Combinatorial Problems," *Asian Journal on Control*, 1(1), June 1999.

[4] S. Musick and R. Malhotra. "Chasing the Elusive Sensor Manager," *Proc. IEEE NAECON*, vol. 1, pp. 606–613, May 1994.

[5] J. Nash. "Optimal Allocation of Tracking Resources," *Proc. IEEE Conf. on Decision and Control*, vol. 1, pp. 1177–1180, December 1977.

[6] W. Schmaedeke. "Information-based Sensor Management," *SPIE Proceedings*, vol. 1955, pp. 156–164, April 1993.

[7] D. Willner, C. B. Chang, and K. P. Dunn, "Kalman Filter Algorithms for a Multi-Sensor System," *Proc. IEEE Conf. on Decision and Control*, pp. 570–574, 1976.

Proc. 2000 American Control Conf., Chicago, IL, June 2000.

Determining Track Loss Without Truth Information for Distributed Target Tracking Applications [1]

Lucy Y. Pao and Weerawat Khawsuk
Electrical and Computer Engineering Department
University of Colorado, Boulder, CO 80309-0425
pao@colorado.edu and weerawat.khawsuk@colorado.edu

Abstract

We explore criteria for filter consistency and develop methods of determining target track lifetimes without truth information. Two statistical tests, a sliding window likelihood function and a quadratic ratio of the Wishart matrix, are applied to a distributed fusion architecture. Simulation studies are used to compare the performance of these tests against a measure of track lifetime based on using the true states.

1. Introduction

The ability to track multiple targets in cluttered environments is needed in many applications such as military surveillance, air traffic control, and mobile robots. While a number of tracking algorithms assuming a centralized processing architecture have been developed, distributed processing architectures are more practical due to considerations such as reliability, survivability, communication bandwidth, and computational resources [4, 11, 12]. However, the merging of state estimates is more difficult in distributed tracking due to the loss of information inherent in forming the track estimates at the local processors. One of the major issues is accounting for the correlation between different local processor estimates for a common target [1, 3].

The distributed fusion architecture of Fig. 1 consists of several local processors and one global processor. Each local processor independently tracks targets in its surveillance region with its own sensors, using centralized algorithms for tracking targets in clutter such as Nearest Neighbor [2], Joint Probabilistic Data Association (JPDA) [2], or Mixture Reduction [9, 13]. The target state estimates from each local processor are passed to a global processor and possibly other local processors. At the global processor, a distributed fusion algorithm combines the local tracks to form global tracks of targets in the entire surveillance region.

Because the tracking ability of the global processor largely depends on the quality of the target estimates it receives from the local processors, distributed tracking systems need a mechanism to determine whether local tracks are lost so that these lost local target estimates do not degrade the ability of the global processor

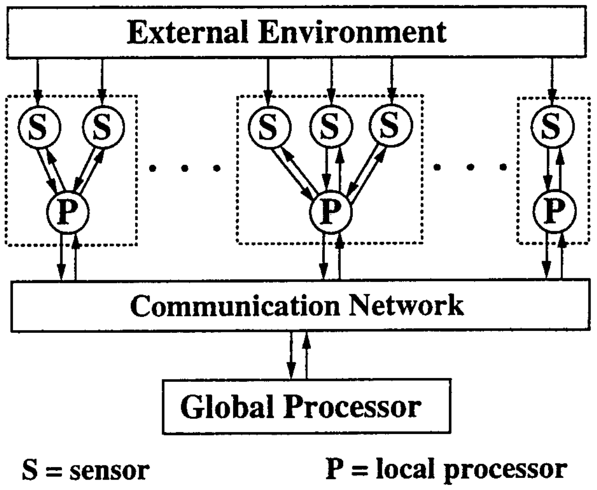


Fig. 1: Distributed Sensor Fusion Architecture.

to maintain tracks of targets. We investigate several methods of determining track loss based only on sensor information that is available to each local processor, and we apply these methods to the popular JPDA algorithm. By eliminating local tracks determined to be lost, the performance of the global processor in a distributed tracking system can be improved.

This paper is organized as follows. The Kalman filter and the JPDA algorithm are reviewed in Section 2, and criteria for filtering consistency are presented in Section 3. Derivation and implementation of two statistical tests — *modified log-likelihood function* and *quadratic ratio Wishart* tests, are investigated in Sections 4 and 5, respectively. Finally, Monte Carlo simulation results are presented and concluding remarks are given in Sections 6 and 7.

2. Kalman Filter and JPDA

Let $x(k)$ and $z(k)$ denote the vectors of the target state and the target originated measurement at the kth time interval. Suppose the target and measurement dynamics are determined by the known time-invariant matrices F, G, and H and random noise vectors $w(k)$ and $v(k)$ as

$$x(k) = Fx(k-1) + Gw(k) \quad (1)$$
$$z(k) = Hx(k) + v(k) \quad (2)$$

where $w(k)$ and $v(k)$ are independent Gaussian random variables with $\mathcal{N}[0, Q(k)]$ and $\mathcal{N}[0, R(k)]$ distributions, respectively.

[1]This work was supported in part under an Office of Naval Research Young Investigator Award (Grant N00014-97-1-0642) and a National Science Foundation Early Faculty CAREER Development Award (Grant CMS-9625086).

The predicted target state and measurement are

$$\hat{x}(k \mid k-1) = F\hat{x}(k-1 \mid k-1) \quad (3)$$
$$\hat{z}(k \mid k-1) = H\hat{x}(k \mid k-1) \quad (4)$$

and the predicted target state and predicted measurement error covariances can be found by

$$P(k \mid k-1) = FP(k-1 \mid k-1)F' + GQ(k)G' \quad (5)$$
$$S(k \mid k-1) = HP(k \mid k-1)H' + R(k). \quad (6)$$

The state estimate and error covariance updates are

$$\nu(k) = z(k) - H\hat{x}(k \mid k-1) \quad (7)$$
$$\hat{x}(k \mid k) = \hat{x}(k \mid k-1) + K(k)\nu(k) \quad (8)$$
$$P(k \mid k) = [I - K(k)H]\, P(k \mid k-1) \quad (9)$$
$$K(k) = P(k \mid k-1)H' [S(k \mid k-1)]^{-1} \quad (10)$$

where $\nu(k)$ and $K(k)$ are the innovation and Kalman gain, respectively. Once the target state and covariance estimates have been updated, they are fed back into the algorithm and the entire process is repeated for the new set of measurements at the next time step.

When tracking in cluttered environments and the origin of measurements is not known, a data association algorithm such as the Joint Probabilistic Data Association (JPDA) method is needed. In JPDA, the combined measurement

$$z(k) = \sum_{j=0}^{m(k)} \beta_j(k) z_j(k) \quad (11)$$

is used in (7) where $z_j(k)$ is the jth measurement at time k, $\beta_j(k)$ is the probability that $z_j(k)$ is the target originated measurement, and $m(k)$ is the number of gated measurements. $j = 0$ denotes the possibility that there are no target originated measurements, with $z_0(k) = \hat{z}(k \mid k-1)$. The updated state covariance is

$$P(k \mid k) = \beta_0(k)P(k \mid k-1) + \{1 - \beta_0(k)\}\, P^C(k \mid k-1) + \tilde{P}(k)$$
$$P^C(k \mid k) = \{I - K(k)H\}\, P(k \mid k-1) \quad (12)$$
$$\tilde{P}(k) = K(k) \left\{ \sum_{j=1}^{m(k)} \beta_j(k)\nu_j(k)\nu_j'(k) - \nu(k)\nu'(k) \right\} K'(k).$$

In practice, the JPDA filter for a distributed multi-target, multi-sensor system can be implemented either sequentially or in parallel [5, 10]. In the presence of clutter measurements, the sequential implementation performs better on the average than the parallel implementation in a centralized processing architecture [5, 10]. In distributed architectures with distributed data association and filtering, the communication requirements between the local and global processors are reduced if the local multi-sensor fusion algorithms are implemented in parallel [12].

3. Filtering Consistency

The Kalman filter is a consistent estimator in the absence of clutter. The consistency of a time-invariant estimator is defined as asymptotic convergence of the estimate to the true value:

$$\lim_{k\to\infty} E\{\hat{x}(k|k) - x(k)\} = 0. \quad (13)$$

In cluttered environments, however, the JPDA filter is not necessarily consistent, and the system loses tracks. Loss of a track occurs when the mean-square error (MSE) of the state estimate consistently exceeds a certain threshold. Because the true MSE is not available to the processor without knowledge of the actual target states, we propose to develop methods for determining track loss in sensor fusion algorithms by exploring filtering consistency.

Common criteria for filtering consistency are [2]:

1. The state errors should be zero mean (unbiased) and error covariance should be consistent with the calculated covariance matrix of a filter.
2. The innovations should have the same property as the state error in 1.
3. The innovations should be uncorrelated in time.

The first criterion is the most important, but it can be tested only in simulations, where the true state $x(k)$ is available for comparison. In practice, only the last two criteria, which are consequences of the first, can be tested since it is possible to monitor the measurements $z(k)$ arriving at each time interval.

4. Likelihood Function Test

Let $Z^k = \{z(1), \ldots, z(k)\}$ denote the target originated measurements up to time k. The joint probability density function (PDF) of Z^k is the *likelihood function* of the system model [2]

$$Pr\{Z^k\} = \frac{\exp\left\{-\frac{1}{2}\sum_{i=1}^{k} \nu'(i)S^{-1}(i)\nu(i)\right\}}{\prod_{i=1}^{k} |2\pi S(i)|^{1/2}}. \quad (14)$$

The exponent of (14), known as the *modified log-likelihood function*, can be computed recursively as

$$\rho(k) \equiv \sum_{i=1}^{k} \nu'(i)S^{-1}(i)\nu(i) = \rho(k-1) + \nu'(k)S^{-1}(k)\nu(k). \quad (15)$$

The individual term $\varepsilon_\nu(k) \equiv \nu'(k)S^{-1}(k)\nu(k)$ is known as the *normalized innovation squared* or *Mahalanobis distance* and is χ^2 distributed with m degrees of freedom (χ^2_m), where m is the dimension of the measurements. $\rho(k)$ is χ^2 distributed with km degrees of freedom (χ^2_{km}). The modified log-likelihood function (15) can be used to test the validity of received measurements [2], where the distribution of a sampling of the function is used to identify unlikely deviations from its expected value.

Generally, the cumulative $\rho(k)$ can not be used for tracks with long time histories because it becomes dominated by old measurements and responds very slowly to recent ones. In practice [2], the memory of $\rho(k)$ must be limited to the relatively recent past. The *sliding window* approach restricts $\rho(k)$ in (15) to be from $k-N+1$ to k as

$$\rho_N(k) \;=\; \sum_{i=k-N+1}^{k} \varepsilon_\nu(i). \tag{16}$$

The *fading-memory* method applies an $\alpha < 1$ factor to $\rho(k)$ at each step, which results in

$$\rho_\alpha(k) \;=\; \alpha\rho_\alpha(k-1) + \varepsilon_\nu(i) \;=\; \sum_{i=1}^{k} \alpha^{k-i}\varepsilon_\nu(i). \tag{17}$$

The $\rho_N(k)$ of the sliding window method has a χ^2_{Nm} distribution, while in the steady state, the fading-memory approach $\rho_\alpha(k)$ is approximately a χ^2 random variable with $m(1+\alpha)/(1-\alpha)$ degrees of freedom with mean $m/(1-\alpha)$ and variance $2m/(1-\alpha^2)$.

In cluttered environments, there are false measurements in addition to the target originated measurements. To reduce computational complexity, gating is used to eliminate measurements unlikely to have originated from the target. The gated measurements are used to compute innovations, and the distribution of these innovations is used to determine if the track is lost. For instance, for a given target, if $\rho_N(k) > r$ consistently where r is such that

$$Pr\left\{\chi^2_{Nm} > r\right\} = 1 - P_G \tag{18}$$

where P_G is the gate probability, then that target track is said to be lost.

Three heuristic models are investigated to represent the innovation distance $\varepsilon_\nu(k)$ in computing $\rho_N(k)$ or $\rho_\alpha(k)$. With the implementation of the JPDA filter, the *combined innovation* (posterior) model [2]

$$\nu(k) \;=\; \sum_{j=1}^{m(k)} \beta_j(k)\nu_j(k) \tag{19}$$

can be used in $\varepsilon^1_\nu(k) = \nu'(k)S^{-1}(k)\nu(k)$. Since the target originated measurement is unknown, the combined innovation is used. The covariance of the combined innovation is "smaller" than $S(k)$, which is the covariance of the "correct" innovation. In our evaluation of the combined innovation model, we replaced $S(k)$ with $q_2S(k)$ where q_2 is the *information reduction factor* for the Probabilistic Data Association (PDA) algorithm [2]. While no such factor for the JPDA has been developed, the q_2 factor for the PDA has been observed to work well in predicting the performance of the multisensor JPDA [5]. However, our evaluation of this combined innovation model shows that it performs poorly in determining track lifetimes. Despite the use of the q_2 factor, when clutter dominates the actual measurement (which generally indicates that the track is becoming lost), the $\beta^2_j(k)$ factors (which are < 1) in $\varepsilon^1_\nu(k)$ become insignificant, contributing lower statistical values $\varepsilon^1_\nu(k)$ well below the threshold.

A model yielding higher statistical values in such cases is a *probabilistic distance* (posterior) model:

$$\varepsilon^2_\nu(k) \;=\; \sum_{j=1}^{m(k)} \beta_j(k)\nu'_j(k)S^{-1}(k)\nu_j(k). \tag{20}$$

Our results show that this model gives higher statistics and leads to track lifetime estimates closer to the actual values than the combined innovation model. However, this model still requires the JPDA process to attain the $\beta_j(k)$ probabilities before the track can be determined to be lost.

To reduce computational complexity, averaging with equal probability leads to an *average distance* (prior) model:

$$\varepsilon^3_\nu(k) \;=\; \frac{1}{m(k)}\sum_{j=1}^{m(k)} \nu'_j(k)S^{-1}(k)\nu_j(k). \tag{21}$$

If the track is determined to be lost based upon this model, then the track can be eliminated without the need to compute the JPDA probabilities.

5. Wishart Ratio Test

Recall that for filtering consistency, the innovation sequence $\nu(k) = z(k) - \hat{z}(k \mid k-1)$ should be uncorrelated in time and $\mathcal{N}\left[0, S(k \mid k-1)\right]$ distributed. Without loss of generality, it is convenient to consider the normalized innovation sequence $\bar{\nu}(k) = S^{-1/2}(k \mid k-1)\nu(k)$ which is $\mathcal{N}\left[0, I\right]$ distributed, because its covariance is the identity matrix and is independent of time.

If the past N samples of the normalized innovation sequence $\{\bar{\nu}(k-N+1), \ldots, \bar{\nu}(k)\}$ are being monitored, the maximum likelihood estimators of the identity covariance matrix are $\hat{I}(k) = \frac{1}{N}\hat{S}(k)$, where

$$\hat{S}(k) \;=\; \sum_{j=k-N+1}^{k} \left[\bar{\nu}(j) - \bar{\nu}(k)\right]\left[\bar{\nu}(j) - \bar{\nu}(k)\right]' \tag{22}$$

$$\bar{\nu}(k) \;=\; \frac{1}{N}\sum_{j=k-N+1}^{k} \bar{\nu}(j). \tag{23}$$

The $\hat{I}(k)$ and $\bar{\nu}(k)$ are the *sample covariance matrix* and *sample mean* of the normalized innovation sequence $\bar{\nu}(k)$ at time k.

The joint PDF of the $\frac{1}{2}m(m+1)$ elements in the lower triangular portion of the random matrix $\hat{S}(k)$ is a *centered Wishart* distribution $\mathcal{W}_m\left(\eta, \hat{S}(k); I\right)$ with

$\eta = N-1$ degrees of freedom, where [8]

$$\mathcal{W}_m\left(\eta, \hat{S}(k); I\right) = \frac{\left|\hat{S}(k)\right|^{\frac{\eta-m-1}{2}} \exp\left\{-\frac{1}{2}tr[\hat{S}(k)]\right\}}{2^{\frac{\eta m}{2}} \pi^{\frac{m(m-1)}{4}} \prod_{j=1}^{m} \Gamma\left(\frac{\eta-j+1}{2}\right)} \quad (24)$$

in the region of the positive definite matrices $\hat{S}(k)$, while outside the region $\mathcal{W}_m\left(\eta, \hat{S}(k); I\right) = 0$. The parameters are the dimension m of $\tilde{\nu}(k)$ (the number of measurement states), the number of degrees of freedom $\eta = N-1$, and the random matrix $\hat{S}(k)$. It is necessary that $\eta \geq m$ such that $\hat{S}(k)$ is not singular.

In hypotheses testing, the Wishart statistics $\hat{S}(k)$ in (22) are complicated and not well developed because it is difficult to establish the confidence domain for a random matrix. In practice, several scalar measures of $\hat{S}(k)$ can be used for testing random matrices [7], such as the trace, the generalized variance, the maximal eigenvalue, the sum of all elements of the matrix, *etc.* Each measure characterizes geometrical parameters of the correlation ellipsoid differently.

Note that for any fixed $m \times 1$ vector L, provided that $L'VL \neq 0$ and the estimator S of the matrix V is Wishart distributed $\mathcal{W}_m(\eta, S; V)$, then the quadratic ratio $L'SL/(L'VL)$ is χ^2_η distributed [6, 7, 8]. If $L'VL = 0$, then $L'SL = 0$ with probability 1. Therefore, testing the random estimator $\hat{S}(k)$ of an identity matrix I is equivalent to testing the quadratic Wishart statistic

$$\ell(k) = \frac{L'\hat{S}(k)L}{L'L} \sim \chi^2_\eta. \quad (25)$$

Without loss of generality, one can choose any fixed vector L such that $L'L = 1$. Therefore, the above statistical test reduces to $\ell(k) = L'\hat{S}(k)L$, and the corresponding threshold test follows along the lines of the likelihood function tests of Section 4.

The choice of an L vector corresponds to different scalar measure characteristics [7]. In the case of choosing $L' = \frac{1}{\sqrt{m}}[1, 1, \cdots, 1]$, the test statistic $\ell(k)$ is the sum of all elements in the random matrix $\hat{S}(k)$. To achieve the maximum value of $\ell(k)$, one can choose an L vector such that it maximizes $\ell(k)$ under the constraints of $L'L = 1$. This vector is the eigenvector of the matrix $\hat{S}(k)$ corresponding to the maximum eigenvalue of $\hat{S}(k)$.

In [6, 7], the quadratic ratio tests perform well in sensor failure detection and are sensitive to false measurements due to higher noise levels. Therefore, it may be a valid statistical test to detect deviations from the expected distribution of innovations when tracking targets in high clutter densities. For tracking in clutter, posterior and prior knowledge of gated measurements can be used to represent the innovation sequence $\nu(k) = z(k) - \hat{z}(k \mid k-1)$. A posterior knowledge yields a *combined innovation* model which utilizes combined innovations (19) to compute a sample covariance matrix (22). However, a prior knowledge of equal probability leads to an *average innovation* model

$$\nu(k) = \frac{1}{m(k)} \sum_{j=1}^{m(k)} \nu_j(k) \quad (26)$$

which has the advantage of determining track loss prior to data association and filtering.

6. Simulation Results

In this section, the statistical tests of Sections 4 and 5 are utilized to determine the track lifetimes of targets without knowledge of the true states. The simulation results represent the performance of a local processor in the distributed architecture of Fig. 1. The tracking scenario consists of two targets moving in 2 dimensions in straight lines corrupted by Gaussian noise with $\mathcal{N}[0, 0.0144I]$.

The distributed tracking system consists of 2 local processors with 2 sensors each, with the multisensor JPDA implemented in parallel at each processor. Sensor noise is also white Gaussian with $\mathcal{N}[0, 0.0144I]$. The clutter density λ is varied between 0.2 and 0.8, resulting in an average of 0.504 to 2.016 clutter points per gate. One hundred Monte Carlo runs are performed in each tracking simulation, and the track lifetime results are averaged for both tracks over both local processors.

Based on truth, the processor is said to lose track of a target if the target originated measurements from both sensors lie outside the gated region for 5 consecutive scans. The actual value of track lifetime is determined as the end of this run of 5 consecutive scans. Similarly, when the statistical tests of the innovations from both sensors exceed the corresponding thresholds for at least 5 consecutive scans, the track is considered lost and the track lifetime is determined.

The likelihood function $\rho(k)$ and quadratic Wishart $\ell(k)$ tests with various implementations, are applied to the tracking system. The likelihood function is implemented with both sliding window $\rho_N(k)$ and fading memory $\rho_\alpha(k)$ approaches. Figure 2 shows the average local track lifetimes versus clutter density for the actual track lifetimes and some of the sliding window $\rho_N(k)$ and quadratic ratio $\ell(k)$ estimates. Results from the fading memory $\rho_\alpha(k)$ approach are not shown as it yields track lifetimes that are far away from the actual values (about 100% error on the average). This is largely due to the fact that the approximate distribution mentioned in Section 4 is only accurate when the tracking filter has reached steady state; however, steady state is often not reached when tracking in cluttered environments.

All three heuristic models of combined innovation $\varepsilon^1_\nu(k)$, probabilistic distance $\varepsilon^2_\nu(k)$, and average distance $\varepsilon^3_\nu(k)$ are implemented with $\rho_N(k)$. Because of the β^2 factor effect, the $\rho_N(k)$ test with $\varepsilon^1_\nu(k)$ yields inaccurate

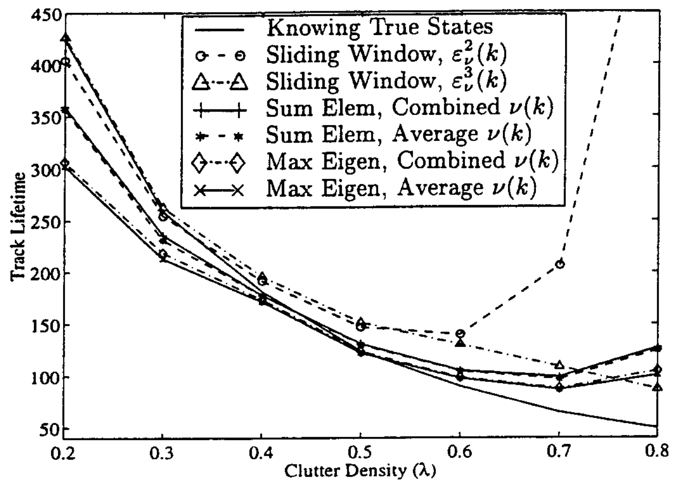


Fig. 2: Track lifetimes of two statistical tests: 1) Sliding window $\rho_N(k)$ with $\varepsilon_\nu^2(k)$ and $\varepsilon_\nu^3(k)$ models, and 2) Sum of all elements and maximum eigenvalue $\ell(k)$ with combined and average $\nu(k)$ models.

track lifetimes and are not shown (also approximately 100% error on average). For the quadratic Wishart $\ell(k)$ test, both methods of choosing an L vector are considered — sum of all elements and maximum eigenvalue of $\hat{S}(k)$, and each method is applied with average (prior) and combined (posterior) innovation models. Both prior and posterior models yield similar track lifetimes.

For tracking in low ($\lambda < 0.4$) clutter densities, the $\rho_N(k)$ tests yield closer track lifetimes to the actual values. The $\rho_N(k)$ performs better because $\varepsilon_\nu^2(k)$ and $\varepsilon_\nu^3(k)$ are mainly dominated by the distance due to the actual measurement rather than clutter measurements. However, the $\ell(k)$ tests yield more accurate track lifetimes when tracking in the clutter density range $0.4 \leq \lambda \leq 0.6$. In this range, more clutter measurements are gated and the distance due to the true target measurement becomes dominated by the clutter points. As a result, the $\rho_N(k)$ yields low statistical values below the threshold even when the system has actually lost track of the targets. For very high clutter densities ($\lambda > 0.6$), all the methods tend to overestimate the track lifetimes, with $\rho_N(k)$ with $\varepsilon_\nu^3(k)$ matching the trend of the actual track lifetimes the best.

7. Conclusions

The statistical tests of the log-likelihood function and quadratic ratio Wishart matrix have been investigated to check the validity of received measurements and evaluate the track lifetimes of targets for a distributed fusion tracking system. Simulation results indicate that the likelihood function gives more accurate track lifetimes with the sliding window rather than the fading memory approach. Further, the probabilistic and average distances are the best models to implement with the likelihood function when tracking in low clutter densities. However, the quadratic ratio test yields better performance for moderate to higher clutter densities, with the maximum eigenvalue approach giving track lifetimes closer to the actual values in this range.

Greater computational efficiency is achieved if the average distance and average innovation models are used in the likelihood function and quadratic ratio tests, respectively. If either test indicates that track loss has already occurred, both models have the advantage of discarding received measurements before passing them to the data association and filtering process.

Simulations show promising results that these two statistical tests can be used to determine track loss for distributed fusion architectures when truth information is not available. It can also be used in centralized fusion architectures by discontinuing the updates on tracks that have been determined to be lost to improve the computational efficiency of the system.

References

[1] Y. Bar-Shalom. "On the Track-to-Track Correlation Problem," *IEEE Trans. Aut. Ctrl.*, 26(2): 571–572, 1981.

[2] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*, Academic Press, Inc., San Diego, 1988.

[3] C. B. Chang and L. C. Youens. "Measurement Correlation for Multiple Sensor Tracking in a Dense Target Environment," *IEEE Trans. Aut. Ctrl.*, 27(6): 1250–1252, 1982.

[4] C. Y. Chong, K. C. Chang, and S. Mori. "Distributed Tracking in Distributed Sensor Networks," *Proc. American Ctrl. Conf.*, pp. 1863–1868, 1986.

[5] C. W. Frei and L. Y. Pao. "Alternatives to Monte-Carlo Simulation Evaluations of Two Multisensor Fusion Algorithms," *Automatica*, 34(1): 103–110, 1998.

[6] Ch. M. Gadzhiev. "Checking Multivariate Model Fit from the Generalized Wishart-Statistic Variance," *Measurement Techniques*, 36(12): 1316-1319, 1993.

[7] Ch. M. Gadzhiev. "Testing the Covariance Matrix of a Renovating Sequence under Operating Control of the Kalman Filter," *Automation and Remote Control*, 57(7): 1046–1052, 1996.

[8] N. L. Johnson and S. Kotz. *Distributions in Statistics: Continuous Multivariate Distributions*, John Wiley & Sons, Inc., New York, 1972.

[9] L. Y. Pao. "Multisensor Multitarget Mixture Reduction Algorithms for Tracking," *J. Guid., Ctrl., & Dyn.*, 17(6): 1205–1211, 1994.

[10] L. Y. Pao and C. W. Frei. "A Comparison of Parallel and Sequential Implementations of a Multisensor Multitarget Tracking Algorithm," *Proc. American Ctrl. Conf.*, pp. 1683–1687, 1995.

[11] L. Y. Pao. "Measurement Reconstruction Approach for Distributed Multisensor Fusion," *J. Guid., Ctrl., & Dyn.*, 19(4): 842–847, 1996.

[12] L. Y. Pao and M. K. Kalandros. "Algorithms for a Class of Distributed Architecture Tracking," *Proc. American Ctrl. Conf.*, pp. 1434–1438, 1997.

[13] D. J. Salmond. "Mixture Reduction Algorithms for Target Tracking in Clutter," *SPIE Signal and Data Processing of Small Targets,* Vol. 1305, pp. 434–445, 1990.

To appear in IEEE Trans. Automatic Control, 45(8), Aug. 2000.

The Optimal Order of Processing Sensor Information in Sequential Multisensor Fusion Algorithms *

Lucy Y. Pao and Lidija Trailović

Department of Electrical and Computer Engineering

University of Colorado at Boulder

Abstract

We examine the order of sensor processing in the sequential Multisensor Probabilistic Data Association (MSPDA) filter for target tracking applications. If two sensors of different qualities are used in the sequential MSPDA filter, the root mean square position error is generally smaller when the worse sensor is processed first. This finding regarding the order of sensor processing is supported by simulations of a target tracking system, and by analyses of first through sixth-order target process models using the Modified Riccati Equation.

1 Introduction

Tracking problems involve processing measurements from a target of interest, and producing, at each time step, an estimate of the target's current position and velocity. Uncertainties in the target motion and in the measured values, usually modeled as additive random noise, lead to corresponding

*This work was supported in part by an Office of Naval Research Young Investigator Award (Grant N00014-97-1-0642) and a National Science Foundation Early Faculty CAREER Award (Grant CMS-9625086). The authors thank Christian Frei of the Automatic Control Laboratory, Swiss Federal Institute of Technology, Zurich, Switzerland for his help in debugging the simulator.

uncertainties in the target state. Additional uncertainty regarding the origin of the received data, which may or may not include measurements from the targets or random clutter (false alarms), leads to the problem of data association [1, 2].

In this paper, we analyze the sequential implementation of the Multisensor Probabilistic Data Association (MSPDA) filtering algorithm [3]. It was shown in [3, 4] that sequential processing of information from sensors of equal quality is superior to parallel processing of the sensor information, in terms of computational efficiency and two performance metrics. In tracking applications, however, sensors are usually of unequal qualities, and tracking performance may be affected by the order of processing sensor information. Thus, we investigate here the optimal order of processing sensor information (in terms of minimizing the root-mean-square position error) in sequential implementations of the MSPDA algorithm when two sensors of different qualities are used. The results of our work are applicable to problems such as tracking aircraft with radar measurements or tracking of submarines with sonar measurements [1, 2].

This paper is organized as follows. In Section 2, we review the sequential implementation of the Multisensor Joint Probabilistic Data Association (MSJPDA) algorithm. Simulation results are then presented in Section 3, followed by more analytical results in Section 4 where we consider the Modified Riccati Equation and discuss results for first through sixth-order target process models. Finally, conclusions are presented in Section 5.

2 Sequential MSJPDA Filtering

The multisensor multitarget tracking problem is to track T targets using N_s sensors in a cluttered environment. Some of the measurements arise from targets, and some from clutter; some targets may not yield any measurements in a particular time interval or for a particular sensor. The probability of detection P_D^i is assumed to be constant across targets for a given sensor i.

The dynamics of the target state $\mathbf{x}^t(k)$ are assumed to be determined by known matrices $\mathbf{F}^t(k)$

and $\mathbf{G}^t(k)$, and random vectors $\mathbf{w}^t(k)$ as follows

$$\mathbf{x}^t(k+1) = \mathbf{F}^t(k)\,\mathbf{x}^t(k) + \mathbf{G}^t(k)\,\mathbf{w}^t(k) \tag{1}$$

where $t = 1, \ldots, T$. The noise vectors $\mathbf{w}^t(k)$ are independent Gaussian random variables with zero mean and known covariance matrices $\mathbf{Q}^t(k)$.

With N_s sensors, let $M_k^i, i = 1, 2, \ldots, N_s$, be the number of measurements from each sensor i at the kth time interval. Assuming a pre-correlation gating process is used to eliminate some of the measurements [1], let m_k^i denote the number of validated measurements from sensor i at time k. The volume of a gate at time k is chosen such that with probability P_G^i the target originated measurements, if there are any, fall into the gate of sensor i. The target originated measurements are determined by

$$\mathbf{z}_{i,l_i}^t(k) = \mathbf{H}_i(k)\,\mathbf{x}^t(k) + \mathbf{v}_i^t(k), \tag{2}$$

where $t = 1, \ldots, T$, $i = 1, \ldots, N_s$, and $1 \leq l_i \leq M_k^i$. Matrices $\mathbf{H}_i(k)$ are known, each $\mathbf{v}_i^t(k)$ is a zero mean Gaussian noise vector uncorrelated with all other noise vectors, and the covariance matrices $\mathbf{R}_i(k)$ of the noise vectors $\mathbf{v}_i^t(k)$ are known. For a given target t and sensor i, it is not known which measurement l_i originates from the target. That is the problem of data association whereby it is necessary to determine which measurements originate from which targets [1]. Measurements not originating from targets are false measurements (or clutter), and they are assumed to be uniformly distributed throughout the surveillance region with a density λ.

A sequential implementation of the MSJPDA algorithm processes the measurements from each sensor one sensor at a time [3, 4]. The measurements of the first sensor are used to compute the intermediate state estimate $\hat{\mathbf{x}}_1^t(k|k)$ and the corresponding covariance $\mathbf{P}_1^t(k|k)$ for each target. The measurements of the next sensor are then used to further improve this intermediate state estimate. In processing each sensor's measurements, the actual association being unknown, the conditional estimate is determined by taking a weighted average over all possible associations. For $1 \leq t \leq T$,

$1 \leq i \leq N_s$, and $0 \leq l_i \leq m_k^i$, let $\beta_{i,l_i}^t(k)$ denote the conditional probability that measurement l_i from sensor i is the true measurement from target t given all measurements received up to time k. $l_i = 0$ denotes the event that the target was not detected at time k. With $\hat{\mathbf{x}}_i^t(k|k)$ and $\mathbf{P}_i^t(k|k)$ as the state estimate and covariance, respectively, after processing the data of the ith sensor, the update equations are

$$\hat{\mathbf{x}}_i^t(k|k) = \hat{\mathbf{x}}_{i-1}^t(k|k) + \mathbf{K}_i^t(k) \sum_{l_i=0}^{m_k^i} \beta_{i,l_i}^t(k)[\mathbf{z}_{i,l_i}(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}_{i-1}^t(k|k)], \qquad i = 1, \ldots, N_s, \tag{3}$$

where $\hat{\mathbf{x}}_0^t(k|k) = \hat{\mathbf{x}}^t(k|k-1)$ and $\hat{\mathbf{x}}_{N_s}^t(k|k) = \hat{\mathbf{x}}^t(k|k)$. With $\mathbf{P}_0^t(k|k) = \mathbf{P}^t(k|k-1)$ and $\mathbf{P}_{N_s}^t(k|k) = \mathbf{P}^t(k|k)$, the update of the covariance matrices is

$$\begin{aligned}\mathbf{P}_i^t(k|k) = {} & \beta_{i,0}^t(k)\mathbf{P}_{i-1}^t(k|k) + \left[1 - \beta_{i,0}^t(k)\right] \left[\mathbf{I} - \mathbf{K}_i^t(k)\mathbf{H}_i(k)\right] \mathbf{P}_{i-1}^t(k|k) + \mathbf{K}_i^t(k) \\ & \times \left[\sum_{l_i=0}^{m_k^i} \beta_{i,l_i}^t(k)\mathbf{z}_{i,l_i}(k)\mathbf{z}_{i,l_i}(k)^T - \sum_{l_i=0}^{m_k^i} \beta_{i,l_i}^t(k)\mathbf{z}_{i,l_i}(k) \sum_{l_i=0}^{m_k^i} \beta_{i,l_i}^t(k)\mathbf{z}_{i,l_i}(k)^T\right] \mathbf{K}_i^t(k)^T, \quad i = 1, \ldots, N_s\end{aligned} \tag{4}$$

A superior performance (in terms of RMS position error, track lifetime, and computational efficiency metrics) of the sequential implementation of the MSJPDA over the parallel implementation was shown [3, 4] when multiple sensors of the same quality were used. If the sensors are not of equal qualities, however, a question that arises is what is the best order to process the sensor data in the sequential implementation.

3 Simulation Results

For comparing sequential implementations of the MSJPDA algorithm using different processing orders of sensors with different qualities, we initially ran Monte Carlo simulations for two sensors tracking two targets. We considered the dynamic target model (1) for $t = 1, 2$, with time-invariant matrices $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$, $\mathbf{Q}$, and $\mathbf{R}_i$. A typical state vector would include position and velocity variables.

Hence, typical **F** and **G** are

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \Delta^2/2 & 0 \\ 1 & 0 \\ 0 & \Delta^2/2 \\ 0 & 1 \end{bmatrix}, \tag{5}$$

for the state vectors $\mathbf{x}^t(k) = [x \; \dot{x} \; y \; \dot{y}]^T(k)$ representing the positions and velocities of the targets at time $k\Delta$, where Δ is the time step between measurements. The two targets are initially 10 units apart and initially move in parallel directions with the same speed, but due to process and acceleration noise, directions and speed vary in time. There are two sensors whose measurements are governed by (2) with

$$\mathbf{H}_1 = \mathbf{H}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{6}$$

The process and measurement noise covariances are

$$\mathbf{Q} = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix}, \; \mathbf{R}_1 = \begin{bmatrix} r_1 & 0 \\ 0 & r_1 \end{bmatrix}, \; \mathbf{R}_2 = \begin{bmatrix} r_2 & 0 \\ 0 & r_2 \end{bmatrix}. \tag{7}$$

The measurements corresponding to the sensor with covariance $\mathbf{R}_1$ are processed first, and measurements from the sensor with covariance $\mathbf{R}_2$ are processed second in the sequential MSJPDA. The initial states of the targets are perfectly known, and each target is always well inside the surveillance region.

To evaluate tracking performance, one hundred Monte Carlo runs were performed for various values of clutter density λ and the average RMS position error over all runs was computed. Figure 1 shows a sampling of our results, where the following parameter values were used: $\Delta = 1$, $P_G = 0.999$, and $P_D = 1.0$. The clutter density λ was varied from 0.1 to 1.0. The system noise was varied ($q = 0.0144$ and $q = 0.0256$), and three pairs of curves were produced to compare sequential algorithm performance when the different sensors of different qualities were applied

$(r_1, r_2 = 0.0064, 0.0256, 0.1024)$. The two sensors used are of different qualities; the better sensor is the one with the "smaller" noise covariance matrix $\mathbf{R}_i$. With these parameter values, the expected number of false measurements per gate, using the steady-state Kalman filter covariances, varies from 0.085 to 4.672. From Figure 1, comparing the trends of the RMS position error when the simulations are run with different system noise covariance parameters q, and with different ratios of r_1 and r_2 in (7), we see that processing the worse sensor first yields smaller RMS position error. We used RMS position error as the performance metric because it provides a measure of tracking accuracy based on comparison with truth information (which is available in the simulations) and also allows us to compare our results with previous work in [3, 4]. The volume of the uncertainty ellipsoid [5] (product of the eigenvalues of the covariance matrix $\mathbf{P}$) was also used as a performance metric in some simulations, and the results showed the same trend: processing the worse sensor first yields a smaller uncertainty ellipsoid. Track lifetimes were also computed in the simulations, but it was found that track lifetime does not seem to be affected by the order of sensor processing.

4 Modified Riccati Equation Analyses

Since the Modified Riccati Equation (MRE) can be used to predict the RMS position (or other) errors of the system [1, 2], we also applied the MRE to more efficiently evaluate whether processing the worse sensor first leads to smaller RMS errors for a wider range of system and scenario parameters. Multisensor extensions of the MRE [3] were used to predict the RMS tracking performance of the dynamic target model of (1)–(2) with $t = 1$ (single target).

For the appropriate time-invariant matrices $\mathbf{F}, \mathbf{G}, \mathbf{H}$ and $\mathbf{Q}, \mathbf{R_1}, \mathbf{R_2}$, the MRE iteration for the sequential MSPDA filter for two sensors tracking one target is

$$\mathbf{P}(k|k-1) = \mathbf{F}\,\mathbf{P}(k-1|k-1)\,\mathbf{F}^T + \mathbf{G}\,\mathbf{Q}\,\mathbf{G}^T \tag{8}$$

$$\mathbf{S_1}(k) = \mathbf{H}\,\mathbf{P}(k|k-1)\,\mathbf{H}^T + \mathbf{R_1} \tag{9}$$

$$\mathbf{K_1}(k) = \mathbf{P}(k|k-1)\,\mathbf{H}^T\,\mathbf{S_1}^{-1}(k) \tag{10}$$

$$\mathbf{P_1}(k|k) = \mathbf{P}(k|k-1)\; - \;C_k^1\,\mathbf{K_1}(k)\,\mathbf{S_1}(k)\,\mathbf{K_1}(k)^T \tag{11}$$

$$\mathbf{S_2}(k) = \mathbf{H}\,\mathbf{P_1}(k|k)\,\mathbf{H}^T\; + \;\mathbf{R_2} \tag{12}$$

$$\mathbf{K_2}(k) = \mathbf{P_1}(k|k)\,\mathbf{H}^T\,\mathbf{S_2}^{-1}(k) \tag{13}$$

$$\mathbf{P}(k|k) = \mathbf{P_1}(k|k)\; - \;C_k^2\,\mathbf{K_2}(k)\,\mathbf{S_2}(k)\,\mathbf{K_2}(k)^T \tag{14}$$

where

$$C_k^1 = P_D P_G - q_1 + q_2(\lambda V_k^1) \tag{15}$$

$$C_k^2 = P_D P_G - q_1 + q_2(\lambda V_k^2) \tag{16}$$

The q_1 and q_2 functions are defined in [1, 2] and depend on the dimension of the system, P_D, λ, and V_k, the volume of the validation region (gate) at time k.

In the time-invariant case considered here, it was found [2] that for most values of P_D and λ, the equations (8)–(14) can be iterated until the covariance $\mathbf{P}(k|k)$ converges to a steady-state covariance matrix $\bar{\mathbf{P}}$. No general stability results are known for the MRE, but numerical convergence and divergence have been observed. In order to obtain a scalar tracking performance metric, the steady-state RMS position error can be extracted [2] from the sum of the diagonal elements of the $\bar{\mathbf{P}}$ matrix corresponding to target position

$$RMS = e(P_D, \lambda) \overset{\text{def}}{=} \sqrt{\sum_{position} \text{diag}\,(\bar{\mathbf{P}})} \tag{17}$$

Using the MRE for a one target-two sensors scenario, we computed the steady state error covariance

$$\bar{\mathbf{P}} = \mathbf{P}(k|k) = \mathbf{P}(k-1|k-1) \tag{18}$$

which is the same definition of RMS position error used in the simulations of Section 3.

4.1 One, Two, and Three-Dimensional MRE

We considered tracking systems in one, two, and three-dimensions, with appropriate system matrices (5), (6), and (7). In addition to numerically iterating the MRE in equations (8)–(14) in order to obtain the RMS position error, we also solved the steady-state equation (18) under particular assumptions. The parameters used were the same as in the simulations discussed in Section 3. The state vector consisted of position, or position and velocity, for the one, two, and three-dimensional tracking scenarios.

We first consider the case when the state vector consists of position and velocity. Under simplified conditions where $\mathbf{Q}$, $\mathbf{R}_1$, and $\mathbf{R}_2$ are diagonal matrices, as in equation (7), and if the initial position covariance matrix $\mathbf{P}(0|0)$ is block diagonal, the steady state position error covariance matrix $\bar{\mathbf{P}}$ will also be block diagonal. Expressions for the steady-state block diagonal components of $\bar{\mathbf{P}}$ under these special circumstances were given in [6] for second, fourth, and sixth order system models where the state vectors consist of position and velocity.

The steady-state solution results for the fourth order system model are shown in Figure 2 for tracking in two dimensions, and compared with the results obtained by MRE iterations (8)–(14). The RMS position error is shown as a function of the clutter density λ for three sets of sensor parameters: $r_1 = r_2$, $r_1 > r_2$, and $r_1 < r_2$. When the two sensors are of different qualities ($r_1 \neq r_2$), the computed RMS position error depends on the order of sensor processing, giving the same trends as observed in the simulations: processing the worse sensor first ($r_1 > r_2$) results in smaller RMS position error (better performance). Similar results were obtained for second and sixth order target process models.

We also evaluated the MRE (8)–(14) with the state vector consisting of position only. The one-dimensional model reduces to a scalar system discussed here, while the two and three-dimensional models have diagonal entries (again, under assumptions that $\mathbf{Q}$, $\mathbf{R}_1$, and $\mathbf{R}_2$ are diagonal, and $\mathbf{P}(0|0)$ is diagonal) defining the RMS position error in (17), being similar to the scalar system. For the scalar system,

$$\mathbf{F} = \mathbf{G} = \mathbf{H} = 1, \qquad \mathbf{R_1} = r_1, \qquad \mathbf{R_2} = r_2, \qquad \mathbf{Q} = q, \qquad\qquad r_1, r_2, q > 0. \tag{19}$$

The solution of $\bar{\mathbf{P}}$ in (18) as a function of q, r_1, r_2, C_k^1, and C_k^2 was discussed in detail in [6]. This was not a closed-form solution, since C_k^1 and C_k^2 in steady state depend on q, r_1, and r_2 in a complicated way, and to obtain numerical solutions, we used the steady-state values of C_k^1 and C_k^2 from MRE iterations (8)–(14). Under our assumptions that $\mathbf{Q}$, $\mathbf{R_1}$, and $\mathbf{R_2}$ are diagonal, and $\mathbf{P}(0|0)$ diagonal, however, it may be possible to map out approximate functions for C_k^1 and C_k^2 in steady state in terms of q, r_1, r_2, and clutter density λ. The system noise q and sensor parameters r_1 and r_2 affect the steady state covariance error $\bar{\mathbf{P}}$, and therefore the RMS position error. The RMS position error as a function of clutter density λ for first, second, and third order system models showed trends that are similar to those of Figure 2.

4.2 Results over Larger Parameter Ranges

So far, we have shown that for several sets of sensor parameters, the system tracking performance in terms of the RMS position error improves if the worse sensor is processed first. It is of interest, however, to investigate how the order of sensor processing affects the RMS position error over a wider range of system parameters: sensor parameters r_1 and r_2, system noise q, and clutter density λ. Hence, we used MRE numerical iterations to efficiently evaluate the RMS position error over large parameter ranges.

We varied the ratio of the sensor parameters r_1/r_2 between 0.01 and 100, while keeping the parameter r_e of the equivalent sensor $\left(\frac{1}{r_e} = \frac{1}{r_1} + \frac{1}{r_2}\right)$ at $r_e = 0.01$. Figure 3 shows for a second-order, two-dimension system, how the RMS position error, normalized to the RMS position error when both sensors are equal ($r_1/r_2 = 1$), varies with r_1/r_2 for $q = 0.01$ and 0.1, and $\lambda = 0.001$, 0.2, 0.5, and 1.0. The results are given only for the parameter values where the MRE iterations converge. Results for other first through sixth order process models are similar. In the three-dimensional case, the range where processing the worse sensor first gives smaller errors is narrower, and the improvements are less significant, as shown in Figure 4, where the lower plot is an expanded view for $0.5 < r_1/r_2 < 2$. The range where $r_1/r_2 > 1$ corresponds to the case when the worse sensor is processed first. As expected, the RMS position error increases with increased system noise q and with increased clutter density λ.

Qualitative behavior over a large range of r_1/r_2 is mostly affected by the system order and noise q. If the system noise q is low, no significant differences in the RMS position error can be observed for different orders of processing sensor information – the curves are almost symmetrical around the center point $r_1/r_2 = 1$. For larger noise q, the RMS position error exhibits a local minimum at or slightly to the right of the $r_1/r_2 = 1$ point. Around this point, there is a range of r_1/r_2 values where the RMS position error is smaller if the worse sensor is processed first. For example, in the two-dimensional, second order case (Figure 3), the error for $r_1/r_2 = 10$ is smaller than the error for $r_1/r_2 = 0.1$, and in three-dimensional, third order case (Figure 4), the error for $r_1/r_2 = 2$ is smaller than the error for $r_1/r_2 = 0.5$. When $r_1/r_2 \gg 1$, a region of r_1/r_2 can be found when processing the better sensor first yields smaller RMS position error, as well as the region where order of processing sensor information does not affect the RMS position error, as can be seen in Figure 4. The shape of the obtained curves and the size of regions highly depend on clutter density λ, system noise q, and system order. This leads to a conclusion that whenever sequential MSJPDA filtering is to be applied with unequal sensors, it would be beneficial to know the ratio r_1/r_2 of sensor characteristics, as well as estimates of λ and q, so that the processing order can be set up giving the minimal RMS position error.

5 Conclusions

We have analyzed the order of sensor processing in the sequential implementation of the MSJPDA filter. Our results indicate that processing the worse sensor first generally yields smaller RMS position error if the two sensors are of unequal but comparable quality. Though counter intuitive at first, this trend was confirmed in one, two, and three-dimensional models (first through sixth-order systems) of the MSPDA filter using both the MRE numerical algorithm and its steady-state solution under particular conditions. A full explanation of this trend is difficult, because of the complexity of the data association process built in the MSPDA algorithm, but one explanation would be that processing the best sensor last improves tracking performance of the sensor system as a whole. Analyses over ranges of sensor parameters show that tracking system performance of the sequential MSJPDA filter, in terms of the RMS position error, favors using sensors of comparable qualities, and that processing the worse sensor first gives better results if the sensor qualities do not differ by a large amount.

References

[1] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, Academic Press Inc., 1988.

[2] T. E. Fortmann, Y. Bar-Shalom, M. Scheffe, and S. Gelfand, "Detection Thresholds for Tracking in Clutter – A Connection Between Estimation and Signal Processing," *IEEE Trans. Aut. Control*, 30(3): 221–229, Mar. 1985.

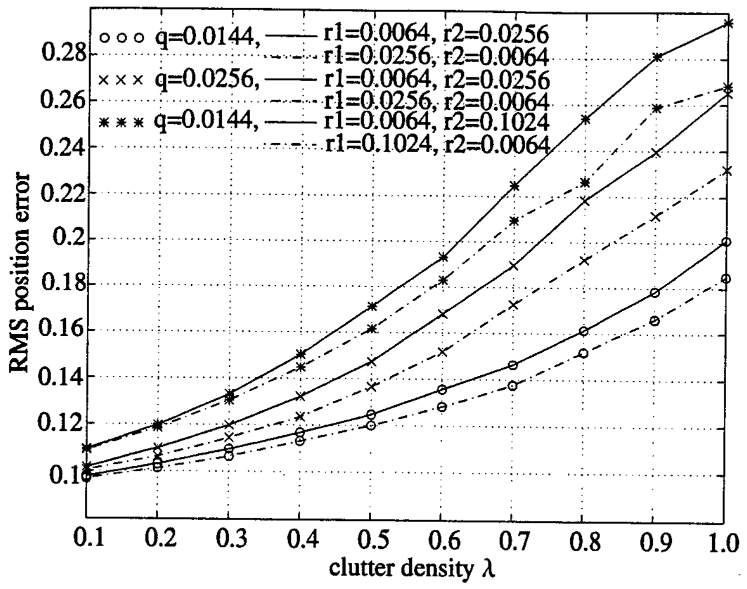
[3] C. W. Frei, *A Comparison of Parallel and Sequential Implementations of a Multisensor Multitarget Tracking Algorithm*, M. S. thesis, Northwestern University, May 1995.

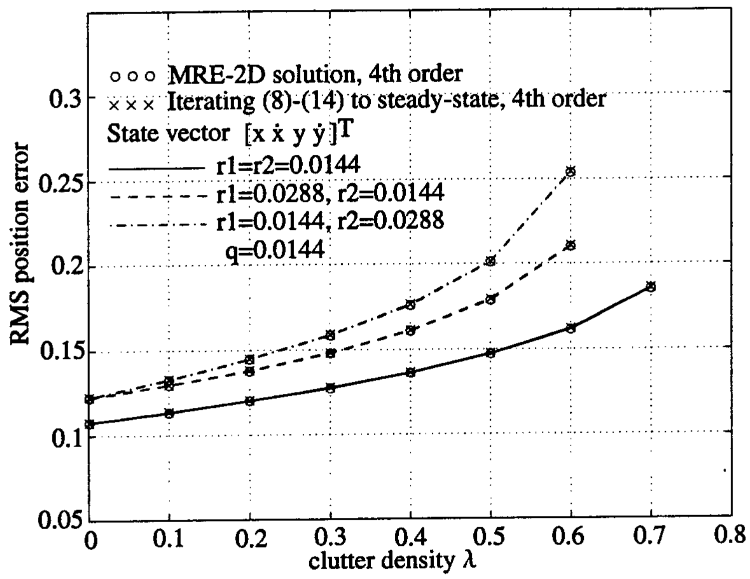
[4] C. W. Frei and L. Y. Pao, "Alternatives to Monte-Carlo Simulation Evaluations of Two Multisensor Fusion Algorithms," *Automatica*, 34(1): 103–110, Jan. 1998.

[5] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*, Addison-Wesley Publishing Company, 1991.

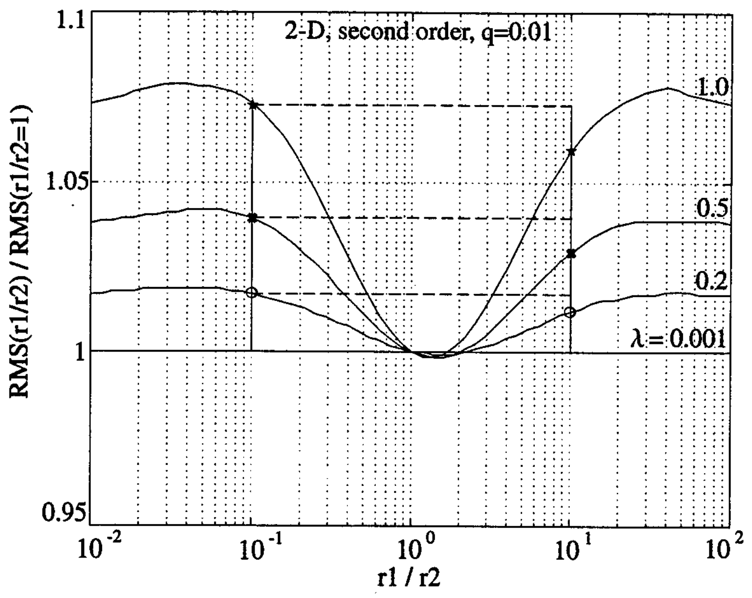
[6] L. Y. Pao and L. Trailović, “On the Order of Processing Sensors in Sequential Implementations of Fusion Algorithms”, *Proc. American Control Conference*, San Diego, CA, pp. 2407–2411, June 1999.

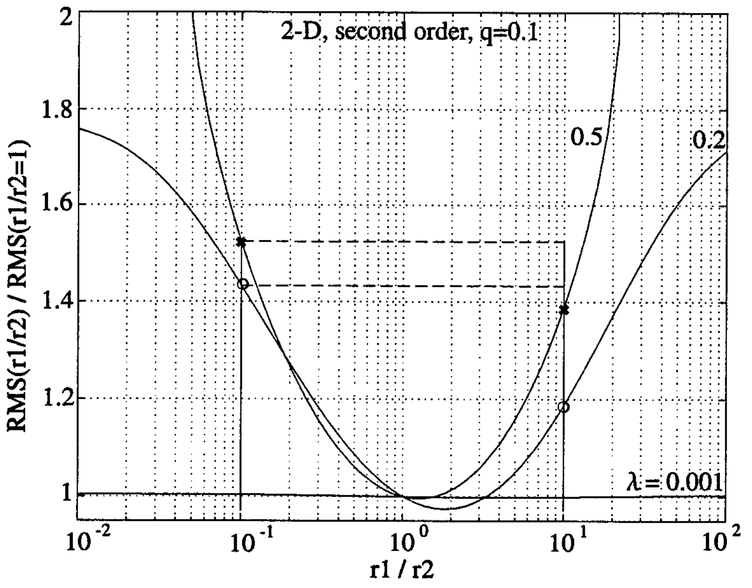
List of Figure Captions

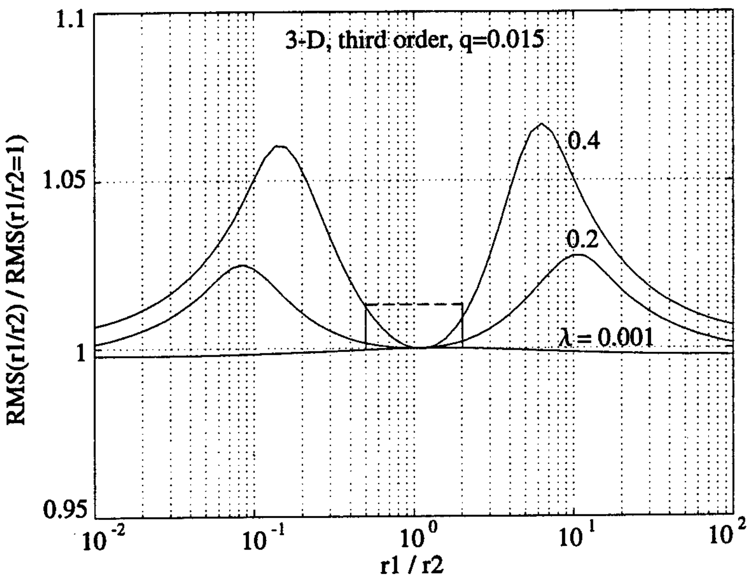

○ ○ ○ q=0.0144, —— r1=0.0064, r2=0.0256
-·-·- r1=0.0256, r2=0.0064
× × × q=0.0256, —— r1=0.0064, r2=0.0256
-·-·- r1=0.0256, r2=0.0064
✱ ✱ ✱ q=0.0144, —— r1=0.0064, r2=0.1024
-·-·- r1=0.1024, r2=0.0064
RMS position error
clutter density λ
0.28
0.26
0.24
0.22
0.2
0.18
0.16
0.14
0.12
0.1
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

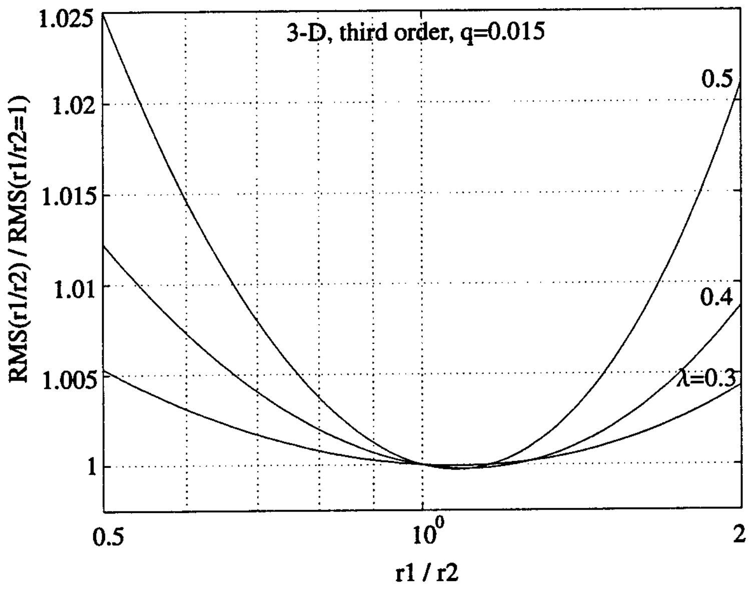


○ ○ ○ MRE-2D solution, 4th order
× × × Iterating (8)-(14) to steady-state, 4th order
State vector $[x\ \dot{x}\ y\ \dot{y}]^T$
r1=r2=0.0144
r1=0.0288, r2=0.0144
r1=0.0144, r2=0.0288
q=0.0144
RMS position error
clutter density λ
0.05
0.1
0.15
0.2
0.25
0.3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


2-D, second order, q=0.01
1.0
0.5
0.2
λ = 0.001
1.1
1.05
1
0.95
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
r1 / r2
RMS(r1/r2) / RMS(r1/r2=1)


2-D, second order, q=0.1
0.5
0.2
λ = 0.001
2
1.8
1.6
1.4
1.2
1
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
r1 / r2
RMS(r1/r2) / RMS(r1/r2=1)


3-D, third order, q=0.015
0.4
0.2
λ = 0.001
1.1
1.05
1
0.95
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
r1 / r2
RMS(r1/r2) / RMS(r1/r2=1)


3-D, third order, q=0.015
0.5
0.4
λ=0.3
1.025
1.02
1.015
1.01
1.005
1
0.5
10^{0}
2
r1 / r2
RMS(r1/r2) / RMS(r1/r2=1)

Submitted in Dec. 1999 for the Proc. AIAA Guidance, Navigation, and Control Conf., Denver, CO, Aug. 2000.
Accepted for publication in April 2000.

THE EFFECTS OF DATA ASSOCIATION ON SENSOR MANAGER SYSTEMS

Michael Kalandros, Student Member AIAA

and

Lucy Y. Pao, Member AIAA

Department of Electrical and Computer Engineering

University of Colorado

Boulder, CO 80309-0425

kalandro@colorado.edu and pao@colorado.edu

Abstract

Surveillance systems tracking multiple targets often do not have the sensing or computational resources to apply all sensors to all targets in the allocated time intervals. Hence, sensor management schemes have recently been proposed to reduce the tracking demands on these systems while minimizing the loss of tracking performance by selecting only enough sensing resources to maintain a desired covariance level for each target. However, covariance control algorithms to date have not addressed the presence of clutter measurements and the need for data association in those cases. This paper presents a method of reducing the effects of data association on covariance control algorithms through the addition of a scalar "loss of information" term. Monte Carlo simulations show that without this term, the covariance control system is unable to maintain the desired covariance, resulting in a much larger actual covariance level and ultimately a much higher rate of track loss. Use of the loss of information term generally restores system performance. Further insights guide the selection of effective covariance goals.

1 Introduction

The application of multisensor fusion to surveillance systems has provided superior tracking performance at the cost of increased sensing and computational demands. Ideally, all available sensors can be applied to all targets to achieve the most accurate state estimate of each one. However, most sensors can track only a finite number of targets in a single sampling period. Because of this, not all targets can be tracked with all sensors and improving tracking accuracy for one target may result in the degradation of track accuracy for a different target. Furthermore, each measurement imposes computational costs on a tracking system with a finite amount of processing capacity. What is needed is a sensor management technique that can balance tracking performance with available resources [11].

Sensor managers can control a variety of system parameters including which sensor combinations are used in a multi-sensor system [14, 15, 7]; sampling frequency or revisit time [1]; and sensor modes or waveforms [9]. In [7], a system is presented that manages system resources by selecting individual sensors to achieve a specified state estimate covariance for each target, rather than attempting to use all sensors on all targets. The system separates sensor management into a covariance control problem and a sensor scheduling problem (see Figure 1). The scheduler prioritizes sensing actions and executes them as time allows. Low priority actions may be delayed until future scans or may be dropped altogether. The effects of such delays have been studied in [13]. The covariance controller maintains the covariance level of each target estimate to within the desired limit while reducing system resource demands.

In many environments, the tracking task is complicated by the presence of clutter measurements and multiple (possibly closely-spaced) targets. When this occurs, each measurement must be correctly associated with its originating target or classified as a false or clutter measurement. This process is known as data association. The resulting uncertainty about measurement origin decreases the accuracy of the state estimate of each track and can ultimately lead to track loss.

Several data association methods have been proposed, including the Probabilistic Data Association Filter (PDAF) [2]. The state estimate covariance calculation for this filter depends on the actual measurements received, making it

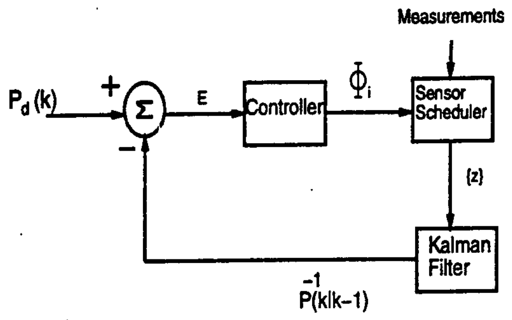


Figure 1: *Block Diagram of a Tracking System with Covariance Controller (Sensor Selection Algorithm).*

stochastic in nature, but can be approximated by the addition of a scalar "loss of information" term to the standard Kalman filter equations [4, 10]. While single-sensor management systems have been designed for the PDAF (e.g. [1, 9]), no sensor managers to date have been designed around the multisensor extension of the PDAF [12, 6]. This paper expands the covariance control algorithms presented in [7] to account for the loss of information encountered in the multisensor PDAF.

The paper is organized as follows. The Kalman filter is summarized in Section 2. The basic covariance control algorithm is described in Section 3. Section 4 describes the Probabilistic Data Association Filter, and the loss of information associated with the PDAF is presented in Section 5. Algorithms using this loss of information are derived in Section 6 and simulation results of these algorithms are presented in Section 7. Section 8 will discuss the implications of those results for the selection of desired covariances before the final conclusions are given in Section 9.

2 Kalman Filter

Assume that the tracking system has N_s sensors. There are 2^{N_s} possible combinations or subsets of those sensors that can be selected by the multisensor manager. The ith possible subset is defined as Φ_i and N_{s_i} is the number of sensors in that combination. The input from each selected sensor is used to update the state estimate of the target.

The sequential Kalman filter is an algorithm for combining multiple inputs from stochastic or linearized systems to form an estimate in a state space representation. The Kalman filter is based on the following assumptions about the target and measurement systems [2, 3]:

$$x(k) = Fx(k-1) + Gu(k-1) + w(k-1) \tag{1}$$

$$z_j(k) = H_j x(k) + v_j(k), \quad j = 1, \ldots, N_{s_i} \tag{2}$$

where $x(k)$ is the current state of the target; F, G, H_j are known system matrices; $u(k)$ is the control signal; and $z_j(k)$ is a measurement of the target from the jth sensor in Φ_i. $w(k)$ is a variable representing process noise or higher-order motion not modeled by F, and $v_j(k)$ is a variable representing measurement noise in sensor j. Both $w(k)$ and $v_j(k)$ are assumed to have zero-mean, white, Gaussian probability distributions.

Since $w(k)$ and $v_j(k)$ are zero-mean noise processes, the target states and measurements in the next time interval can be predicted by

$$\hat{x}(k \mid k-1) = F\hat{x}(k-1 \mid k-1) + Gu(k-1) \tag{3}$$

$$\hat{z}_j(k) = H_j \hat{x}(k \mid k-1) \tag{4}$$

The input $u(k)$ is considered known and will be omitted in future equations because it can be easily reinserted. The quantity $\nu_j(k) = \hat{z}_j(k) - z_j(k)$ is known as the innovation. The covariance of the state and the innovation predictions are

$$P(k \mid k-1) = FP(k-1 \mid k-1)F' + Q(k-1) \tag{5}$$

$$S_1(k) = H_1 P(k \mid k-1) H_1' + R_1(k) \tag{6}$$

$$S_j(k) = H_j P_{j-1}(k \mid k) H_j' + R_j(k), \quad j = 2, \ldots, N_{s_i} \tag{7}$$

respectively, where $Q(k)$ is the process noise covariance and $R_j(k)$ is the measurement noise covariance for the jth sensor. $P_j(k \mid k)$ is the updated covariance resulting from sensor j as defined in eq. (10) below.

The sequential algorithm runs a separate Kalman filter for each sensor in the combination, propagating its estimate to the next filter [16]:

$$\begin{aligned}
\hat{x}_1(k \mid k) &= \hat{x}(k \mid k-1) + K_1(k)\big(z_1(k) - H_1\hat{x}(k \mid k-1)\big) \\
\hat{x}_j(k \mid k) &= \hat{x}_{j-1}(k \mid k) + K_j(k)\big(z_j(k) - H_j\hat{x}_{j-1}(k \mid k)\big), \quad j = 2, \ldots, N_{s_i} \\
\hat{x}(k \mid k) &= \hat{x}_{N_{s_i}}(k \mid k)
\end{aligned} \tag{8}$$

where

$$\begin{aligned}
K_1(k) &= P(k \mid k-1) H_1' S_1^{-1}(k) \\
K_j(k) &= P_{j-1}(k \mid k) H_j' S_j^{-1}(k), \quad j = 2, \ldots, N_{s_i}
\end{aligned} \tag{9}$$

The state covariance is updated for each filter by

$$\begin{aligned}
P_1(k \mid k) &= (I - K_1(k)H_1)P(k \mid k-1) \\
P_j(k \mid k) &= \big(I - K_j(k)H_j\big)P_{j-1}(k \mid k), \quad j = 2, \ldots, N_{s_i} \\
P(k \mid k) &= P_{N_{s_i}}(k \mid k)
\end{aligned} \tag{10}$$

Once the state and covariance estimates have been updated, they are fed back into the algorithm and the entire process is repeated for the new set of measurements at the next time step. Alternatively, the covariance update can be calculated in a single step using the inverses of the covariance matrices [3]:

$$P^{-1}(k \mid k) = P^{-1}(k \mid k-1) + \sum_{j \epsilon \Phi_i} H_j' R_j^{-1} H_j \tag{11}$$

where

$$J_i = \sum_{j=1}^{N_{s_i}} H_j' R_j^{-1} H_j, \quad i = 1, \ldots, 2^{N_s} \tag{12}$$

is the sensor information gain for the ith combination of sensors.

3 Covariance Control

The covariance control approach to sensor management assigns a desired state estimate covariance goal to each target and selects combinations of sensors to meet those goals at each sampling period. By attempting to meet these goals rather than using all sensors on all targets, the demands on sensors and computational resources can be reduced.

Figure 1 shows the block diagram of the tracking system proposed in [7]. Control of the covariance of the system is implemented via a sensor selection algorithm. The sensor selection can be determined based on the difference between the inverses of the predicted covariance in (5) and the desired covariance $P_d(k)$. Replacing the updated covariance matrix in (11) with the desired covariance and solving for the necessary sensor information gain, we see that we want J_i to equal E, where

$$E = P_d^{-1}(k) - P^{-1}(k \mid k-1) \tag{13}$$

The selected sensor combinations are then passed to a scheduler which executes the requests as time and resources permit. The sensors then pass measurements to the Kalman filter, which produces updated state and covariance estimates, as well as state and covariance predictions for the next sampling period, for each target.

In actual target tracking applications a number of other tasks are also performed, including data association (when clutter measurements or closely-spaced targets are present), track initiation and deletion, and registration [3].

These tasks compete for sensor and processor time as well, and would also fall under the control of the sensor manager. In the initial development of sensor managers in [7], only the state estimation task is considered. With this assumption, the controller's job is to regulate the sensing resources used by the Kalman filter to reduce the demands on the tracking system due to state estimation.

The covariance control algorithm used in this paper is the Eigenvalue/Minimum Sensors Algorithm [7]. It requires that the sensors used produce an updated covariance that is within the desired covariance at all times. This will result in the difference, $P_d - P_i$, where P_i is the updated covariance after using sensor combination i, having all positive eigenvalues (as well as the difference $J_i - E$). While adding sensors will eventually achieve this goal, the computational demand on the Kalman filtering algorithm will increase linearly with the number of sensors. Since the goal is also to reduce the computational load on the tracking system, the sensor combination with the fewest number of sensors that produces all positive eigenvalues in the covariance error should be used at each scan. The sensor information matrices J_i can be calculated off-line and stored in an on-line library to reduce the computational demand of sensor selection.

4 Probabilistic Data Association Filter (PDAF)

Many sensing systems (such as radar) can receive false measurements in addition to returns from targets of interest. These clutter measurements can be due to noise in the sensing system itself, multi-path reflections from the target, or returns from objects that are not targets of interest. Clutter measurements are typically assumed to be uniformly distributed and thus the number of false measurements for a given area will be a Poisson distribution parameterized by the clutter density, λ.

Probabilistic Data Association [2] begins by defining the normalized distance squared between each measurement $z_\ell(k)$ (the sensor index j is suppressed here to simplify the notation) and its predicted value $\hat{z}(k)$ as

$$d(\nu_\ell) = \nu'_\ell(k) S^{-1}(k) \nu_\ell(k), \quad \ell = 1 \ldots M_k \tag{14}$$

where M_k is the number of measurements received by an individual sensor at time k. This distance is used to eliminate measurements that are unlikely to be target-originated by defining a gate as a region around the expected value of the measurement such that $d(\nu_\ell) \leq \gamma^2$. The measurements inside the gate are processed by the PDAF while measurements outside the gate are ignored. The gate size, γ is chosen so that the true measurement will be found inside the gate with some high probability, usually $> 99\%$. The volume of this region at time k can be calculated as

$$V_k = c_m \gamma^m |S(k)|^{1/2} \tag{15}$$

where m is the dimension of the measurement and c_m is the volume of an m-dimensional unit sphere.

The PDAF derives an estimate for the state by replacing the innovation with a weighted sum of gated innovations, known as the combined innovation,

$$\nu = \sum_{\ell=1}^{m_k} \nu_\ell \beta_\ell \tag{16}$$

where β_ℓ is the probability that measurement z_ℓ is the true measurement of the target and m_k is the number of gated measurements at time k. These probabilities can be calculated as

$$\beta_\ell(k) = \begin{cases} \frac{e_\ell}{b + \sum_{n=1}^{m_k} e_n} & \ell = 1 \ldots m_k \\ \\ \frac{b}{b + \sum_{n=1}^{m_k} e_n} & \ell = 0 \end{cases} \tag{17}$$
$$e_\ell = e^{-\frac{d(\nu_\ell)}{2}}$$

$$b = \frac{\lambda|2\pi S(k)|^{1/2}(1 - p_D p_G)}{p_D}$$

where p_D is the probability of detecting the target and p_G is the probability that the detected target will fall within the measurement gate described above.

The state covariance resulting from using the combined innovation is

$$\begin{aligned} P(k|k) &= P(k|k-1) - (1-\beta_0)K(k)H(k)P(k|k-1) + \tilde{P}(k) \\ \tilde{P}(k) &= K(k)\left[\sum_{\ell=1}^{m_k} \beta_\ell \nu_\ell \nu'_\ell - \nu(k)\nu'(k)\right]K'(k) \end{aligned} \tag{18}$$

All other values in the Kalman Filter are calculated as discussed in Section 2.

5 Loss of Information in the PDAF

The presence of β_0 and $\tilde{P}(k)$ in eq. (18) change the calculation of the covariance from a deterministic equation into a stochastic one (because both terms depend on the actual data received). In [4], a method of predicting tracking system performance without resorting to Monte Carlo simulations is proposed. It calculates the expected value of the covariance, given the previous conditions of the filter. The resulting equations are then evaluated for each sampling period to predict average RMS error, track lifetime, etc. We propose to use the single period performance estimate to improve the accuracy of sensor selection for covariance control algorithms. [4] begins with the expected value of the covariance calculated in eq. (18)

$$E[P(k|k)|Z^{k-1}, P(k|k-1)] = P(k|k-1) - (1 - E[\beta_0])K(k)H(k)P(k|k-1) + E[\tilde{P}(k)] \tag{19}$$

where

$$\begin{aligned} E[\beta_0] &= 1 - p_D p_G && (20) \\ E[\tilde{P}(k)] &= K(k) E\left[\sum_{l=1}^{m_k} \beta_l \nu_l \nu_l' - v(k)v'(k)\right] K'(k) \\ &= (q_1 - q_2) K(k) H(k) P(k|k-1) && (21) \end{aligned}$$

The variables q_1 and q_2 are scalar, and expressions for them are derived in [4]. With some final cancelations, and considering that $E[\beta_0] \approx q_1$ under typical conditions,

$$E[P(k|k)|Z^{k-1}, P(k|k-1)] = (I - q_2 K(k) H(k)) P(k|k-1) \qquad (22)$$

[4] further shows that q_2 varies between 0 and 1. Because it reduces the effectiveness of the sensor measurements, q_2 is known as the "loss of information" term. The calculation of q_2 is computationally demanding and will be calculated off-line when used in the covariance control algorithms.

Clearly, if the loss of information due to data association is ignored in covariance control techniques, the sensor manager will overestimate the effect of each sensor on the state estimate covariance and the sensor selections will consistently fail to meet the covariance goals. To counter the loss of information effect on sensor management, this research uses the average performance of the data association algorithms in the presence of clutter and incorporates the resulting scalar term into the sensor models.

6 The Augmented Covariance Control Algorithm

In [4] the q_2 term is shown to be a function of the expected number of gated measurements (which in turn is a function of the innovation covariance, $S(k)$) and the probability of detection. Using this information, it is possible to pre-compute a table of possible q_2 values for use in an online algorithm. The values from this table are shown in

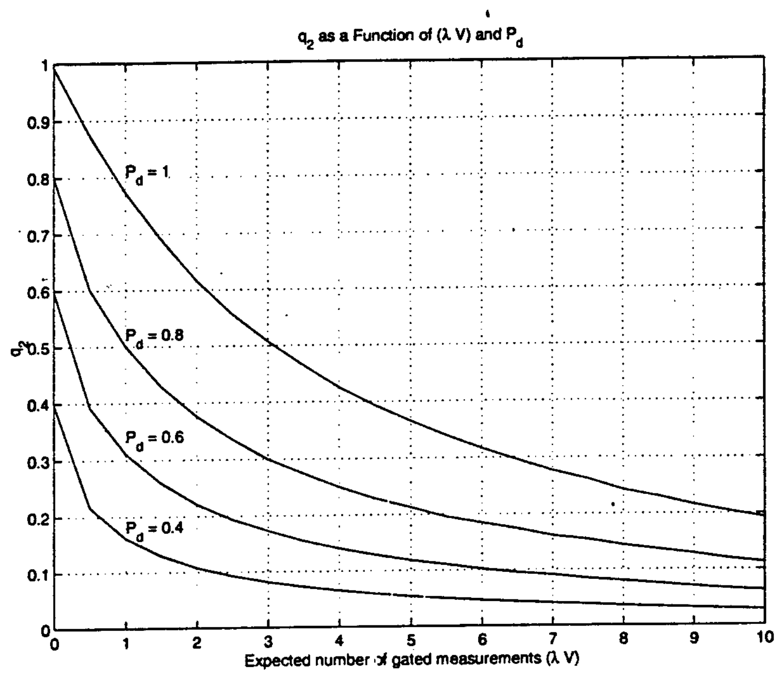


Figure 2: *Loss of Information as a function of probability of detection and the expected number of measurements.*

Figure 2.

The algorithm begins by calculating the predicted state of the target using eq. (3) and the covariance of that prediction using eq. (5). It then estimates the loss of information term by calculating the expected number of measurements, $\lambda V(k)$, and using that information to linearly interpolate the q_2 term from the previously calculated table. Note that while the expectation is exact for the first sensor in Φ_i, it is only an approximation for the sensors that follow, since each depend non-linearly on the updated covariance resulting from the previous sensors. A method called hybrid approximation [5, 10], is a more exact representation, but is more computationally demanding. However, it is expected that the value of q_2 will be underestimated for all but the first sensor, since the algorithm only calculates the q_2 term based on the original prediction covariance. It does not account for the smaller *a priori* covariance (P_{j-1} in eq. (10)) seen by all of the sensors after the first one.

Once the loss of information terms have been calculated, eq. (22) is used to estimate the effect of each sensor

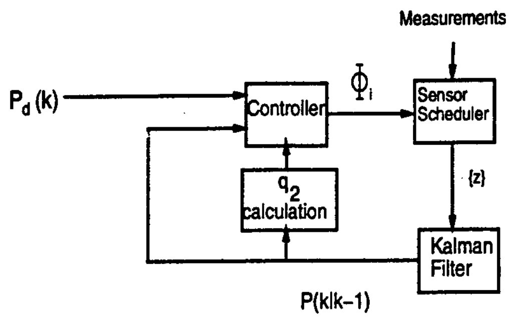


Figure 3: *Block Diagram of the augmented Covariance Controller System.*

on the target covariance. This information is used by the Eigenvalue/Minimum Sensors algorithm to choose an appropriate sensor combination for the target. The sensor combinations are formed a single sensor at a time using a greedy search heuristic to reduce the computational demand of sensor selection (see [8] for a more in-depth discussion of the computational demand of sensor selection and methods of reducing that demand). The Eigenvalue/Minimum Sensors algorithm itself is slightly modified from the one presented in [7, 8]. Because the PDAF is derived from the standard Kalman filter algorithms, and the scalar loss of information term can not be applied to the covariance update of the information filter, the information filter equations (eq. (11)) are abandoned and replaced with those of the standard Kalman filter (eq. (10)). Instead of the difference in eq. (13), the inputs to the augmented covariance controller consist of the desired covariance, the prediction covariance, and the loss of information terms (see Figure 3).

7 Simulation Results

This section presents the results from Monte Carlo simulations using the covariance control techniques discussed in this paper. The tracking situation presented is two targets moving in the $x - y$ plane in nominally straight lines corrupted by acceleration or control noise. This noise is zero-mean, white, and Gaussian with covariance matrices of 0.05 times the identity matrix. The target states to be tracked (estimated) consist of both the position and velocity of each target, i.e. $[x\ \dot{x}\ y\ \dot{y}]'$. The desired covariance for Target 1 is diagonal with a position variance of 0.02 and a velocity variance of 1. The desired covariance for Target 2 is also diagonal with a position variance of 0.02 but has a smaller velocity variance of 0.5. Each simulation is run for 500 scans.

For the purpose of these simulations, the targets are assumed to be non-interacting, which means that the distance between the two is large enough that the measurement of one will not lie in the gated region of the other. While this is not always the case in actual tracking applications, interacting targets result in a non-uniform clutter density which is not modeled by the q_2 term. Future work will determine the effect of closely-spaced targets on the current covariance control algorithm and develop methods to reduce or eliminate that effect.

The simulations compare two tracking systems, each consisting of 8 sensors measuring both the x and y position of each target. While the sensors' accuracy in a given direction varies, they have roughly equal overall abilities, defined as each having a position information matrix with equal volume, where the position information matrix for sensor j is defined as J_{p_j}, such that

$$\begin{aligned} J_{p_j} &= H_p J_j H_p', \quad j = 1, \dots, N_s \\ H_p &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{aligned} \tag{23}$$

J_j is the sensor information gain as defined in eq. (12), applied here to individual sensors only, and H_p extracts the position "information" from the overall sensor information gain. Both systems manage the sensing demand with the Eigenvalue/Minimum Sensors covariance control technique, but while one uses the loss of information term to model the effects of data association, the other does not. The measurement noise for each sensor is 0.04 times the identity matrix. Clutter density λ is the same for each sensor. The simulations vary clutter density from $\lambda = 0.2$ to $\lambda = 1.0$ in steps of 0.2 and are averaged over 200 runs each. This clutter density range leads the expected number of gated measurements varying from 5 to 27 measurements for target 1 and from 3 to 15 measurements for target 2, assuming that the desired covariance goals are met for each target.

The simulations record the actual minimum eigenvalue of the difference between the desired and updated covariance as well as that of the difference predicted by the sensor selection algorithm with and without the loss of information term. The predicted difference is used by the covariance controller to select the appropriate sensor combination. Because of this, successful covariance control depends on the accuracy of that prediction. The minimum eigenvalue of the covariance difference $(P_d - P_i)$ is used as the metric since the Eigenvalue/Minimum Sensors algorithm attempts to drive all eigenvalues greater than zero, thus the smallest eigenvalue will be positive if the algorithm achieves this goal and negative if it does not. The smallest eigenvalue of the covariance difference is normalized by that of the desired covariance to provide a more intuitive measurement of the accuracy of the control algorithm. The covariance metric is only evaluated while the target is actually being tracked and does not reflect covariance trends after the target is lost. Track loss is declared when target-originated measurements have not been gated for 5 consecutive scans or when the actual RMS position error exceeds 10 times the standard deviation of the position error estimate for 5 consecutive scans in either the x or the y direction, where the standard deviation is the square root of the state estimate variances corresponding to the x and y position estimates, respectively.

Figure 4 shows the average covariance difference (normalized by the minimum eigenvalue of the desired covariance matrix) for both targets without the use of the q_2 term. When it is not used, the minimum eigenvalue of the actual covariance is 20 to 170 times that of the desired covariance, meaning the covariance goal is not achieved at any clutter level. Furthermore, the minimum eigenvalue of the predicted covariance difference is always positive, meaning that the controller thinks that it will achieve the desired covariance at every scan.

Figure 5 shows the average normalized covariance difference for both targets when the q_2 term is used. The desired covariance goals are easily achieved, resulting in a positive definite covariance difference, in all but the highest clutter level. This difference is never more than 45% of the minimum eigenvalue of the desired covariance. Additionally, the predicted difference between the desired and updated covariance is never more than 25% of the minimum eigenvalue

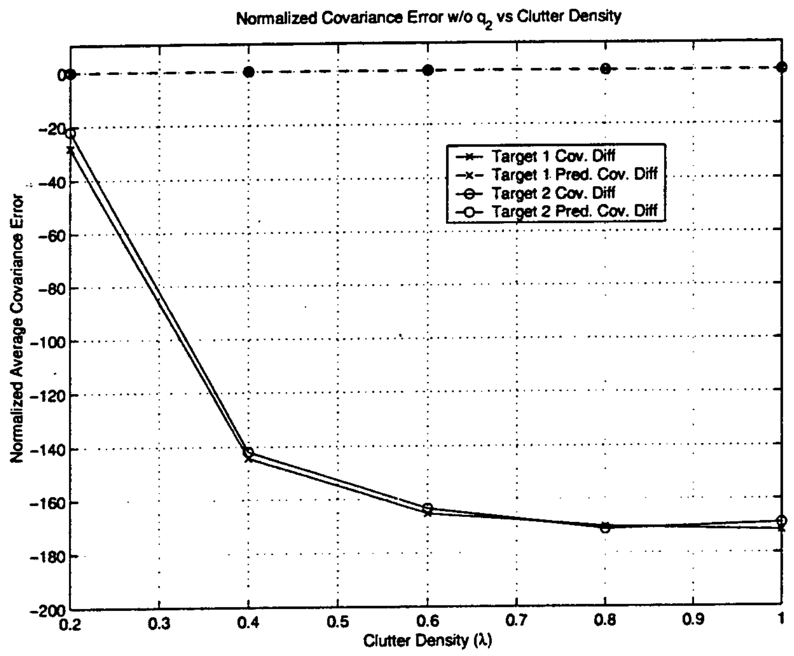


Figure 4: *Covariance differences for Targets* 1 *and* 2 *(normalized by the minimum eigenvalue of the desired covariance for each) when the sensor management algorithm does not take into account the effects of data association.*

of P_d. Finally, the error between the predicted difference and the actual difference is also never more than 25% of the smallest eigenvalue of the desired covariance. It is interesting to note that even though the average predicted covariance difference when $\lambda = 0.6$ and $\lambda = 1.0$ is negative, the average number of sensors used is below the maximum of 8 (shown in Figure 6). This, coupled with the fact that the average predicted covariance is negative less than 10% of the time, implies that the average difference is affected by large covariance errors relatively rarely, but that the size of those errors can be quite large.

Figure 6 is a graph of the average number of sensors used while the target is being tracked versus the clutter density. Even though the predicted covariance difference is generally positive for both algorithms (meaning that each algorithm expects to meet the desired covariance goal), the covariance control algorithm without the q_2 term

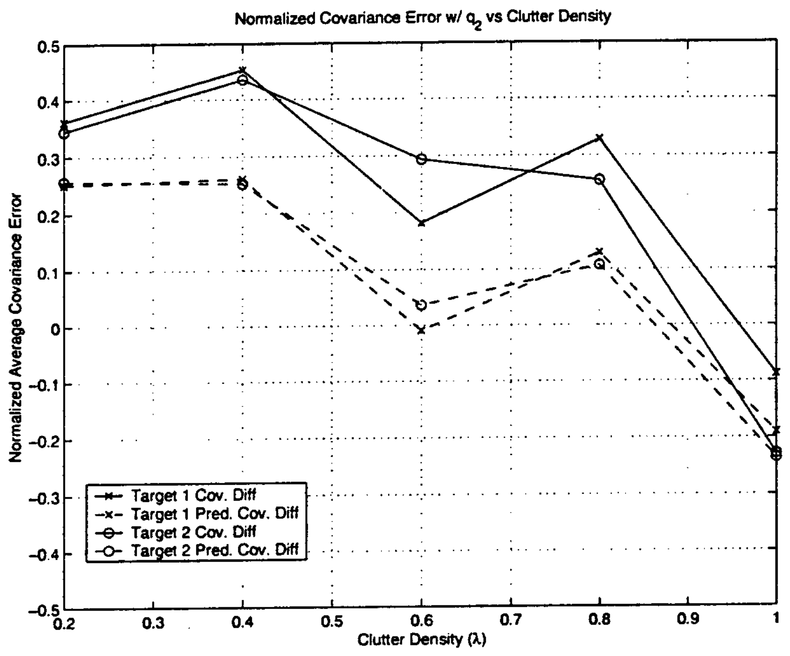


Figure 5: *Covariance difference for Targets* 1 *and* 2 *(normalized by the minimum eigenvalue of the desired covariance for each) when the sensor management algorithm uses the loss of information factor q_2 to model the effects of data association.*

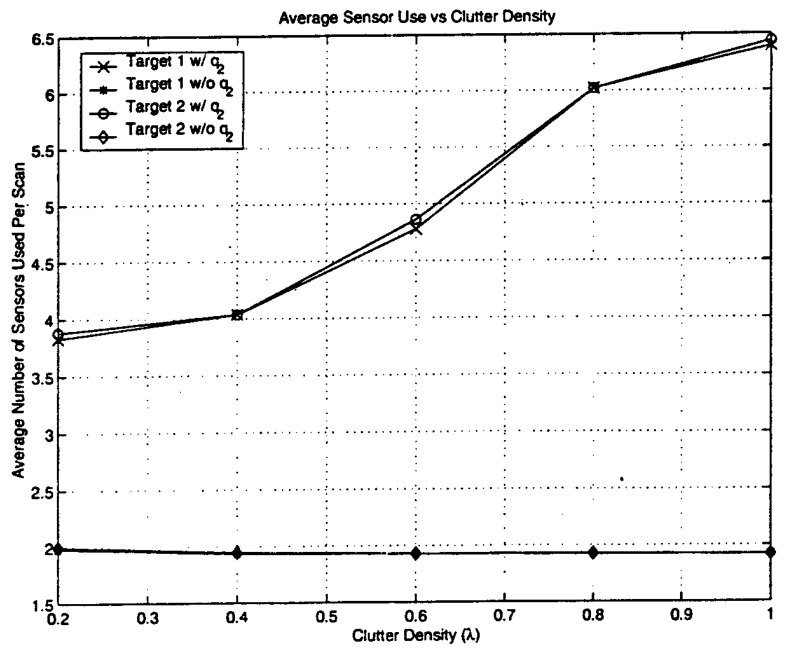


Figure 6: *Without the q_2 term, the sensor manager assigns fewer sensors to the tracking task and thus fails to achieve the desired covariance goals.*

uses only two sensors on average at each time scan, while many more sensors are selected by the covariance control algorithm using the q_2 parameter. Thus without the loss of information term, the covariance control algorithm overestimates the sensors' abilities and fails to allocate enough sensing resources to accomplish the tracking goals. With the q_2 term, the covariance control algorithm apparently ***underestimates*** sensor abilities, shown in Figure 5 by the consistently overly pessimistic estimation of the covariance difference by the algorithms (dashed lines) when compared to the actual differences (solid lines). As described in Section 6, this performance is expected, since only the original prediction covariance is used to calculate q_2 for each target/sensor combination, while the sequential Kalman filter reduces the covariance of the state estimate after each sensor is used. This effect can be reduced by recalculating the q_2 term after each sensor is picked. However this practice will increase the computational demand of the covariance controller and make it less robust to possible changes in sensor execution order by the scheduler.

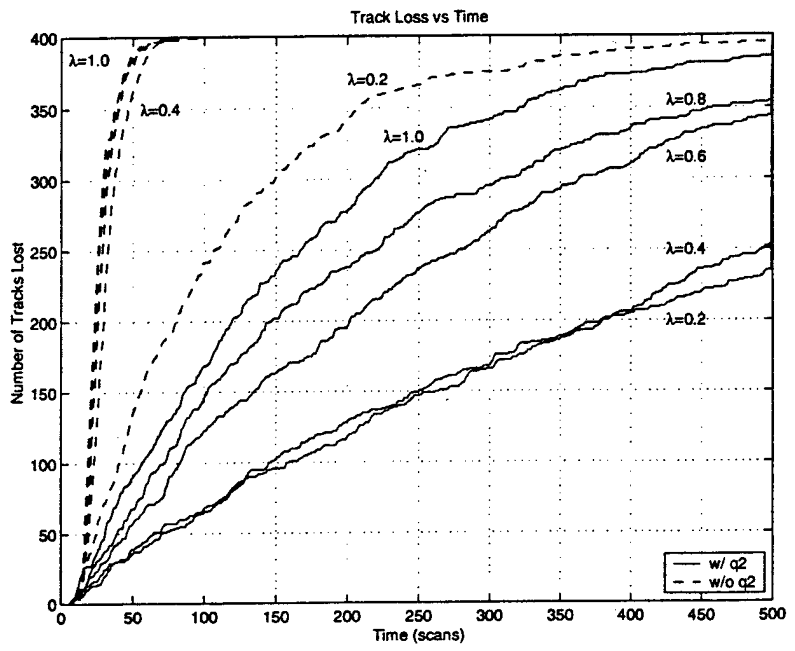


Figure 7: *Combined track loss for both targets with and without the q_2 term.*

Figure 7 shows the cumulative number of tracks lost at each time scan over the 200 Monte Carlo runs at each clutter density level. Note that track lifetime is dramatically improved by the use of the loss of information term. From [4, 10], the probability of track loss is a function of the number of gated measurements and thus a function of the prediction covariance and the clutter density. Since this probability grows with the number of gated measurements (and thus with the size of the state estimate covariance) the poorer track loss performance of the tracking system without the q_2 term is a result of the large state estimate covariances produced by overestimating the effectiveness of each sensor. Note that in all cases, track loss occurs even though the average number of sensors is less than the maximum (8) and in the case of the q_2-based algorithm, both the predicted and actual covariances generally meet the desired covariance goals. Thus control of the state estimate covariance does not guarantee the elimination of track loss.

8 Implications for the Selection of P_d

The strength of the covariance control approach is that a target-specific covariance goal can be set independently of other parameters in the tracking task. This goal can be the reduction of the state estimate covariance to accurately fire a weapon at a target or to avoid confusing the identity of closely spaced targets. In these examples, the desired covariance is determined by external factors (missile lock requirements or target spacing). When these external factors are missing, the selection of P_d is less obvious. While ultimately the covariance of each target should be as small as possible, limited resources require that some covariances will be larger than others. Obviously for lower priority targets the desired covariance should be "large" to avoid taking valuable sensing resources from higher priority objects. When data association is not required, this covariance can be set arbitrarily high. In the presence of clutter measurements, however, a large covariance increases the gate size and ultimately leads to a high probability of track loss. This drastically reduces the freedom in choosing P_d.

Additionally, the variation in sensor abilities in different dimensions can complicate the derivation of a realistic and effective desired covariance. States that are not directly measured (the velocity states $\dot{x}$ and $\dot{y}$ in the simulations above) are relatively insensitive to the use of sensors, since most of the sensing information goes to those states that are directly sensed. Furthermore, correlations between states can impose additional constraints on the actual covariance that are more demanding than the desired covariance goal.

In these simulations, tracking performance for both targets is roughly the same, in spite of the difference in the desired velocity variance for each. This is mostly because the desired velocity variances are set much higher than those generally reached by the actual state estimate covariance. While the desired velocity variances were 1.0 and 0.5 for targets 1 and 2, respectively, the actual average velocity variance was much closer to 0.1 for the tracking system that used the q_2 term. They were much higher in the system that did not use the loss of information term, but the size of the position variance differences in those cases was generally as large or larger than the difference in the desired velocity variances. Thus because the desired velocity variances were set higher than the velocity variances indirectly imposed by the desired position variance settings, the overall desired covariance goal was essentially the same for both targets.

If the desired velocity variance is set lower, to 0.03 for example, then the number of sensors used will increase dramatically. Since most of the information from the sensors will go to the position estimate, the actual position variances will be $1-2$ orders of magnitude smaller than the desired value of 0.02, while the velocity variances will be just under 0.03. Thus the setting of an aggressive velocity variance limit results in a much more demanding *de facto* limitation on the position variance. However, unless external factors require such a precise estimate of the velocities, it is unwise to set the velocity variances so low, since the sensors can not directly affect those values. Even though the velocity variance does contribute to the size of the gated volume for each sensor and thus can increase the number of gated measurements, it is more efficient to limit the gated volume by reducing the size of the position variances, which are more sensitive to the use of sensors than those of the velocity.

While a rigorous derivation of efficient desired covariance goals remains an unsolved problem, general constraints can be used to approximately calculate an efficient desired covariance. Obviously the shape of the desired covariance should be small in the states that are directly measured to reduce the effects of data association, while large in states that are not measured to avoid unnecessarily high sensing demands. The desired covariance should also be scalable to reflect a range of target priorities. The update equation of the information filter in eq. (11) can be used to calculate a rough desired covariance in the absence of external goals.

$$\begin{aligned} P_{d_0}^{-1} &= \epsilon I + \sum_{j=1}^{N_s} c_j H_j' R_j^{-1} H_j \\ P_d &= \alpha P_{d_0} \end{aligned} \tag{24}$$

where $0 < c_j < 1$ is the relative number of targets that sensor j can track compared to the overall tracking capacity of the system. The sum $\sum_{j=1}^{N_s} c_j H_j' R_j^{-1} H_j$ reflects the information about each state that is measured by the sensors.

The factor c_j provides an estimate of the availability of each sensor during a given scan – the more capacity a sensor has the more likely it will be available for a sensing assignment at any given moment. Since this sum generally results in a singular matrix, the scaled identity matrix ϵI is added to insure that P_{d_0} is positive definite. By setting $0 < \epsilon << 1$, the portions of the desired covariance corresponding to states that are not directly measured by the sensors are set to a large value to avoid expending resources on them. P_{d_0} can then be multiplied by the scalar factor α to vary the size of the covariance to reflect the priority of each target; α will be small for high priority targets and large for low priority targets.

9 Conclusion

While single-sensor management systems have been designed to work with the Probabilistic Data Association Filter, no sensor managers exist for multisensor extensions of this tracking algorithm. In particular, the use of the PDAF results in a loss of effectiveness for each sensor due to the uncertainty in origin of each measurement. While the effect of this uncertainty on the state estimate covariance depends on the actual received measurements, and is thus generally stochastic, it can be approximated by the addition of a scalar loss of information term to the standard Kalman filter covariance update equation. One multi-sensor management approach is covariance control, which attempts to maintain a desired state estimate covariance goal for each target. This paper augments an existing covariance control technique through the addition of the loss of information term, allowing the algorithm to evaluate the reduced effectiveness of each sensor. Monte Carlo simulations show that without this term, the covariance control system is unable to maintain the desired covariance, resulting in a much larger actual covariance level. This in turn leads to a much higher rate of track loss. Use of the loss of information term restores this performance, allowing the control system to generally achieve the desired covariance goal and reducing the overall rate of track loss.

A comprehensive approach to the selection of efficient covariance goals remains an open question. In the absence of externally set covariance goals, the desired covariance should be small in directions that are directly measured to avoid a high rate of track loss. In directions that are not measured by the sensors, the desired covariance should be large to avoid wasting sensing resources.

Acknowledgments

This work is supported in part by an Office of Naval Research Young Investigator Award (Grant N00014-97-1-0642), the Data Fusion Corporation, and the Colorado Advanced Software Institute.

References

[1] S. S. Ahmeda I. Harrison, and M. S. Woolfson, "Tracking Multiple Targets with a Phased Array Radar," *Proc. of Radar Systems*, Edinburgh, UK, pp. 601-605, 1997.

[2] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, Academic Press, Inc., San Diego, 1988.

[3] Y. Bar-Shalom and X. Li, *Estimation and Tracking: Principles, Techniques, and Software*, Artech House, Boston, 1993.

[4] T. E. Fortmann, Y. Bar-Shalom, M. Scheffe, and S. Gelfand, "Detection Thresholds for Tracking in Clutter – A Connection between Estimation and Signal Processing," *IEEE Trans. on Automatic Control*, vol. AC-30, no. 3, pp. 221-228, March 1985.

[5] C. W. Frei, "A Comparison of Parallel and Sequential Implementations of a Multisensor Multitarget Tracking Algorithm," Masters Thesis, Northwestern University, 1995.

[6] C. W. Frei and L. Y. Pao, "Alternatives to Monte Carlo Simulation Evaluations of Two Multisensor Fusion Algorithms," *Automatica*, vol. 34, no. 1, pp. 103-110, Jan. 1998.

[7] M. Kalandros and L. Y. Pao, "Controlling Target Estimate Covariance in Centralized Multisensor Systems," *Proc. American Control Conf.*, Philadelphia, PA, pp. 2749–2753, 1998.

[8] M. Kalandros, L.Y. Pao, and Y.C. Ho. "Randomization and Super-Heuristics in Choosing Sensor Sets for Target Tracking Applications", *Proc. IEEE Conference on Decision and Control*, Phoenix, AZ, pp. 1803-1808, Dec. 1999.

[9] D. J. Kershaw and R. J. Evans, "Waveform Selective Probabilistic Data Association," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 33, pp. 1180-1188, Oct. 1997.

[10] X. R. Li and Y. Bar-Shalom, "Stability Evaluation and Track Life of the PDAF for Tracking in Clutter," *IEEE Trans. on Automatic Control*, vol. 36, no. 5, May 1991.

[11] S. Musick and R. Malhotra, "Chasing the Elusive Sensor Manager," *Proceedings of the IEEE 1994 NAECON*, Dayton, OH, vol. 1, pp. 606-613, 1994.

[12] L. Y. Pao "Centralized Multisensor Fusion Algorithms for Tracking Applications," *Control Engineering Practice*, vol. 2, no. 5, pp. 875-887, Oct. 1994.

[13] L. Y. Pao and M. K. Kalandros, "The Effects of Delayed Sensor Requests on Sensor Manager Systems," *Proc. AIAA Guidance, Navigation, and Control Conf.*, Boston, MA, pp.1127-1135, Aug. 1998.

[14] J. Nash, "Optimal Allocation of Tracking Resources," *Proceedings of the 1977 IEEE Conference on Decision and Control*, New Orleans, LA, vol. 1, pp. 1177-1180, 1977.

[15] W. Schmaedeke, "Information-based Sensor Management," *SPIE Proceedings*, vol. 1955, April 1993.

[16] D. Willner, C. B. Chang, and K. P. Dunn, "Kalman Filter Algorithms for a Multi-Sensor System," *Proceedings of the 1976 IEEE Conference on Decision and Control*, 1976.

Submitted to

IEEE Transactions on Aerospace and Electronic Systems

Covariance Control for Multisensor Systems

MICHAEL KALANDROS and LUCY Y. PAO
Department of Electrical and Computer Engineering
University of Colorado
Boulder, CO 80309-0425
kalandro@colorado.edu and pao@colorado.edu

Abstract

The increased use of multisensor surveillance systems has provided superior tracking performance at the cost of increased computational demand. To mitigate this demand, sensor managers, which generate and prioritize the actions of those sensors, have been developed that extremize a function of target covariance, priority, and threat level. However, it is difficult to impose a target-specific covariance goal. This paper presents a method of controlling the individual covariance of each target estimate.

1 Introduction

The application of multisensor fusion to surveillance systems has provided superior tracking performance at the cost of increased computational demand. As the number of targets and sensors increases, tracking systems can very quickly become overloaded by the incoming data. Furthermore, as the number of available sensors and sensor modes increases, it is easy to overwhelm human operators, such as fighter pilots, as well. What is needed is an automated method of balancing tracking performance with system resources. Such a system is known as a sensor manager.

Sensor managers are a general class of systems that generate sensing actions, then prioritize and schedule those actions [5]. Sensing actions can include the tasking of sensors to illuminate a target, the selection of sensor modes, or scanning an area for unknown targets. These actions will be selected to achieve various goals such as maintaining a target track, optimizing the chance of detecting new targets, identifying detected targets, and minimizing electromagnetic emissions to reduce the chance of detection by the enemy. Because of finite sensing and computational capacity as well as strict time constraints, the sensing tasks must be scheduled. The priority of a sensor action depends on the threat level of an individual target, as well as other less well-defined situational awareness issues.

Sensor manager systems can be roughly divided between two categories, descriptive and normative. Descriptive systems are advantageous when objective information is absent. Approaches in this category include fuzzy reasoning, evidential reasoning, and expert systems [7]. Normative systems use objective performance metrics to guide sensing behavior. These metrics, in turn, provide a ready-made method of comparing the performance of competing approaches. Such approaches typically use utility, decision, or information theoretic metrics [5, 7].

While some authors (e.g. [7]) recommend a unified approach to the sensor management problem, the disparate tasks of the sensor manager (target tracking, target detection, target identification, etc.) demand individual study before they can be managed as a group. This paper will focus entirely on sensor management as applied to target tracking. As such, the tracking task provides objective information that can be easily used in a normative system: the target estimate covariance.

To date, most sensor management techniques have treated this as an optimization problem, where the goal is to apply combinations of sensors to each target to minimize a cost function generated using target priority, threat level, and the covariance of each target state estimate [6]. A variation of this is to maximize a cost functional based on the increase in state information from each sensor combination [8]. In [3], a sequence of navigational measurements for U.S. Navy ballistic missile submarines are scheduled using a cost function based on the covariance of the submarines' current position estimate and the risk of enemy exposure involved in making those measurements. Additionally, neural nets have been applied to the sensor management tracking problem with mixed results [5].

A drawback of these approaches is that it is difficult to impose a target-specific covariance goal, such as reducing the covariance of a target estimate to accurately fire a weapon. While coarse control of the covariance can be achieved by adjusting the priority of a target, it is not clear what priority level will achieve the desired covariance. Underestimating the required priority will result in failure to achieve the covariance goal, while overestimating the required priority will have a deleterious effect on the covariance of other targets. Attempting to achieve covariance goals by manipulating the priority of multiple targets at the same time only intensifies the problem. A more effective method would be to attempt to minimize a cost functional based on the difference between the desired covariance and actual covariance values. The methods described in [6] and [8] rely on the positive definiteness of the covariance matrices, an assumption that would be routinely violated when using the covariance difference metric. Both methods use the determinant of the covariance in their metrics. While the absolute value of the determinant would eliminate some of the problems, an eigenvalue at zero (meaning that the covariance goal has been met in some direction) would result in a determinant equal to zero, which could mask a large error in another direction. Thus a new sensor allocation algorithm is needed to maintain specific covariance goals.

This paper explores the use of different metrics for the evaluation of the error between the desired and actual covariance in addition to proposing a new architecture for the sensor manager system. The system we are proposing separates the system into a covariance controller and a sensor scheduler. The covariance controller can assign sensor combinations to each target to meet a desired covariance level. The scheduler prioritizes sensing actions and executes them as time allows. Low priority actions may be delayed until future scans or may be dropped altogether.

In this paper, the sensor scheduler is relegated to a "black box" without specifying its operations. This convention allows the use of various scheduling practices such as those presented in [7] or [10]. As mentioned above, one of the expected effects of the separate sensor scheduler is the delay of the execution of sensing requests. This arises due to scheduling delays and the limited computational resources of the tracking system. Because of this, not all sensor requests can be executed in a single sampling period, causing sensor requests to accumulate in the command queue. This results in future requests being delayed as well.

This article is arranged as follows. Section 2 details our approach to the sensor management problems. It includes a review of the equations of the Kalman filter and the motivation behind our sensor selection algorithms. The system architecture and the actual sensor selection algorithms follow in Sections 3 and 4. Selected simulation results for these algorithms are presented in Section 5. The effects of sensor request delay are presented in Sections 6 and 7. Sections 8 and 9 quantify the effects of sensor request delay for the scalar Kalman filters in the continuous and discrete time domains, respectively. Finally, Section 10 summarizes our conclusions based on this work.

2 Preliminaries

2.1 Kalman Filter

The sequential Kalman filter is an algorithm for combining multiple inputs from stochastic or slightly non-linear systems to form an estimate in a state space representation. The Kalman filter is based on the following assumptions about the target and measurement systems [1, 2]:

$$\begin{aligned} x(k) &= Fx(k-1) + Gu(k-1) + w(k-1) && (1) \\ z_j(k) &= H_j x(k) + v_j(k), \quad j = 1, \ldots, N_s && (2) \end{aligned}$$

where $x(k)$ is the current state of the target; F, G, H_j are known system matrices; $u(k)$ is the control signal; $z_j(k)$ is a measurement of the target from sensor j; and there are N_s sensors. $w(k)$ is a variable representing process noise or higher-order motion not modeled by F, and $v_j(k)$ is a variable representing measurement noise in sensor j. Both $w(k)$ and $v_j(k)$ are assumed to have zero-mean, white, Gaussian probability distributions.

Since $w(k)$ and $v_j(k)$ are zero-mean noise processes, the target states and measurements in the next time interval can be predicted by

$$\begin{aligned} \hat{x}(k \mid k-1) &= F\hat{x}(k-1 \mid k-1) + Gu(k-1) && (3) \\ \hat{z}_j(k) &= H_j\hat{x}(k \mid k-1) && (4) \end{aligned}$$

The input $u(k)$ is considered known and will be omitted in future equations because it can be easily reinserted. The quantity $\nu_j(k) = \hat{z}_j(k) - z_j(k)$ is known as the innovation. The prediction covariance of the state and innovation can be found by

$$\begin{aligned} P(k \mid k-1) &= FP(k-1 \mid k-1)F' + Q(k-1) && (5) \\ S_j(k) &= H_jP(k \mid k-1)H_j' + R_j(k) && (6) \end{aligned}$$

respectively, where $Q(k)$ is the process noise covariance and $R_j(k)$ is the measurement noise covariance.

With N_s sensors, there are 2^{N_s} possible combinations or subsets of those sensors that can be used by the covariance controller. The ith possible subset is defined as Φ_i where N_{s_i} is the number of sensors in that combination. For a given i, the sequential algorithm runs a separate Kalman filter for each sensor, propagating its estimate to the next filter [9]:

$$\begin{aligned} \hat{x}_1(k \mid k) &= \hat{x}(k \mid k-1) + K_1(k)\big(z_1(k) - H_1\hat{x}(k \mid k-1)\big) \\ \hat{x}_j(k \mid k) &= \hat{x}_{j-1}(k \mid k) + K_j(k)\big(z_j(k) - H_j\hat{x}_{j-1}(k \mid k)\big), \quad j\epsilon\Phi_i \\ \hat{x}(k \mid k) &= \hat{x}_{N_{s_i}}(k \mid k) && (7) \end{aligned}$$

where

$$\begin{aligned} K_1(k) &= P(k \mid k-1)H_1'S_1^{-1}(k) \\ K_j(k) &= P_{j-1}(k \mid k)H_j'S_j^{-1}(k), \quad j\epsilon\Phi_i && (8) \end{aligned}$$

The state covariance is updated for each filter by

$$\begin{aligned} P_1(k \mid k) &= (I - K_1(k)H_1)P(k \mid k-1) \\ P_j(k \mid k) &= \big(I - K_j(k)H_j\big)P_{j-1}(k \mid k), \quad j\epsilon\Phi_i \\ P(k \mid k) &= P_{N_{s_i}}(k \mid k) && (9) \end{aligned}$$

Once the state and covariance estimates have been updated, they are fed back into the algorithm and the entire process is repeated for the new set of measurements at the next time step. Alternatively, the covariance update can be calculated in a single step using the inverses of the covariance matrices [2]:

$$P^{-1}(k \mid k) \;=\; P^{-1}(k \mid k-1) + \sum_{j \epsilon \Phi_i} H_j' R_j^{-1} H_j \tag{10}$$

Since the sum $\sum_{j \epsilon \Phi_i} H_j' R_j^{-1} H_j$ is used frequently, we shall define it as the sensor information gain:

$$J_i \;=\; \sum_{j \epsilon \Phi_i} H_j' R_j^{-1} H_j, \qquad i = 1, \ldots, 2^{N_s} \tag{11}$$

where J_i is the sensor information gain for the ith combination of sensors.

2.2 Sensor Selection

The sensor selection can be determined based on the difference between the inverses of the predicted covariance in eq. (5) and the desired covariance $P_d(k)$. Replacing the updated covariance matrix in eq. (10) with the desired covariance and solving for the necessary sensor information gain, we see that we want J_i to equal ΔP, where

$$\Delta P \;=\; P_d^{-1}(k) - P^{-1}(k \mid k-1) \tag{12}$$

To achieve the desired covariance, the sensor information gain will ideally equal the difference between the inverses of the actual and desired covariance matrices. Generally, none of the sensor information gains will exactly equal this difference; thus $J_i - \Delta P$ will typically *not* be zero for any i. An algorithm is needed to select a set of sensors that will make $J_i - \Delta P$ as "small" as possible, allowing the desired covariance to be reasonably well approximated.

While J_i will always be positive definite, the same can not be said for ΔP. However, generally the desired covariance will be smaller than the predicted covariance (requiring the use of sensors) so for the purpose of this discussion we will assume that ΔP is at least positive semi-definite.

With the above assumption, both J_i and ΔP can be represented as ellipsoids where the square-root of the eigenvalues are the half lengths of the major and various minor axes and the eigenvectors indicate the direction of those axes. The two ellipsoids shown in Figure 1a are the same size, but J_i really does not meet the requirements of ΔP. Ideally, the ellipsoid formed by J_i should completely enclose the ellipsoid formed by ΔP (Figure 1b) which ensures that the actual covariance, P, is within the desired covariance, P_d. However, if ΔP is much smaller than J_i, then too much of the sensor resources are probably being applied to that target.

One method is to compare the largest eigenvalue of ΔP with the smallest eigenvalue of J_i. Obviously, this eliminates many candidate sensor combinations that might otherwise work (including the example in Figure 1b). This technique is far too restrictive, forcing too many sensor resources to be applied to that target.

Another method is to multiply each unit eigenvector, $\nu_{\Delta P}$, of ΔP by the square root of its eigenvalue, $\lambda_{\Delta P}$, forming ν_{axis}, an axis vector of the ellipsoid. Then project each axis vector onto each eigenvector of J_i:

$$\nu_{\text{axis}} \;=\; \sqrt{\lambda_{\Delta P}}\,\nu_{\Delta P} \tag{13}$$

$$\sqrt{\lambda_{J_i}} \;\geq\; \nu_{\text{axis}} \cdot \nu_{J_i} \tag{14}$$

where λ_{J_i} and $\lambda_{\Delta P}$ are the eigenvalues of J_i and ΔP, respectively, and ν_{J_i} and $\nu_{\Delta P}$ are the eigenvectors.

The algorithm will reject those sensor combinations that have any projections that exceed the square roots of the eigenvalues of J_i (Figure 2a), violating constraint (14). Of those sensor combinations that are not rejected, the one

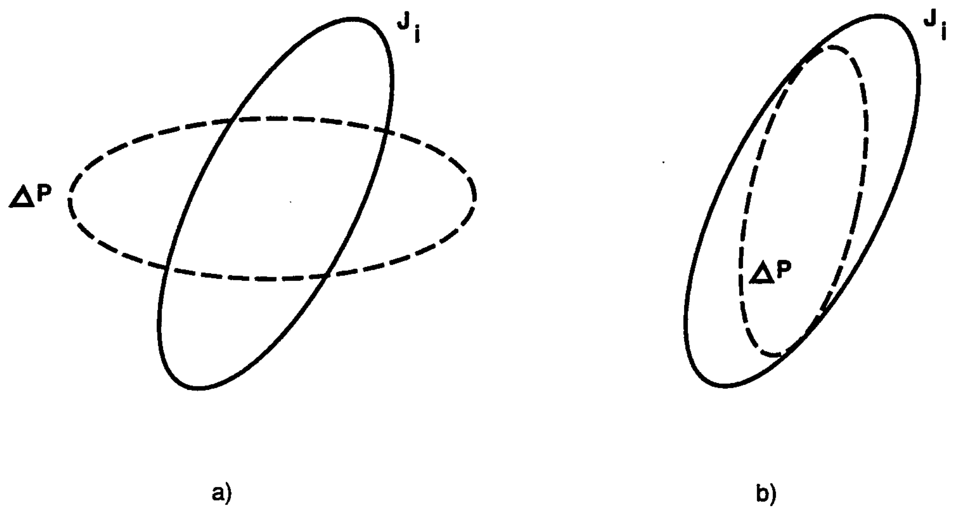


Figure 1: a) These ellipses are the same size, but represent very different covariance matrices. b) The ellipse formed by J_i encloses that formed by ΔP.

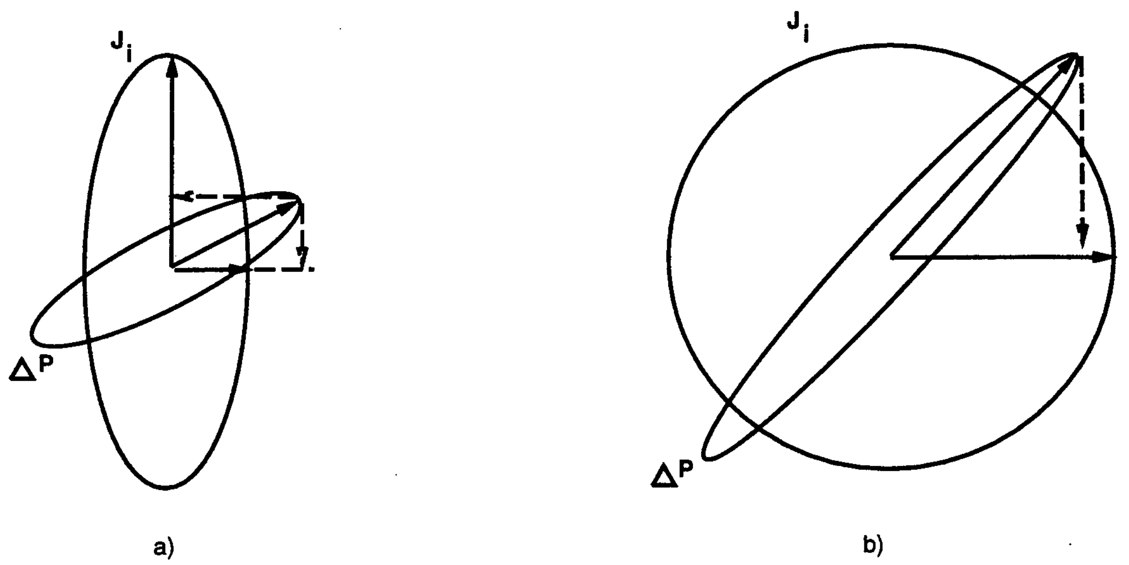


Figure 2: a) Projecting the eigenvectors of the ΔP matrix onto the those of J_i indicates if that ellipse exceeds the other. b) The projection method fails in this case.

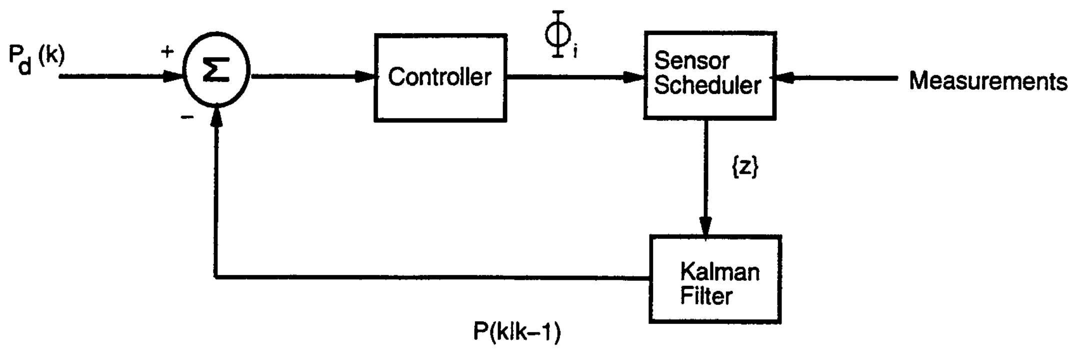


Figure 3: Block Diagram of a Tracking System with Covariance Controller

with the fewest sensors will be used. However, this technique does not ensure enclosure of ΔP by J_i, as shown in Figure 2b.

A method that guarantees enclosure and is not overly restrictive is to represent the two ellipsoids in polar coordinates. Select a set of angles (closely spaced to avoid missing the ends of long, thin ellipsoids). Then compare the magnitude of the ellipsoid formed by ΔP with that formed by J_i at each angle. If it is bigger, then sensor combination i is rejected. Of course, the number of angles required increases exponentially with the dimension of the covariance matrices, making this one of the most computationally demanding of the comparison methods.

A less demanding approach that also ensures enclosure of ΔP is to examine the difference $J_i - \Delta P$. If the difference is positive semi-definite, then the ellipsoid formed by ΔP is enclosed by that of J_i. This seems to be the best available compromise between reliability and computational demands. This method is therefore the basis of one of the actual sensor selection algorithms described later in this paper.

3 Architecture

Figure 3 shows the block diagram of the proposed tracking system. The Kalman filter can be thought of as the plant while the sensor scheduler acts as a system delay. Control of the covariance of the system is implemented via a sensor selection algorithm. The sensor selection is determined based on the difference between the predicted covariance for the next sampling period and the desired covariance. Note that the only input to the controller is the difference between the desired and actual target estimate covariances. Alternatively, the desired and actual target estimate covariances can be replaced by their inverses, resulting in ΔP being the input into the controller. Actual target tracking and estimation are performed by the Kalman filter. The controller's job is to regulate the use of sensing resources by the Kalman filter to reduce the computational load on the tracking system.

Because the controller is separate from the Kalman filter, it can run at a slower speed than the Kalman filter, allowing several iterations of the tracking algorithm to be performed before a new sensor combination is considered by the controller. This can be done by predicting the covariance at the next controller sampling period using the

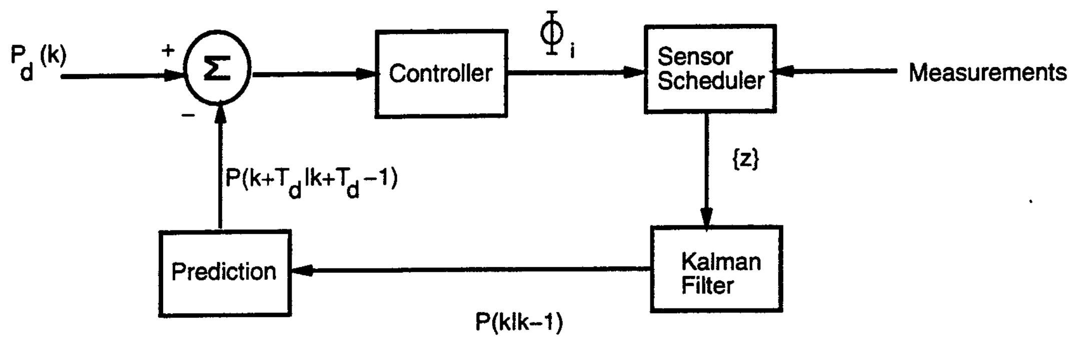


Figure 4: Block Diagram of a Tracking System with Covariance Controller and Prediction

architecture shown in Figure 4. In such a system, one sampling period of the controller will correspond to multiple sampling periods of the Kalman filter.

4 Sensor Selection Algorithms

The first sensor selection algorithm requires that the sensors used produce an updated covariance that is within the desired covariance at all times. This will result in the difference, $P_d - P_i$, where P_i is the updated covariance using sensor combination i, having all positive eigenvalues (as well as the difference $J_i - \Delta P$). While adding sensors will eventually achieve this goal, the computational demand will increase linearly with the number of sensors. Since the goal is also to reduce the computational load on the tracking system, the sensor combination with the fewest number of sensors that produces all positive eigenvalues in the covariance error should be used at each scan. We shall call this the Eigenvalue/Minimum Sensors Algorithm.

To reduce the on-line computational load on the system, the algorithm can be divided into on-line and off-line components. The off-line component precalculates J_i for each sensor combination. The use of the precalculated information gains and eq. (10) instead of eq. (9) reduces the on-line computational demand by eliminating the calculation of matrix inverses during the Kalman gain calculation for each sensor (see eq. (8)). Instead, only the inverse of the predicted covariance must be computed each scan. The on-line component calculates $J_i - \Delta P$ for each i and selects those which are positive definite. This ensures that the updated covariance matrix will be within the desired covariance limits. Of those combinations that meet this criteria, the one with the fewest number of sensors is selected. This criteria could easily be extended to a cost functional approach when using sensors that have varying computational demands associated with their use. Similarly, the sensors could also be weighted by their electromagnetic output or by the risk involved with their use (e.g. as proposed in [3]). Use of these metrics would allow the maintaining of desired covariance goals while minimizing the exposure to enemy forces.

Another method of rationing sensor resources is to view positive eigenvalues in the covariance error as excess resources applied to a target and negative eigenvalues as too little resources applied to that target. As such, the goal of the sensor selection algorithm should be to minimize the norm of the covariance error, $P_d - P(k|k)$. This is the Matrix Norm Algorithm. This technique does not guarantee that the resulting covariance will be within the desired covariance limits since the algorithm does not take the sign of the eigenvalues into account. One major drawback to this approach is that the norm of the inverse covariance error used in the Eigenvalue/Minimum Sensors Algorithm, $J_i - \Delta P$, can not be used in the place of $P_d - P(k|k)$ (since $P^{-1}(k|k) - P_d^{-1} \neq (P(k|k) - P_d)^{-1}$), precluding the use of eq. (10) and the precalculated library of J_i's to reduce the computational complexity of the algorithm.

A third technique, the Norm/Sensors algorithm, relaxes the requirements of the Matrix Norm technique, allowing the norm of the covariance difference to vary within a predefined boundary $\pm\delta$, selecting the sensor combination that uses the fewest sensors while keeping the covariance within that boundary.

5 Simulation Results

The following figures are the results of computer simulations of the three covariance control algorithms for multi-sensor tracking systems. A "dumb" system that simply always uses all its sensor resources is also included for a performance comparison. Of the simulations that have been performed, we have chosen to present a few cases which best showcase the distinctions between the various systems. Each system uses three sensors that measure the position states x and y with different noise covariance values in the x and y directions. Sensor 1 has a measurement noise covariance of 1 and 0.05 in the x and y directions, respectively. Sensor 2 has a noise covariance of 0.05 and 1. Sensor 3 has a noise covariance of 0.22 in both directions (all sensor noise covariance matrices are diagonal). Hence, Sensor 1 is very accurate in the y direction, Sensor 2 is very accurate in the x direction, and Sensor 3 is moderately accurate in both directions. However, overall, the sensors are approximately equally accurate in that the determinants of the noise matrices for the sensors are about equal.

A single object nominally moving in the positive y direction of an $x-y$ space is tracked using a sequential Kalman filter. The target state consists of $[x, \dot{x}, y, \dot{y}]^T$. Its motion is corrupted by a zero-mean, white, Gaussian noise with a covariance of $0.12I$ (I = identity matrix). A desired estimate covariance, P_d, is defined and follows a step pattern, starting as a diagonal matrix with eigenvalues $[0.2, 0.3, 0.2, 0.3]$ at scan 0 and decreasing to a matrix with eigenvalues $[0.05, 0.25, 0.13, 0.22]$ at scan 25. The boundary size δ for the Norm/Sensors algorithm is 0.2.

Figure 5 shows the covariance error using the two metrics described in the proposed algorithms: the smallest eigenvalue of the difference between the desired and actual covariances, and the 2-norm of that difference. Figure 6 shows the number of sensors used per scan – corresponding to the computational work load imposed on each tracking system.

Compare the performance of the "dumb" system to that of the Eigenvalue/Minimum Sensors algorithm. While both systems always meet the desired covariance goal, note that in the first half of the tracking task, the Eigenvalue/Minimum Sensors algorithm uses fewer sensors than the "dumb" system, yet suffers very little loss in covariance error performance. In the second half of the tracking task, both systems use all of the sensor resources to meet the desired covariance. While the "dumb" system wastes sensor resources by using all sensors for each scan, the sensor selection algorithm is able to balance tracking performance goals with system demands, allocating maximum resources only when necessary.

In the first half of the tracking task, the two norm-based algorithms choose the same sensors while in the second half, each algorithm chooses a different sensor combination. In each case, the Norm/Sensors algorithm uses the fewest sensors, while the Eigenvalue/Minimum Sensors algorithm requires the most of all of the sensor selection algorithms. However, except for the "dumb" system, the Eigenvalue/Minimum Sensors algorithm provides the best tracking performance, always selecting sensors so that the covariance is within the desired covariance, hence leading to the smallest RMS errors between the state estimate and the truth. The Norm/Sensors algorithm typically allows the largest covariance. The Matrix Norm algorithm's performance generally falls between the other two techniques.

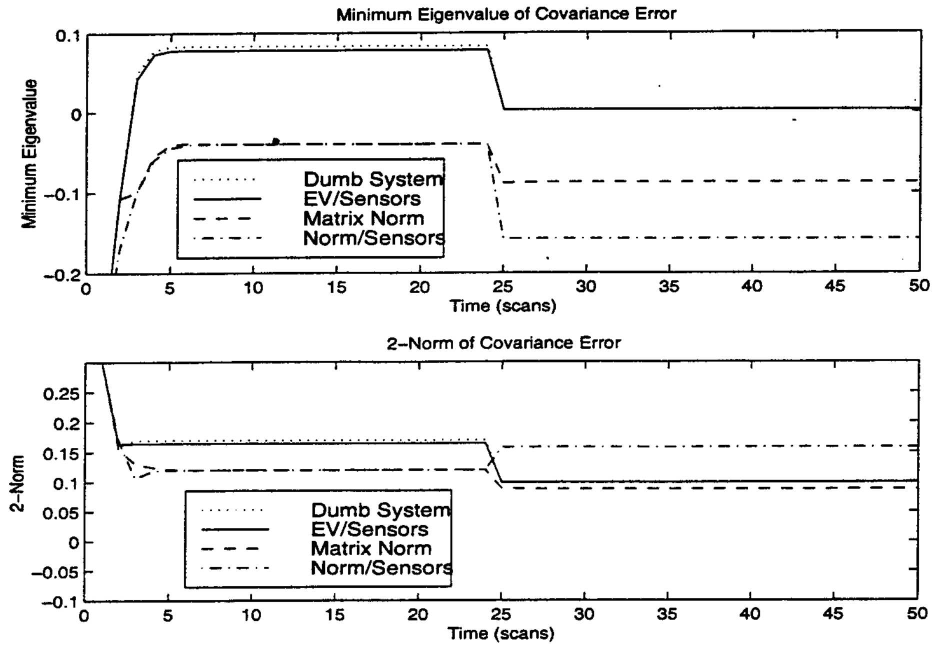


Figure 5: Comparison of the Covariance Tracking Accuracy of Sensor Selection Systems with no Delay

These algorithms represent a continuum of tradeoffs of computational demand versus tracking accuracy. Of the sensor selection algorithms, the Eigenvalue/Minimum Sensors algorithm will, in general, use the most sensor resources, since it has the strictest covariance requirement (the covariance must be less than the desired covariance in all directions). The Matrix Norm may choose fewer sensors, since it does not require the covariance to be within the desired covariance. Finally, the Norm/Sensors algorithm should choose the fewest sensors, but generally allows the largest covariance.

6 The Effect of Sensor Request Delay

Similar to the effects of delay on dynamic control systems, the tracking performance of the sensor selection algorithms when there is delay becomes less stable. The problems that delay causes are due to the fact that the covariance controller makes sensor selections based on the current covariance. For example, Figure 7, shows the evolution of the covariance of a scalar system with time using a single sensor selection. Assume that the controller decides on a sensor selection and executes it at time zero. Then 0.2 seconds later, if the desired covariance P_d changes, the controller selects a different set of sensors based on the difference between P_d and the variance at that time. If the execution of this change is delayed for 0.2 seconds, then since the covariance difference $P_d - P_{k|k-1}$ at $t = 0.4$ is different from the covariance difference at $t = 0.2$, either excessive or insufficient sensor resources have probably been assigned to this tracking task.

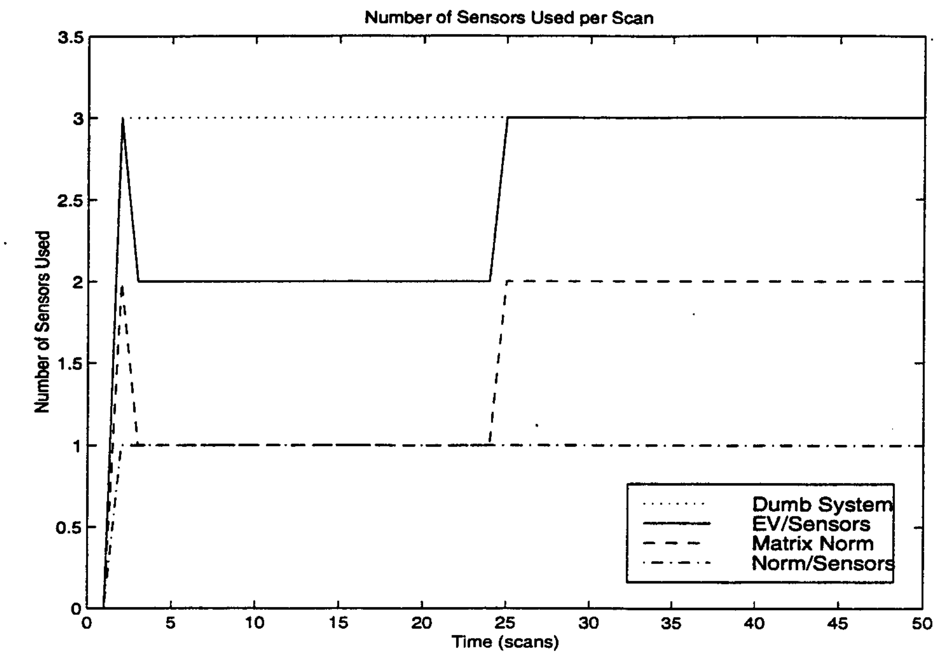


Figure 6: Comparison of the Efficiency of Sensor Selection Systems with no Delay

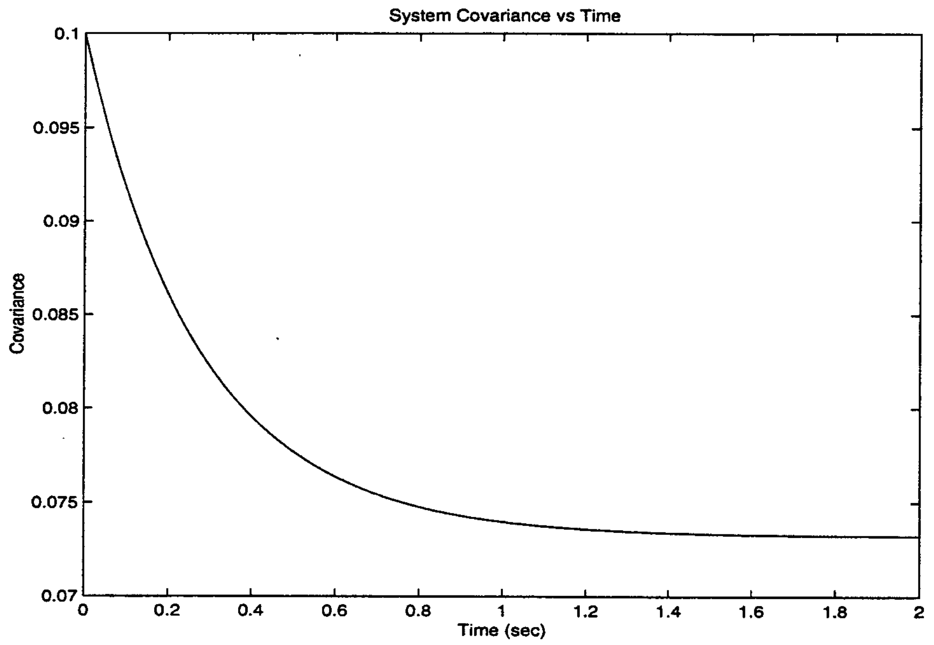


Figure 7: The Covariance of a System Converges Quickly to Its Steady-state Value.

Sensor request delay should not be confused with measurement delay, where the measurements do not represent the present state of the target. In this case further processing of the measurements is needed to account for this. In sensor request delay, the tasking of the sensor lags behind the requests. Once the measurements are made, they represent the current state of the target.

The effects of delay can be ameliorated by predicting the covariance estimate after the delay, allowing the sensor selection algorithm to make its decision based on what the predicted covariance will actually be when the delayed sensor selection is executed. To implement this prediction scheme, we simply iterate the covariance prediction and update equations of the Kalman filter for the projected length of the delay, using the assumed sensor selections for that time period (using the architecture in Figure 4). Correctly assuming the delay and sensor selections will restore most of the performance reduction caused by the delay.

7 Simulation Results for the Effect of Delay

The effect of delay on the system is also simulated, using the Eigenvalue/Minimum Sensors Algorithm. The algorithms and systems simulated are defined as follows:

- Dumb System – a "dumb" system that uses all three sensors throughout the tracking task and has no delay;
- No Delay – a "smart" system running the EV/Minimum Sensors Algorithm;
- 5 Scan Delay – a system running the EV/Minimum Sensors Algorithm, but whose choices are delayed by five scans;
- 5 Scan Delay w/ Prediction – a system with its sensor choices delayed by 5 scans, but that compensates by predicting the correct covariance at the end of that delay.

In all the systems, no sensors are selected initially, and the systems with delay will not make any target measurements for the first 5 scans.

Figures 8 and 9 show a plot of the smallest eigenvalue of the difference between the desired and actual covariances (the various curves have been shifted slightly both vertically and horizontally to improve the readability of the figures). The plot shows the poor performance observed when a delay is added to the sensor selection system and not accounted for in the algorithm – the actual covariance both over- and under- shoots the desired covariance. Notice that the predictive system recovers quickly from the delay-induced errors. Once again, since the "dumb" system uses all three sensors each scan, its covariance is always contained within the desired covariance ellipsoid.

Figure 10 shows the number of sensors used in each scan – again corresponding to the computational work load imposed on each tracking system. The "dumb" system uses the most system resources since it uses all of the sensors throughout the tracking task. The "smart" system without delay and the predictive system use the fewest – increasing the number of sensors only briefly when the desired covariance is reduced. The delayed system without the predictive compensation uses less resources than the "dumb" system, but more than the undelayed or predictive systems. While uncompensated delays can severely degrade system performance, predictive compensation of those delays can restore most of that performance.

8 Analysis of Delay in the Continuous Kalman Filter

Since it is not always possible to accurately model the delay, a more detailed analysis of the effect of delay is needed. In general, there is no closed form solution for the estimate covariance trajectory. The scalar continuous version of

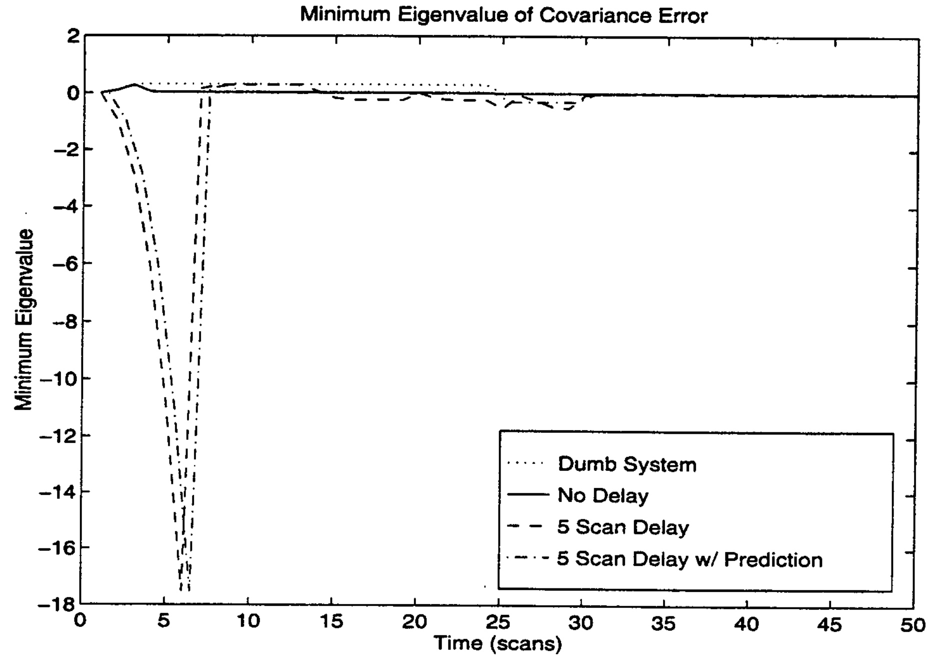


Figure 8: The Effect of Delay on the Covariance Tracking Accuracy of Minimum Sensor Systems

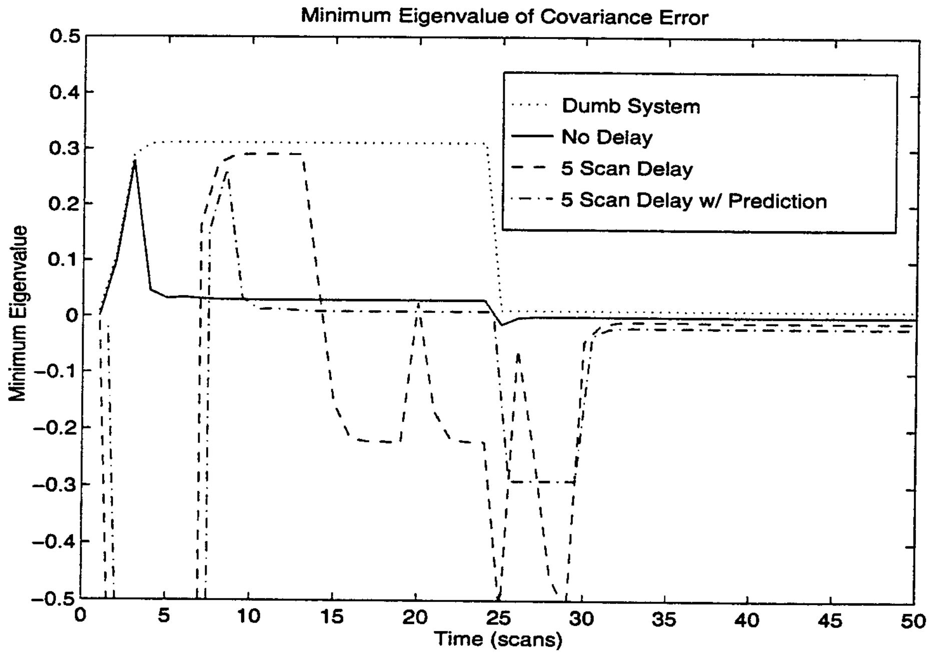


Figure 9: A Close-up of Figure 8

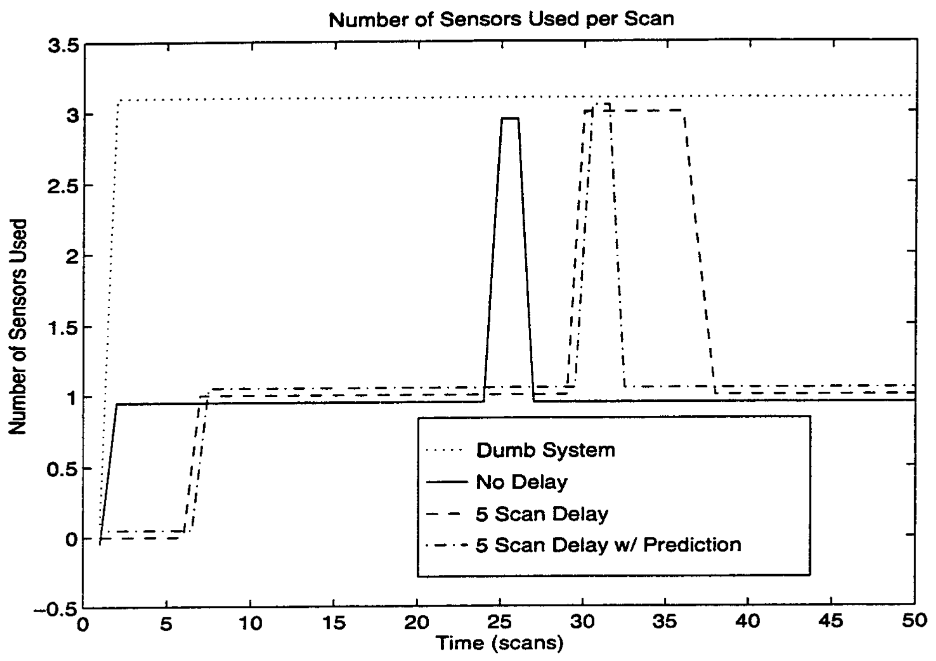


Figure 10: The Effect of Delay on the Efficiency of Minimum Sensor Systems

the Kalman filter (Kalman-Bucy), however, does have a closed form solution. In this section, we provide an analysis based on this closed form solution; and in the next section, we will extend those results to the discrete-time Kalman filter.

We begin with the scalar version of the Kalman-Bucy filter:

$$\dot{x}(t) = -ax(t) + w(t) \tag{15}$$

$$z(t) = x(t) + v(t) \tag{16}$$

where x is the target state variable to be tracked and z is a measurement of that state. Note that when $a > 0$, the nominal system is stable. The state evolves according to a stochastic linear differential equation corrupted by white, zero mean noise $w(t)$ with variance q. The measurement is also corrupted by white, zero mean noise $v(t)$ with variance r. The state estimate is then

$$\dot{\hat{x}}(t) = -a\hat{x}(t) + \frac{p(t)}{r}[z(t) - \hat{x}(t)] \tag{17}$$

where $p(t)$ is the state estimate variance (equivalent to covariance in the scalar case) such that [2]:

$$p(t) = p_1 + \frac{p_1 + p_2}{Ce^{2\alpha t} - 1} \tag{18}$$

$$\alpha = \sqrt{a^2 + \frac{q}{r}} \tag{19}$$

$$\begin{aligned} p_0 &= p(0) \\ p_1 &= r(\alpha - a) \\ p_2 &= r(\alpha + a) \\ C &= \frac{p_0 + p_2}{p_0 - p_1} \end{aligned}$$

Notice that as t goes to infinity, the second term in eq. (18) goes to zero. Thus p_1 is the steady-state value of the state estimate variance. We now define the error in the estimate of the updated state variance due to a delay in sensor request execution as

$$\begin{aligned} \Delta p(t,d) &= p(t) - p(t+d) \\ &= \frac{2r\alpha C e^{2\alpha t}(e^{2\alpha d} - 1)}{(Ce^{2\alpha t} - 1)(Ce^{2\alpha(t+d)} - 1)} \\ &= \frac{2r\alpha C e^{2\alpha t}(e^{2\alpha d} - 1)}{C^2 e^{4\alpha t} e^{2\alpha d} - Ce^{2\alpha t}(e^{2\alpha d} + 1) + 1} \end{aligned} \tag{20}$$

where $t, d \geq 0$. This is a valid assumption since the equation only describes the variance after $t = 0$ and a sensing request can never be executed before it is requested. Divide the numerator and denominator by $e^{4\alpha t}$ to get

$$\Delta p(t,d) = e^{-2\alpha t} \frac{2r\alpha C(e^{2\alpha d} - 1)}{C^2 e^{2\alpha d} - Ce^{-2\alpha t}(e^{2\alpha d} + 1) + e^{-4\alpha t}} \tag{21}$$

It is now easy to see that as t increases, $\Delta p(t,d)$ goes to zero. If the convergence is monotonic, then the sensitivity of the variance estimate to a sensor request delay will always decrease with time. If this is the case, then a lower controller scan rate (compared to the Kalman filter scan rate) will result in a more robust performance of the sensor selection algorithms.

If the convergence to zero is monotonic, the sign of the derivative should not change (if $\Delta p(t,d)$ is negative, it is always increasing to zero; if it is positive, it is always decreasing to zero).

$$\begin{aligned} \frac{\partial}{\partial t}\Delta p(t,d) &= -2\alpha e^{-2\alpha t} \frac{2r\alpha C(e^{2\alpha d} - 1)}{C^2 e^{2\alpha d} - Ce^{-2\alpha t}(e^{2\alpha d} + 1) + e^{-4\alpha t}} \\ &\quad - e^{-2\alpha t} \frac{2r\alpha C(e^{2\alpha d} - 1)[2\alpha C e^{-2\alpha t}(e^{2\alpha d} + 1) - 4\alpha e^{-4\alpha t}]}{(C^2 e^{2\alpha d} - Ce^{-2\alpha t}(e^{2\alpha d} + 1) + e^{-4\alpha t})^2} \\ &= \frac{-4r\alpha^2 e^{-2\alpha t}(e^{2\alpha d} - 1)(C^3 e^{2\alpha d} - Ce^{-4\alpha t})}{(C^2 e^{2\alpha d} - Ce^{-2\alpha t}(e^{2\alpha d} + 1) + e^{-4\alpha t})^2} \end{aligned} \tag{22}$$

Since $e^{2\alpha d}$ is always greater than 1 and the denominator is squared, only the factor $C^3 e^{2\alpha d} - Ce^{-4\alpha t}$ will affect the sign of the derivative. The assumption that $t, d \geq 0$ leads to $|C| > 1$ as a sufficient but not necessary condition for the monotonicity of $\Delta p(t,d)$. This behavior can be seen in Figure 11. A more precise requirement for monotonicity is $C^2 e^{2\alpha d} > 1$ since $e^{-4\alpha t}$ is never larger than 1.

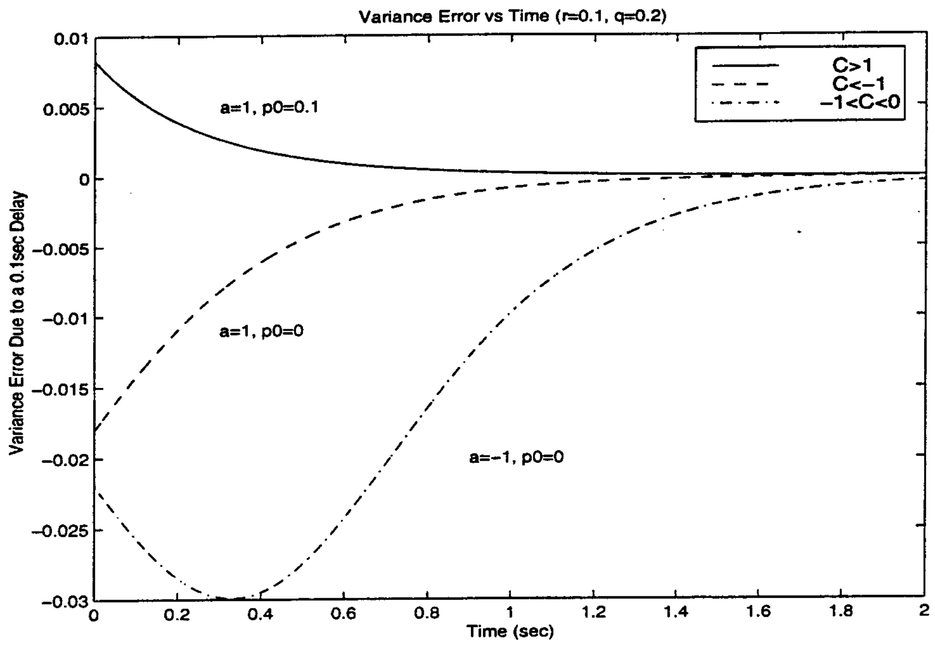


Figure 11: Examples of the qualitative performance of variance evolution with time for various C. For the case where $C > 1$, $p_0 > p_1$ and for both cases where $C < 0$, $p_0 < p_1$.

8.1 Analysis of C

The next question becomes: When is $|C| < 1$?

$$\begin{aligned} C &= \frac{p_0 + p_2}{p_0 - p_1} \\ &= \frac{p_0 + r\alpha + ra}{p_0 - r\alpha + ra} \end{aligned} \tag{23}$$

Observe from eq. (19) that $\alpha \geq |a|$. Then, regardless of the sign of a, the numerator of C is always positive. This in turn means that the sign of C is solely related to the relationship between the initial and final variances of the state estimate. Thus if $p_0 > p_1$, then C is positive and if $p_0 < p_1$, then C is negative.

For the case of $0 < C < 1$, the following must hold

$$\begin{aligned} p_0 + r\alpha + ra &< p_0 - r\alpha + ra \\ r\alpha &< -r\alpha \end{aligned} \tag{24}$$

Since $r\alpha$ is always positive, this condition cannot exist. Therefore, $p_0 > p_1 \Rightarrow C > 0 \Rightarrow C > 1$. Thus, the variance of all scalar systems will converge monotonically when the initial variance is larger than the steady-state variance.

For the case of $-1 < C < 0$, the following must hold

$$p_0 + r\alpha + ra < -(p_0 - r\alpha + ra) \tag{25}$$

$$p_0 < -ra$$

Since p_0 and r are always positive, this implies that a must be less than zero ($x(t)$ is unstable) for this to occur. Therefore, when $p_0 < p_1$, $a > 0 \Rightarrow C < -1$. This, combined with the monotonicity of all systems with $C > 0$ indicates that the variance error due to delay of a stable target will always converge to zero monotonically with time. When $p_0 < p_1$ and $a < 0$, $p_0 > -ra \Rightarrow C < -1$. When $p_0 < -ra$ holds (meaning $(-1 < C < 0)$, then $C^2 e^{2\alpha d} > 1$ is required for monotonicity.

8.2 Prediction of Variance Via Delay Estimation

Our simulation studies have shown that the effects of delay can be almost completely eliminated by predicting the actual variance after the delay. The sensor manager can then select sensor combinations based on $p(t+d)$ rather than $p(t)$. We now look at the effect of errors in the estimate of the delay. Define the variance prediction error due to an error, δ, in the delay estimate as

$$\begin{aligned} \Delta\hat{p}(t,d,\delta) &= p(t+d) - p(t+d+\delta) \\ &= p(t') - p(t'+\delta) \\ &= \Delta p(t',\delta) \end{aligned} \tag{26}$$

Thus a redefinition of variables allows us to use the variance error from before. Note, however, that δ can be positive or negative.

A useful aspect of variance prediction to determine is whether it is better to overestimate or underestimate the actual delay. To examine this, assume a delay estimate error of δ. If $\delta > 0$, the delay has been overestimated; if $\delta < 0$, the delay has been underestimated. Looking at the variance prediction error,

$$\Delta p(t',\delta) = \frac{2r\alpha C e^{2\alpha t'}(e^{2\alpha\delta} - 1)}{(Ce^{2\alpha t'} - 1)(Ce^{2\alpha(t'+\delta)} - 1)} \tag{27}$$

the denominator consists of two factors of the form $Ce^{(\cdot)} - 1$. Since the exponents are always positive, when $C > 1$, each factor is always positive. When $C < 0$, each factor is always negative. Thus the denominator is always positive and does not affect the sign of the variance estimate error. In the numerator, the only factors that can be negative are C and $(e^{2\alpha\delta} - 1)$. Notice that when $\delta < 0$, the factor $(e^{2\alpha\delta} - 1)$ is negative and when $\delta > 0$, the factor is positive. Of course when $\delta = 0$, the variance is predicted exactly and the error is zero. Thus, when $C > 0$, the sign of the variance prediction error is the same as that of δ. When $C < 0$, the sign is opposite that of δ.

Assume for the moment, that $\delta \geq 0$. Hence, when $C > 0$, $\Delta p(t',\delta) \geq 0$; and when $C < 0$, $\Delta p(t',\delta) \leq 0$. Since, all other variables being equal, we can expect the variance error due to δ, $\Delta p(t',\delta)$, to have the opposite sign of $\Delta p(t',-\delta)$, the sum of the two will have the same sign as the larger of the two errors. That sum becomes

$$\begin{aligned} \Delta p(t',\delta) + \Delta p(t',-\delta) &= \frac{2r\alpha C e^{2\alpha t'}(e^{2\alpha\delta} - 1)}{(Ce^{2\alpha t'} - 1)(Ce^{2\alpha(t'+\delta)} - 1)} + \frac{2r\alpha C e^{2\alpha t'}(e^{-2\alpha\delta} - 1)}{(Ce^{2\alpha t'} - 1)(Ce^{2\alpha(t'-\delta)} - 1)} \\ &= \frac{2r\alpha C e^{6\alpha t'}(2 - e^{-2\alpha\delta} - e^{2\alpha\delta})(C^2 - e^{-4\alpha t'})}{(Ce^{2\alpha t'} - 1)(Ce^{2\alpha(t'+\delta)} - 1)(Ce^{2\alpha t'} - 1)(Ce^{2\alpha(t'-\delta)} - 1)} \end{aligned} \tag{28}$$

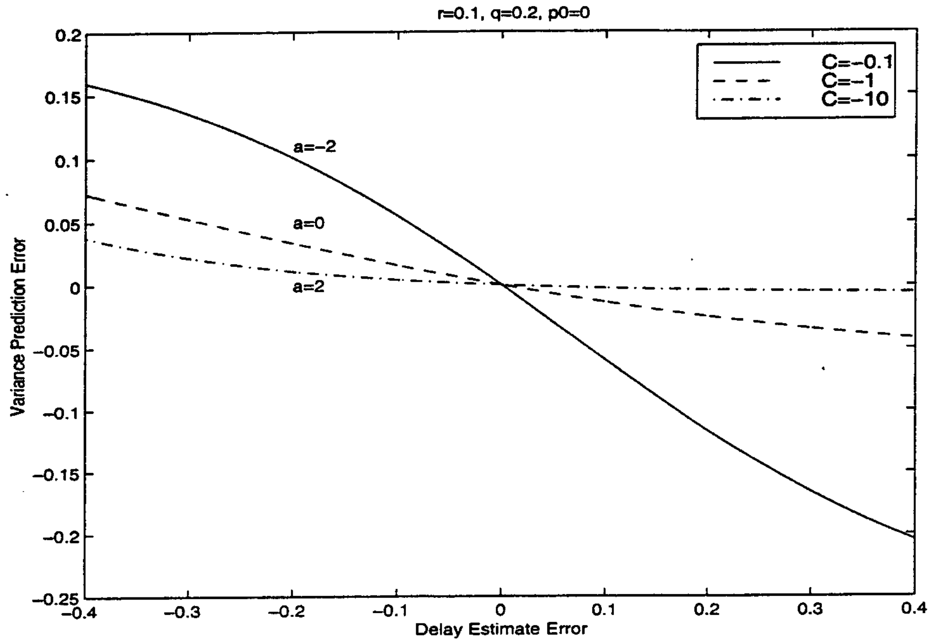


Figure 12: The effect of delay estimation error on variance prediction shows that as $|C|$ increases, the benefits of overestimating the delay increase as well. Here, r, q, and $p0$ are fixed and a is varied to yield different values of C. The actual delay, d, is 0.4 seconds.

Again, the denominator consists of four factors that are all positive or all negative depending on C, so the denominator is always positive. If we look at the factor $(2 - e^{-2\alpha\delta} - e^{2\alpha\delta})$, we find the derivative with respect to δ is

$$\frac{\partial(2 - e^{-2\alpha\delta} - e^{2\alpha\delta})}{\partial\delta} = 2\alpha(e^{-2\alpha\delta} - e^{2\alpha\delta}) \tag{29}$$

For $\delta \geq 0$ this derivative is always negative except at $\delta = 0$, which represents the local maximum. The value of $(2 - e^{-2\alpha\delta} - e^{2\alpha\delta})$ at this point is zero. Thus, this factor is always less than or equal to zero.

The factors left to control the sign of the sum $\Delta p(t', \delta) + \Delta p(t', -\delta)$ are C and $C^2 - e^{-4\alpha t'}$. If $|C| > 1$ then $C^2 - e^{-4\alpha t'}$ is always positive. If $C > 0$ (and thus > 1), then $\Delta p(t', \delta) > 0$, $\Delta p(t', -\delta) < 0$ and the sum in eq. (28) is negative – indicating that the error due to $-\delta$ is greater than the error due to δ. When $C < -e^{-2\alpha t'} < 0$ (and thus $\Delta p(t', \delta) < 0$ and $\Delta p(t', -\delta) > 0$), the sum in eq. (28) is positive, indicating that once again, the error due to $-\delta$ is greater. Since $t' = t + d$, the relation $C < -e^{-2\alpha d}$ is a sufficient condition to ensure the superiority of overestimating the delay. Figure 12 shows the effect of different values of C on $\Delta p(t', \delta) + \Delta p(t', -\delta)$. In this case, when $C = -0.1$, the variance prediction error is actually slightly larger when the delay is overestimated than when it is underestimated, but as $|C|$ increases, it becomes much more advantageous to overestimate the delay. Notice that when $C = -10$, the prediction error is very near zero regardless of how much the delay is overestimated.

9 Extension to Discrete Kalman Filter

Since most tracking systems are discrete time systems, it is desirable to extend these results to the scalar discrete Kalman filter (repeated below with scalar notation for clarity).

$$
\begin{aligned}
x_{k+1} &= f x_k + w_k && (30)\\
z_k &= h x_k + v_k && (31)
\end{aligned}
$$

The state estimate becomes

$$
\begin{aligned}
\hat{x}_{k+1|k} &= f \hat{x}_{k|k} && (32)\\
\hat{x}_{k+1|k+1} &= \hat{x}_{k+1|k} + K_{k+1}(z_{k+1} - h\hat{x}_{k+1|k})
\end{aligned}
$$

The notation $\hat{x}_{k+1|k}$ means "the estimate of x at time $k+1$ given measurements through time k". The system variance is calculated as follows:

$$
\begin{aligned}
p_{k+1|k} &= f^2 p_{k|k} + q && (33)\\
s_{k+1} &= h^2 p_{k+1|k} + r\\
K_{k+1} &= \frac{p_{k+1|k} h}{s_{k+1}}\\
p_{k+1|k+1} &= (1 - K_{k+1} h) p_{k+1|k}
\end{aligned}
$$

where q and r are again the target and measurement noise variances, respectively. The prediction variance, $p_{k+1|k}$ (abbreviated as p_{k+1} below to keep the equations readable), can be described using the following Riccati equation:

$$
\begin{aligned}
p_{k+1} &= \frac{A p_k + qr}{h^2 p_k + r} && (34)\\
A &= f^2 r + h^2 q && (35)
\end{aligned}
$$

Defining a function u_k such that [4]

$$
p_k = \frac{u_{k+1} - r u_k}{h^2 u_k}, \qquad (36)
$$

with initial conditions $u_0 = 1$ and $u_1 = h^2 p_0 + r$, $u_k = 0$ for $k < 0$, and substituting it into eq. (34), yields

$$
\begin{aligned}
\frac{u_{k+2} - r u_{k+1}}{h^2 u_{k+1}} &= \frac{A u_{k+1} + (h^2 qr - Ar) u_k}{h^2 u_{k+1}} && (37)\\
u_{k+2} &= (A + r) u_{k+1} + (h^2 qr - Ar) u_k + \delta_{k+2} + (h^2 p_0 - A)\delta_{k+1} && (38)
\end{aligned}
$$

where the delta functions have been added to satisfy the initial conditions u_0 and u_1. The Z-transform of this equation is

$$U(z) = \frac{z(z+h^2p_0-A)}{z^2-(A+r)z-(h^2qr-Ar)} = \frac{R_1}{1-\lambda_1 z^{-1}} + \frac{R_2}{1-\lambda_2 z^{-1}} \tag{39}$$

where

$$\lambda_1 = \frac{A+r+\sqrt{(A+r)^2+4(h^2qr-Ar)}}{2} = \frac{f^2r+h^2q+r+\sqrt{(f^2r+h^2q+r)^2-4f^2r^2}}{2} \tag{40}$$

$$\lambda_2 = \frac{A+r-\sqrt{(A+r)^2+4(h^2qr-Ar)}}{2} = \frac{f^2r+h^2q+r-\sqrt{(f^2r+h^2q+r)^2-4f^2r^2}}{2} \tag{41}$$

$$R_1 = \frac{\lambda_1+h^2p_0-A}{\lambda_1-\lambda_2} \tag{42}$$

$$R_2 = -\frac{\lambda_2+h^2p_0-A}{\lambda_1-\lambda_2} \tag{43}$$

Taking the inverse Z-Transform gives $u_k = R_1\lambda_1^k + R_2\lambda_2^k$. Substituting this result back into eq. (36),

$$p_k = \frac{R_1\lambda_1^k(\lambda_1-r)+R_2\lambda_2^k(\lambda_2-r)}{h^2(R_1\lambda_1^k+R_2\lambda_2^k)} = \frac{R_1(\lambda_1-r)(\frac{\lambda_1}{\lambda_2})^k+R_2(\lambda_2-r)}{h^2(R_1(\frac{\lambda_1}{\lambda_2})^k+R_2)} \tag{44}$$

Let us take a closer look at λ_1, λ_2, R_1, and R_2. Since we expect the variance to be real, both λ_1 and λ_2 should be real. For this to be true, the expression under the radical should be positive in eqs. (40) and (41).

$$(f^2r+h^2q+r)^2-4f^2r^2 = (f^4r^2-2f^2r^2+2f^2h^2qr+r^2-2h^2qr+h^4q^2)+4h^2qr = (f^2r-r+h^2q)^2+4h^2qr \tag{45}$$

Since the last equation is obviously positive, both λ_1 and λ_2 are real. Furthermore, since $f^2r+h^2q+r>0$ and $f^2r^2 \geq 0$, then $(f^2r+h^2q+r)^2 \geq (f^2r+h^2q+r)^2-4f^2r^2$, thus λ_1 and λ_2 are both positive as well, with $\lambda_1 > \lambda_2$.

To see when R_1 is positive,

$$R_1 = \frac{\lambda_1+h^2p_0-A}{\lambda_1-\lambda_2} > 0 \tag{46}$$

$$h^2p_0 > -\lambda_1 + A \tag{47}$$

$$p_0 > \frac{f^2r-r+h^2q-\sqrt{(f^2r-r+h^2q)^2+4h^2qr}}{2h^2} \tag{48}$$

Since the right-hand term is obviously negative, R_1 is always greater than zero. Evaluating R_2 yields

$$\begin{aligned} R_2 &= -\frac{\lambda_2 + h^2 p_0 - A}{\lambda_1 - \lambda_2} \\ &= -\frac{h^2 p_0 + \frac{r - A - \sqrt{(f^2 r + h^2 q + r)^2 - 4 f^2 r^2}}{2}}{\lambda_1 - \lambda_2} \\ &= -\frac{h^2 (p_0 - p_f)}{\lambda_1 - \lambda_2} \end{aligned} \tag{49}$$

where $p_f = \frac{\lambda_1 - r}{h^2}$ is the final variance (as shown below). It is easy to see that $R_2 > 0$ when $p_f > p_0$ (the variance of the target estimate is increasing) and $R_2 < 0$ when $p_0 > p_f$ (target estimate variance is decreasing). Another useful relationship is

$$\begin{aligned} R_1 + R_2 &= \frac{(\lambda_1 + h^2 p_0 - A) - (\lambda_2 + h^2 p_0 - A)}{\lambda_1 - \lambda_2} \\ &= \frac{\lambda_1 - \lambda_2}{\lambda_1 - \lambda_2} \\ &= 1 \end{aligned} \tag{50}$$

Using this, along with the fact that R_1 is always positive, it is easy to show that $R_1 c + R_2 > 1 > 0$ for all $c > 1$.

Since $\lambda_1 > \lambda_2 > 0$, note that as $k \to \infty$ in eq. (44), $p_k \to \frac{\lambda_1 - r}{h^2} = p_f$. It is then possible to substitute p_f into eq. (44) to get

$$p_k = p_f - \frac{R_2(\lambda_1 - \lambda_2)}{h^2(R_1 \alpha^k + R_2)} \tag{51}$$

where

$$\alpha = \frac{\lambda_1}{\lambda_2} \tag{52}$$

Now again define the variance error due to a delay d as

$$\begin{aligned} \Delta p(k,d) &= p_k - p_{k+d} \\ &= \frac{-\frac{R_1 R_2}{h^2}(\lambda_1 - \lambda_2)\alpha^k(\alpha^d - 1)}{(R_1 \alpha^k + R_2)(R_1 \alpha^{k+d} + R_2)} \end{aligned} \tag{53}$$

It is easy to see that this error goes to zero as $k \to \infty$. Furthermore, setting $d = 1$ provides the slope of the variance, which shows that the variance is monotonic with time (as is expected). The slope of the variance error of the discrete system can be defined as $\Delta p(k,d) - \Delta p(k+1,d)$.

$$\Delta p(k,d) - \Delta p(k+1,d) = \frac{-\frac{R_1 R_2}{h^2}(\lambda_1 - \lambda_2)\alpha^k(\alpha^d - 1)(\alpha - 1)(R_1^2 \alpha^{2k+d+1} - R_2^2)}{(R_1 \alpha^k + R_2)(R_1 \alpha^{k+d} + R_2)(R_1 \alpha^{k+1} + R_2)(R_1 \alpha^{k+d+1} + R_2)} \tag{54}$$

Notice that each of the factors in the denominator will always be positive, and all of the factors in the numerator, except for $R_1^2\alpha^{2k+d+1} - R_2^2$, do not change sign with k. This leaves two cases to evaluate:

Case 1, $R_2 < 0$: Recalling that when $R_2 < 0$, $R_1 > |R_2|$, we see that the variance error due to delay will always be monotonic when the target estimate variance is decreasing. This matches the behavior of the scalar continuous time Kalman filter variance.

Case 2, $R_2 > 0$: Since $R_1^2\alpha^{2k+d+1} - R_2^2 > R_1^2 - R_2^2$, the constraint $R_1^2 - R_2^2 > 0$ is a sufficient condition for the positiveness of $R_1^2\alpha^{2k+d+1} - R_2^2$.

$$\begin{aligned} R_1^2 - R_2^2 &= \frac{(\lambda_1^2 - \lambda_2^2) + 2(\lambda_1 - \lambda_2)(h^2 p_0 - A)}{(\lambda_1 - \lambda_2)^2} \\ &= \frac{(\lambda_1 + \lambda_2) + 2(h^2 p_0 - A)}{(\lambda_1 - \lambda_2)} \\ &= \frac{r - A + 2h^2 p_0}{(\lambda_1 - \lambda_2)} \\ &= \frac{-r(f^2 - 1) + h^2(2p_0 - q)}{(\lambda_1 - \lambda_2)} \qquad (55) \end{aligned}$$

This establishes the following sufficient conditions on the monotonicity of the convergence of the variance error when $p_0 < p_f$:

$$|f| \le 1 \qquad (56)$$

$$p_0 \ge \frac{q}{2} \qquad (57)$$

or more precisely,

$$p_0 \ge \frac{r(f^2 - 1) + h^2 q}{2h^2} \qquad (58)$$

Looking again at whether it is better to overestimate or underestimate the delay when it is not known, define $k' = k + d$ and let $\delta > 0$ be the error in the delay estimate. Then

$$\begin{aligned} \Delta p(k', \delta) + \Delta p(k', -\delta) &= \frac{-\frac{R_1 R_2}{h^2}(\lambda_1 - \lambda_2)\alpha^{k'}}{R_1\alpha^{k'} + R_2}\left(\frac{\alpha^{\delta} - 1}{R_1\alpha^{k'+\delta} + R_2} + \frac{\alpha^{-\delta} - 1}{R_1\alpha^{k'-\delta} + R_2}\right) \\ &= \frac{-\frac{R_1 R_2}{h^2}(\lambda_1 - \lambda_2)\alpha^{k'}}{R_1\alpha^{k'} + R_2}\left(\frac{(R_1\alpha^{k'} - R_2)(2 - \alpha^{\delta} - \alpha^{-\delta})}{(R_1\alpha^{k'+\delta} + R_2)(R_1\alpha^{k'-\delta} + R_2)}\right) \qquad (59) \end{aligned}$$

Recalling eq. (53), the sign of $\Delta p(k, \delta)$ is opposite that of R_2, and the sign of $\Delta p(k, -\delta)$ is the same as R_2. Thus it is better to overestimate the delay when the sign of eq. (59) is the same as R_2 (meaning the magnitude of $\Delta p(k, -\delta)$ is greater than that of $\Delta p(k, \delta)$). From previous analysis, the term $(2 - \alpha^{\delta} - \alpha^{-\delta})$ is always negative, leaving only $(R_1\alpha^{k'} - R_2)$ to affect the sign of the sum over time. Since $\alpha^{k'} \ge 1$, if $R_1 - R_2 > 0$, then the sign of eq. (59) is the same as R_2, and thus it is better to overestimate, rather than underestimate, the length of the sensor request delay.

$$\begin{aligned} R_1 - R_2 &= \frac{\lambda_1 + \lambda_2 + 2h^2 p_0 - 2A}{\lambda_1 - \lambda_2} \\ &= \frac{r - A + 2h^2 p_0}{\lambda_1 - \lambda_2} \\ &= \frac{r(1 - f^2) + h^2(2p_0 - q)}{\lambda_1 - \lambda_2} \end{aligned} \tag{60}$$

This leads to the same conditions required for montonicity in eqs. (56)-(58).

10 Conclusions

Several sensor selection algorithms have been proposed for maintaining a target's state estimate covariance near a desired level without over-taxing the computational resources of a tracking system. The proposed algorithms maintain a specific desired covariance for each target while reducing the resource demands of current unmanaged or "dumb" systems. Simulation results indicate that the three sensor selection algorithms presented in this paper clearly outperform "dumb" systems in terms of resource efficiency. Other sensor manager functions including prioritizing and scheduling are assumed to be performed separately and will impact the covariance control algorithms in the form of request execution delays.

As in dynamic systems, delay dramatically reduces the performance of the control algorithm, but if it can be accurately modeled, most of the performance can be restored. However, the effect of unmodeled delay decreases with time (as the covariance of the system converges to a steady-state value). Furthermore, when the actual delay is unknown or varies over time, overestimating the delay will generally produce smaller covariance prediction errors than underestimating the delay. Underestimating the delay is better in continuous-time tracking systems only when increasing the estimate covariance of a target with an unstable dynamic model. In discrete-time systems, a similar trend is found in that overestimating the delay usually leads to smaller covariance prediction errors.

Based on these observations, strategies to reduce the effects of delay on covariance estimates could include reducing the scan rate of the controller, i.e. allowing the Kalman filter to run longer in between changing the sensor combination. However, this strategy is limited by the desired responsiveness of the system. If the scan rate of the controller is reduced, the time required to change the target covariance increases. Another strategy is to consistently overestimate the delay when attempting to predict the covariance. The drawback to this method is that it increases the computational demand on the controller.

Acknowledgments

This work was supported in part by the Data Fusion Corporation, the Colorado Advanced Software Institute, an Office of Naval Research Young Investigator Award (Grant N00014-97-1-0642), and a National Science Foundation Early Faculty CAREER Award (Grant CMS-9625086). The authors would also especially like to thank Woody Kober and John K. Thomas of the Data Fusion Corporation for suggesting this area of research and for their feedback in the development of the sensor selection algorithms, and Nathan Baltz of the University of Colorado for his editorial comments.

References

[1] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*, Academic Press, Inc., San Diego, 1988.

[2] Y. Bar-Shalom and X. Li. *Estimation and Tracking: Principles, Techniques, and Software*, Artech House, Boston, 1993.

[3] T. Kerr, "Modeling and Evaluating an Empirical INS Difference Monitoring Procedure Used to Sequence SSBN NAVAID Fixes", *Navigation: Journal of the Institute of Navigation*, vol. 28, no. 4, pp. 263 - 285, Washington, DC, 1982.

[4] V. Kocic and G. Ladas, *Global Behavior of Nonlinear Difference Equations of Higher Order with Applications,* pp. 177-88, Kluwer Academic Publishers, Dordrecht, Netherlands, 1993.

[5] S. Musick and R. Malhotra, "Chasing the Elusive Sensor Manager," *Proceedings of the IEEE 1994 NAECON*, vol. 1, pp. 606-613, Dayton, OH, IEEE: New York, NY, 1994.

[6] J. Nash, "Optimal Allocation of Tracking Resources," *Proceedings of the 1977 IEEE Conference on Decision and Control*, vol. 1, pp. 1177-1180, New Orleans, LA, IEEE: New York, NY, 1977.

[7] R. Popoli, "The Sensor Management Imperative", *Multitarget-Multisensor Tracking: Applications and Advances Volume 2*, Y. Bar-Shalom, ed., pp 325-390, Artech House, Boston, 1992.

[8] W. Schmaedeke, "Information-based Sensor Management," SPIE Proceedings, vol. 1955, April 1993.

[9] D. Willner, C. B. Chang, and K. P. Dunn, "Kalman Filter Algorithms for a Multi-Sensor System," *Proceedings of the 1976 IEEE Conference on Decision and Control*, 1976.

[10] Z. Zhang and K.J. Hintz "OGUPSA Sensor Scheduling Architecture and Algorithm," *Proceedings of SPIE Signal Processing, Sensor Fusion, and Target Reocgnition V*, vol. 2755, pp. 296-303, 1996.

ALLOCATION OF SENSING RESOURCES IN DISTRIBUTED MULTIPROCESSOR SYSTEMS

by

NATHAN THOMAS BALTZ

B.S., University of Arkansas, 1997

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirement for the degree of

Master of Science

Department of Electrical and Computer Engineering

1999

This thesis entitled:

Allocation of Sensing Resources in Distributed Multiprocessor Systems

written by Nathan Baltz

has been approved for the Department of Electrical and Computer Engineering

by

Lucy Pao

Lucy Y. Pao

Clifford T. Mullis

C. T. Mullis

Date April 30, 1999

The final copy of this thesis has been examined by the signators, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Baltz, Nathan (MS, Electrical and Computer Engineering)

Allocation of Sensing Resources in Distributed Multiprocessor Systems

Thesis directed by Professor Lucy Pao

There is a need for rigorous analytical methods for managing modern sensors so that computing, communication, and sensing resources are optimally allocated to achieve a desired level of estimation performance. We consider distributed sensor management issues related to maintaining estimation performance. Other issues related to sensor management such as detection performance, multitarget tracking, and cluttered measurements were not considered in this thesis.

This thesis provides an analysis of error covariance control techniques for distributed multiprocessor, multisensor systems. The distributed algorithms developed for allocating sensing resources manage the rate and resolution at which information from various nodes' sensors is processed. The algorithms are robust in the respect that they preserve a level of nodal autonomy of sensor usage while also maintaining a desired level of estimation performance.

Dedication

I would like to thank my family for their support and encouragement, for my mother's patience with me when she was writing her masters thesis during the time when we were growing up. Now that I have written my masters thesis I have tremendous respect for parents who are going to school to better themselves. Finally, I want to thank my God for the gifts he has blessed me.

Acknowledgments

My advisor Lucy Pao has been a wonderful person to work with in the short time I have been here. Her dedication and professionalism was unwavering even while raising a family. Mike Kalandros, a class mate and coworker, has been a tremendous asset. During the two years we worked together, Mike has helped me innumerable times.

We also worked closely and received funding from John Thomas and Woody Kober at the Data Fusion Corporation (DFC). Their guidance helped us to look at more practical aspects of the problem that we had not considered. This work was also funded in part by the Colorado Advanced Software Institute (CASI), and an Office of Naval Research (ONR) Young Investigator Award (Grant N00014-97-1-0642).

Dr. Tom Mullis and Dr. Penina Axlerad have both been a pleasure to work with. I must acknowledge and thank Tom and Penny for taking the time to read my thesis, be on my committee, and for giving much needed feedback.

Other classmates and co-workers Craig Cutforth, Wat Khawsuk, Christopher Lee, Steve Johnson, Sara Bradburn, Matt Tucker, Adam Bennett, Teri Piatt, Drew Engelmann, and Mark Lau have been instrumental in good discussions involving current research ideas in controls and signal processing.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Sensor Management

Sensor management is concerned with improving or optimizing the measurement process in a tracking system [17]. Different sensors may have different controllable parameters. When a sensor parameter may be changed in real time it is called an agile sensing resource. Sensors can be classified as active or passive, serial or parallel, and broadband or narrowband. Similar to multi-user communications systems, multi-target tracking systems use concepts of multiple access such as Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA), and Code Division Multiple Access (CDMA) to efficiently allocate sensing, communication, and computational resources.

When managing a monopulse ESA radar some parameters we may be concerned with are radar beam shape, electromagnetic emissions, average energy, average power, modulated waveform, modulating waveform, carrier frequency, pulse period, and sample period [6]. Although not all radars can change these parameters in real time, at some point in the design process these parameters must be chosen. Other non-controllable parameters that affect detection and estimation performance are Radar Cross Section (RCS) and channel noise. These parameters together determine the detection and estimation performance in a tracking system.

An infrared CCD array is an example of a passive, narrow band or broadband, parallel sensor. Parameters associated with this sensor are its pixel frequency response, image resolution, and sample rate. After the CCD array is designed, the frequency response and image resolution are generally fixed, while the sample period can be a variable parameter depending on the hardware. Quantization effects such as finite image resolution and measurement or channel noise affect subsequent estimates.

Rate and resolution are two fundamental concepts in signal processing. We define rate as the inverse of the sensor sample period and resolution as the inverse of a positive definite error covariance matrix, the so called Fisher information. Most sensor management techniques have considered rate and resolution separately [7,8,11,13,15, 19,20,21,22].

Because the types of sensors are so varied, there are several levels of detail in which one can model a sensor. In a similar manner there are also different levels of detail in a sensor manager. These will be coarsely divided and labeled micro and macrosensor management. One description of microsensor and macrosensor management is contained in [17]. The microsensor manager handles the details of how the sensor is to achieve the sensing task provided by the macrosensor manager [17]. In managed data fusion, the sensor manager uses feedback from the signal processing and information processing systems in order to control the sensors' manageable resources.

Sensing actions are broadly divided between searching for new targets and tracking existing targets. Separation of these two actions has been enabled by special sensors that allow dynamic allocation of sensing tasks to the sensor time line. The classic example that illustrates this is the difference between mechanically scanned antenna (MSA) radars and electronically scanned array (ESA) radars. The benefits of dynamic time allocation in MSA radars can not be fully achieved because of the time constant associated with physically moving the antenna. This has caused searching and tracking to be coupled in MSA radar systems [17]. The benefits of dynamic time allocation in ESA radars, however, can more fully be made use of because the cost associated with switching between sensing tasks is negligible. This has allowed the separation of searching and tracking in some ESA radar systems. We therefore begin our efforts by focusing on tracking one target with the available sensing resources.

Several sensor management systems have been proposed for centralized systems [13,19] based on the optimization of a cost function generated using target priority, the covariance of each target state estimate, and the cost of using specific sensor combinations. Two problems associated with using these techniques are that 1) using target priority is a coarse adjustment for maintaining tracking performance, and 2) they do not consider using sample rate to maintain tracking performance.

The drawback of the above approaches in addressing tracking performance motivated the development of the algorithms presented in this thesis. These methods are based upon maintaining desired covariance goals [8,14,15]. Using these desired covariance goals we are able to develop algorithms that use both sensor rate and sensor resolution to jointly optimize the target error covariance. In this approach, the algorithms are implemented in a specific architecture that separates the sensor manager into a controller, which selects the sensor combinations based on their ability to achieve this goal, and a sensor scheduler that prioritizes sensing actions and executes them as time allows. Low priority actions may be delayed until future scans or may be dropped altogether. The covariance controller maintains the covariance level of each target estimate to within some desired level while reducing system resource demands.

The sensor scheduler is relegated to a "black box" without specifying its operations. One of the expected effects of the separate sensor scheduler is the delay of the execution of sensing requests. This arises due to scheduling delays and the limited computational resources of the tracking system [16]. Because of this, not all sensor requests can be executed in a single scan, causing sensor requests to accumulate in the command queue. This results in future requests being delayed as well.

Distributed sensing systems have significant advantages over centralized sensing systems. Having a distributed network increases survivability. When one node or nodes' sensing resources fail, the remaining nodes can reallocate sensing resources so that the mission goals are maintained. Depending on the distributed network architecture, both the communication bandwidth and computational complexity may be reduced compared with that of the centralized system where all measurement data goes to the central processor. This communication and computational load reduction can be achieved by using non-fully connected networks which usually results in worse tracking performance. In a fully connected network however, the tracking performance is not degraded because each node has full information of targets of interest. Another important advantage is increased target detectability and observability. The radar cross section (RCS) for one node may be so small that it could cause a missed detection. Having many nodes at different look angles can allow for better detectability. Some nodes sensors may only provide information about a subset of the state vector. Having different nodes' sensors observing the same target can allow target observability.

1.2 Problem Statement

The application of distributed multiprocessor, multisensor fusion to surveillance systems has provided superior tracking performance at the cost of increased communication and computational demand. As the number of platforms, targets, and sensors increase, tracking systems can very quickly become overloaded by the incoming data. Sensor management systems that can balance tracking performance with system resources have been proposed to combat this problem. Such systems generate sensing actions, then prioritize and schedule those actions [11].

1.3 Approach

Due to the advantages of distributed networks, we wish to develop robust distributed sensor management techniques. A block diagram of a decentralized tracking system is shown in Figure 1. This model shows the different components of the system including sensing and communications facilities along with signal and information processors and a sensor manager.

This block diagram illustrates the sensor control problem, where each nodes' sensor manager uses rate and resolution to maintain the state information matrix $\mathbf{P}^{-1}(t_{n+1}|t_n)$ within an elliptical annulus described by a desired update information matrix and a desired prediction information matrix given by $\mathbf{P}_{du}^{-1}(t_n)$ and $\mathbf{P}_{dp}^{-1}(t_n)$, respectively. These desired information matrices are common goals among all nodes. This philosophy emphasizes the development of algorithms whereby all nodes work together to achieve a common goal. In contrast to the centralized sensor manager, the decentralized sensor management problem is further complicated due to the presence of feedback of sensor information from other sensing nodes.

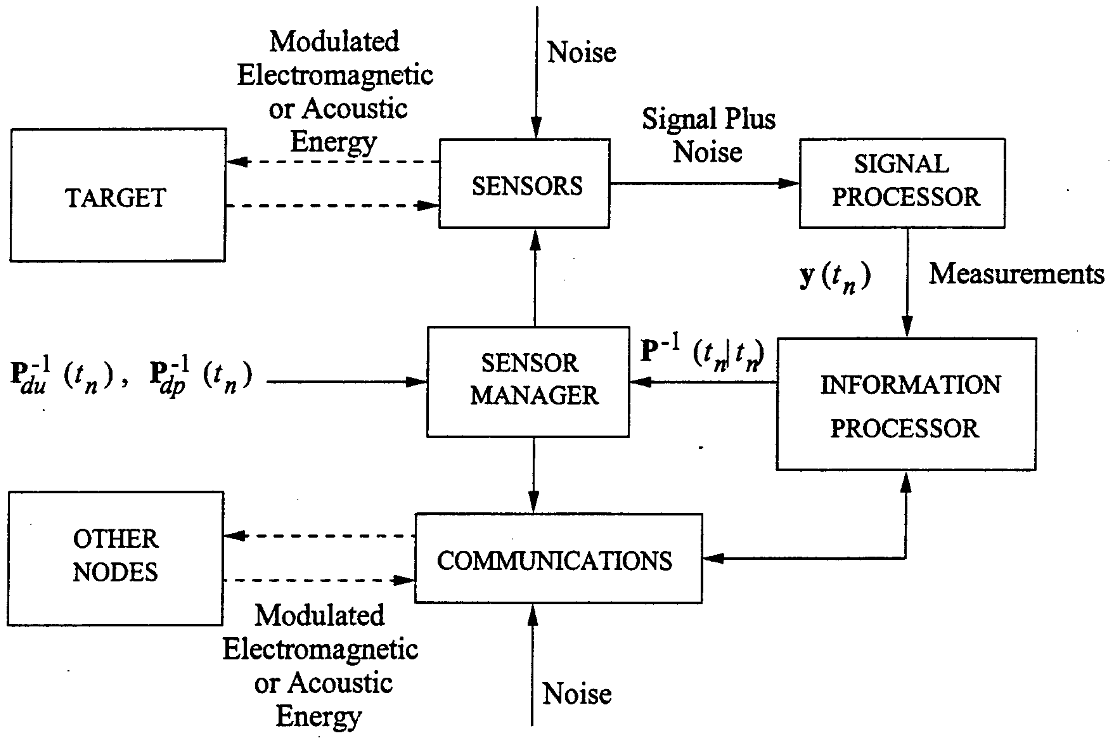


Figure 1 Components of a decentralized tracking system

The elliptical annulus can have several interpretations: the first interpretation is to treat the desired information goals as upper and lower bounds on the state information matrix and the second interpretation is to treat each desired information matrix as lower bounds on the state information matrix. The first interpretation elicits the idea that we want to maintain the state information matrix inside the elliptical annulus, i.e. $\mathbf{P}_{du}^{-1}(t_n) > \mathbf{P}^{-1}(t_{n+1}|t_n) > \mathbf{P}_{dp}^{-1}(t_n)$. The second interpretation of the desired information matrices is that each is treated as a lower bound, i.e. $\mathbf{P}^{-1}(t_{n+1}|t_n) > \mathbf{P}_{dp}^{-1}(t_n)$ and $\mathbf{P}^{-1}(t_n|t_n) > \mathbf{P}_{du}^{-1}(t_n)$. This work uses the second interpretation of the elliptical annulus so that we have more information than desired. If we use covariance matrices rather than information matrices, then the lower bounds become upper bounds on the state covariance matrix.

We present two robust distributed algorithms that maintain the desired covariance while using a minimal amount of internodal communication. In addition to exploring the distributed problem, we also investigated a number of aspects of the centralized sensor management problem in order to clarify which potential approaches to pursue for decentralized sensor management algorithms.

Many issues must be considered when designing a sensor manager. Rate and resolution are the primary issues when we are concerned with maintaining a certain level of estimation performance. Two other important performance criteria that a sensor manager might consider are detection performance and electromagnetic (EM) power emissions. The detector is contained in the signal processor in the block diagram from Figure 1. In some scenarios it may be desirable to put limits on the EM power emissions such as in covert operations. For all of these issues we must consider the communication and computational load not only in the detection and filtering algorithms but also the additional communication and computation involved with the sensor management algorithms.

The block diagram of the sensor manager is shown in Figure 2. The sample period, $T(t_n)$, is a function of time and can change for each new sample. The resolution, $\Gamma(t_n)$, is a subset of the available sensors at the *i*th node. The rate and resolution computation can be performed at a reduced rate from the sensor sample rate(s). This is one method for reducing the computation in the sensor manager. The sensor scheduler

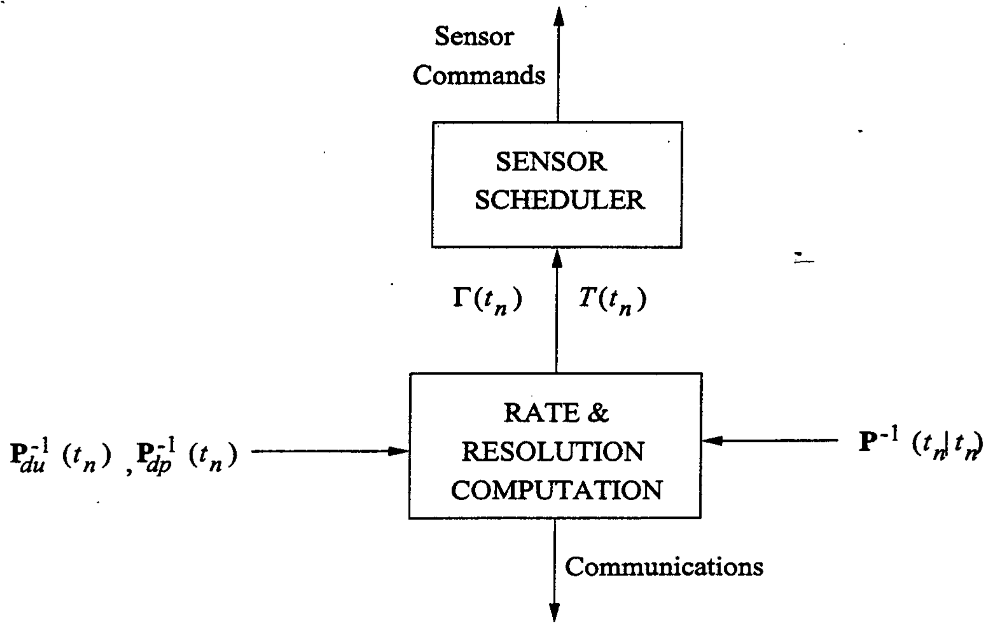

Sensor Commands
SENSOR SCHEDULER
$\Gamma(t_n)$
$T(t_n)$
$\mathbf{P}_{du}^{-1}(t_n)$, $\mathbf{P}_{dp}^{-1}(t_n)$
RATE & RESOLUTION COMPUTATION
$\mathbf{P}^{-1}(t_n|t_n)$
Communications

Figure 2 Sensor manager block diagram

takes the rate and resolution information and applies it to the sensor time lines. The sensor scheduler can cause problems in maintaining a desired covariance. The sensor scheduler can cause both delay and dropout of sensor requests. Delays can happen when two sample rates are not commensurate. When tracking one target, as considered in this thesis, the sensor scheduler will have lesser affects than if there are multiple targets that must be tracked.

1.4 Contributions

Although this work has been directed towards a masters thesis, that does not preclude original thought. There are several original contributions contained in this thesis. The main contribution of this work is the covariance control technique described in Section 3.2, where a desired elliptical annulus is used to allocate sensing resources.

Using different metrics to control the prediction and update covariance are also contributions. Several new metrics have been developed for choosing sensors that are based upon the Singular Value Decomposition (SVD). Rate optimization polynomials were also developed that are used for computing the optimal sample period in the Kalman Filter for achieving a specific covariance goal. These contributions apply to centralized and decentralized multisensor systems.

The ordered nodes algorithm and extended ordered nodes algorithm are contributions to the area of distributed multiprocessor sensor management. These specific algorithms apply many of the covariance control techniques developed in this thesis.

1.5 Overview

The chapters are organized as follows. Chapter 2 reviews some theoretical background material. Using linear system theory, we review discrete tracking models from a continuous time linear system. These models are used for developing covariance control techniques where the sample period is a variable control parameter.

Chapter 3 addresses sensor manager issues for centralized systems. The Kalman filter equations are reviewed and used in developing covariance control techniques. The sensor manager controls covariance through adjustment of sensor rates and sensor resolutions. These techniques are developed as background material for the more difficult problem of allocating sensor resources in decentralized systems.

Chapter 4 introduces sensor manager techniques for decentralized systems. The DKF is used in the development of two distributed sensor manager algorithms. The Ordered Nodes algorithm and Extended Ordered Nodes algorithms are detailed and simulations are presented to demonstrate the performance of these algorithms.

Finally, chapter 5 gives some conclusions about covariance control techniques used in sensor managers and discusses issues for further investigation.

Chapter 2

THEORETICAL BACKGROUND

2.1 Linear System Theory

The following development will establish the notation used in the remaining sections and chapters. In order to model the stochastic nature of a continuous time random process, an *N*th order stochastic differential equation is used with both stochastic and deterministic inputs. The matrix coefficients are subscripted to indicate that they are the continuous time parameters. The matrix coefficients are assumed known and possibly time varying. In an actual application, some of the coefficients may be unknown and must be computed a priori and possibly estimated in real time from data received from the sensors.

$$\dot{\boldsymbol{x}}(t) = \mathbf{A}_c(t)\boldsymbol{x}(t) + \mathbf{B}_c(t)\boldsymbol{w}_c(t) + \mathbf{C}_c(t)\boldsymbol{u}(t) \tag{2.1}$$

$$\boldsymbol{y}(t) = \mathbf{D}_c(t)\boldsymbol{x}(t) + \mathbf{E}_c(t)\boldsymbol{v}_c(t) + \mathbf{F}_c(t)\boldsymbol{u}(t) \tag{2.2}$$

The deterministic input $\boldsymbol{u}(t)$, appearing in both (2.1) and (2.2), models control inputs and feed through terms, respectively. For example, in an aircraft, satellite, or robot, these terms could model forces exerted by thrusters, gyros, or motors. In estimation problems, these control inputs are usually removed and, when the inputs are deterministic, do not degrade the quality of the estimates. Proceeding in this fashion and relabeling the matrix coefficients we have the state and measurement equations.

$$\dot{x}(t) = \mathbf{A}_c(t)x(t) + \mathbf{B}_c(t)w_c(t) \tag{2.3}$$

$$y(t) = \mathbf{C}_c(t)x(t) + \mathbf{D}_c(t)v_c(t) \tag{2.4}$$

Using the matrix integrating factor $e^{-\int_{t_0}^{t}\mathbf{A}_c(\tau)d\tau}$ to multiply each side of (2.3), we can solve the system of differential equations. Assuming the matrices in (2.3) are constant, the matrix integrating factor is then $e^{-\int_{t_0}^{t}\mathbf{A}_c(\tau)d\tau} = e^{(t_0-t)\mathbf{A}_c}$. The solution is [12]

$$e^{(t_0-t)\mathbf{A}_c}(\dot{x}(t) - \mathbf{A}_c x(t)) = \frac{d}{dt}(e^{(t_0-t)\mathbf{A}_c}x(t)) = e^{(t_0-t)\mathbf{A}_c}\mathbf{B}_c w_c(t)$$

$$\int_{t_0}^{t}\frac{d}{d\tau}(e^{(t_0-\tau)\mathbf{A}_c}x(\tau))d\tau = \int_{t_0}^{t}e^{(t_0-\tau)\mathbf{A}_c}\mathbf{B}_c w_c(\tau)d\tau$$

$$\begin{aligned} e^{(t_0-t)\mathbf{A}_c}x(t) - e^{(t_0-t_0)\mathbf{A}_c}x(t_0) &= \int_{t_o}^{t}e^{(t_0-\tau)\mathbf{A}_c}\mathbf{B}_c w_c(\tau)d\tau \\ x(t) &= e^{(t-t_0)\mathbf{A}_c}x(t_0) + e^{(t-t_0)\mathbf{A}_c}\int_{t_0}^{t}e^{(t_0-\tau)\mathbf{A}_c}\mathbf{B}_c w_c(\tau)d\tau \\ x(t) &= e^{(t-t_0)\mathbf{A}_c}x(t_0) + \int_{t_0}^{t}e^{(t-\tau)\mathbf{A}_c}\mathbf{B}_c w_c(\tau)d\tau \end{aligned} \tag{2.5}$$

With the substitution of variables $t_{n+1} = t$, $t_n = t_o$, and $T = t_{n+1} - t_n$, we establish the linear difference equation

$$\begin{aligned} x(t_{n+1}) &= e^{\mathbf{A}_c T}x(t_n) + \int_{t_n}^{t_{n+1}} e^{\mathbf{A}_c(t_{n+1}-\tau)}\mathbf{B}_c w_c(\tau)d\tau \\ x(t_{n+1}) &= \mathbf{A}x(t_n) + \mathbf{B}w(t_n) \end{aligned} \tag{2.6}$$

where $\mathbf{A} = e^{\mathbf{A}_c T}$, $\mathbf{B} = \mathbf{I}$, and $w(t_n) = \int_{t_n}^{t_{n+1}} e^{\mathbf{A}_c(t_{n+1}-\tau)}\mathbf{B}_c w_c(\tau)d\tau$. Depending on the assumption imposed on the noise term in the integral, two models can be developed. The first model lets $\mathbf{B} = \mathbf{I}$ so that the noise term entering the system is just the last term in (2.6). This model is the so called discretized-continuous model. The second model assumes the continuous time white noise input is piece-wise constant during each integration period. This allows us to remove the noise term from the integral, simplifying the computation of the integral. This model is the so called direct-discrete model. These models are used to simulate the motion of a target. The process noise covariance for each of the above models is derived in the following sections for different state vectors.

The state and measurement equations for the discrete system are

$$x(t_{n+1}) = \mathbf{A}x(t_n) + \mathbf{B}w(t_n) \tag{2.7}$$

$$y(t_n) = \mathbf{C}x(t_n) + \mathbf{D}v(t_n) \tag{2.8}$$

where $y(t_n) = \left[y_1^{\mathrm{T}}(t_n) \ \ldots \ y_M^{\mathrm{T}}(t_n)\right]^{\mathrm{T}}$, $\mathbf{C} = \left[\mathbf{C}_1^{\mathrm{T}} \ \ldots \ \mathbf{C}_M^{\mathrm{T}}\right]^{\mathrm{T}}$, and

$v(t_n) = \left[v_1^T(t_n) \ \ldots \ v_M^T(t_n)\right]^{\mathrm{T}}$, with M sensors each taking measurements of length m_i.

The process noise $w(t_n)$ and measurement noise $v(t_n)$ are assumed to have the following statistical properties.

$$\mu_w = \mathrm{E}[w(t_n)] = 0 \tag{2.9}$$

$$\mathrm{E}[w(t_m)w(t_n)^{\mathrm{T}}] = \sigma^2\mathbf{C}_{ww}\delta_{m-n} \tag{2.10}$$

$$\mu_v = \mathrm{E}[v(t_n)] = 0 \tag{2.11}$$

$$\mathrm{E}[v(t_m)v(t_n)^{\mathrm{T}}] = \rho^2\mathbf{C}_{vv}\delta_{m-n} \tag{2.12}$$

$$\mathrm{E}[w(t_m)v(t_n)^{\mathrm{T}}] = 0 \tag{2.13}$$

where δ_{m-n} is the Kroneker delta function given as

$$\delta_{m-n} = \begin{cases} 1, m = n \\ 0, m \neq n \end{cases} \tag{2.14}$$

The volume of the covariance matrices can be scaled by changing the σ and ρ.

$$\mathbf{Q} = \mathrm{E}[\mathbf{B}w(t_m)w(t_n)^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}] = \sigma^2\mathbf{B}\mathbf{C}_{ww}\mathbf{B}^{\mathrm{T}}\delta_{m-n} \tag{2.15}$$

$$\mathbf{R} = \mathrm{E}[\mathbf{D}v(t_m)v(t_n)^{\mathrm{T}}\mathbf{D}^{\mathrm{T}}] = \rho^2\mathbf{D}\mathbf{C}_{vv}\mathbf{D}^{\mathrm{T}}\delta_{m-n} \tag{2.16}$$

2.2 Target Tracking Models

When we observe a small enough trajectory of the motion of a target, it appears to be smooth and continuous in nature. One approach is to begin with a continuous state equation and derive the associated discrete state and measurement equations.

A band limited Gaussian white noise process is one approximation for the motion of a target. The sampling frequency needed to recover all the information of

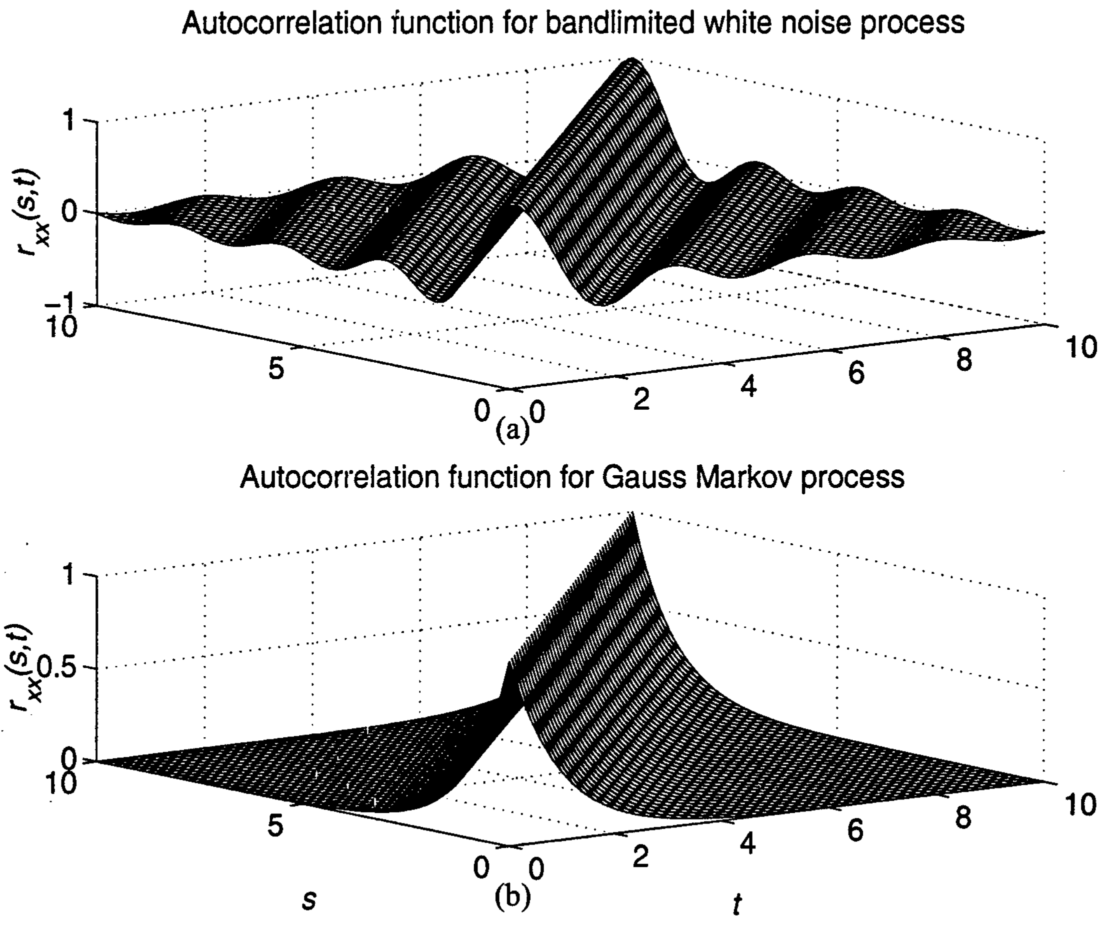


Figure 3 Autocorrelation functions of two stationary processes: (a) bandlimited white noise process and (b) Gauss-Markov process

such a process is twice the highest frequency component of the band limited white noise process. Implicit in this statement is that an infinite set of samples are obtained from the random process in order to completely recover the signal. When only a finite set of samples are received, the error in estimating the signal *between* samples decreases as more samples are received.

The autocorrelation functions of two Wide Sense Stationary (WSS) processes are shown in Figure 3. These processes are WSS because the mean is constant and the autocorrelation can be written as $r_{xx}(s, t) = r_{xx}(\tau)$ where $\tau = s - t$. Figure 3a shows the sinc function given by $r_{xx}(s, t) = \frac{\sin(\omega(s-t))}{\omega(s-t)}$. Figure 3b shows an exponential autocorrelation function given by $r_{xx}(s, t) = \sigma^2 \exp(-a|s-t|)$.

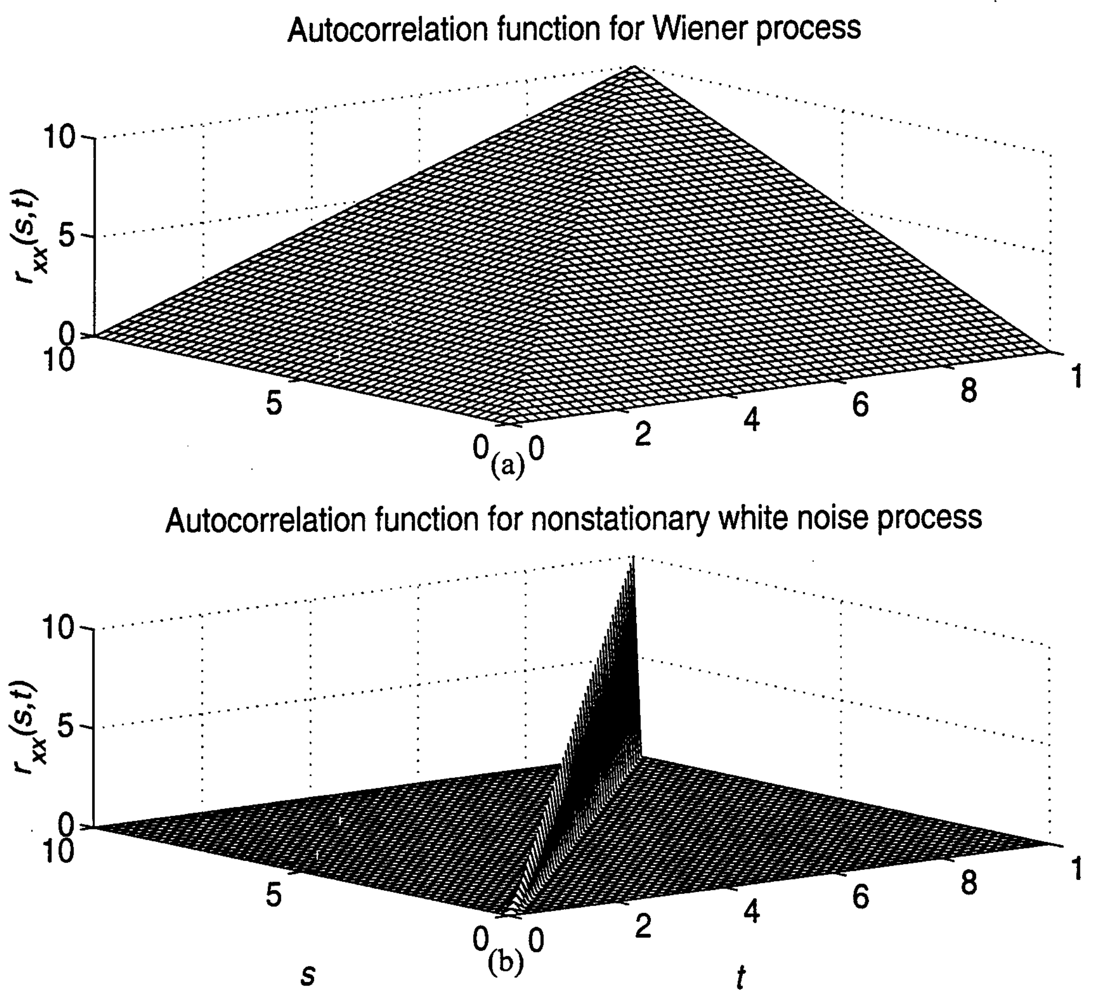


Figure 4 Autocorrelation functions of two nonstationary processes: (a) Wiener process and (b) ramped white noise process

Another stochastic process is obtained by integrating a white Gaussian noise signal. This random process is the so-called Brownian motion or Wiener process, named after English Botanist Robert Brown who first discovered such motion in nature and Norbert Wiener who gave a rigorous mathematical study of such a random process, respectively. The Wiener process autocorrelation function is $r_{xx}(s, t) = \sigma^2 min(s, t)$. The autocorrelation function for the Weiner process is plotted in Figure 4 (a). This process is no longer stationary because the variance $\sigma_x^2(t) = r_{xx}(t, t) = \sigma^2 t$, changes with time. This simple model might be appropriate for modeling the motion of a target in one dimension because it models the growing uncertainty.

Figure 4b shows the autocorrelation function of a ramped white noise process. The signal is $x(t) = \sigma\sqrt{t}w_c(t)$, with $t \geq 0$. The autocorrelation of this process is

$$r_{xx}(s, t) = \mathrm{E}[x(s)x(t)] = \sigma^2\sqrt{s}\sqrt{t}\mathrm{E}[w_c(s)w_c(t)] = \sigma^2 t\delta_{s-t} \qquad (2.17)$$

The variance of this process is $\sigma_x^2(t) = r_{xx}(t, t) = \sigma^2 t$. Notice that the autocorrelation functions in each of the above plots have the same variance, but since the Wiener process is the integral of white noise it produces smoother trajectories than the nonstationary white noise process.

2.2.1 First Order Models

The choice of the coefficients in (2.3) and (2.4) will determine the properties of the autocorrelation function of the process. The first model we develop is a Wiener process. Using (2.3) and (2.4) let the coefficients be

$$\mathbf{A}_c = 0, \mathbf{B}_c = \sigma, \mathbf{C}_c = 1, \mathbf{D}_c = 0 \qquad (2.18)$$

With these coefficients the state and measurement equations for a Wiener process are

$$\dot{x}(t) = \sigma w_c(t) \tag{2.19}$$

$$y(t) = x(t) \tag{2.20}$$

Using (2.5), the solution of the state vector is

$$x(t) = x(t_0) + \sigma \int_{t_0}^{t} w_c(\tau) d\tau \tag{2.21}$$

where t_0 is the initial time. The process noise entering the system is the integral of white noise. Assuming $t_o = 0$, let the initial state be a zero mean gaussian random variable i.e. $x(0) \sim N(0, \rho^2)$. The mean of the Wiener process is $\mu_x(t) = \mathrm{E}[x(t)] = 0$. The autocorrelation of this Wiener process is

$$\begin{aligned} r_{xx}(s,t) &= \mathrm{E}[x(s)x(t)] \\ &= \mathrm{E}\left[\left(x(0) + \sigma \int_0^s w(\beta) d\beta\right)\left(x(0) + \sigma \int_0^t w(\tau) d\tau\right)\right] \\ &= \mathrm{E}\left[\sigma^2 \int_0^s \int_0^t w(\beta) w(\tau) d\tau d\beta\right] + \mathrm{E}[x(0)^2] \\ &= \sigma^2 \int_0^s \int_0^t \mathrm{E}[w(\beta) w(\tau)] d\tau d\beta + \rho^2 \\ &= \sigma^2 \int_0^s \int_0^t \delta(\tau - \beta) d\tau d\beta + \rho^2 \\ &= \sigma^2 min(s,t) + \rho^2 \end{aligned} \tag{2.22}$$

where $s, t \geq 0$. This autocorrelation function is shown in Figure 4a with $\sigma = 1$ and $\rho = 0$. When the initial estimate has zero variance ($\rho^2 = 0$), it does not contribute to the autocorrelation function. The Wiener process is nonstationary and its second moment, $r_{xx}(t,t) = \sigma^2 t + \rho^2$, increases linearly with time.

Let $y(t_n)$ be noiseless measurements of the Wiener process at the discrete times $t_1 < t_2 < \ldots < t_n$. We will form least-squares estimates of this process based upon all measurements up to the present time. Assuming a linear estimator of the form

$$\hat{y}(t_n) = \sum_{k=1}^{n} a_k(t) y(t_k) = \boldsymbol{a}_n^{\mathrm{T}}(t) \boldsymbol{y}_n \tag{2.23}$$

where $\boldsymbol{a}_n(t) = \begin{bmatrix} a_1(t) & \ldots & a_n(t) \end{bmatrix}^{\mathrm{T}} \in \Re^{n \times 1}, \forall t > 0$ is a vector of functions or filter coefficients and $\boldsymbol{y}_n = \begin{bmatrix} y(t_1) & \ldots & y(t_n) \end{bmatrix}^{\mathrm{T}} \in \Re^{n \times 1}$ are the discrete set of measurements. Notice that the vector length of the filter coefficients $\boldsymbol{a}_n(t)$ and measurement vector $\boldsymbol{y}_n$ increase as more samples are obtained. The entire signal can be estimated from all the samples from the Wiener process. Letting the error in the estimate be $e(t) = y(t) - \hat{y}(t_n)$, the mean square error (MSE) of the estimator in (2.23) is

$$\begin{aligned} f(\boldsymbol{a}_n(t)) &= \mathrm{E}[e^2(t)] \\ &= \mathrm{E}[(y(t) - \hat{y}(t_n))^2] \\ &= \mathrm{E}[(y(t) - \boldsymbol{a}_n^{\mathrm{T}}(t)\boldsymbol{y}_n)^2] \\ &= \mathrm{E}[y(t)^2 - 2\boldsymbol{a}_n^{\mathrm{T}}(t)\boldsymbol{y}_n y(t) + \boldsymbol{a}_n^{\mathrm{T}}(t)\boldsymbol{y}_n\boldsymbol{y}_n^{\mathrm{T}}\boldsymbol{a}_n(t)] \\ &= \mathrm{E}[y(t)^2] - 2\boldsymbol{a}_n^{\mathrm{T}}(t)\mathrm{E}[\boldsymbol{y}_n y(t)] + \boldsymbol{a}_n^{\mathrm{T}}(t)\mathrm{E}[\boldsymbol{y}_n\boldsymbol{y}_n^{\mathrm{T}}]\boldsymbol{a}_n(t) \end{aligned} \tag{2.24}$$

To solve for the optimal coefficients in $\boldsymbol{a}_n(t)$, we compute the gradient of the MSE with respect to the coefficients of $\boldsymbol{a}_n(t)$ and set it equal to zero.

$$\nabla_{\boldsymbol{a}} f(\boldsymbol{a}_n(t)) = -2\mathrm{E}[\boldsymbol{y}_n y(t)] + 2\mathrm{E}[\boldsymbol{y}_n\boldsymbol{y}_n^{\mathrm{T}}]\boldsymbol{a}_n(t) = 0 \tag{2.25}$$

Since there is zero measurement noise this allows us to use the autocorrelation function in (2.22) to compute the expectations in (2.25). With this, the first autocorrelation is the vector

$$\mathbf{r}_n(t) = \mathrm{E}[\boldsymbol{y}_n y(t)] = \begin{bmatrix} \mathrm{E}[y(t_1)y(t)] & \ldots & \mathrm{E}[y(t_n)y(t)] \end{bmatrix}^{\mathrm{T}} \tag{2.26}$$

and the second autocorrelation function is the matrix produced by the expectation of the outer product of the measurement vectors

$$\mathbf{R}_n = \mathrm{E}[y_n y_n^{\mathrm{T}}] = \mathrm{E}\left[\begin{bmatrix} y(t_1) \\ \dots \\ y(t_n) \end{bmatrix}\begin{bmatrix} y(t_1) \\ \dots \\ y(t_n) \end{bmatrix}^{\mathrm{T}}\right] \tag{2.27}$$

$$(\mathbf{R}_n)_{ij} = \mathrm{E}[y(t_i)y(t_j)] \quad i, j \in \{1, \dots, n\}$$

Evaluating the expectations using $\mathrm{E}[y(s)y(t)] = min(s, t)$, we have

$$\mathbf{r}_n(t) = \mathbf{R}_n \boldsymbol{a}_n(t)$$

$$\begin{bmatrix} min(t, t_1) \\ min(t, t_2) \\ \dots \\ min(t, t_n) \end{bmatrix} = \begin{bmatrix} t_1 & t_1 & \dots & t_1 \\ t_1 & t_2 & \dots & t_2 \\ \dots & \dots & \dots & \dots \\ t_1 & t_2 & \dots & t_n \end{bmatrix} \begin{bmatrix} a_1(t) \\ a_2(t) \\ \dots \\ a_n(t) \end{bmatrix} \tag{2.28}$$

This symmetric matrix can be factored into a lower triangular matrix filled with ones and an upper triangular matrix filled with the time differences between adjacent samples

$$\mathbf{R}_n = \begin{bmatrix} t_1 & t_1 & \dots & t_1 \\ t_1 & t_2 & \dots & t_2 \\ \dots & \dots & \dots & \dots \\ t_1 & t_2 & \dots & t_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ \dots & 1 & 0 & \dots \\ 1 & \dots & 1 & 0 \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} t_1 & t_1 & \dots & t_1 \\ 0 & t_2 - t_1 & \dots & t_2 - t_1 \\ \dots & 0 & \dots & \dots \\ 0 & \dots & 0 & t_n - t_{n-1} \end{bmatrix} \tag{2.29}$$

The inverse of $\mathbf{R}_n$ is the product of the following two banded matrices.

$$\mathbf{R}_n^{-1} = \begin{bmatrix} t_1^{-1} & -(t_2 - t_1)^{-1} & 0 & 0 & 0 \\ 0 & (t_2 - t_1)^{-1} & -(t_3 - t_2)^{-1} & \dots & 0 \\ \dots & 0 & (t_3 - t_2)^{-1} & \dots & 0 \\ 0 & \dots & 0 & \dots & -(t_n - t_{n-1})^{-1} \\ 0 & 0 & \dots & 0 & (t_n - t_{n-1})^{-1} \end{bmatrix} \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ -1 & 1 & 0 & \dots & 0 \\ \dots & -1 & \dots & 0 & \dots \\ 0 & \dots & -1 & 1 & 0 \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix} \tag{2.30}$$

Using (2.28) the optimal coefficients are $\boldsymbol{a}_n(t) = \mathbf{R}_n^{-1}\mathbf{r}_n(t)$. The optimal least-squares estimate is

$$\hat{y}(t_n) = \boldsymbol{a}_n^{\mathrm{T}}(t)\boldsymbol{y}_n = \mathbf{r}_n^{\mathrm{T}}(t)\mathbf{R}_n^{-1}\boldsymbol{y}_n \tag{2.31}$$

Using (2.24) and (2.31), the MSE of this estimator is

$$\begin{aligned} f(\boldsymbol{a}_n(t)) &= \mathrm{E}[(y(t)-\hat{y}(t_n))^2] \\ &= \mathrm{E}[y(t)^2] - 2\boldsymbol{a}_n^{\mathrm{T}}(t)\mathrm{E}[\boldsymbol{y}_n y(t)] + \boldsymbol{a}_n^{\mathrm{T}}(t)\mathrm{E}[\boldsymbol{y}_n\boldsymbol{y}_n^{\mathrm{T}}]\boldsymbol{a}_n(t) \\ &= min(t,t) - 2\mathbf{r}_n^{\mathrm{T}}(t)\mathbf{R}_n^{-1}\mathbf{r}_n(t) + \mathbf{r}_n^{\mathrm{T}}(t)\mathbf{R}_n^{-1}\mathbf{R}_n\mathbf{R}_n^{-1}\mathbf{r}_n(t) \\ &= t - \mathbf{r}_n^{\mathrm{T}}(t)\mathbf{R}_n^{-1}\mathbf{r}_n(t) \end{aligned} \tag{2.32}$$

Using the MSE in (2.32), the error variance of the entire process can be computed for each additional sample. After simplifying (2.32) the smoothed, filtered, and predicted estimates have the following error variances.

$$f(\boldsymbol{a}_n(t)) = \begin{cases} t - t^2/t_1, & 0 \le t \le t_1 \\ \dfrac{-(t-t_{k+1})(t-t_k)}{t_{k+1}-t_k}, & t_k \le t \le t_{k+1} \\ t - t_n, & t_n \le t \end{cases} \tag{2.33}$$

Since there is zero measurement noise, the variance of the estimation error will go to zero for each sample of the Wiener process. The prediction variance increases linearly until the next sample is received, and the smoothed or interpolated estimates have a quadratic variance between samples. This process is illustrated in Figure 5. The slope of the prediction variance is equal to the variance of the continuous time white noise. In this simulation the variance of the white noise is one and the slope of the prediction variance in Figure 5 is one. If we impose a bound on the prediction estimate variance, this immediately places restrictions on the maximum sample period. This gives us the intuitive feeling that the longer we wait to sample, the more the error variance will grow.

When there is both measurement and process noise, the error variance will never go completely to zero. This must be taken into account when designing the estimation algorithm. This leads to the development of the Kalman filter. The Kalman filter allows for uncertainty in both the state equation model and the measurements.

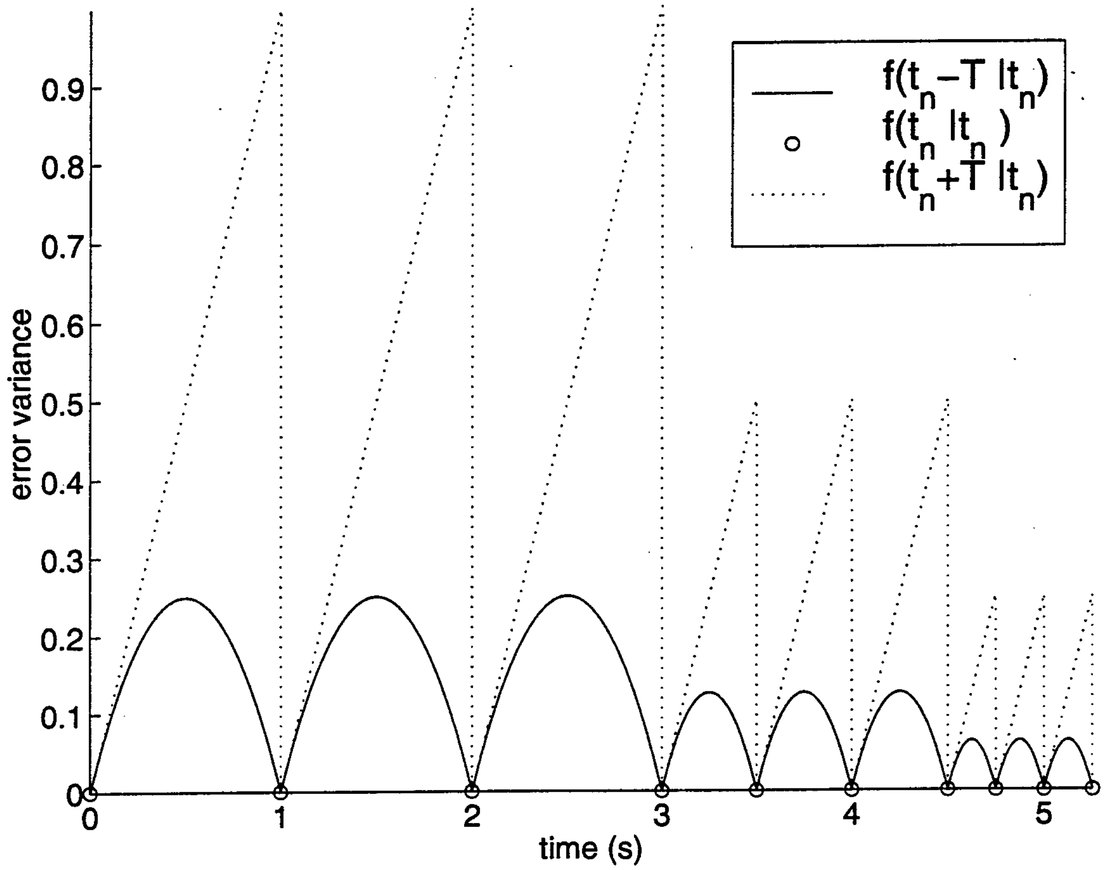


Figure 5 Error variance of smoothed, filtered, and predicted Wiener process

Based on these two uncertainties, the Kalman filter computes optimal estimates by weighting old estimates with new measurements appropriately. This concludes the development of the Wiener process. The Wiener process is also a Gauss-Markov process because the state has Gaussian density and the process is Markov.

A different Gauss-Markov process can be generated by letting the state transition matrix be a nonzero scalar. The coefficients for the continuous time process are

$$\mathbf{A}_c = -a, \mathbf{B}_c = 1, \mathbf{C}_c = 1, \mathbf{D}_c = 0 \tag{2.34}$$

With these values, the state and measurement equations have the form

$$\dot{x}(t) = -ax(t) + w_c(t) \tag{2.35}$$

$$y(t) = x(t) \tag{2.36}$$

Let the white noise input have variance $r_{ww}(s, t) = E[w_c(s)w_c(t)] = 2a\sigma^2\delta(s-t)$. Using the coefficients defined in (2.34) with the solution of the general differential equation in (2.5) gives

$$x(t) = e^{-at}x(0) + \int_0^t e^{(\tau - t)a}w_c(\tau)d\tau \tag{2.37}$$

where $t_o = 0$. The mean of this process is $\mu_x(t) = \mathrm{E}[x(t)] = e^{-at}\mathrm{E}[x(0)]$. Notice that this process will be stationary only if the mean of $x(0)$ is zero, otherwise the mean will exponentially decay or increase depending on the sign of a. With the initial state variance $\mathrm{E}[x(0)^2] = \sigma^2$ the autocorrelation function of $x(t)$ is

$$\begin{aligned} r_{xx}(s, t) &= \mathrm{E}[x(s)x(t)] \\ &= \mathrm{E}\left[\left(e^{-as}x(0) + \int_0^s e^{(\tau - s)a}w_c(\tau)d\tau\right)\left(e^{-at}x(0) + \int_0^t e^{(\beta - t)a}w_c(\beta)d\beta\right)\right] \\ &= \mathrm{E}\left[e^{-a(s+t)}\int_0^s\int_0^t e^{(\tau+\beta)a}w_c(\beta)w_c(\tau)d\beta d\tau\right] + \sigma^2 e^{-a(s+t)} \\ &= e^{-a(s+t)}\sigma^2\left(2a\int_0^s\int_0^t e^{(\tau+\beta)a}\delta(\tau-\beta)d\beta d\tau + 1\right), \text{ let } u = min(s,t) \\ &= e^{-a(s+t)}\sigma^2\left(2a\int_0^u e^{2\tau a}d\tau + 1\right) \\ &= e^{-a(s+t)}\sigma^2\left(e^{2\tau a}\Big|_0^u + 1\right) \\ &= e^{-a(s+t)}\sigma^2(e^{2ua} - 1 + 1) \\ &= \sigma^2 e^{-a(s+t-2u)} \\ &= \sigma^2 e^{-a|s-t|} \end{aligned} \tag{2.38}$$

The plot of this autocorrelation function was shown in Figure 3b.

2.2.2 Second Order Models

The Wiener process model developed in Section 2.2.1 can be extended to second order models. The general technique is to let some *n*th order derivative of a random process equal white noise. Using (2.3) and (2.4), let the coefficients be

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \mathbf{B}_c = \sigma\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{C}_c = \begin{bmatrix} 1 & 0 \end{bmatrix}, \mathbf{D}_c = \alpha \tag{2.39}$$

The state vector is $x(t) = \begin{bmatrix} p(t) & v(t) \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} p(t) & \dot{p}(t) \end{bmatrix}^{\mathrm{T}}$ where *p(t)* and *v(t)* are the position and velocity, respectively. With these coefficients for the continuous time model, we get

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \sigma \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} w_c(t) \tag{2.40}$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) + \alpha v_c(t) \tag{2.41}$$

In this model, the velocity is a Wiener process and the acceleration is white noise. The observability matrix for this system is

$$O(\mathbf{A}_c, \mathbf{C}_c) = \begin{bmatrix} \mathbf{C}_c \\ \mathbf{C}_c\mathbf{A}_c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.42}$$

Since this matrix is full rank, the system states, position and velocity, are observable.

Solving this differential equation and forming the difference equation, we have

$$\begin{aligned}
x(t_{n+1}) &= e^{\begin{bmatrix}0&1\\0&0\end{bmatrix}T}x(t_n)+\sigma\int_{t_n}^{t_{n+1}}e^{(t_{n+1}-\tau)\begin{bmatrix}0&1\\0&0\end{bmatrix}}\begin{bmatrix}0&0\\0&1\end{bmatrix}w_c(\tau)d\tau \\
&= \begin{bmatrix}1&T\\0&1\end{bmatrix}x(t_n)+\sigma\int_{t_n}^{t_{n+1}}\begin{bmatrix}1&t_{n+1}-\tau\\0&1\end{bmatrix}\begin{bmatrix}0&0\\0&1\end{bmatrix}\begin{bmatrix}w_1(\tau)\\w_2(\tau)\end{bmatrix}d\tau \\
&= \begin{bmatrix}1&T\\0&1\end{bmatrix}x(t_n)+\sigma\int_{t_n}^{t_{n+1}}\begin{bmatrix}1&t_{n+1}-\tau\\0&1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}w_2(\tau)d\tau \qquad (2.43)\\
&= \begin{bmatrix}1&T\\0&1\end{bmatrix}x(t_n)+\sigma\int_{t_n}^{t_{n+1}}\begin{bmatrix}t_{n+1}-\tau\\1\end{bmatrix}w_2(\tau)d\tau \\
&= \mathbf{A}x(t_n)+w(t_n)
\end{aligned}$$

The noise term entering the linear system has the following covariance.

$$\begin{aligned}
\mathrm{E}[w(t_n)w(t_n)^{\mathrm{T}}] &= \mathrm{E}\left[\sigma\int_{t_n}^{t_{n+1}}\begin{bmatrix}t_{n+1}-\tau\\1\end{bmatrix}w_2(\tau)d\tau\left(\sigma\int_{t_n}^{t_{n+1}}\begin{bmatrix}t_{n+1}-\beta\\1\end{bmatrix}w_2(\beta)d\beta\right)^{\mathrm{T}}\right] \\
&= \sigma^2\int_{t_n}^{t_{n+1}}\int_{t_n}^{t_{n+1}}\begin{bmatrix}t_{n+1}-\tau\\1\end{bmatrix}\begin{bmatrix}t_{n+1}-\beta & 1\end{bmatrix}\mathrm{E}[w_2(\beta)w_2(\tau)]d\beta d\tau \\
&= \sigma^2\int_{t_n}^{t_{n+1}}\int_{t_n}^{t_{n+1}}\begin{bmatrix}t_{n+1}-\tau\\1\end{bmatrix}\begin{bmatrix}t_{n+1}-\beta & 1\end{bmatrix}\delta(\tau-\beta)d\beta d\tau \qquad (2.44)\\
&= \sigma^2\int_{t_n}^{t_{n+1}}\begin{bmatrix}(t_{n+1}-\tau)^2 & t_{n+1}-\tau\\ t_{n+1}-\tau & 1\end{bmatrix}d\tau, \qquad t_{n+1}-\tau = \gamma \\
&= -\sigma^2\int_T^0\begin{bmatrix}\gamma^2&\gamma\\\gamma&1\end{bmatrix}d\gamma = \sigma^2\begin{bmatrix}T^3/3 & T^2/2\\ T^2/2 & T\end{bmatrix}
\end{aligned}$$

This model is referred to as a discretized-continuous model because it was derived from a continuous time state equation.

Another approximation can be obtained by assuming the white noise to be constant within the sampling interval. Since the noise term is constant within the integration period, it may be removed from the integral. The state equation in (2.43) becomes

$$\begin{aligned} x(t_{n+1}) &= \begin{bmatrix}1 & T\\0 & 1\end{bmatrix}x(t_n) + \sigma w_2(t_n)\int_{t_n}^{t_{n+1}}\begin{bmatrix}t_{n+1}-\tau\\1\end{bmatrix}d\tau \\ &= \begin{bmatrix}1 & T\\0 & 1\end{bmatrix}x(t_n) - \sigma w_2(t_n)\int_{T}^{0}\begin{bmatrix}\gamma\\1\end{bmatrix}d\gamma \\ &= \begin{bmatrix}1 & T\\0 & 1\end{bmatrix}x(t_n) + \sigma\begin{bmatrix}T^2/2\\T\end{bmatrix}w_2(t_n) \\ &= \mathbf{A}x(t_n) + \mathbf{B}w(t_n) \end{aligned} \tag{2.45}$$

and the covariance of the noise term is

$$\begin{aligned} \mathrm{E}[\mathbf{B}w(t_n)(\mathbf{B}w(t_n))^{\mathrm{T}}] &= \mathrm{E}\left[\sigma\begin{bmatrix}T^2/2\\T\end{bmatrix}w_2(t_n)\left(\sigma\begin{bmatrix}T^2/2\\T\end{bmatrix}w_2(t_n)\right)^{\mathrm{T}}\right] \\ &= \sigma^2\begin{bmatrix}T^2/2\\T\end{bmatrix}\mathrm{E}[w_2(t_n)^2]\begin{bmatrix}T^2/2 & T\end{bmatrix} \\ &= \sigma^2\begin{bmatrix}T^4/4 & T^3/2\\T^3/2 & T^2\end{bmatrix} \end{aligned} \tag{2.46}$$

This is called a direct discretized model. Notice that in the discretized continuous model the integral of the outer product (dyad) causes the matrix to be full rank. However, in the direct discretized model, the covariance is approximated simply by an outer product of two vectors which makes the matrix singular. Both models are approximations to simulate how the covariance of position and velocity change with the sample period. In the discretized continuous model, an integral must be evaluated to compute the input. In the direct discretized model, the input is simply a piece wise white noise signal multiplied with a gain.

The next model is used for tracking the position of a target in the *x-y* plane. This model generates a Wiener process in two dimensions. For the model described by

(2.3) and (2.4), let

$$\mathbf{A}_c = 0, \mathbf{B}_c = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{C}_c = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{D}_c = \begin{bmatrix} \alpha & \gamma \\ \gamma & \rho \end{bmatrix} \tag{2.47}$$

The state vector is $x(t) = \begin{bmatrix} p_x(t) & p_y(t) \end{bmatrix}^{\mathrm{T}}$ where $p_x(t)$ and $p_y(t)$ are the positions in the x and y dimensions, respectively. With these coefficients for the continuous time model, we get

$$\dot{x}(t) = w_c(t) \tag{2.48}$$

$$y(t) = x(t) + \begin{bmatrix} \alpha & \gamma \\ \gamma & \rho \end{bmatrix} v_c(t) \tag{2.49}$$

When the noise vector entering the system is uncorrelated, then the motion in each dimension can be modeled independently. This technique allows for modeling of target motion in each dimension and can reduce the size of the state equations.

The observability matrix for this system is

$$O(\mathbf{A}_c, \mathbf{C}_c) = \begin{bmatrix} \mathbf{C}_c \\ \mathbf{C}_c\mathbf{A}_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}} \tag{2.50}$$

Since the observability matrix is full rank the state vector is observable. The observabililty matrix indicates if a system is observable, however, it does not take into account measurement noise which is needed to figure the quality of the observations. A better measure would be to consider a matrix function $\mathbf{P}(\mathbf{A}_c, \mathbf{B}_c, \mathbf{C}_c, \mathbf{D}_c)$ that takes into account all the coefficients and provides covariance information of each of the states.

2.2.3 Third Order Models

The previous models can be extended to a third order system. Using (2.3) and

(2.4), let the matrix coefficients be

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{B}_c = \sigma\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{C}_c = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \mathbf{D}_c = \alpha \tag{2.51}$$

The state vector is $\boldsymbol{x}(t) = \begin{bmatrix} p(t) & v(t) & a(t) \end{bmatrix}^{\mathrm{T}}$ where *p(t), v(t),* and *a(t)* are the position, velocity, and acceleration, respectively. In this system, the acceleration is a Wiener process requiring its derivative (jerk) to be white noise. With these coefficients for the continuous time model, we get

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\boldsymbol{x}(t) + \sigma\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\boldsymbol{w}_c(t) \tag{2.52}$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\boldsymbol{x}(t) + \alpha v_c(t) \tag{2.53}$$

The observability matrix for this system is

$$O(\mathbf{A}_c, \mathbf{C}_c) = \begin{bmatrix} \mathbf{C}_c \\ \mathbf{C}_c\mathbf{A}_c \\ \mathbf{C}_c\mathbf{A}_c^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.54}$$

Since this matrix is full rank, the system states position, velocity, and acceleration are observable. Solving this differential equation and forming the difference equation, we have

$$x(t_{n+1}) = e^{\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix}T} x(t_n) + \sigma \int_{t_n}^{t_{n+1}} e^{(t_{n+1}-\tau)\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix}} \begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix} w_c(\tau)d\tau$$

$$= \begin{bmatrix}1&T&T^2/2\\0&1&T\\0&0&1\end{bmatrix} x(t_n) + \sigma \int_{t_n}^{t_{n+1}} \begin{bmatrix}1&t_{n+1}-\tau&(t_{n+1}-\tau)^2/2\\0&1&t_{n+1}-\tau\\0&0&1\end{bmatrix}\begin{bmatrix}0\\0\\1\end{bmatrix} w_3(\tau)d\tau \quad (2.55)$$

$$= \begin{bmatrix}1&T&T^2/2\\0&1&T\\0&0&1\end{bmatrix} x(t_n) + \sigma \int_{t_n}^{t_{n+1}} \begin{bmatrix}(t_{n+1}-\tau)^2/2\\t_{n+1}-\tau\\1\end{bmatrix} w_3(\tau)d\tau$$

The process noise covariance for this discretized-continuous model is

$$\mathbf{Q}(T) = \mathrm{E}[w(t_n)w(t_n)^{\mathrm{T}}]$$

$$= \mathrm{E}\left[\sigma \int_{t_n}^{t_{n+1}} \begin{bmatrix}(t_{n+1}-\tau)^2/2\\t_{n+1}-\tau\\1\end{bmatrix} w_3(\tau)d\tau \left(\sigma \int_{t_n}^{t_{n+1}} \begin{bmatrix}(t_{n+1}-\beta)^2/2\\t_{n+1}-\beta\\1\end{bmatrix} w_3(\beta)d\beta\right)^{\mathrm{T}}\right]$$

$$= \sigma^2 \int_{t_n}^{t_{n+1}}\int_{t_n}^{t_{n+1}} \begin{bmatrix}(t_{n+1}-\tau)^2/2\\t_{n+1}-\tau\\1\end{bmatrix} \begin{bmatrix}(t_{n+1}-\beta)^2/2 & t_{n+1}-\beta & 1\end{bmatrix} \delta(\tau-\beta)d\beta d\tau \quad (2.56)$$

$$= \sigma^2 \int_{t_n}^{t_{n+1}} \begin{bmatrix}(t_{n+1}-\tau)^4/4 & (t_{n+1}-\tau)^3/2 & (t_{n+1}-\tau)^2/2\\(t_{n+1}-\tau)^3/2 & (t_{n+1}-\tau)^2 & t_{n+1}-\tau\\(t_{n+1}-\tau)^2/2 & t_{n+1}-\tau & 1\end{bmatrix} d\tau, \qquad t_{n+1}-\tau = \gamma$$

$$= -\sigma^2 \int_T^0 \begin{bmatrix}\gamma^4/4 & \gamma^3/2 & \gamma^2/2\\\gamma^3/2 & \gamma^2 & \gamma\\\gamma^2/2 & \gamma & 1\end{bmatrix} d\gamma = \sigma^2 \begin{bmatrix}T^5/20 & T^4/8 & T^3/6\\T^4/8 & T^3/3 & T^2/2\\T^3/6 & T^2/2 & T\end{bmatrix}$$

Notice that $\mathbf{Q}(T)$ is full rank and that σ^2 is the variance of the white noise entering the system. The other covariance model we can develop from (2.55) is computed by assuming the white noise to be constant during each sample interval. The difference equation for this direct-discretized model is

$$x(t_{n+1}) = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} x(t_n) + \sigma w_3(t_n) \begin{bmatrix} T^3/6 \\ T^2/2 \\ T \end{bmatrix} \tag{2.57}$$

The process noise covariance in this direct discretized model is

$$\mathbf{Q}(T) = \mathrm{E}[\mathbf{B}w(t_n)(\mathbf{B}w(t_n))^{\mathrm{T}}] = \sigma^2 \begin{bmatrix} T^6/36 & T^5/12 & T^4/6 \\ T^5/12 & T^4/4 & T^3/2 \\ T^4/6 & T^3/2 & T^2 \end{bmatrix} \tag{2.58}$$

Notice that this matrix is rank deficient. The white noise variance is the same as in the discretized-continuous model.

2.2.4 Higher Order Models

The previous models can be extended to higher order systems. For example, we can model the motion of a target in 2 dimensions with 4 states $x, \dot{x}, y$ and $\dot{y}$. Modeling the motion in x and y as independent and using a direct-discrete model we get

$$\mathbf{A} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{B} = \sigma \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} \alpha & \beta \\ \beta & \gamma \end{bmatrix} \tag{2.59}$$

where correlated measurements are taken in x and y. The process noise covariance in this model is

$$\mathbf{Q}(T) = \mathrm{E}[\mathbf{B}w(t_n)(\mathbf{B}w(t_n))^{\mathrm{T}}] = \sigma^2 \begin{bmatrix} T^4/4 & T^3/2 & 0 & 0 \\ T^3/2 & T^2 & 0 & 0 \\ 0 & 0 & T^4/4 & T^3/2 \\ 0 & 0 & T^3/2 & T^2 \end{bmatrix} \qquad (2.60)$$

The state equations developed in this chapter for discrete systems, with coefficients $\mathbf{A}$ and $\mathbf{B}$, are used in simulations to model target motion. The state transition matrices and process noise covariances are used in sensor management techniques for solving for the optimum sample rate.

2.3 Estimating Process Noise Variance

The simulation tracking models derived in Section 2.2 have shown a development of the state transition matrices and process noise covariance matrices. The state transition matrix is a function of the sample period. The process noise covariance is a function of the sample period T and variance σ^2. Both T and σ^2 are design parameters. The choice of the variance of the continuous time process noise is used for "tuning" the model to the actual motion of the target [3].

This variance may be estimated based upon observed data and a priori knowledge concerning target dynamics. One technique developed by Singer [21] uses a mixed probability density for modeling target motion. For example if we model a target's acceleration in one dimension as a random variable and we have information concerning the target's maximum acceleration, the mixed density would be

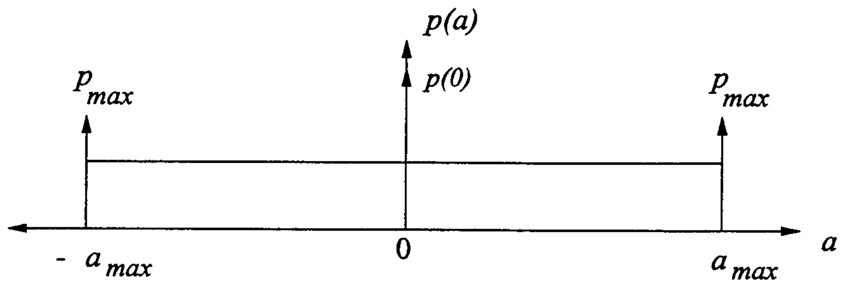


Figure 6 Mixed density for the target acceleration in one dimension

where the height of the continuous portion of the density is computed so that the area integrates to one. The acceleration random variable is zero mean with variance

$$\begin{aligned}
\sigma_a^2 &= \mathrm{E}[(a-\mu_a)^2] \\
&= \int_{-a_{max}}^{a_{max}} a^2 p(a) da \\
&= \int_{-a_{max}}^{a_{max}} a^2 (p_{max}\delta(a-a_{max}) + p_{max}\delta(a+a_{max}) + p(0)\delta(a))da + \\
&\quad \int_{-a_{max}}^{a_{max}} a^2 \frac{(1-p(0)-2p_{max})}{2a_{max}} da \\
&= 2p_{max}a_{max}^2 + \frac{(1-p(0)-2p_{max})}{2a_{max}}\left(\frac{a^3}{3}\right)\Bigg|_{-a_{max}}^{a_{max}} \\
&= 2p_{max}a_{max}^2 + \frac{(1-p(0)-2p_{max})}{2a_{max}}\frac{2a_{max}^3}{3} \\
&= \frac{a_{max}^2}{3}(1+4p_{max}-p(0))
\end{aligned} \tag{2.61}$$

Using this model the variance is shown to be a function of the maximum acceleration and the probabilities of the Dirac delta functions from the mixed probability density.

Chapter 3

CENTRALIZED SENSOR MANAGEMENT

The state error covariance of the Kalman filter can be described by the discrete Riccati equation. The discrete matrix Riccati equations are defined and then two optimization problems are introduced. We then list and give some properties of seven different covariance control metrics for either maintaining the prediction covariance or the update covariance.

The effect of sensor resolution on the steady-state solution to the Riccati equation is examined. The results of this analysis are then used to develop steady-state covariance control through the selection of sensor combinations. The analysis is then extended to the effect of sample period on the discrete Riccati equation for specific target models. This leads to the use of the sample period to maintain the prediction error covariance close to the desired prediction error covariance. The last two sections provide solutions to the scalar and matrix Riccati equations.

3.1 Centralized Kalman Filter

Within a certain set of assumptions, the Kalman filter is an optimal adaptive filter for combining old estimates with new measurements. The state and measurement equations are

$$x(t_{n+1}) = \mathbf{A}(t_n)x(t_n) + \mathbf{B}(t_n)w(t_n) \tag{3.1}$$

$$y(t_n) = \mathbf{C}(t_n)x(t_n) + \mathbf{D}(t_n)v(t_n) \tag{3.2}$$

The three types of estimation are prediction, filtering, and smoothing. In prediction problems we form estimates of a future state of the process, in filtering problems the estimates are of present values, and smoothing refers to forming estimates of past values of the state vector. These three types of estimation are given in (3.3), (3.4), and (3.5), respectively.

$$\mathrm{E}[x(t_n+T)|y(t_n), y(t_{n-1}), \ldots] \equiv \hat{x}(t_n+T|t_n, t_{n-1}, \ldots) \tag{3.3}$$

$$\mathrm{E}[x(t_n)|y(t_n), y(t_{n-1}), \ldots] \equiv \hat{x}(t_n|t_n, t_{n-1}, \ldots) \tag{3.4}$$

$$\mathrm{E}[x(t_n-T)|y(t_n), y(t_{n-1}), \ldots] \equiv \hat{x}(t_n-T|t_n, t_{n-1}, \ldots) \tag{3.5}$$

where ($\equiv$) means defined to be. The $T > 0$ indicates the future or past times at which the prediction or smoothed values are being estimated. Since the random process in (3.1) is a 1st order Markov process, we can uses Bayes' rule for conditional densities to simplify the expectations in (3.3), (3.4), and (3.5). Bayes' rule for the filtering problem in (3.4) can be simplified as follows

$$\begin{aligned} p(x_n|y_n, y_{n-1}, y_{n-2}, \ldots) &= \frac{p(x_n, y_n, y_{n-1}, y_{n-2}, \ldots)}{p(y_n, y_{n-1}, y_{n-2}, \ldots)} \\ &= p(x_n|y_n) = \frac{p(x_n, y_n)}{p(y_n)} \end{aligned} \tag{3.6}$$

so that the conditioning is only on the most recent measurement. A similar simplification can be done for the prediction and smoothed estimates. In the Kalman filter, the above simplification allows filter estimates to be formed from the latest received measurements.

We assume $\mathrm{E}[w(t_n)] = \mathrm{E}[v(t_n)] = \mathbf{0}$, $\mathrm{E}[\mathbf{B}w(t_n)w^{\mathrm{T}}(t_n)\mathbf{B}^{\mathrm{T}}] = \mathbf{Q}(t_n)$,

$\mathrm{E}[\mathbf{D}\boldsymbol{v}(t_n)\boldsymbol{v}^{\mathrm{T}}(t_n)\mathbf{D}^{\mathrm{T}}] = \mathbf{R}(t_n)$, and $\mathrm{E}[\boldsymbol{w}(t_n)\boldsymbol{v}^{\mathrm{T}}(t_n)] = \mathbf{0}$. The process noise covariance can be modeled in several different ways. The first method is called the discretized continuous model and the second method is called the direct discretized model. These two models were reviewed in Section 2.1 and Section 2.2. One form of the prediction and update equations in the Kalman filter is

Prediction Equations:

$$\hat{x}(t_n|t_{n-1}) = \mathbf{A}(t_n)\hat{x}(t_{n-1}|t_{n-1}) \tag{3.7}$$

$$\hat{y}(t_n|t_{n-1}) = \mathbf{C}(t_n)\hat{x}(t_n|t_{n-1}) \tag{3.8}$$

$$\mathbf{P}(t_n|t_{n-1}) = \mathbf{A}(t_n)\mathbf{P}(t_{n-1}|t_{n-1})\mathbf{A}^{\mathrm{T}}(t_n) + \mathbf{B}(t_n)\mathbf{Q}(t_n)\mathbf{B}^{\mathrm{T}}(t_n) \tag{3.9}$$

Update Equations:

$$\mathbf{P}^{-1}(t_n|t_n) = \mathbf{P}^{-1}(t_n|t_{n-1}) + \mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n)\mathbf{C}(t_n) \tag{3.10}$$

$$\mathbf{K}(t_n|t_n) = \mathbf{P}(t_n|t_n)\mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n) \tag{3.11}$$

$$\hat{x}(t_n|t_n) = \hat{x}(t_n|t_{n-1}) + \mathbf{K}(t_n)(y(t_n) - \hat{y}(t_n|t_{n-1})) \tag{3.12}$$

where,

$$\mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n)\mathbf{C}(t_n)$$

$$= \begin{bmatrix} \mathbf{C}_1(t_n) \\ \\ \mathbf{C}_M(t_n) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{R}_1^{-1}(t_n) & & \\ & \cdots & \\ & & \mathbf{R}_M^{-1}(t_n) \end{bmatrix} \begin{bmatrix} \mathbf{C}_1(t_n) \\ \\ \mathbf{C}_M(t_n) \end{bmatrix} = \sum_{j \in \Gamma} \mathbf{C}_j^{\mathrm{T}}(t_n)\mathbf{R}_j^{-1}(t_n)\mathbf{C}_j(t_n) \tag{3.13}$$

with Γ a subset of sensors of size $|\Gamma| = M$. This is the so called information matrix filter [3] because it uses the inverse of the information update from (3.10) to compute the Kalman gain in (3.11). This recursion defines the Kalman filter. The state update in (3.12) can also be expressed as

$$\hat{x}(t_n|t_n) = (\mathbf{I} - \mathbf{P}(t_n|t_n)\mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C})\mathbf{A}\hat{x}(t_{n-1}|t_{n-1}) + \mathbf{P}(t_n|t_n)\mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}y(t_n) \tag{3.14}$$

where the time dependence on **C** and **R** have been suppressed. The process noise covariance $\mathbf{Q}(t_n)$ and gain $\mathbf{B}(t_n)$ do not appear directly in (3.14). The matrices $\mathbf{Q}(t_n)$ and $\mathbf{B}(t_n)$ are subsumed into the state error covariance. This is a "coefficient" adaptive filter because the gains between the previous state estimate $\hat{x}(t_{n-1}|t_{n-1})$ and the new measurement $y(t_n)$ change based upon the state error covariance $\mathbf{P}(t_n|t_n)$.

The Kalman filter recursion *can not* be propagated off-line because it is dependent on new measurements. The state error covariance or state information matrix *can only* be propagated off-line given that it is known what sensors are used for each sample and what sample periods are used.

The discrete matrix Riccati equation (DMRE) is obtained by combining equations (3.9) and (3.10). In Section 3.6 and Section 3.7, solutions for the discrete scalar Riccati equation and DMRE are developed. A closed form solution of the DMRE is helpful because it allows quick computation of the steady state error covariance. Let us assume constant coefficients. With $\mathbf{P}(n+1) \equiv \mathbf{P}(t_{n+1}|t_n)$ and $\mathbf{P}(n) \equiv \mathbf{P}(t_n|t_{n-1})$, substituting the inverse of (3.10) into (3.9) and using the Matrix Inversion Lemma [3] to compute the inverse of the matrix addition, we get

$$\begin{aligned}\mathbf{P}(n+1) &= \mathbf{A}(\mathbf{P}^{-1}(n) + \mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C})^{-1}\mathbf{A}^{\mathrm{T}} + \mathbf{Q} \\ &= \mathbf{Q} + \mathbf{A}\mathbf{P}(n)\mathbf{A}^{\mathrm{T}} - \mathbf{A}\mathbf{P}(n)\mathbf{C}^{\mathrm{T}}(\mathbf{R} + \mathbf{C}\mathbf{P}(n)\mathbf{C}^{\mathrm{T}})^{-1}\mathbf{C}\mathbf{P}(n)\mathbf{A}^{\mathrm{T}}\end{aligned} \tag{3.15}$$

This is the DMRE for the prediction error covariance. In a similar manner, we can take equation (3.9) and substitute its inverse into (3.10). With $\mathbf{P}^{-1}(n) \equiv \mathbf{P}^{-1}(t_n|t_n)$ and $\mathbf{P}^{-1}(n-1) \equiv \mathbf{P}^{-1}(t_{n-1}|t_{n-1})$, we can write (3.10) as

$$\begin{aligned}\mathbf{P}^{-1}(n) &= (\mathbf{A}\mathbf{P}(n-1)\mathbf{A}^{\mathrm{T}} + \mathbf{Q})^{-1} + \mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C} \\ &= \mathbf{Q}^{-1} + \mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C} - \mathbf{Q}^{-1}\mathbf{A}(\mathbf{P}^{-1}(n-1) + \mathbf{A}^{\mathrm{T}}\mathbf{Q}^{-1}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Q}^{-1}\end{aligned} \tag{3.16}$$

where the Matrix Inversion Lemma [3] was used in a similar manner as in (3.15). Comparing these two equations, we notice that $\mathbf{P}(n)$ appears four times on the right side of (3.15), however $\mathbf{P}^{-1}(n-1)$ only appears once on the right side of (3.16). This shows how the information matrix recursions in (3.16) are more computationally efficient than the prediction error covariance recursions in (3.15), because the constant matrix terms in (3.16) can be computed off-line and stored. Propagation of these equations is one technique for finding a steady state solution for the DMRE.

3.2 Centralized Covariance Control Approach

The approach taken is to specify two desired covariance matrices, one for the desired prediction covariance $\mathbf{P}_{dp}$ and another for the desired update covariance $\mathbf{P}_{du}$. We can form two separate minimization problems by considering (3.9) and (3.10) separately: one minimization for computing the optimum rate and the other for computing the optimum resolution. The minimization problem associated with finding the optimum rate is

$$T_o(t_n) = \arg \underset{\forall T > 0}{min} \; f(\mathbf{P}(t_n + T|t_n), \mathbf{P}_{dp}(t_n)) \tag{3.17}$$

where $\mathbf{P}(t_n + T|t_n) = \mathbf{P}(t_{n+1}|t_n)$ is the prediction covariance given in (3.9) and $T_o(t_n)$ is the optimal sample period at time t_n based upon some metric $f(\bullet)$. The prediction covariance is a function of the sample period. Due to sensor limitations concerning maximum sample rate, the minimization also specifies a minimum sample period given by $T_{\min}$. This minimization is over the uncountable, infinite set T.

The minimization problem associated with finding the optimum sensor resolution (subset of sensors) is

$$\Gamma_o(t_n) = \arg \underset{\Gamma}{min} \; g(\mathbf{P}^{-1}(t_n|t_n), \mathbf{P}_{du}^{-1}(t_n)) \tag{3.18}$$

where $\mathbf{P}^{-1}(t_n|t_n)$ is given in (3.10) and is dependent on the choice of sensors Γ as determined by the sum in (3.13). The optimal set $\Gamma_o(t_n)$ at time t_n is based upon the minimization of some metric $g(\bullet)$. This minimization is over the countable, finite set Γ. When covariance matrices and information matrices are symmetric and positive definite, they can be represented by the level curves of there associated quadratic forms, where the square root of the eigenvalues are the lengths of the semi-axes and the eigenvectors indicate the directions of those axes, i.e. given $\mathbf{P} > 0$ the ellipse of $\mathbf{P}$ is $ellipse(\mathbf{P}) \equiv \{x; x^{\mathrm{T}}\mathbf{P}^{-1}x = 1\}$. All subsequent ellipses are generated using this function. Two ellipses are plotted in Figure 7 showing an elliptical annulus for the two desired covariance matrices.

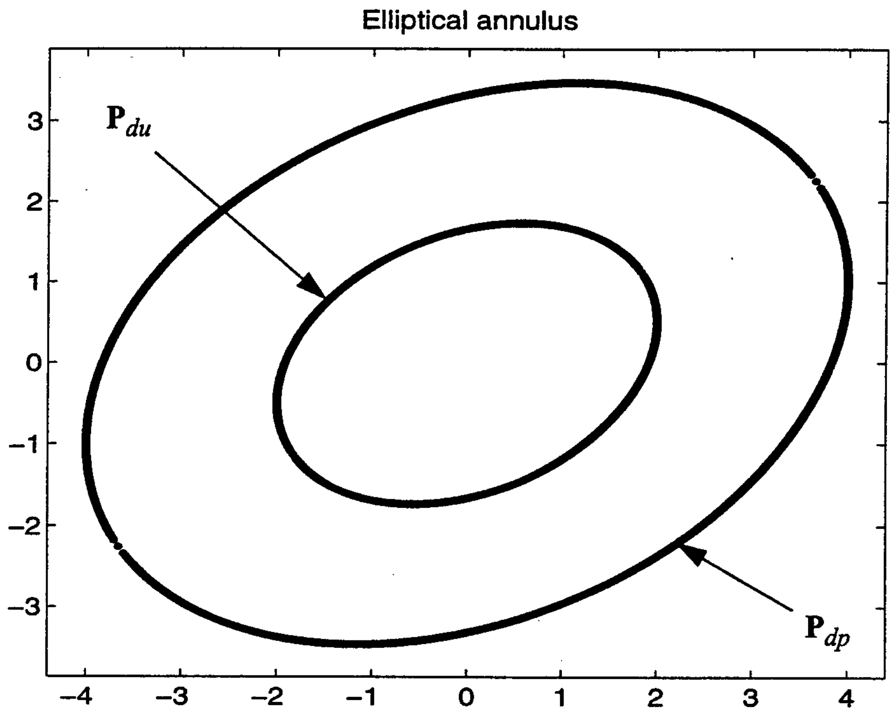


Figure 7 Elliptical annulus describing desired update covariance and desired prediction covariance

Since many data analysis algorithms require rigorous upper bounds on estimation error, we primarily considered covariance control techniques that drive the actual covariances inside the desired covariances. Other techniques involve maintaining the actual error covariance within certain bounds of the desired error covariance. The plots in Figure 8 show several of these ellipses where the dark solid line indicates a desired

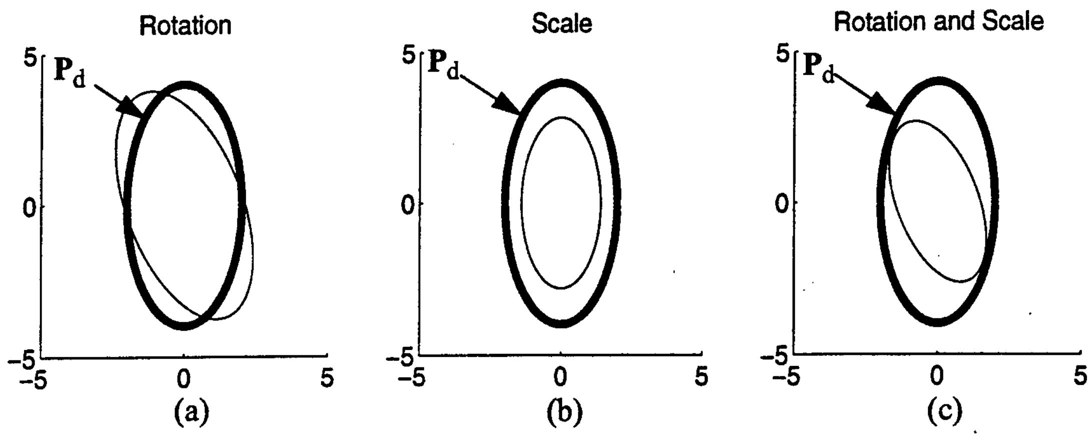


Figure 8 Illustration of (a) rotation, (b) scaling, and (c) rotation and scaling covariance and the thin solid line indicates different update or prediction covariances. Plot (a) shows a rotation of the desired covariance, plot (b) shows a scaling of the desired covariance, and plot (c) shows a combination of these two where the desired covariance is rotated and scaled.

Some metrics we have considered for comparing two covariance matrices are the matrix 2-norm, Frobenius norm, traces, determinants, relative entropy, and metrics based upon the singular value decomposition (SVD). In general, each metric may be considered for controlling either the prediction or update covariance (information) matrices. Let $\mathbf{P}_d$ be a symmetric positive definite desired prediction or update covariance matrix i.e. $\mathbf{P}_d = \mathbf{P}_d^{\mathrm{T}} > 0$. Let $\mathbf{P}$ be a symmetric positive semidefinite prediction or update covariance matrix, i.e. $\mathbf{P} = \mathbf{P}^{\mathrm{T}} \geq 0$.

Each metric is a function of these two matrices. The first four metrics use the matrix difference, $\mathbf{M} = \mathbf{P}_d - \mathbf{P}$. The last metric uses the matrix product, $\mathbf{N} = \mathbf{P}\mathbf{P}_d^{-1}$. The two metrics in (3.27) and (3.28) use the SVD where the decompositions for the desired covariance and update covariance are

$$\mathbf{P}_d = \mathbf{U}_d\mathbf{S}_d\mathbf{U}_d^* \qquad (3.19)$$

and

$$\mathbf{P} = \mathbf{USU^*} \tag{3.20}$$

respectively. The *singular value vectors* for the desired and update covariances are defined to be

$$\Sigma_d = diag(\mathbf{S}_d) \in \Re^{N \times 1} \tag{3.21}$$

and

$$\Sigma = diag(\mathbf{S}) \in \Re^{N \times 1} \tag{3.22}$$

respectively, where *diag*(•) denotes a vector containing the diagonal elements of the matrix contained in the argument. Denoting the absolute value as |•|, the determinant function as *det*(•), and the trace function as *tr*(•), the seven metrics we considered are:

$$\textit{2-norm},\ G(\mathbf{M}) = \|\mathbf{M}\|_2 = \sigma_1(\mathbf{M}) \tag{3.23}$$

$$\textit{Frobenius norm},\quad F(\mathbf{M}) = \|\mathbf{M}\|_F = \sqrt{\sum_{i=1}^{N} \sigma_i^2(\mathbf{M})} = \sqrt{tr(\mathbf{M}^2)} \tag{3.24}$$

$$\textit{Trace function},\ T(\mathbf{M}) = |tr(\mathbf{M})| = \left|\sum_{i=1}^{N} \lambda_i(\mathbf{M})\right| = \left|\sum_{i=1}^{N} \mathbf{M}_{ii}\right| \tag{3.25}$$

$$\textit{Determinant function},\ A(\mathbf{M}) = |det(\mathbf{M})| = \left|\prod_{i=1}^{N} \lambda_i(\mathbf{M})\right| \tag{3.26}$$

$$\textit{Projection metric},\ J(\mathbf{P}, \mathbf{P}_d) = \frac{N - tr(|\mathbf{U}_d\mathbf{U}^*|)}{N} + \|\Sigma_d - \Sigma\|_2 \tag{3.27}$$

$$\textit{Scale Invariant Projection metric},\ K(\mathbf{P}, \mathbf{P}_d) = \frac{N - tr(|\mathbf{U}_d\mathbf{U}^*|)}{N} + \left(1 - \frac{\Sigma_d^T\Sigma}{\|\Sigma_d\|\|\Sigma\|}\right) \tag{3.28}$$

$$\textit{Kullback-Liebler metric},\ L(\mathbf{N}) = \frac{1}{2}(tr(\mathbf{N}) - N - \ln(det(\mathbf{N}))) \tag{3.29}$$

The seven metrics excluding (3.27) and (3.28) are commonly used and their properties are well documented. Metrics (3.27) and (3.28) were specifically developed for covariance control based upon maintaining the direction of the semiaxes of an ellipse and the length of those axes. Since these two metrics are new we prove several properties of these metrics in this section.

The first two functions are norms because they satisfy the following properties [23]

$$\begin{aligned} &1)\ f(\mathbf{A}) \geq 0, \text{ with equality iff } \mathbf{A} = \mathbf{0} \\ &2)\ f(\alpha\mathbf{A}) = |\alpha| f(\mathbf{A}) \\ &3)\ f(\mathbf{A}+\mathbf{B}) \leq f(\mathbf{A}) + f(\mathbf{B}) \end{aligned} \tag{3.30}$$

where $\mathbf{A}, \mathbf{B} \in \Re^{N \times N} \equiv S$ and $f(\bullet): S \rightarrow \Re$. The last property is the triangle inequality. The remaining functions are not norms because they do not satisfy these three properties. All seven functions are greater than or equal to zero. Metrics also have a rigorous definition. Metrics are a binary operator defined by a mapping $g(\bullet,\bullet): (S, S) \rightarrow \Re$ that measures the "distance" between two points in S. Metrics have the following properties [23]

$$\begin{aligned} &1)\ g(\mathbf{A}, \mathbf{B}) \geq 0, \text{ with equality iff } \mathbf{A} = \mathbf{B} \\ &2)\ g(\mathbf{A}, \mathbf{B}) = g(\mathbf{B}, \mathbf{A}) \\ &3)\ g(\mathbf{A}, \mathbf{C}) \leq g(\mathbf{A}, \mathbf{B}) + g(\mathbf{B}, \mathbf{C}) \end{aligned} \tag{3.31}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \Re^{N \times N} \equiv S$. Since, the last three functions do not satisfy all the conditions of a metric, they might be more appropriately called "pseudometrics". However, since all the functions satisfy some of the conditions, for simplicity, we will generically call each of the functions metrics or "distance" measures.

The seven metrics can be used in combination with checking the condition $\lambda_{min}(\mathbf{M}) > 0$, which will insure that the error covariance is less than the desired error covariance. These metrics are not equivalent when comparing the difference in covariance matrices versus the difference in their corresponding information matrices.

To illustrate how these metrics perform, we considered a suite of 7 sensors with their associated sensor information matrices in $\Re^{2x2}$. Each metric in (3.23)-(3.29) was computed for all 2^7 sensor combinations. The set of sensors that minimized each respective metric was chosen as the optimal sensor combination. Figure 9 shows the set of 7 measurement information ellipses using thin solid lines and the level curve of the desired information update as a thick solid line. The ellipses representing the information matrix for each sensor are labeled as $\mathbf{R}_i^{-1}$, with i=1, 2,..., 7 denoting the sensor number.

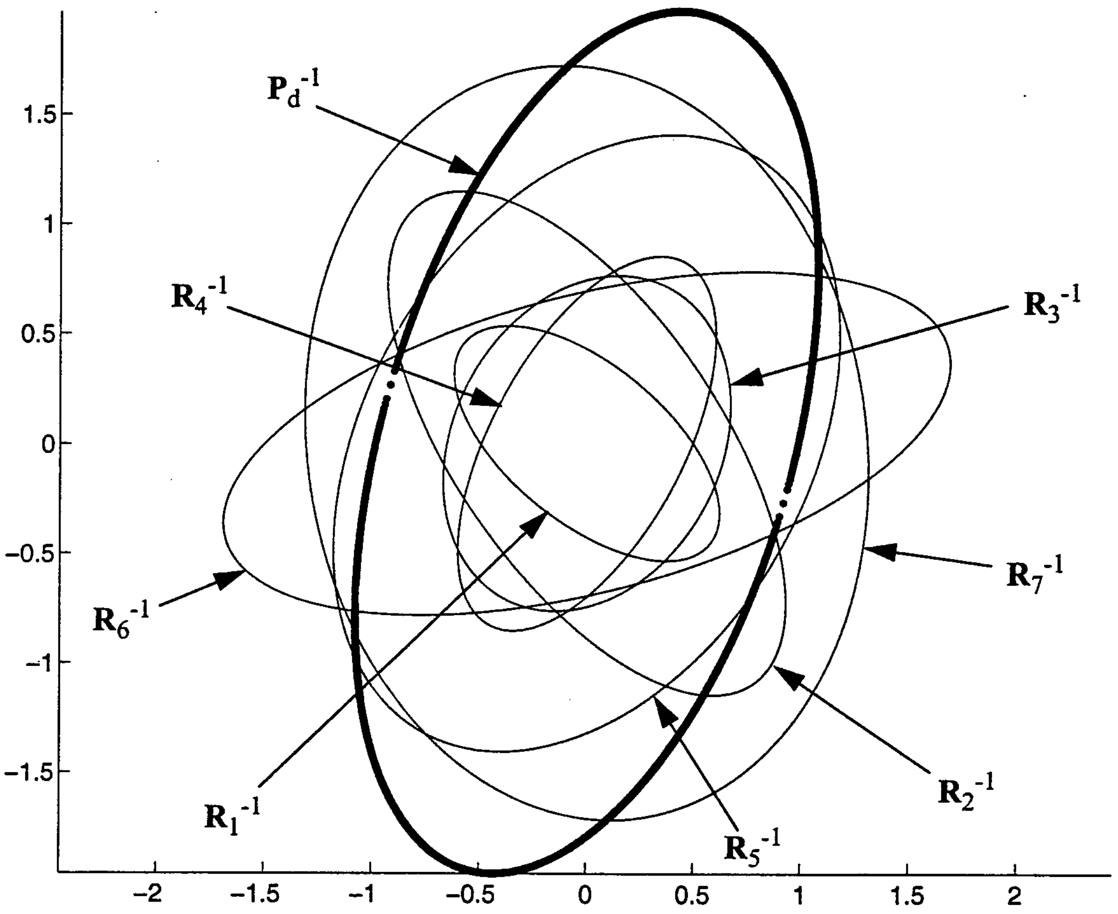


Figure 9 Ellipses of desired information and sensor information matrices

Figure 10 shows the optimal sensor combination at the top of each subplot for the seven metrics. The ellipses associated with the desired information matrix is indicated by the *thick* solid lines and the ellipses associated with the optimal sensor combination is indicated by the *thin* solid lines.

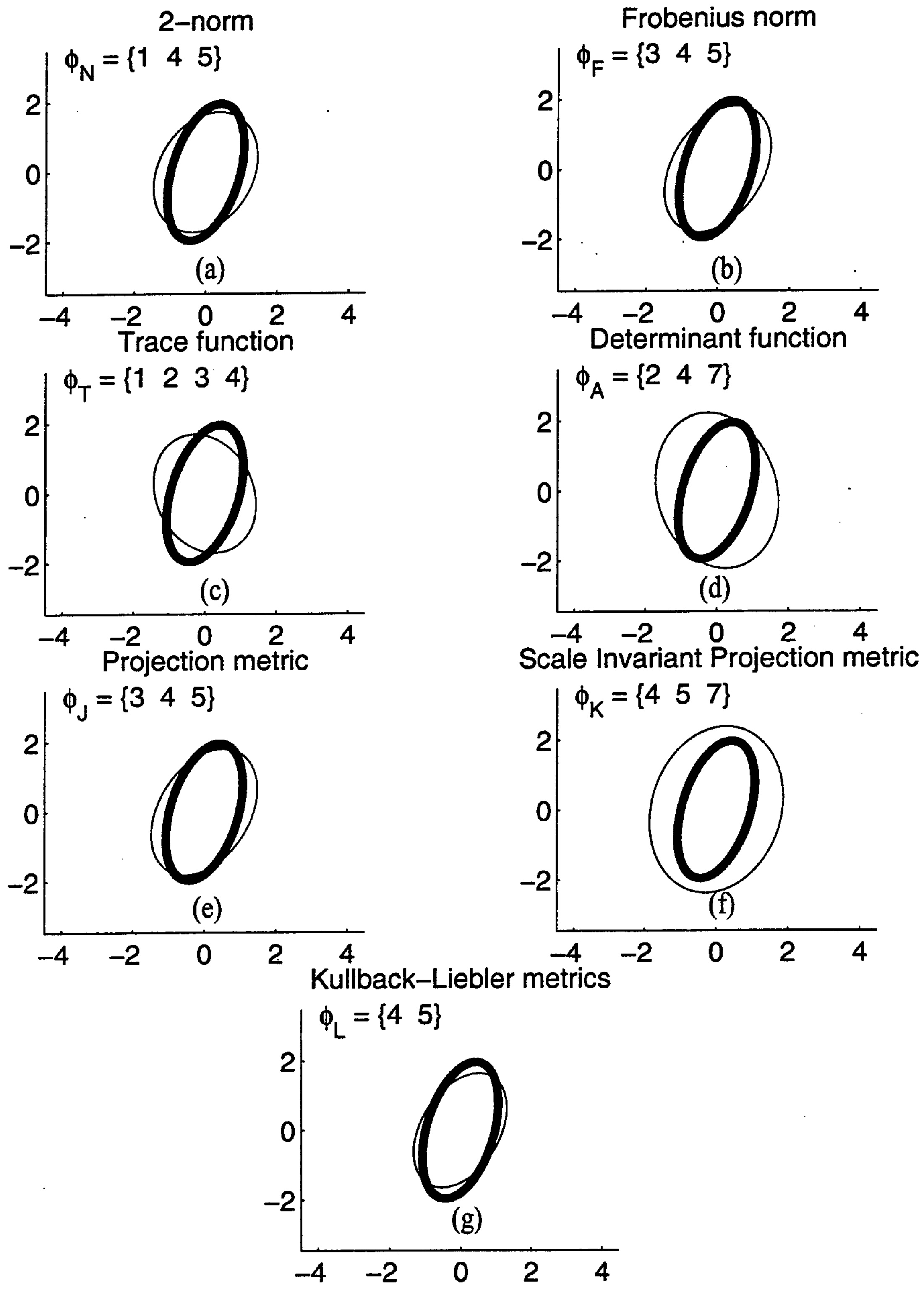


Figure 10 Illustration of covariance control measures: (a) 2-norm, (b) Frobenius norm, (c) Trace function, (d) Determinant function, (e) Projection metric, (f) Scale Invariant Projection metric, and (g) Kullback-Liebler metric

The *2-norm* and the *Frobenius norm* exhibit similar properties when selecting sets of sensors. The Frobenius norm of a particular set of matrices is always greater than or equal to the 2-norm of that same set. This is because the 2-norm is equal to the largest singular value while the Frobenius norm adds all of the singular values, including the largest.

The *trace function* has the lowest computational complexity because it only requires the sum of the diagonal elements of $\mathbf{M}$. This computation is order N.

The *determinant function* is based on using the determinant and computes the area of the differences. Plot d of Figure 10 shows the area metric. This metric exhibits a property where the selected ellipse is closest to being tangent (at two points) to the desired ellipse. When the two level curves are tangent at two points, the matrix difference is not full rank and the quadratic form will be either positive semidefinite or negative semidefinite, depending on which level curve is contained within the other. Because the semidefinite matrix has an eigenvalue equal to zero, the computation of the area metric in (3.26) is zero. The area metric is suitable for minimizing one of the singular values, however, a small minimum eigenvalue can mask the larger eigenvalues, when multiplied together.

The *projection metric* combines a trace operation with a vector norm. The vectors Σ_d and Σ contain the singular values of $\mathbf{P}_d$ and $\mathbf{P}$, respectively, and are called the *singular value vectors*. The term $\|\Sigma_d - \Sigma\|_2$ measures the distance between the singular value vectors. This metric is labeled as being a projection because the trace operation takes each singular vector associated with one matrix (ordered according to largest to smallest singular value) and projects them onto the singular vectors of the other matrix. The trace and vector norm are both order N computations. Most of the computation of this metric is involved with computing the SVD of each of the covariance matrices. The SVD of the desired covariances $\mathbf{P}_d$ can be computed off-line so that the computation involved with computing the SVD could be reduced by half during real-time operation. The projection metric has the following properties

1. $J(\mathbf{P}, \mathbf{P}_d) = \dfrac{N - tr(|\mathbf{U}_d\mathbf{U}^*|)}{N} + \|\Sigma_d - \Sigma\|_2 \geq 0$ with equality iff $\mathbf{P} = \mathbf{P}_d$

2. $J(\alpha\mathbf{P}, \alpha\mathbf{P}_d) = \dfrac{N - tr(|\mathbf{U}_d\mathbf{U}^*|)}{N} + |\alpha|\|\Sigma_d - \Sigma\|_2 \quad \forall\alpha \in \Re$

3. $J(\mathbf{P}, \mathbf{P}_d) = J(\mathbf{P}_d, \mathbf{P})$

Proof:

Let the norm and trace functions be denoted by $h(\Sigma_d, \Sigma) = \|\Sigma_d - \Sigma\|_2$ and $g(\mathbf{U}_d, \mathbf{U}) = (N - tr(|\mathbf{U}_d\mathbf{U}^*|))/N$, respectively. The matrices composed of the singular vectors in the SVD are $\mathbf{U}_d = \begin{bmatrix} \boldsymbol{u}^*_{d1} & \dots & \boldsymbol{u}^*_{dN} \end{bmatrix}^*$ and $\mathbf{U} = \begin{bmatrix} \boldsymbol{u}^*_1 & \dots & \boldsymbol{u}^*_N \end{bmatrix}^*$. Using the Cauchy-Schwarz inequality we can show the function $g(\mathbf{U}_d, \mathbf{U})$ is bounded by zero

$$tr(|\mathbf{U}_d\mathbf{U}^*|) = \sum_{i=1}^{N} |\boldsymbol{u}_{di}\boldsymbol{u}_i^*| \leq \sum_{i=1}^{N} \|\boldsymbol{u}_{di}\|\|\boldsymbol{u}_i\| = N \tag{3.32}$$

with equality iff $\boldsymbol{u}_{di}$ and $\boldsymbol{u}_i$ are each linearly dependent. This proves the first property.

The second property is true because $g(\mathbf{U}_d, \mathbf{U})$ is invariant to scaling of $\mathbf{P}$ and $\mathbf{P}_{du}$; and $h(\Sigma_d, \Sigma)$ is a norm with the property $h(\alpha\Sigma_d, \alpha\Sigma) = |\alpha|h(\Sigma_d, \Sigma)$. The second property shows that the projection metric is an affine function when scaling both covariances. The third property can be shown by recognizing that $tr(|\mathbf{U}_d\mathbf{U}^*|) = tr(|\mathbf{U}\mathbf{U}_d^*|)$ and that $\|\Sigma_d - \Sigma\|_2 = \|\Sigma - \Sigma_d\|_2$.

The *scale invariant projection metric* attempts to match the alignment and shape of the ellipsoids of $\mathbf{P}$ and $\mathbf{P}_d$. The singular value vectors Σ and Σ_d describe the "shape" of the ellipsoids (relative magnitudes of the eigenvalues) and the vectors contained in $\mathbf{U}$ and $\mathbf{U}_d$ define the relative orientation of these ellipsoids in state space.

The term $1-\frac{\Sigma_d^T\Sigma}{\|\Sigma_d\|\|\Sigma\|}$ is minimized when the "shape" of the two ellipsoids are the same and $(N-tr(|\mathbf{U}_d\mathbf{U}^*|))/N$ is minimized when the ellipsoids are aligned. Four properties of the scale invariant projection metric are

1. $K(\mathbf{P}, \mathbf{P}_d)=\frac{N-tr(|\mathbf{U}_d\mathbf{U}^*|)}{N}+\left(1-\frac{\Sigma_d^T\Sigma}{\|\Sigma_d\|\|\Sigma\|}\right)\geq 0$ with equality iff $\mathbf{P}_d=\alpha\mathbf{P}$

2. $2>K(\mathbf{P}, \mathbf{P}_d)\geq 0$

3. $K(\alpha\mathbf{P}, \beta\mathbf{P}_d) = K(\mathbf{P}, \mathbf{P}_d) \quad \forall\alpha, \beta\in\Re$

4. $J(\mathbf{P}, \mathbf{P}_d) = J(\mathbf{P}_d, \mathbf{P})$

Proof:

Let the cosine and trace functions be denoted by $h(\Sigma_d, \Sigma) = 1-\frac{\Sigma_d^T\Sigma}{\|\Sigma_d\|\|\Sigma\|}$ and $g(\mathbf{U}_d, \mathbf{U}) = (N-tr(|\mathbf{U}_d\mathbf{U}^*|))/N$, respectively. The trace function $g(\mathbf{U}_d, \mathbf{U})$ is the same as in the projection metric. The cosine function $h(\Sigma_d, \Sigma)$, is non-negative because $1-\frac{\Sigma_d^T\Sigma}{\|\Sigma_d\|\|\Sigma\|} = 1-\cos(\theta)\geq 0$ with equality iff $\Sigma_d = \alpha\Sigma$. The cosine of the angle between the singular value vectors is non-negative because the singular value vectors lie in the all positive orthant. This proves the first property.

The second property may be shown by maximizing $g(\mathbf{U}_d, \mathbf{U})$ and $h(\Sigma_d, \Sigma)$, respectively. When the product $\mathbf{U}_d\mathbf{U}^*$ has zeros on the diagonal then the function $g(\mathbf{U}_d, \mathbf{U})$ achieves a maximum of one. Now let $\Sigma_d = \begin{bmatrix}1 & \dots & 1\end{bmatrix}^T$ and $\Sigma = \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}^T$. Then $\lim_{N\to\infty}\frac{\Sigma_d^T\Sigma}{\|\Sigma_d\|\|\Sigma\|} = 0$, so that $h(\Sigma_d, \Sigma)$ achieves a maximum equal to 1. This means that the metric $K(\mathbf{P}, \mathbf{P}_d)$ is upper bounded by 2 as the vectors become infinite in length. When the matrices belong to a finite dimensional space the upper bounds will be less than 2. For example with $\mathbf{P}, \mathbf{P}_d\in\Re^{2x2}$, the maximum angle between the singular value vectors is 45 degrees in which case the maximum of $h(\Sigma_d, \Sigma)$ is $1-\cos(45) = 1-(\sqrt{2})/2\cong 0.2929$ and the maximum of $K(\mathbf{P}, \mathbf{P}_d)$ is 1.2929.

The third property is true because scaling either matrix only scales the singular value vectors which does not change the angle between the vectors. The scale invariant projection metric could be useful for choosing the resolution of measurements independent of the rate of measurements. The last property is true because $tr(|\mathbf{U}_d\mathbf{U}^*|) = tr(|\mathbf{U}\mathbf{U}_d^*|)$ and $h(\Sigma_d, \Sigma) = h(\Sigma, \Sigma_d)$.

The *Kullback-Liebler metric* measures the relative entropy between two random vectors with associated densities. We use it to measure the relative entropy between a desired density and the actual density of the state vector. The Kullback-Liebler metric is derived from the definition of relative entropy. The relative entropy is the N fold integration

$$L(f, g) = \int \dots \int g(x_1, \dots, x_N) \ln\left(\frac{g(x_1, \dots, x_N)}{f(x_1, \dots, x_N)}\right) dx_1 \dots dx_N \tag{3.33}$$

where $f(\boldsymbol{x})$ is the desired density and $g(\boldsymbol{x})$ is the actual density of random vector $\boldsymbol{x}$. Let $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ have the following Gaussian densities

$$f(\boldsymbol{x}) = (2\pi)^{-N/2} det(\mathbf{P}_d)^{-1/2} \exp\left(-\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\mathbf{P}_d^{-1}\boldsymbol{x}\right) \tag{3.34}$$

$$g(\boldsymbol{x}) = (2\pi)^{-N/2} det(\mathbf{P})^{-1/2} \exp\left(-\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\mathbf{P}^{-1}\boldsymbol{x}\right) \tag{3.35}$$

Using a simpler notation to represent the N fold integration and N arguments we have

$$\begin{aligned} L(f, g) &= \int g(\boldsymbol{x}) \ln\left(\frac{g(\boldsymbol{x})}{f(\boldsymbol{x})}\right) d\boldsymbol{x} \\ &= \int g(\boldsymbol{x}) \ln g(\boldsymbol{x}) d\boldsymbol{x} - \int g(\boldsymbol{x}) \ln f(\boldsymbol{x}) d\boldsymbol{x} \end{aligned} \tag{3.36}$$

Expressing the quadratic forms in $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ as $p(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\mathbf{P}_d^{-1}\boldsymbol{x}$ and

$q(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\mathbf{P}^{-1}\boldsymbol{x}$, respectively, the first integral in (3.36) is

$$\begin{aligned}\int g(\boldsymbol{x})\ln g(\boldsymbol{x})d\boldsymbol{x} &= \int g(\boldsymbol{x})\ln[(2\pi)^{-N/2}det(\mathbf{P})^{-1/2}\exp(q(\boldsymbol{x}))]d\boldsymbol{x}\\ &= \int q(\boldsymbol{x})g(\boldsymbol{x})d\boldsymbol{x}-\frac{N}{2}\ln(2\pi)\int g(\boldsymbol{x})d\boldsymbol{x}-\frac{1}{2}\ln(det(\mathbf{P}))\int g(\boldsymbol{x})d\boldsymbol{x}\\ &= -\frac{1}{2}trace(\mathbf{P}^{-1}\mathbf{P})-\frac{N}{2}\ln(2\pi)-\frac{1}{2}\ln(det(\mathbf{P}))\\ &= -\frac{N}{2}-\frac{N}{2}\ln(2\pi)-\frac{1}{2}\ln(det(\mathbf{P}))\end{aligned} \tag{3.37}$$

The second integral in (3.36) is

$$\begin{aligned}-\int g(\boldsymbol{x})\ln(f(\boldsymbol{x}))d\boldsymbol{x} &= -\int g(\boldsymbol{x})\ln[(2\pi)^{-N/2}det(\mathbf{P}_d)^{-1/2}\exp(p(\boldsymbol{x}))]d\boldsymbol{x}\\ &= -\int p(\boldsymbol{x})g(\boldsymbol{x})d\boldsymbol{x}+\frac{N}{2}\ln(2\pi)\int g(\boldsymbol{x})d\boldsymbol{x}+\frac{1}{2}\ln(det(\mathbf{P}_d))\int g(\boldsymbol{x})d\boldsymbol{x}\\ &= \frac{1}{2}trace(\mathbf{P}\mathbf{P}_d^{-1})+\frac{N}{2}\ln(2\pi)+\frac{1}{2}\ln(det(\mathbf{P}_d))\end{aligned} \tag{3.38}$$

where $\int q(\boldsymbol{x})g(\boldsymbol{x})d\boldsymbol{x}$ and $\int p(\boldsymbol{x})g(\boldsymbol{x})d\boldsymbol{x}$ are computed by using properties of the trace operation. The trace of a scalar function is the same function and the trace commutes with expectations.

$$\begin{aligned}\int q(\boldsymbol{x})g(\boldsymbol{x})d\boldsymbol{x} &= \mathrm{E}[q(\boldsymbol{x})] = trace(\mathrm{E}[q(\boldsymbol{x})])\\ &= -\frac{1}{2}(\mathrm{E}[trace(\boldsymbol{x}^{\mathrm{T}}\mathbf{P}^{-1}\boldsymbol{x})])\\ &= -\frac{1}{2}(\mathrm{E}[trace(\mathbf{P}^{-1}\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}})])\\ &= -\frac{1}{2}trace(\mathbf{P}^{-1}\mathrm{E}[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}])\\ &= -\frac{1}{2}trace(\mathbf{P}^{-1}\mathbf{P}) = -\frac{N}{2}\end{aligned} \tag{3.39}$$

Performing the same computation on the second integral we get

$$\begin{aligned}\int p(\boldsymbol{x})g(\boldsymbol{x})d\boldsymbol{x} &= \mathrm{E}[p(\boldsymbol{x})] = trace(\mathrm{E}[p(\boldsymbol{x})]) \\ &= -\frac{1}{2}(\mathrm{E}[trace(\boldsymbol{x}^{\mathrm{T}}\mathbf{P}_d^{-1}\boldsymbol{x})]) \\ &= -\frac{1}{2}(\mathrm{E}[trace(\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\mathbf{P}_d^{-1})]) \\ &= -\frac{1}{2}trace(\mathrm{E}[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}]\mathbf{P}_d^{-1}) \\ &= -\frac{1}{2}trace(\mathbf{P}\mathbf{P}_d^{-1})\end{aligned} \tag{3.40}$$

Recognizing that the new argument can be expressed as $\mathbf{P}\mathbf{P}_d^{-1}$, the final integral in (3.36) has the form

$$\begin{aligned}L(\mathbf{P}\mathbf{P}_d^{-1}) &= -\frac{N}{2}-\frac{1}{2}\ln(det(\mathbf{P})) + \frac{1}{2}trace(\mathbf{P}{\mathbf{P}_d}^{-1}) + \frac{1}{2}\ln(det(\mathbf{P}_d)) \\ &= \frac{1}{2}(trace(\mathbf{P}\mathbf{P}_d^{-1}) - N - \ln(det(\mathbf{P}\mathbf{P}_d^{-1})))\end{aligned} \tag{3.41}$$

which corresponds to the metric given in (3.29). This metric can be shown to be positive. Let the eigenvalues of $\mathbf{P}\mathbf{P}_d^{-1}$ be λ_i. The equation in (3.41) is then

$$L(\mathbf{P}\mathbf{P}_d^{-1}) = \frac{1}{2}\left(\sum_{i=1}^{N}\lambda_i - N - \sum_{i=1}^{N}\ln(\lambda_i)\right) = \sum_{i=1}^{N}[\lambda_i - \ln(\lambda_i)] - N \tag{3.42}$$

When the eigenvalues of the matrix product are positive the argument of the summation $\lambda_i - \ln(\lambda_i)$ is always greater than one requiring the sum to be greater than *N*. When the eigenvalues are all one then the metric is zero. This metric was developed from information theoretic principles and combines the trace and determinant functions. Relative information is used in [19] to compute sensor information gains. The development in [19] does not use a desired covariance, but rather tries to maximize the one step information gain. Without the use of a desired covariance, more sensing resources could be allocated to tracking a target then might otherwise be needed.

3.3 Rate and Resolution Analysis

The covariance prediction in (3.9) and the information update in (3.10) are used individually to study how the *sampling rate* and *sensor resolution* contribute to the error covariance. These equations, with arbitrary sample times, are

$$\mathbf{P}^{-1}(t_n|t_n) = \mathbf{P}^{-1}(t_n|t_{n-1}) + \mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C} \tag{3.43}$$

$$\mathbf{P}(t_{n+1}|t_n) = \mathbf{A}(T)\mathbf{P}(t_n|t_n)\mathbf{A}^{\mathrm{T}}(T) + \mathbf{Q}(T) \tag{3.44}$$

When the sensor measurements are uncorrelated, the information gain $\mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C}$ has a block diagonal structure. The information gain can be represented as

$$\mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 \\ \cdots \\ \mathbf{C}_M \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{R}_1^{-1} & & 0 \\ & \cdots & \\ 0 & & \mathbf{R}_M^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{C}_1 \\ \cdots \\ \mathbf{C}_M \end{bmatrix} \tag{3.45}$$

where $\mathbf{C}_i \in \Re^{m_i \times N}$, $\mathbf{R}_i^{-1} \in \Re^{m_i \times m_i}$, and $i \in \{1, \ldots, M\}$. When all the sensors are used then $\mathbf{C} \in \Re^{p \times N}$ and $\mathbf{R}^{-1} \in \Re^{p \times p}$ where $p = \sum_{i=1}^{M} m_i$. When we consider using different sensor combinations, the dimensions of the sensor information matrix $\mathbf{R}^{-1}$ and measurement matrix $\mathbf{C}$ will change depending on which sensors are used and their associated dimensions.

The information update in (3.43) increases monotonically with the addition of more sensors, i.e. $\mathbf{P}^{-1}(t_n|t_n) \geq \mathbf{P}^{-1}(t_n|t_{n-1})\ \forall \mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C} \geq 0$. Figure 11a illustrates a set of monotonically increasing ellipses for information matrices in $\Re^{2 \times 2}$. The sets S_i have the properties $S_1 \subset \ldots \subset S_i \subset \ldots \subset S_N$ and $|S_i| = i$ where $|S_i|$ denotes number of sensors in the *i*th set. The innermost ellipse in plot (a) of Figure 11 corresponds to the ellipse of the information update using only one sensor. Each successive ellipse proceeding outwards is a result of using one additional sensor. The outermost ellipse uses all six sensors.

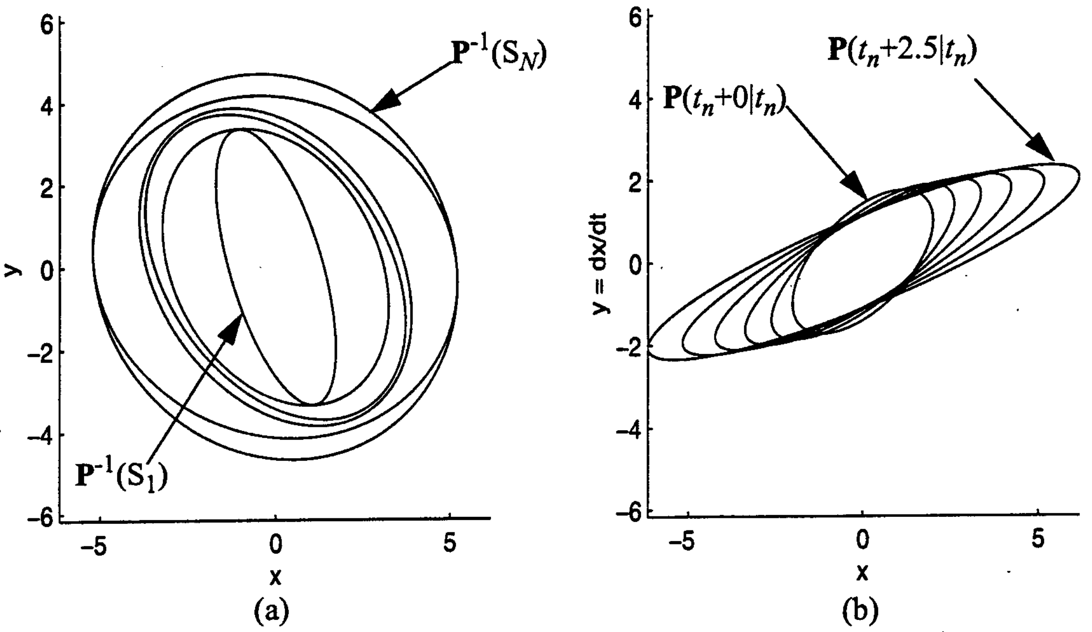


Figure 11 Monotonically increasing ellipses with varying resolution and rate - (a) Ellipses representing matrices achieving higher resolution (smaller covariance) as the number of sensors increases, (b) Prediction error covariance ellipses as a function of the sample period

The prediction covariance increases monotonically with the sample period if $\mathbf{P}(t_n + T_2|t_n) \geq \mathbf{P}(t_n + T_1|t_n)\ \forall T_2 > T_1$. This implies that the eigenvalues of $\mathbf{P}(t_n + T_2|t_n) - \mathbf{P}(t_n + T_1|t_n) \geq 0$ are greater than or equal to zero. The initial update covariance is one factor that determines if the prediction covariance increases monotonicly. Figure 11b shows non-monotonicly "growing" ellipses where $\mathbf{A}(T)$, $\mathbf{Q}(T)$, and $\mathbf{P}(t_n|t_n)$ are given by

$$\mathbf{A}(T) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \mathbf{Q}(T) = \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix}\sigma_c^2, \mathbf{P}(t_n|t_n) = \begin{bmatrix} 4 & 2 \\ 2 & 3 \end{bmatrix} \tag{3.46}$$

The 'innermost' ellipse of plot (b) corresponds to letting $T = 0$. Each successive ellipse proceeding outwards increases the sample period by 0.5. The 'outside' ellipse of plot (b) corresponds to letting $T = 2.5$.

Control over both the *sampling rate* (T^{-1}) and *sensor resolution* ($\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}$) provides two methods for maintaining the desired covariance. The first method is to fix T and then solve for an optimal $\mathbf{R}^{-1}$ which represents the best subset of sensors to use from the available suite of sensors. The second method is to fix the sensor resolution $\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}$ and then solve for an optimal T. These covariance control techniques are outlined in the following two subsections.

3.3.1 Steady State Covariance Control Using Sensor Resolution

In a centralized, open loop covariance control scheme, where the rate is fixed, sets of sensors are chosen at each interval such that the steady state error covariance is made to be close in some sense to some desired covariance. During the transient period the error signals are nonstationary and the error convergence is completely determined by the Riccati equation associated with the Kalman filter.

Using this formulation we can find the theoretically optimal sensor set for maintaining the desired covariance given a fixed sampling rate. Using the Riccati equation in (3.15), it is possible to solve for the measurement covariance. We first choose a nominal sample period T, such that the state transition matrix $\mathbf{A}(T)$ and process noise covariance matrix $\mathbf{Q}(T)$ are held constant. If the full state vector is measured such that $\mathbf{C} = \mathbf{I}$, we can solve for an optimal measurement covariance $\mathbf{R}$ or sensor resolution $\mathbf{R}^{-1}$ corresponding to the desired covariance. Now let the steady state prediction error covariance equal the desired prediction error covariance

$\mathbf{P}(n+1) = \mathbf{P}(n) = \mathbf{P}_{dp}$.

$$\begin{aligned}\mathbf{P}_{dp} &= \mathbf{Q} + \mathbf{A}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} - \mathbf{A}\mathbf{P}_{dp}\mathbf{C}^{\mathrm{T}}(\mathbf{R} + \mathbf{C}\mathbf{P}_{dp}\mathbf{C}^{\mathrm{T}})^{-1}\mathbf{C}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} \\ \mathbf{A}\mathbf{P}_{dp}(\mathbf{R} + \mathbf{P}_{dp})^{-1}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} &= \mathbf{Q} + \mathbf{A}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} - \mathbf{P}_{dp} \\ (\mathbf{R} + \mathbf{P}_{dp})^{-1} &= (\mathbf{A}\mathbf{P}_{dp})^{-1}(\mathbf{Q} + \mathbf{A}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} - \mathbf{P}_{dp})(\mathbf{A}\mathbf{P}_{dp})^{-\mathrm{T}}\end{aligned} \tag{3.47}$$

Computing the inverse of the last equation in (3.47) we have the optimal covariance denoted by $\mathbf{R}_o$

$$\mathbf{R}_o = \mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}}(\mathbf{Q} + \mathbf{A}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} - \mathbf{P}_{dp})^{-1}\mathbf{A}\mathbf{P}_{dp} - \mathbf{P}_{dp} \qquad (3.48)$$

The desired prediction covariance $\mathbf{P}_{dp}$ must be chosen such that $\mathbf{R}_o$ is positive definite. If the desired covariance results in an $\mathbf{R}_o$ that is not positive definite, then the sampling rate is not adequate for maintaining the desired prediction covariance matrix. When faster convergence of the covariance to steady-state is desired, then the sample rate may be increased so that faster convergence is achieved. Once the optimal measurement covariance is found, a search over all possible sensor combinations is performed to determine which set of sensors yields a combined $\mathbf{R}$ or $\mathbf{R}^{-1}$ that is "closest" (according to some metric) to $\mathbf{R}_o$ or $\mathbf{R}_o^{-1}$, respectively.

When $\mathbf{C} \in \Re^{p \times N}, p < N$, the full state vector is not measured and we can not compute the required inverses to solve for $\mathbf{R}_o$. We can, however, use the pseudoinverse to get an estimate of the optimal measurement covariance matrix. Let $\mathbf{M} = \mathbf{A}\mathbf{P}_{dp}\mathbf{C}^{\mathrm{T}}$ and $\mathbf{M}^{\dagger} = (\mathbf{M}^{\mathrm{T}}\mathbf{M})^{-1}\mathbf{M}^{\mathrm{T}}$. Using the pseudoinverse, the optimal measurement covariance that gets us closest to the desired prediction covariance is

$$\mathbf{R}_o = (\mathbf{M}^{\dagger}(\mathbf{Q} + \mathbf{A}\mathbf{P}_{dp}\mathbf{A}^{\mathrm{T}} - \mathbf{P}_{dp})\mathbf{M}^{\dagger\mathrm{T}})^{-1} - \mathbf{C}\mathbf{P}_{dp}\mathbf{C}^{\mathrm{T}} \qquad (3.49)$$

The mapping in (3.49) is from $\mathbf{P}_{dp} \in \Re^{N \times N}$ to $\mathbf{R}_o \in \Re^{p \times p}$. This allows us to solve for $p((p+1)/2)$ parameters in $\mathbf{R}_o$ to maintain a desired covariance matrix $\mathbf{P}_{dp}$ with $N(N+1)/2$ parameters.

Given a fixed sample rate, the measurement resolution can be used to control the covariance. The above technique is an open loop covariance control scheme where a single sensor resolution is computed that drives the steady state covariance close to the desired. Figure 12 shows the convergence of the error variance for a scalar system.

Since we are only controlling the sensor resolution it is sufficient to only specify one desired variance. This desired variance could be treated as either a desired prediction variance or a desired update variance. In this simulation the desired variance is treated as a desired prediction variance which explains why the peaks of the sawtooth waveform converge to the desired prediction variance. At $T = 10s$ the desired variance is

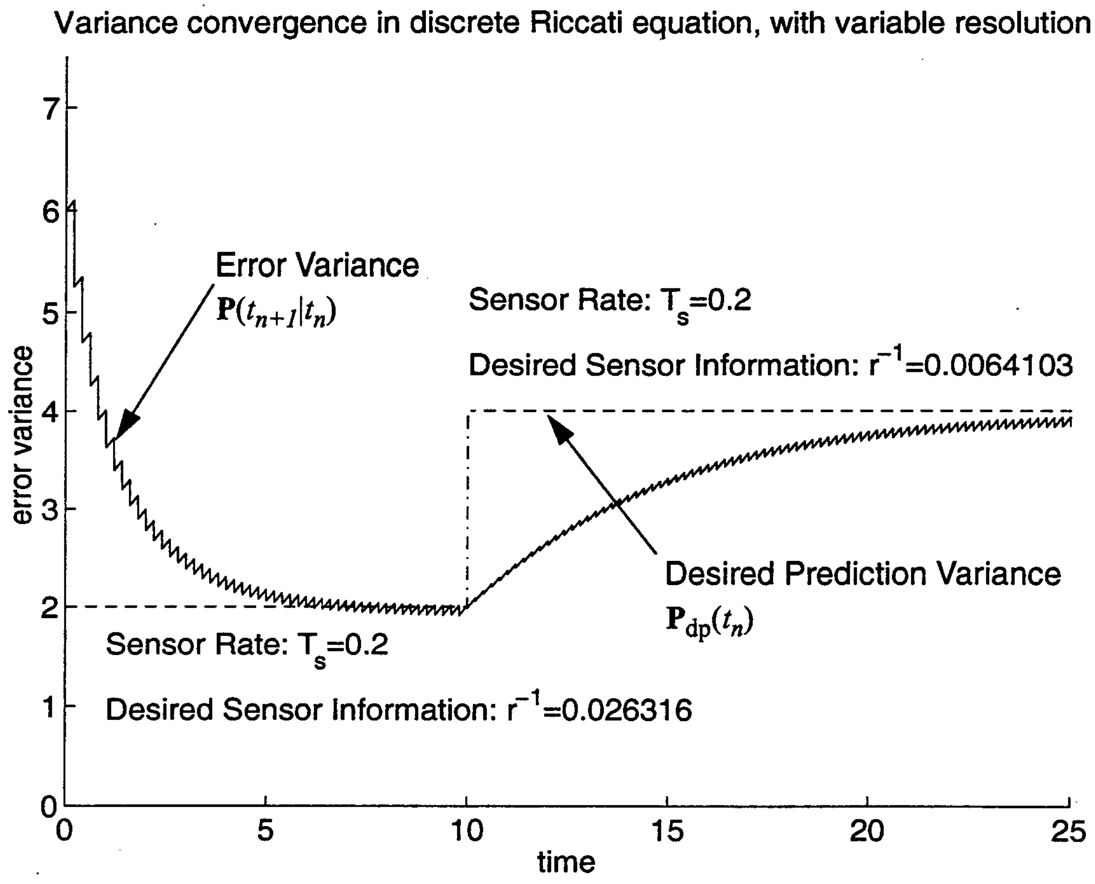


Figure 12 Variance convergence using variable resolution.

increased to *4*. The sample rate stays constant throughout the entire simulation. When the desired variance is increased, the optimal solution for the resolution decreases. The new sensor resolution is used and causes the prediction variance to converge to the desired prediction variance at steady state.

3.3.2 Covariance Control Using Sample Period

The state covariance matrix may also be controlled through choice of the sensor sample period. In this method, we assume that a sensor measurement has provided a state covariance update. The optimal sample period is then determined by analyzing the difference between the desired and prediction covariance. This optimization can be performed on a per-sample basis to obtain the best covariance control or performed at a lower rate to reduce computational complexity, but would also reduce the sensor manager's ability to control the covariance.

The prediction covariance is $\mathbf{P}(t_{n+1}|t_n) = \mathbf{A}(T)\mathbf{P}(t_n|t_n)\mathbf{A}^{\mathrm{T}}(T) + \mathbf{Q}(T)$ where matrices $\mathbf{A}(T)$ and $\mathbf{Q}(T)$ are functions of the sample period $T = t_{n+1} - t_n$. The difference between the desired covariance and the prediction error covariance is

$$\begin{aligned} \mathbf{M}_N(T) &= \mathbf{P}_{dp}(t_n) - \mathbf{P}(t_n + T|t_n) \\ &= \mathbf{P}_{dp}(t_n) - \mathbf{A}(T)\mathbf{P}(t_n|t_n)\mathbf{A}^{\mathrm{T}}(T) - \mathbf{Q}(T) \end{aligned} \tag{3.50}$$

where $\mathbf{M}_N(T) \in \Re^{N \times N}$ and the goal is to determine T such that $\mathbf{M}_N(T) = \mathbf{0}$ or is as "close" to zero as possible. One metric to consider is the minimum singular value. The characteristic equation of $\mathbf{M}_N(T)$ is

$$f(s) = det(s\mathbf{I} - \mathbf{M}_N(T)) = \sum_{k=0}^{N} b_k(T)s^k = \prod_{k=1}^{N} (s - \lambda_k(T)) \tag{3.51}$$

where the eigenvalues are all real because $\mathbf{M}_N(T)$ is symmetric. Letting $s = 0$ we have $b_0(T) = \prod_{k=1}^{N} -\lambda_k(T)$. The roots of the polynomial $b_0(T)$ correspond to the values of T that make the product of the eigenvalues equal to zero. The degree of the polynomial $b_0(T)$ will depend on which model is used. For example, given an Nth order model with state vector composed of N derivatives $\mathbf{x}(t_n) = \left[x(t_n)\ x^{(1)}(t_n)\ \dots\ x^{(N-1)}(t_n)\right]^{\mathrm{T}}$, the $b_0(T)$ polynomial in the discretized-continuous model has degree N^2. With the same state vector, the $b_0(T)$ polynomial in the direct-discretized model has degree $2\sum_{i=1}^{N} i$.

When the update covariance has the property $\mathbf{P}_{dp}(t_n) > \mathbf{P}(t_n|t_n)$, we can *always* find a positive value for the sample period. Figure 13 provides a graphical argument of why there must exist a positive value of *T*. The inner-most ellipse represents the update covariance and the dark solid ellipse represents the desired prediction covariance. Given $\mathbf{P}_{dp}(t_n) > \mathbf{P}(t_n|t_n)$, as $T \rightarrow \infty$ the ellipse of $\mathbf{P}(t_n + T|t_n)$ can either

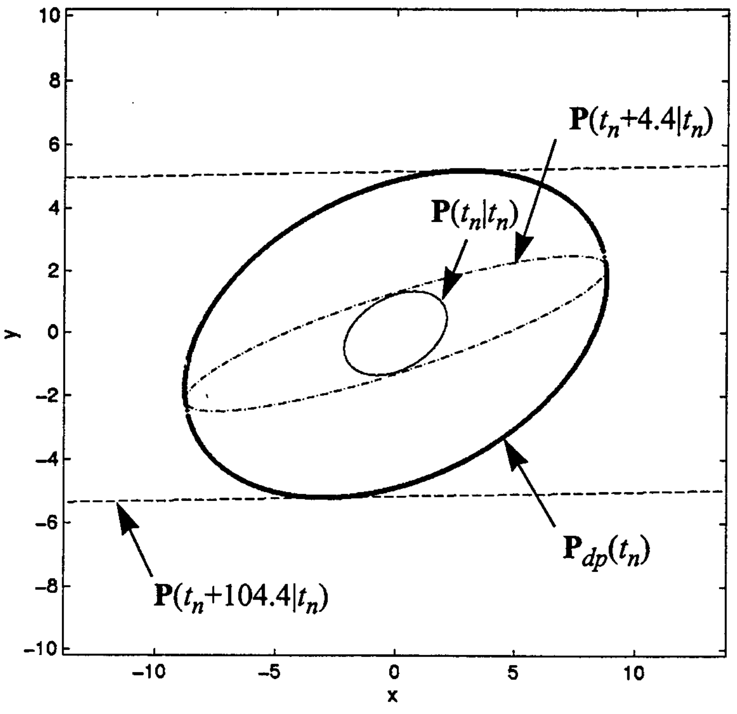


Figure 13 Two solutions for the sample period when the desired prediction covariance is greater than the update covariance

always stay inside of $\mathbf{P}_{dp}(t_n)$ or equal or exceed $\mathbf{P}_{dp}(t_n)$. Since $\mathbf{P}(t_n + T|t_n)$ increases without bound as $T \rightarrow \infty$, then it must exceed $\mathbf{P}_{dp}(t_n)$ for some positive *T*. At *T*=4.4 seconds and *T*=104.4 seconds, the prediction covariance ellipse is tangent to the desired ellipse. These values of *T* are thus two positive roots of the polynomial $b_o(T)$. The minimum positive root of $b_o(T)$ is the largest sample period *T* for which

$\mathbf{P}(t_n + T|t_n)$ remains completely within the ellipse of $\mathbf{P}_{dp}(t_n)$, in this case 4.4 seconds.

Given a fixed sensor resolution, the sample rate can be used to control the covariance. Figure 14 shows the convergence of the error variance for a scalar system. The same desired variances are used in this simulation as the one shown in Figure 12. This desired variance could be treated as either a desired prediction variance or a desired update variance. At $T = 10s$, the desired variance is increased to *4*. The sensor resolution stays constant throughout the entire simulation. When the desired variance is increased, the optimal solution for the sample rate decreases. The new sample

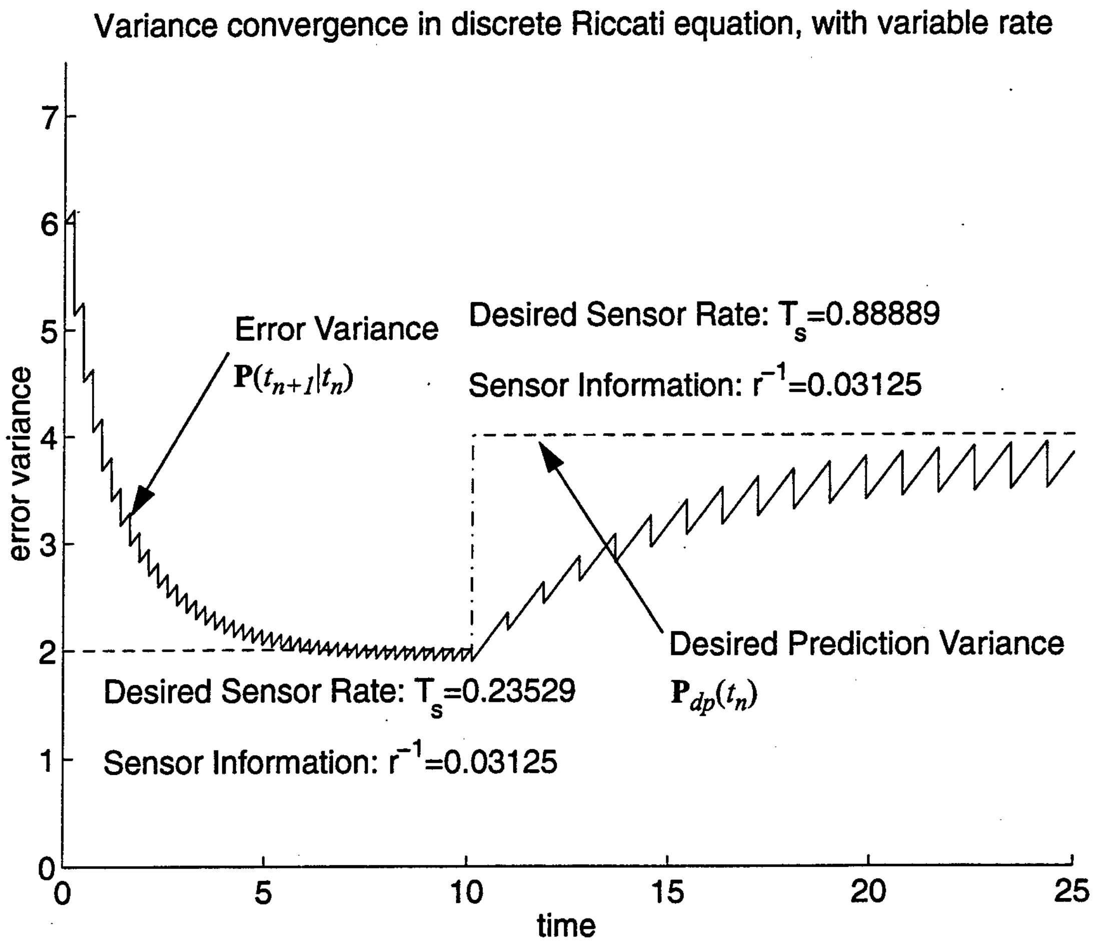


Figure 14 Variance convergence using variable rate

rate is used and causes the prediction variance to converge to the desired prediction variance at steady state.

Here we develop rate optimization polynomials for 1st and 2nd order target models using both the discretized-continuous model and the direct-discrete model. The specific coefficients for 1*st*, 2*nd*, and 3*rd* order polynomials are given in Appendix A. Let the elements of the desired prediction covariance and update covariance be $\mathbf{P}_{dp}(t_n)_{ij} = d_{ij}$ and $\mathbf{P}(t_n|t_n)_{ij} = p_{ij}$, respectively. With a first-order model tracking target position in a single coordinate, the difference in (3.50) is simply

$$\mathbf{M}_1(T) = d_{11} - a^2 p_{11} - T\sigma_c^2 \tag{3.52}$$

Letting $\mathbf{M}_1(T)$ equal zero and solving for the optimal sample period we have $T = (d_{11} - a^2 p_{11})/\sigma_c^2$. This equation shows how the desired variance, update variance, and white noise variance affect the solution of the sample period. When the desired variance is decreased, the sample period must also decrease. In a similar way, when the update variance is larger, the computed sample period will also be smaller. The direct-discretized model uses a different estimate of the covariance. The difference in (3.50) is

$$\mathbf{M}_1(T) = d_{11} - a^2 p_{11} - T^2\sigma_c^2 \tag{3.53}$$

Solving for the sample period, we have $T = \sqrt{(d_{11} - a^2 p_{11})/\sigma_c^2}$. The sample period for this model is just the square root of that computed using the discretized-continuous model.

For a 2*nd* order model tracking a target's position and velocity, we have

$$\mathbf{M}_2(T) = \begin{bmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{bmatrix} - \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ T & 1 \end{bmatrix} - \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix} \sigma_c^2$$

$$= \begin{bmatrix} d_{11} - p_{11} - 2Tp_{12} - p_{22}T^2 - (T^3\sigma_c^2)/3 & d_{12} - p_{12} - p_{22}T - (T^2\sigma_c^2)/2 \\ d_{12} - p_{12} - p_{22}T - (T^2\sigma_c^2)/2 & d_{22} - p_{22} - T\sigma_c^2 \end{bmatrix}$$

as a function of the sample period. The characteristic equation is

$$\begin{aligned} f(s) &= det(s\mathbf{I} - \mathbf{M}_2(T)) \\ &= b_2(T)s^2 + b_1(T)s + b_0(T) = (s - \lambda_1(T))(s - \lambda_2(T)) \end{aligned} \tag{3.54}$$

where the polynomial coefficients in T are

$$\begin{aligned} b_2(T) &= 1 \\ b_1(T) &= e_3T^3 + e_2T^2 + e_1T + e_0 \\ b_0(T) &= c_4T^4 + c_3T^3 + c_2T^2 + c_1T + c_0 \end{aligned} \tag{3.55}$$

The coefficients of $b_1(T)$ are

$$\begin{aligned} e_3 &= \sigma_c^2/3 \\ e_2 &= p_{22} \\ e_1 &= 2p_{12} + \sigma_c^2 \\ e_0 &= p_{11} + p_{22} - d_{11} - d_{22} \end{aligned} \tag{3.56}$$

and the coefficients of $b_0(T)$ are

$$\begin{aligned} c_4 &= \sigma_c^4/12 \\ c_3 &= \sigma_c^2(p_{22} - d_{22})/3 \\ c_2 &= (p_{12} + d_{12})\sigma_c^2 - d_{22}p_{22} \\ c_1 &= (p_{11} - d_{11})\sigma_c^2 + 2(d_{12}p_{22} - p_{12}d_{22}) \\ c_0 &= -d_{12}^2 + d_{11}d_{22} - d_{22}p_{11} + 2d_{12}p_{12} - p_{12}^2 - d_{11}p_{22} + p_{11}p_{22} \end{aligned} \tag{3.57}$$

and the eigenvalues of $\mathbf{M}_2(T)$ are $\lambda_1(T)$ and $\lambda_2(T)$. To find when the minimum eigenvalue equals zero we let $s = 0$, so that the characteristic polynomial is just the product of the eigenvalues $f(0) = det(-\mathbf{M}_2(T)) = b_0(T) = \lambda_1(T)\lambda_2(T)$. The smallest positive root of $b_0(T)$ will be the shortest sample period that minimizes the minimum singular value of $\mathbf{M}_2(T)$. Explicit expression for the eigenvalues of $\mathbf{M}_2(T)$

are given by the quadratic formula

$$\begin{aligned} \lambda_1(T) &= \frac{-b_1(T) + \sqrt{b_1^2(T) - 4b_0(T)}}{2} \\ \lambda_2(T) &= \frac{-b_1(T) - \sqrt{b_1^2(T) - 4b_0(T)}}{2} \end{aligned} \tag{3.58}$$

From (3.58) the trace will equal zero when $b_1(T) = 0$. When each eigenvalue of $\mathbf{M}_2(T)$ is "close" i.e. $\lambda_1(T) \cong \lambda_2(T)$, then a large trace will indicate poor covariance control and a small trace will indicate good covariance control. However, when the eigenvalues of $\mathbf{M}_2(T)$ are "close" to negatives of one another i.e. $\lambda_1(T) \cong -\lambda_2(T)$, then the trace could remain small as the magnitudes of $\lambda_1(T)$ and $\lambda_2(T)$ grow, giving a false impression of good covariance control.

Experimentally we observed that the matrix $\mathbf{M}_N(T)$ would usually have N real roots, where each root corresponds to the value of T that makes each eigenvalue equal zero. A subject of further study is to find the conditions on $\mathbf{A}(T)$, $\mathbf{Q}(T)$, $\mathbf{P}(t_n|t_n)$, and $\mathbf{P}_d$ that make $\mathbf{M}_N(T)$ have exactly N real roots.

The desired prediction covariance matrix acts as a threshold. If the sample period T, is chosen from the continuous set $T \in [T_{min}, \infty)$, then when the desired covariance is reduced, the sample rate increases; and when the desired covariance is increased, the sample rate is decreased. However, if the sample period T, is chosen from a discrete set $T \in nT_{min}$, $n \in \{1, 2, \dots\}$, then for some small changes in the desired covariance, the sample period may not change due to the thresholds imposed on the selection of sample period.

3.4 Sensor Delay

The sensor scheduler and specific sensors can cause various effects on the sensor commands concerning rate and resolution. The first effect is *sensor delay.* There are two types of sensor delay. The first type of delay is measurement delay. This happens when a sensor either executes or obtains a measurement but the controlling node does not receive the measured data until some later time. This causes the measurement to be delayed in time. The other type of delay happens when a sensor scheduler obtains a sensor request at time t_o but does not execute this request until a later time $t_1 > t_o$.

Figure 15 shows four sensor time lines each having different delay properties. Each sensor has the same sample rate and each sensor receives measurements at the same times. The pluses (+) on each sensor time line indicate when the sensor manager requests a measurement, the dots (•) indicate when the measurement execution starts, and the circles (o) indicate when the measurements are received.

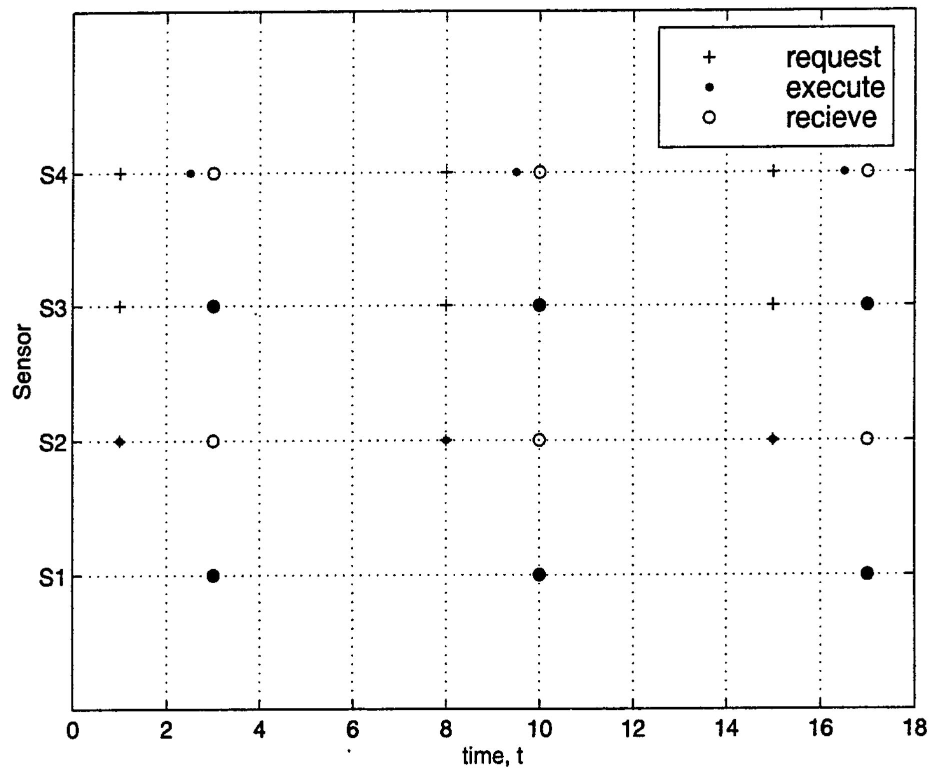


Figure 15 Illustration of measurement delay and sensor request delay

The first sensor S1 has no measurement or sensor request delay. The second sensor S2 has measurement delay but no sensor request delay. The third sensor S3 has no measurement delay but has a sensor request delay. The fourth sensor S4 has both measurement delay and sensor request delay.

These two types of delay must be considered when designing a sensor manager. When these delays are known a priori they may be taken into account in the sensor manager by requesting sensing resources at an earlier time.

3.5 Sensor Dropout

The second type of adverse effects is *sensor dropout.* Sensor dropout is described by an extended loss of sensing resources due to sensor reallocation or sensor failure. This can cause degradation in the ability to maintain a desired estimation performance. The general idea is that when we lose the ability to use a sensor, the rate of the remaining sensors must increase in order to compensate for the reduction in sensor information. A simple illustration can be made using a scalar state equation and vector measurements. The state and measurement equations are

$$x(t_{n+1}) = ax(t_n) + bw(t_n) \tag{3.59}$$

$$\mathbf{y}(t_n) = \mathbf{C}x(t_n) + \mathbf{D}\mathbf{v}(t_n) \tag{3.60}$$

with $a, b > 0$, $\mathbf{C} = \begin{bmatrix}1 & \dots & 1\end{bmatrix}^{\mathrm{T}}$, and $\mathbf{D} = diag\left(\begin{bmatrix}d_1^{1/2} & \dots & d_n^{1/2}\end{bmatrix}\right)$. This is a single state estimation problem using multiple sensors, where the measurements are contained in the vector $\mathbf{y}(t_n)$. The noise terms, $w(t_n)$ and $\mathbf{v}(t_n)$ have the following probability densities

$$w(t_n) \sim N(0, T\sigma^2), b = 1$$
$$q = \mathrm{E}[w(t_n)w(t_n)] = T\sigma^2$$

$$\mathbf{v}(t_n) \sim N(\mathbf{0}, \mathbf{I})$$
$$\mathbf{R} = \mathrm{E}[\mathbf{D}\mathbf{v}(t_m)(\mathbf{D}\mathbf{v}(t_n))^{\mathrm{T}}] = \mathbf{D}\mathbf{D}^{\mathrm{T}}\rho^2\delta(t_n - t_m)$$

Using (3.15) with the above coefficients we have

$$p(t_{n+1}|t_n) = \frac{a^2}{p^{-1}(t_n|t_{n-1}) + \mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C}} + q \tag{3.61}$$

Let the quadratic form in the denominator of (3.61) be $\frac{1}{r_e} \equiv \mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C} = \frac{1}{d_1} + \ldots + \frac{1}{d_n}$. This equation is similar to the "resistors in parallel" equation. Notice that the effective measurement variance r_e increases as sensors dropout. The total information $r_e^{-1}(t_n)$ decreases when sensors are removed. Letting $p(t_{n+1}|t_n) = p(t_n|t_{n-1}) = p_{dp}$, and solving for a desired q in (3.61) we get

$$q = p_{dp} - \frac{a^2}{p_{dp}^{-1} + r_e^{-1}(t_n)} \tag{3.62}$$

With a given sensor combination, we can compute a sensor sample rate that will achieve the desired variance goal. Using $q = T\sigma^2$ in (3.62) the sample rate is

$$T^{-1}(t_n) = \frac{\sigma^2(p_{dp}^{-1} + r_e^{-1}(t_n))}{1 + p_{dp}r_e^{-1}(t_n) - a^2} \tag{3.63}$$

Notice that the process noise variance is linearly related to the sample rate. We now give an example showing how increasing the measurement rate can compensate for sensor dropout to maintain a desired variance goal.

Example:

The system coefficients, the noise variances, and desired prediction variance is

$$\begin{gathered} a = 1, b = 1, \mathbf{C} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}}, d_i = 0.1 \\ \sigma^2 = 4, \rho^2 = 1, p_{dp} = 5 \end{gathered} \tag{3.64}$$

where $i \in \{1, \ldots, 7\}$ indexes seven different sensors having equal measurement variances. Using these coefficients the simulation removes one sensor at each 20*th* sample. Figure 16a and Figure 16b show what happens to the resolution and rate, respectively, when each sensor drops out. The effective sensor resolution decreases linearly because each sensor has the same variance.

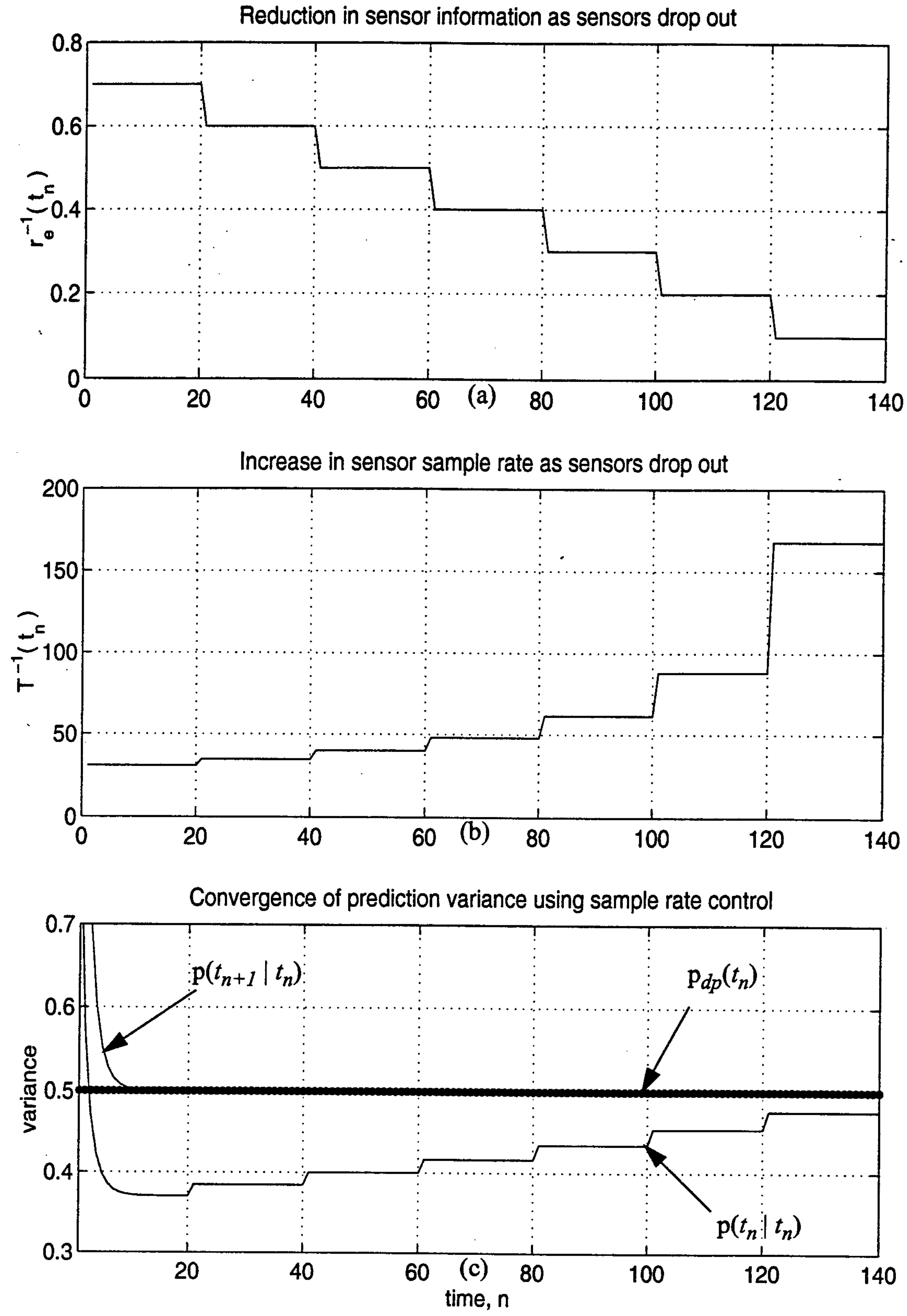


Figure 16 Compensation of sensor drop out using sensor rate

Figure 16c shows the desired prediction variance, the prediction variance, and the update variance. Once the prediction variance has converged to the desired prediction variance it is maintained there by adjusting the sensor rate of the remaining set of sensors. This gives a technique for controlling the variance when sensors dropout. This concludes the delay and dropout analysis.

3.6 Scalar Riccati Difference Equation

We now review a solution to the scalar Riccati difference equation. The general solution technique is to change the 1*st* order nonlinear difference equation into a 2*nd* order linear difference equation. This technique is promising since we have methods of solving linear difference equations this technique is promising.

A discrete Riccati equation can be defined for both the covariance prediction recursion and the update covariance recursion. Using (3.9) and (3.10), we can write the prediction and update variances as a recursive "bilinear" transform. Rewriting the prediction covariance from (3.15) in scalar form we have

$$\begin{aligned} p(t_{n+1}|t_n) &= q + a^2 p(t_n|t_{n-1}) - \frac{a^2c^2p^2(t_n|t_{n-1})}{c^2p(t_n|t_{n-1}) + r} \\ &= \frac{q(c^2p(t_n|t_{n-1}) + r) + a^2p(t_n|t_{n-1})(c^2p(t_n|t_{n-1}) + r) - a^2c^2p^2(t_n|t_{n-1})}{r + c^2p(t_n|t_{n-1})} \\ &= \frac{(a^2r + c^2q)p(t_n|t_{n-1}) + rq}{c^2p(t_n|t_{n-1}) + r} \end{aligned} \tag{3.65}$$

This nonlinear difference equation is called a *bilinear transformation* because it is linear in the numerator and denominator. Other names for the bilinear transformation are Möbius transformation, or linear fractional transformation [1].

The information update monotonically increases and thus the update covariance will always be smaller than the prediction covariance. The update variance recursion in (3.16) takes a similar form

$$\begin{aligned} p(t_{n+1}|t_{n+1}) &= (p(t_{n+1}|t_n)^{-1} + c^2r^{-1})^{-1} \\ p(t_{n+1}|t_{n+1}) &= \frac{1}{(a^2p(t_n|t_n)+q)^{-1} + c^2r^{-1}} \\ p(t_{n+1}|t_{n+1}) &= \frac{a^2rp(t_n|t_n)+rq}{a^2c^2p(t_n|t_n)+qc^2+r} \end{aligned} \tag{3.66}$$

The update covariance can always be related to the prediction variance through the equation $p(t_{n+1}|t_n) = a^2p(t_n|t_n)+q$. Notice that (3.65) and (3.66) have the same general form but different coefficients.

Consider the Riccati difference equation with coefficients $e(n), f(n), g(n), h(n) \in \Re$, where in general these could be time varying sequences.

$$p(n+1) = \frac{e(n)p(n)+f(n)}{g(n)p(n)+h(n)} \tag{3.67}$$

This equation is in the form of a recursive bilinear transform. Equation (3.67) can be converted into a *2nd* order linear difference equation by making the substitution [9]

$$p(n) = \frac{u(n+1)}{g(n)u(n)} - \frac{h(n)}{g(n)} = \frac{u(n+1)-h(n)u(n)}{g(n)u(n)} \tag{3.68}$$

into equation (3.67), with initial conditions $u(0) = 1$, $u(1) = g(0)p(0)+h(0)$, and $u(m) = 0, \forall m < 0$. With this substitution we get

$$\begin{aligned} \frac{u(n+2)-u(n+1)h(n+1)}{g(n+1)u(n+1)} &= \frac{\frac{e(n)u(n+1)-e(n)u(n)h(n)}{g(n)u(n)} + f(n)}{\frac{u(n+1)-h(n)u(n)}{u(n)} + h(n)} \\ &= \frac{e(n)u(n+1)-e(n)u(n)h(n)+g(n)f(n)u(n)}{g(n)u(n+1)} \end{aligned} \tag{3.69}$$

Canceling the $u(n+1)$ terms in the denominator and multiplying by $g(n+1)$ gives us

$$u(n+2)-u(n+1)h(n+1) = \frac{e(n)g(n+1)u(n+1)-e(n)u(n)h(n)g(n+1)+g(n)g(n+1)f(n)u(n)}{g(n)} \tag{3.70}$$

$$\begin{aligned} u(n+2) = {} & \left(\frac{e(n)g(n+1)+h(n+1)g(n)}{g(n)}\right)u(n+1)+ \\ & \left(\frac{(f(n)g(n)-e(n)h(n))g(n+1)}{g(n)}\right)u(n) \end{aligned} \tag{3.71}$$

With constant coefficients (3.71) reduces to

$$u(n+2) = (e+h)u(n+1)+(fg-eh)u(n)+\delta_{n+2}+(gp(0)-e)\delta_{n+1} \tag{3.72}$$

where $n = -2, -1, \ldots$ and the Kronecker delta functions have been included to account for the initial conditions on *u(n)* and *p(n)*. The solution to this second-order linear difference equation can be found by using *Z*-transform techniques. Taking the *Z*-transform and solving for $U(z)/z$ we get

$$z^2U(z)-z(e+h)U(z)-(fg-eh)U(z) = z^2+(gp(0)-e)z \tag{3.73}$$

$$\begin{aligned} \frac{U(z)}{z} &= \frac{z+(gp(0)-e)}{z^2-(e+h)z-(fg-eh)} = \frac{z+(gp(0)-e)}{(z-\lambda_1)(z-\lambda_2)} = \frac{R_1}{z-\lambda_1}+\frac{R_2}{z-\lambda_2} \\ U(z) &= \frac{R_1}{1-\lambda_1 z^{-1}}+\frac{R_2}{1-\lambda_2 z^{-1}} \end{aligned} \tag{3.74}$$

where the residues in the partial fraction expansion are

$$R_1 = \frac{\lambda_1+gp(0)-e}{\lambda_1-\lambda_2}, R_2 = \frac{-(\lambda_2+gp(0)-e)}{\lambda_1-\lambda_2} \tag{3.75}$$

Using the coefficients *a*, *c*, *q*, and *r* from either (3.65) or (3.66) it can be shown that the roots of the characteristic equation from (3.74) $z^2-(e+h)z-(fg-eh) = 0$ are strictly positive. Thus, we can assume without loss of generality, that $\lambda_1 > \lambda_2 > 0$. The solution of (3.74) is then $u(n) = R_1\lambda_1^n + R_2\lambda_2^n$. Substituting this result into (3.68) we get

$$
\begin{aligned}
p(n) &= \frac{R_1\lambda_1^{n+1} + R_2\lambda_2^{n+1} - hR_1\lambda_1^n - hR_2\lambda_2^2}{gR_1\lambda_1^n + gR_2\lambda_2^n} \\
&= \frac{R_1\lambda_1^n(\lambda_1 - h) + R_2\lambda_2^n(\lambda_2 - h)}{gR_1\lambda_1^n + gR_2\lambda_2^n} \\
&= \frac{R_1(\lambda_1 - h)\left(\frac{\lambda_1}{\lambda_2}\right)^n + R_2(\lambda_2 - h)}{gR_1\left(\frac{\lambda_1}{\lambda_2}\right)^n + gR_2}
\end{aligned}
\qquad (3.76)
$$

The steady state variance is $p_f = \lim_{n \to \infty} p(n) = (\lambda_1 - h)/g$. The first term in the numerator of the final expression in (3.76) can be factored into $gR_1 p_f \alpha^n$ where $\alpha = \lambda_1/\lambda_2$, and the second term in the numerator may be factored into $gR_2 p_f + R_2(\lambda_2 - \lambda_1)$. Thus,

$$
\begin{aligned}
p(n) &= \frac{gR_1 p_f \alpha^n + (gR_2 p_f + R_2(\lambda_2 - \lambda_1))}{gR_1\alpha^n + gR_2} \\
&= \frac{p_f(gR_1\alpha^n + gR_2) + R_2(\lambda_2 - \lambda_1)}{gR_1\alpha^n + gR_2} \\
&= p_f + \frac{R_2(\lambda_2 - \lambda_1)}{gR_1\alpha^n + gR_2}
\end{aligned}
\qquad (3.77)
$$

The second term in the last equation of (3.77) goes to zero as $n \to \infty$, and the variance converges to the steady state variance p_f.The ratio of the poles, α, will determine how fast the variance sequence *p(n)* will converge. The parameters (*a, q, c, r*) from (3.65) determine α and p_f.

3.7 Matrix Riccati Difference Equation

Since most tracking applications involve forming estimates of more than one state (positions, velocities, and accelerations), there is motivation for the development of closed form solutions to the discrete matrix Riccati equation (DMRE). This solution allows for fast computation of the steady state covariance. As with the scalar Riccati equation, a DMRE can be defined for the prediction or update covariances using (3.9) and (3.10).

The *Nth* order DMRE in (3.15) can be changed into a 2*Nth* order linear homogeneous matrix difference equation together with a nonlinear equation [12]. Consider the linear and nonlinear matrix difference equations

$$\begin{aligned} \mathbf{Y}(n+1) &= \mathbf{M}\mathbf{Y}(n) \\ \begin{bmatrix} \mathbf{U}(n+1) \\ \mathbf{V}(n+1) \end{bmatrix} &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{Q} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-T} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U}(n) \\ \mathbf{V}(n) \end{bmatrix} \end{aligned} \tag{3.78}$$

$$\mathbf{P}(n+1) = \mathbf{V}(n+1)\mathbf{U}^{-1}(n+1) \tag{3.79}$$

respectively, where $\mathbf{Y}(n) = \begin{bmatrix} \mathbf{U}(n) \\ \mathbf{V}(n) \end{bmatrix}$ and $\mathbf{M} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{Q} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-T} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$ with initial conditions $\mathbf{U}(0) = \mathbf{I}$ and $\mathbf{V}(0) = \mathbf{P}(0)$. These two equations together form a recursive “bilinear” transformation.

This system of equations will be shown to be equivalent to (3.15). Simplifying the matrix **M** in (3.78), and letting $\mathbf{U}(n) = \mathbf{I}$ and $\mathbf{V}(n) = \mathbf{P}(n)$ we have

$$\begin{bmatrix} \mathbf{U}(n+1) \\ \mathbf{V}(n+1) \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{-T} & \mathbf{A}^{-T}\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C} \\ \mathbf{Q}\mathbf{A}^{-T} & \mathbf{Q}\mathbf{A}^{-T}\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C} + \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{P}(n) \end{bmatrix} \tag{3.80}$$

Let each subblock of this matrix be $\mathbf{E} = \mathbf{A}^{-T}$, $\mathbf{F} = \mathbf{Q}\mathbf{A}^{-T}$, $\mathbf{G} = \mathbf{A}^{-T}\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}$, and $\mathbf{H} = \mathbf{Q}\mathbf{A}^{-T}\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C} + \mathbf{A}$ so that we can express the matrix in (3.80) as

$$\mathbf{M} = \begin{bmatrix} \mathbf{E} & \mathbf{G} \\ \mathbf{F} & \mathbf{H} \end{bmatrix} \tag{3.81}$$

The recursive bilinear transformation is $\mathbf{P}(n+1) = (\mathbf{H}\mathbf{P}(n) + \mathbf{F})(\mathbf{G}\mathbf{P}(n) + \mathbf{E})^{-1}$, where we see that each matrix factor is affine in $\mathbf{P}(n)$. When the initial covariance is invertible i.e $det(\mathbf{P}(n)) \neq 0$ then $\mathbf{U}(n+1)$ is

$$\begin{aligned} \mathbf{U}(n+1) &= \mathbf{G}\mathbf{P}(n) + \mathbf{E} \\ &= (\mathbf{A}^{-T}\mathbf{C}^{T}\mathbf{R}^{-1}\mathbf{C}\mathbf{P}(n) + \mathbf{A}^{-T}) \\ &= \mathbf{A}^{-T}(\mathbf{C}^{T}\mathbf{R}^{-1}\mathbf{C} + \mathbf{P}^{-1}(n))\mathbf{P}(n) \end{aligned} \tag{3.82}$$

Grouping matrices as indicated by the parenthesis in (3.83), we can express $\mathbf{U}^{-1}(n+1)$ using the Matrix Inversion Lemma.

$$\begin{aligned} \mathbf{U}^{-1}(n+1) &= [(\mathbf{A}^{-T}\mathbf{C}^{T})\mathbf{R}^{-1}(\mathbf{C}\mathbf{P}(n)) + \mathbf{A}^{-T}]^{-1} \\ &= \mathbf{A}^{T} - \mathbf{C}^{T}(\mathbf{R} + \mathbf{C}\mathbf{P}(n)\mathbf{C}^{T})^{-1}\mathbf{C}\mathbf{P}(n)\mathbf{A}^{T} \end{aligned} \tag{3.83}$$

Solving for $\mathbf{V}(n+1)$ we have

$$\begin{aligned} \mathbf{V}(n+1) &= \mathbf{H}\mathbf{P}(n) + \mathbf{F} \\ &= ((\mathbf{Q}\mathbf{A}^{-T}\mathbf{C}^{T}\mathbf{R}^{-1}\mathbf{C} + \mathbf{A})\mathbf{P}(n) + \mathbf{Q}\mathbf{A}^{-T}) \\ &= \mathbf{A}\mathbf{P}(n) + \mathbf{Q}\mathbf{A}^{-T}\mathbf{C}^{T}\mathbf{R}^{-1}\mathbf{C}\mathbf{P}(n) + \mathbf{Q}\mathbf{A}^{-T} \\ &= \mathbf{A}\mathbf{P}(n) + \mathbf{Q}\mathbf{A}^{-T}(\mathbf{C}^{T}\mathbf{R}^{-1}\mathbf{C} + \mathbf{P}^{-1}(n))\mathbf{P}(n) \\ &= \mathbf{A}\mathbf{P}(n) + \mathbf{Q}\mathbf{U}(n) \end{aligned} \tag{3.84}$$

Finally, using (3.83) together with (3.84) to compute $\mathbf{P}(n+1)$ $=\mathbf{V}(n+1)\mathbf{U}^{-1}(n+1)$ we get

$$\begin{aligned} \mathbf{P}(n+1) &= \mathbf{V}(n+1)\mathbf{U}^{-1}(n+1) \\ &= (\mathbf{A}\mathbf{P}(n) + \mathbf{Q}\mathbf{U}(n))\mathbf{U}^{-1}(n) \\ &= \mathbf{A}\mathbf{P}(n)\mathbf{U}^{-1}(n) + \mathbf{Q} \\ &= \mathbf{Q} + \mathbf{A}\mathbf{P}(n)\mathbf{A}^{T} - \mathbf{A}\mathbf{P}(n)\mathbf{C}^{T}(\mathbf{R} + \mathbf{C}\mathbf{P}(n)\mathbf{C}^{T})^{-1}\mathbf{C}\mathbf{P}(n)\mathbf{A}^{T} \end{aligned} \tag{3.85}$$

which is the DMRE as given in (3.15). Since (3.78) is a linear homogeneous difference equation, the solution can be found for any arbitrary time step.

Using the above solution, the steady state solution is easily obtained. The matrices in (3.78) are Symplectic matrices. Symplectic matrices have the property that $\mathbf{M}^{T}\mathbf{J}\mathbf{M} = \mathbf{J}$ [10], where $\mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix}$ and $det(\mathbf{M}) = 1$. The eigenvalues of Symplectic matrices come in reciprocal pairs, i.e. with proper ordering $\lambda_i(\mathbf{M}) = \lambda_i(\mathbf{M}^{-1})$. These properties and others are developed in Appendix B. The solution to the linear difference equation in (3.78) is simply

$$\mathbf{Y}(n) = \mathbf{M}^n \mathbf{Y}(0) \tag{3.86}$$

Since $\mathbf{M} \in \Re^{2N \times 2N}$, there will always be an even number of eigenvalues. For simplicity in this review of the solution to the DMRE, consider $\mathbf{M}$ having distinct eigenvalues. Expanding $\mathbf{M}$ using an eigendecomposition produces

$$\mathbf{M}^n = (\mathbf{TDT}^{-1})^n = \mathbf{TD}^n\mathbf{T}^{-1} \tag{3.87}$$

Constructing the eigendecomposition so that the first N eigenvalues are inside the unit circle, the partitioned eigenvalue and eigenvector matrices have the form

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_1^{-1} \end{bmatrix}, \mathbf{T} = \begin{bmatrix} \mathbf{T}_1 & \mathbf{T}_2 \\ \mathbf{T}_3 & \mathbf{T}_4 \end{bmatrix} \tag{3.88}$$

where $|\mathbf{D}_1|_{ii} < 1$. Another property of symplectic matrices is that there always exists a symplectic basis that takes $\mathbf{M}$ into a canonical form [10]. The specific canonical form depends on the multiplicity of the eigenvalues. When the eigenvalues are distinct, the canonical form is the diagonal matrix in (3.88). When the eigenvectors form a symplectic basis [10], then the matrix $\mathbf{T}$ in the eigendecomposition is a symplectic matrix. The inverse is $\mathbf{T}^{-1} = -\mathbf{J}\mathbf{T}^{\mathrm{T}}\mathbf{J}$ and can be represented in terms of the matrix blocks contained in $\mathbf{T}$

$$\mathbf{T}^{-1} = -\mathbf{J}\mathbf{T}^{\mathrm{T}}\mathbf{J} = \begin{bmatrix} \mathbf{T}_4^{\mathrm{T}} & -\mathbf{T}_2^{\mathrm{T}} \\ -\mathbf{T}_3^{\mathrm{T}} & \mathbf{T}_1^{\mathrm{T}} \end{bmatrix} \tag{3.89}$$

Using (3.88) and (3.89), we obtain a closed form solution for the state covariance as a function of the sample period n.

$$\mathbf{Y}(n) = \begin{bmatrix}\mathbf{U}(n)\\ \mathbf{V}(n)\end{bmatrix} = \begin{bmatrix}\mathbf{T}_1 & \mathbf{T}_2\\ \mathbf{T}_3 & \mathbf{T}_4\end{bmatrix}\begin{bmatrix}\mathbf{D}_1^n & \mathbf{0}\\ \mathbf{0} & \mathbf{D}_1^{-n}\end{bmatrix}\begin{bmatrix}\mathbf{T}_4^T & -\mathbf{T}_2^T\\ -\mathbf{T}_3^T & \mathbf{T}_1^T\end{bmatrix}\begin{bmatrix}\mathbf{I}\\ \mathbf{P}_0\end{bmatrix}$$
$$= \begin{bmatrix}\mathbf{T}_1\mathbf{D}_1^n\mathbf{T}_4^T-\mathbf{T}_2\mathbf{D}_1^{-n}\mathbf{T}_3^T & -\mathbf{T}_1\mathbf{D}_1^n\mathbf{T}_2^T+\mathbf{T}_2\mathbf{D}_1^{-n}\mathbf{T}_1^T\\ \mathbf{T}_3\mathbf{D}_1^n\mathbf{T}_4^T-\mathbf{T}_4\mathbf{D}_1^{-n}\mathbf{T}_3^T & -\mathbf{T}_3\mathbf{D}_1^n\mathbf{T}_2^T+\mathbf{T}_4\mathbf{D}_1^{-n}\mathbf{T}_1^T\end{bmatrix}\begin{bmatrix}\mathbf{I}\\ \mathbf{P}_0\end{bmatrix} \quad (3.90)$$

$$\mathbf{P}(n) = \mathbf{V}(n)\mathbf{U}^{-1}(n)$$
$$= ((\mathbf{T}_3\mathbf{D}_1^n\mathbf{T}_4^T-\mathbf{T}_4\mathbf{D}_1^{-n}\mathbf{T}_3^T)+(\mathbf{T}_4\mathbf{D}_1^{-n}\mathbf{T}_1^T-\mathbf{T}_3\mathbf{D}_1^n\mathbf{T}_2^T)\mathbf{P}_0) \quad (3.91)$$
$$((\mathbf{T}_1\mathbf{D}_1^n\mathbf{T}_4^T-\mathbf{T}_2\mathbf{D}_1^{-n}\mathbf{T}_3^T)+(\mathbf{T}_2\mathbf{D}_1^{-n}\mathbf{T}_1^T-\mathbf{T}_1\mathbf{D}_1^n\mathbf{T}_2^T)\mathbf{P}_0)^{-1}$$

The solution in (3.91) provides an easy way of solving for the steady state covariance, $\mathbf{P}_f$. Note that at steady state $\lim_{n\to\infty}\mathbf{D}_1^n = \mathbf{0}$. Making this substitution, (3.91) becomes

$$\mathbf{P}(n) = (\mathbf{T}_4\mathbf{D}_1^{-n}\mathbf{T}_1^T\mathbf{P}_0-\mathbf{T}_4\mathbf{D}_1^{-n}\mathbf{T}_3^T)(\mathbf{T}_2\mathbf{D}_1^{-n}\mathbf{T}_1^T\mathbf{P}_0-\mathbf{T}_2\mathbf{D}_1^{-n}\mathbf{T}_3^T)^{-1}$$
$$= (\mathbf{T}_4\mathbf{D}_1^{-n}(\mathbf{T}_1^T\mathbf{P}_0-\mathbf{T}_3^T))(\mathbf{T}_2\mathbf{D}_1^{-n}(\mathbf{T}_1^T\mathbf{P}_0-\mathbf{T}_3^T))^{-1}$$
$$\mathbf{T}_4\mathbf{D}_1^{-n}(\mathbf{T}_1^T\mathbf{P}_0-\mathbf{T}_3^T)(\mathbf{T}_1^T\mathbf{P}_0-\mathbf{T}_3^T)^{-1}\mathbf{D}_1^n\mathbf{T}_2^{-1} \quad (3.92)$$
$$\mathbf{P}_f = \mathbf{T}_4\mathbf{T}_2^{-1}$$

provided $det(\mathbf{T}_2) \neq 0$. The parameters (**A**, **Q**, **C**, **R**) from the DMRE determine the eigenvalues in **D** and the steady-state covariance $\mathbf{P}_f$. Ideally we would like each eigenvalue of $\mathbf{D}_1$ to be small (so that we would have fast convergence of the covariance), and $\mathbf{P}_f$ to converge to the desired prediction covariance, $\mathbf{P}_{dp}$.

When we consider managing sensing resources the parameters **A** and **Q** are functions of the sample rate and **C** and **R** are functions of the choice of sensors. When either the sample rate or sensor resolution change, the eigenvalues in **D** and the steady state covariance $\mathbf{P}_f$ will also change.

Chapter 4

DECENTRALIZED SENSOR MANAGEMENT

4.1 Decentralized Kalman Filter

The choice of decentralized estimation algorithm will help determine the development of the decentralized sensor management algorithms. The DKF is used as the basis for the decentralized system designs described below. This algorithm is used for implementing state estimation among multiple processors in distributed networks. The DKF algorithm developed in [18] is fully distributed in the sense that it does not rely on any central clock, communication, or processing infrastructure. Within a certain set of assumptions, the DKF is mathematically equivalent to a centralized Kalman filter where all measurements are received at one node.

As mentioned in the introduction, decentralized sensing networks have a number of advantages over centralized sensing networks that have the same sensing resources. The most important advantage is survivability. When one node or sensor fails, the entire network can continue to function by reallocating sensing and processing load to the remaining nodes. A second advantage is reduced computational complexity on a per-processor basis. Another advantage is that each node can share the sensing load.

Networks can be either static or dynamic. A static network is one in which the connectivity between nodes does not change, while a dynamic network is one in which the connectivity between nodes can change with time. Some networks are inherently dynamic such as in cellular communications.

Communications channels can be classified as simplex, half duplex, or full duplex. A simplex channel allows information to travel only one way. A duplex channel allows information to travel in both directions. Half duplex means information only travels one way at a time (such as in CB radios) and full duplex allows two way simultaneous information transmission. In our models, we assume a full duplex transmission. The communication load on a per processor basis is defined as the maximum number of full duplex channels required by a single processor. The total communication load is defined as the total number of channels that exist.

The network that has been used in this research is a dynamic clique network. Figure 17 is a model of a full duplex dynamic clique communications network with *M* processors. This network has a communication load of *M-1* channels per processor where *M* is the number of processors. This model is a general network that can be used to model any type of centralized, hierarchical, or decentralized network by removing communication links between processing nodes.

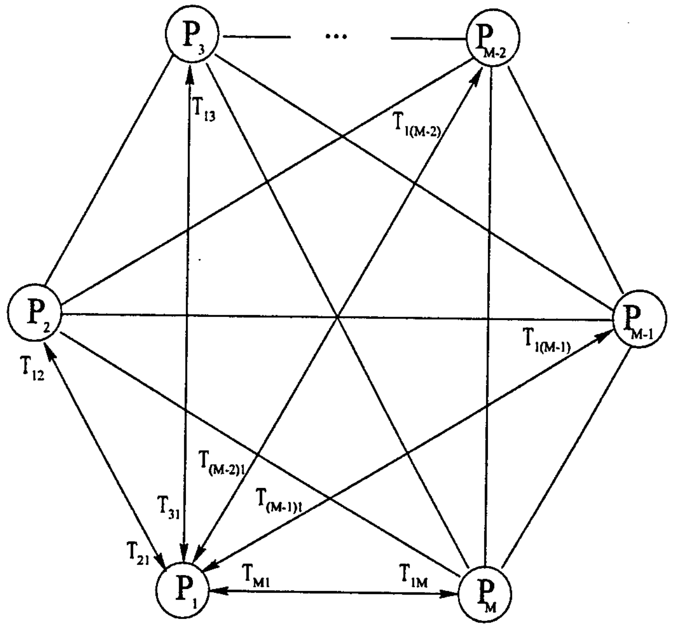


Figure 17 Dynamic clique network

When information is broadcast from one node to the other nodes, the state and covariance information must be transformed into the state space of the receiving nodes. These transformations are modeled in Figure 17 as $\mathbf{T}_{ij}$, where i is the transmitting node and j is the receiving node. The internodal transformations are required because sensors can be located at different points in space and have different look angles [4]. Let $\mathbf{T}_{ij} \in \{C\}$ with C a space of suitable internodal transformations (consisting of rotations and translations). In order to focus better on the problem of sensor management, the model we use assumes $\mathbf{T}_{ij} = \mathbf{I}, \forall i, j$. This implies that all nodes have the same state space. The three main assumptions are

1. Clique network
2. All nodes have the same (global) state space
3. Zero delay communication

The first assumption ensures that each node receives full information of the target state. Assumption 2 ensures that each node has a common state representation so that received information is of the same form. The third assumption insures that each node receives present information. The last assumption can be relaxed to the extent that the communication delays are less than the sample period during a particular interval.

We now review the DKF [18]. The first step is to define the state and measurement equations for each node in the system. Each node in the clique then has the following state and measurement equations

$$\boldsymbol{x}(t_{n+1}) = \mathbf{A}_i(t_n)\boldsymbol{x}(t_n) + \mathbf{B}_i(t_n)\boldsymbol{w}_i(t_n) \tag{4.1}$$

$$\boldsymbol{y}_i(t_n) = \mathbf{C}_i(t_n)\boldsymbol{x}(t_n) + \mathbf{D}_i(t_n)\boldsymbol{v}_i(t_n) \tag{4.2}$$

where $\boldsymbol{x}(t_n) \in \Re^{N \times 1}$, $\boldsymbol{y}_i(t_n) \in \Re^{m_i \times 1}$, $\boldsymbol{w}_i(t_n) \in \Re^{N \times 1}$, and $\boldsymbol{v}_i(t_n) \in \Re^{m_i \times 1}$. The dimensions on these vectors require the system matrices to have dimensions $\mathbf{A}_i(t_n) \in \Re^{N \times N}$, $\mathbf{B}_i(t_n) \in \Re^{N \times N}$, $\mathbf{C}_i(t_n) \in \Re^{m_i \times N}$, and $\mathbf{D}_i(t_n) \in \Re^{m_i \times m_i}$. Although the state equation coefficients in (4.1) have been indexed with *i* for each node, they are assumed to be the same across all processors following from the assumption of each node having the same state space. With these definitions, each node makes estimates based strictly on its own measurements and then assimilates the measurements from other processors. The prediction and update equations based on local measurements are similar to the centralized equations.

Prediction Equations:

$$\tilde{x}_i(t_n|t_{n-1}) = \mathbf{A}(t_n)\tilde{x}_i(t_{n-1}|t_{n-1}) \tag{4.3}$$

$$\tilde{y}_i(t_n|t_{n-1}) = \mathbf{C}_i(t_n)\tilde{x}_i(t_n|t_{n-1}) \tag{4.4}$$

$$\mathbf{P}_i(t_n|t_{n-1}) = \mathbf{A}_i(t_n)\mathbf{P}_i(t_{n-1}|t_{n-1})\mathbf{A}_i(t_n)^{\mathrm{T}} + \mathbf{Q}_i(t_n) \tag{4.5}$$

Update Equations:

$$\tilde{\mathbf{P}}_i^{-1}(t_n|t_n) = \mathbf{P}_i^{-1}(t_n|t_{n-1}) + \mathbf{C}_i^{\mathrm{T}}(t_n)\mathbf{R}_i^{-1}(t_n)\mathbf{C}_i(t_n) \tag{4.6}$$

$$\mathbf{K}_i(t_n) = \tilde{\mathbf{P}}_i(t_n|t_n)\mathbf{C}_i^{\mathrm{T}}(t_n)\mathbf{R}_i^{-1}(t_n) \tag{4.7}$$

$$\tilde{x}_i(t_n|t_n) = \tilde{x}_i(t_n|t_{n-1}) + \mathbf{K}_i(t_n)(y_i(t_n) - \tilde{y}_i(t_n|t_{n-1})) \tag{4.8}$$

where the tilde (~) denotes partial estimates based strictly on new local observations and old data from other nodes. The local state updates at each node may be expressed as

$$\begin{aligned}\tilde{x}_i(t_n|t_n) = {} & (\mathbf{I} - \tilde{\mathbf{P}}_i(t_n|t_n)\mathbf{C}_i^{\mathrm{T}}(t_n)\mathbf{R}_i^{-1}(t_n)\mathbf{C}_i(t_n))\mathbf{A}_i(t_n)\tilde{x}_i(t_{n-1}|t_{n-1}) \\ & + \tilde{\mathbf{P}}_i(t_n|t_n)\mathbf{C}_i^{\mathrm{T}}(t_n)\mathbf{R}_i^{-1}(t_n)y_i(t_n)\end{aligned} \tag{4.9}$$

where this form illustrates that the new state update combines the old state update with the new local measurement. Once each node goes through a measurement cycle, each node then broadcasts the state estimates and covariance matrices.

4.1.1 Assimilation of Variance

The centralized measurement covariance has a block diagonal structure, $\mathbf{R} = blockdiag\begin{bmatrix}\mathbf{R}_1 & \dots & \mathbf{R}_M\end{bmatrix}$, because the measurement noise terms across each processor are assumed to be mutually uncorrelated. The centralized information update is

$$\mathbf{P}^{-1}(t_n|t_n) = \mathbf{P}^{-1}(t_n|t_{n-1}) + \mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n)\mathbf{C}(t_n) \tag{4.10}$$

and the decentralized covariance updates are

$$\tilde{\mathbf{P}}_i^{-1}(t_n|t_n) = \mathbf{P}_i^{-1}(t_n|t_{n-1}) + \mathbf{C}_i^{\mathrm{T}}(t_n)\mathbf{R}_i^{-1}(t_n)\mathbf{C}_i(t_n) \tag{4.11}$$

In a fully connected (clique) network, each node has the same initial information i.e. $\mathbf{P}^{-1}(t_n|t_{n-1}) = \mathbf{P}_i^{-1}(t_n|t_{n-1})$. The block structure of the measurement matrix $\mathbf{C}(t_n)$ and covariance matrix $\mathbf{R}(t_n)$ allows us to relate the centralized sensor information matrix to the decentralized sensor information matrices by

$$\begin{aligned}\mathbf{C}^{\mathrm{T}}(t_n)\mathbf{R}^{-1}(t_n)\mathbf{C}(t_n) &= \begin{bmatrix}\mathbf{C}_1(t_n)\\ \dots\\ \mathbf{C}_M(t_n)\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}\mathbf{R}_1^{-1}(t_n) & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \dots & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{R}_M^{-1}(t_n)\end{bmatrix} \begin{bmatrix}\mathbf{C}_1(t_n)\\ \dots\\ \mathbf{C}_M(t_n)\end{bmatrix}\\ &= \sum_{j=1}^{M} \mathbf{C}_j^{\mathrm{T}}(t_n)\mathbf{R}_j^{-1}(t_n)\mathbf{C}_j(t_n)\end{aligned} \tag{4.12}$$

There are two forms of the assimilation of variance. The first form is found by substituting (4.12) into the centralized information update equation (4.10).

$$\mathbf{P}_i^{-1}(t_n|t_n) = \mathbf{P}_i^{-1}(t_n|t_{n-1}) + \sum_{j=1}^{M} \mathbf{C}_j^{\mathrm{T}}(t_n)\mathbf{R}_j^{-1}(t_n)\mathbf{C}_j(t_n) \tag{4.13a}$$

where this is the assimilated information update computed locally at node i. Solving for the sensor information matrix, $\mathbf{C}_i^{\mathrm{T}}(t_n)\mathbf{R}_i^{-1}(t_n)\mathbf{C}_i(t_n)$, in (4.6) and substituting this into the summation in (4.13a) we get a second form for the assimilation of variance.

$$\mathbf{P}_i^{-1}(t_n|t_n) = \mathbf{P}_i^{-1}(t_n|t_{n-1}) + \sum_{j=1}^{M} \tilde{\mathbf{P}}_j^{-1}(t_n|t_n) - \mathbf{P}_j^{-1}(t_n|t_{n-1}) \qquad (4.13b)$$

Because we have developed (4.13a) and (4.13b) from the centralized Kalman filter equations, the assimilation of variance equations are equivalent to the centralized information update given in (4.10). These two forms are mathematically identical, however, the two forms differ in what quantities are communicated and what processing is performed at the each node after the communication.

4.1.2 Assimilation of State

The details of the development of the assimilation of state can be found in [18]. Similar to the covariance update equations, there are also two mathematically equal state update equations. The first state update equation uses the measurements from each local node.

$$\hat{x}_i(t_n|t_n) = \mathbf{P}_i(t_n|t_n)\left(\mathbf{P}_i^{-1}(t_n|t_{n-1})\hat{x}_i(t_n|t_{n-1}) + \sum_{j=1}^{M} \mathbf{C}_j^{\mathrm{T}}(t_n)\mathbf{R}_j^{-1}(t_n)y_j(t_n)\right) \qquad (4.14a)$$

The second state update equation uses the local state updates to form the global state update.

$$\hat{x}_i(t_n|t_n)=\mathbf{P}_i(t_n|t_n)\left(\mathbf{P}_i^{-1}(t_n|t_{n-1})\hat{x}_i(t_n|t_{n-1})+\sum_{j=1}^{M}\tilde{\mathbf{P}}_j^{-1}(t_n|t_n)\tilde{x}_j(t_n|t_n)-\mathbf{P}_j^{-1}(t_n|t_{n-1})\hat{x}_j(t_n|t_{n-1})\right) \qquad (4.14b)$$

These equations for the assimilation of state are not important for sensor management purposes since the state error covariance is of primary concern. We do, however, consider these equations in Section 4.1.3 to consider several different communication methods looking at communication and computational load.

4.1.3 Communication and Computation Load

The choice of which DKF equations to use in a distributed system should be made partially in consideration of the following analysis of the different communication methods. Using the assimilation equations of state and covariance, we define the following four methods, where the state vector is length N, the measurement vector at the ith node is length m_i, and the nodes are indexed $i \in \{1, \ldots, M\}$.

1. Method **I** uses (4.13b) and (4.14b) and pre-computes $\tilde{\mathbf{P}}_i^{-1}(t_n|t_n) - \mathbf{P}_i^{-1}(t_n|t_{n-1}) \in \Re^{N \times N}$ and $\tilde{\mathbf{P}}_i^{-1}(t_n|t_n)\tilde{x}_i(t_n|t_n) - \mathbf{P}_i^{-1}(t_n|t_{n-1})\hat{x}_i(t_n|t_{n-1}) \in \Re^{N \times 1}$ before transmitting.
2. Method **II** uses (4.13a) and (4.14a) and pre-computes $\mathbf{C}_j^{\mathrm{T}}(t_n)\mathbf{R}_j^{-1}(t_n)\mathbf{C}_j(t_n) \in \Re^{N \times N}$ and $\mathbf{C}_i^{\mathrm{T}}(t_n)\mathbf{R}_i^{-1}(t_n)y_i(t_n) \in \Re^{N \times 1}$ before transmitting.
3. Method **III** uses (4.13b) and (4.14b) with no pre-transmission computation. The transmitted data for this method is the local information matrix $\tilde{\mathbf{P}}_i^{-1}(t_n|t_n) \in \Re^{N \times N}$, and the local estimate $\tilde{x}_i(t_n|t_n) \in \Re^{N \times 1}$. All assimilation computations are performed at the receiving nodes.
4. Method **IV** uses (4.13a) and (4.14a) with no pre-transmission computation. The data transmitted for this method is the measurement matrix $\mathbf{C}_i(t_n) \in \Re^{m_i \times N}$, the sensor information matrix $\mathbf{R}_i^{-1}(t_n) \in \Re^{m_i \times m_i}$, and the measurement vector $y_i(t_n) \in \Re^{m_i}$. All assimilation computations are performed at the receiving nodes.

Table 1 shows the maximum transmission and reception loads for each of the four different methods. The maximum load occurs when every node receives a measurement from its sensors and then immediately transmits the respective information to every other node. We developed Table 1 using the fact that symmetric matrices in $\Re^{N \times N}$ have $(N^2 + N)/2$ different entries. Methods **I**, **II**, and **III** have maximum reception loads that are $M - 1$ times their respective transmission loads. Method **IV** has a different maximum reception load for each node because it depends on the length of the other nodes' measurement vectors. Methods **I**, **II**, and **III** have the same transmission and reception loads.

Table 1: Transmission and reception load at the *i*th node for different communication methods

Method	Maximum Transmission Load	Maximum Reception Load
I	$O[(N^2+3N)/2]$	$O[(M-1)((N^2+3N)/2)]$
II	$O[(N^2+3N)/2]$	$O[(M-1)((N^2+3N)/2)]$
III	$O[(N^2+3N)/2]$	$O[(M-1)((N^2+3N)/2)]$
IV	$O[(m_i^2+3m_i)/2+m_iN]$	$O\left[\sum_{j\neq i}^{M}\left(\frac{(m_j^2+3m_j)}{2}+m_jN\right)\right]$

Table 2 and Table 3 present a comparison of the computational load for each communication scheme. When possible, multiplies and adds are considered as one floating point operation (flop); otherwise, an addition and a multiplication are considered as separate operations. The computation is divided between pre-transmission computation and post-reception computation. Each method shows the computation involved with computing the state and covariance. Notice how in methods **III** and **IV** the pre-transmission computation is zero. These two methods allow for faster transmission of measurement data at the expense of increased post-reception computation. Method **IV** makes use of local computations given in (4.6) and (4.8) to reduce the computation given in (4.13a) and (4.14a).

Table 2: Pre-transmission computational load at the *i*th node for different communication methods

Method	Maximum Pre-Transmission Computation	
	State	Covariance
I	$O[2N^2+N]$	$O[(N^2+N)/2]$
II	$O[m_i^2+Nm_i]$	$O\left[Nm_i^2+\frac{N^2+N}{2}m_i\right]$
III	$O[0]$	$O[0]$
IV	$O[0]$	$O[0]$

Table 3: Post-reception computational load at the *i*th node for different communication methods

Method	Maximum Post-Reception Computation	
	State	Covariance
I	$O[MN+2N^2]$	$O[M(N^2+N)/2]$
II	$O[MN+2N^2]$	$O[M(N^2+N)/2]$
III	$O[2N^2(M+1)+2MN]$	$O[(M-1)(N^2+N)]$
IV	$O\left[2N^2+MN+\sum_{j=1}^{M} m_j^2+m_jN\right]$	$O\left[\sum_{j=1}^{M} m_j^2N+(m_j+1)\left(\frac{N^2+N}{2}\right)\right]$

Method **IV** has the following advantages:

1. When transmission of data is not received by other nodes due to channel disturbances, the pre-transmission computation is wasted at the local processors. Since there is no pre-transmission computation in method **IV**, computing resources can be used for some other tasks.
2. Since there is zero pre-computational load in method **IV**, the entire assimilation process can be better optimized at each receiver yielding reduced computation over the other methods. For example, in method **II**, the multiplies and adds are computed at two different nodes.
3. When measuring a subspace of the full state vector, $m_i < N$. For small m_i, the transmission and reception load can be reduced considerably over that of the other methods.
4. Having raw measurement data can help in various processes [2].

The disadvantage of using method **IV** is a higher computational demand. We now use Tables 1, 2, and 3 to illustrate the trade off between communication and computation.

Example:

There are seven processors (*M=7*), six states (*N=6*), and measurements of length two ($m_i = 2$) at each node. Let *S* be the number of bits per symbol. Method **IV** has transmit and receive communication loads of $17 \times S$ bits and $102 \times S$ bits, respectively. In methods **I,II**, and **III** the transmit and receive communication loads are $27 \times S$ bits and $162 \times S$ bits, respectively. Thus, when using method **IV**, the transmit communication load savings is $10 \times S$ bits and the receive communication load savings is $60 \times S$ bits at each node, during each sample interval.

The per-processor computational load (including all pre-transmission and post-reception computation) of method **I**, **II**, **III**, and **IV** is *360* flops, *343* flops, *912* flops, and *835* flops, respectively. In general, methods **I** and **II** will have lower per processor computational loads than methods **III** and **IV** because a larger portion of the total computation is performed in parallel at each processor before transmission. This shows that for these parameters, method **IV** has the disadvantage of having increased computation over that of methods **I** and **II**. When the communication channels are very reliable, methods **I** and **II** are appealing because of the reduced total computational load.

4.2 Decentralized Covariance Control Approach

The optimal solution for controlling the covariance in a decentralized network must consider all combinations of sensors and sampling rates across all nodes, but this combinatorial search is very computationaly expensive. A solution to this problem is to allow each node to make sensing decisions independently of the other nodes, however, this makes it difficult to coordinate sensing efforts among the nodes. Communication schemes where all nodes reach a consensus before performing sensing actions can restore that coordination. An example of this are schemes where each node "bids" for sensing actions based on its abilities and current sensing load. The drawback to these methods is that they add a rather high communication burden on what may already be a heavily loaded system.

We have developed decentralized covariance control algorithms that avoid adding excessive communication overhead by only allowing one node to make sensing decisions at a time. Although this will usually not result in the optimal solution, these algorithms provide a coordinated sensing effort while maintaining a relatively high level of nodal autonomy. The systems presented here consist of a network of M processors each controlling a suite of m_i sensors. All nodes are initialized to have the same desired covariance matrices. When one or more nodes request that the desired covariance should be changed, either all nodes change the desired covariances or none of the nodes change the desired covariance.

4.2.4 Ordered Nodes Algorithm

The *ordered nodes* algorithm imposes a random ordering on the networked processors. This ordering determines the sensing order and when the last node in the order finishes a sensing task, the first node resumes the process by performing another sensing task. The time it takes to complete one cycle of each node using a sensor will be called the *intranodal sample period.* The time from one node receiving a measurement to the next node receiving its measurement will be called the *internodal sample period.* In steady state, when each node uses the same suite of sensors, the intranodal sample period will be constant across all nodes. The internodal sample period, however, will be different because each node might have different sensors providing more or less information regarding the state. The sum of the internodal sample periods equals the intranodal sample period. The ordered nodes algorithm works as follows:

Algorithm: Ordered Nodes

Variables: $i, j \in \{1, \ldots, M\}$, $\mathbf{P}_{dp}(t_n)$, $\mathbf{P}_{du}(t_n)$, T_{min}

Start:

I. A global random nodal ordering is selected for all nodes.

II. The *i*th node acquires a target, providing the first measurements and measurement covariances to all remaining nodes.

1. *Computation of internodal sample period*: Based on the update covariance $\mathbf{P}_i(t_n|t_n)$ at the *i*th node and $\mathbf{P}_{dp}(t_n)$, the *j*th node ($j = mod(i, M) + 1$) computes the internodal sample period $T_j(t_n)$ and the associated inverse prediction covariance $\mathbf{P}_j^{-1}(t_n + T_j|t_n)$. If $T_j < T_{min}$ then $T_j = T_{min}$, where T_{min} is the smallest internodal period allowable due to communication and sensor limitations.

2. *Computation of sensor resolution*: Node j uses $\mathbf{P}_j^{-1}(t_n + T_j|t_n)$, $\mathbf{P}_{du}^{-1}(t_n)$, and the available suite of sensors to compute the optimal sensor set Φ_o and the associated information update $\mathbf{P}_j^{-1}(t_{n+1}|t_{n+1})$.
3. *Measurement*: When $t = t_n + T_j(t_n)$, node j uses the optimal set of sensors Φ_o.
4. *Communication*: Communicate measurements and measurement covariances to all other nodes. Let $i = j$.
5. Goto step 1 and repeat process.

End.

This algorithm provides good nodal autonomy at the expense of less efficient use of sensing resources. The only communication is the measurement data and the measurement covariance. In the Ordered Nodes Algorithm the sensing load for each node is reduced compared with one node doing all the sensing. By interleaving measurements from each node the intranodal sample rate is reduced by a factor of M.

When the internodal sample periods $T_j(t_n)$ and the optimal sets of sensors $\Phi_j(t_n)$ become periodic, the coefficients of the state and measurement equations also become periodic with a period equal to the intranodal sample period. We now give two examples showing how this algorithm works by choosing system parameters and showing some simulation results.

Example 1:

Three nodes track one target in a single coordinate. Figure 3.2a shows the sensor usage for the three nodes where nodes 1, 2, and 3 have 2, 3, and 4 sensors respectively. This simulation used the following coefficients for the three nodes.

$$\begin{gathered} \mathbf{A}_i = 1, \mathbf{Q}_i(t_n) = \sigma^2 T(t_n), i \in \{1,2,3\} \\ \mathbf{C}_1(t_n) \in \Re^{2\times 1}, \mathbf{C}_2(t_n) \in \Re^{3\times 1}, \mathbf{C}_3(t_n) \in \Re^{4\times 1} \\ \mathbf{R}_1 = diag[13, 23], \mathbf{R}_2 = diag[6, 22, 42], \mathbf{R}_3 = diag[10, 15, 18, 35] \end{gathered} \tag{4.15}$$

The measurement matrices at each node have elements that are 0 or 1, denoting whether or not each sensor is used. Taking all permutations gives us all possible measurement matrices:

$$\mathbf{C}_1(t_n) \in \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\} \tag{4.16}$$

$$\mathbf{C}_2(t_n) \in \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\} \tag{4.17}$$

$$\mathbf{C}_3(t_n) \in \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\} \tag{4.18}$$

The size of each of these sets is 2^{m_i} where m_i is the number of sensors at the ith node. The sensor resolution is then determined by the quadratic form

$$\mathbf{C}_i^{\mathrm{T}}(t_n)\mathbf{R}_i^{-1}\mathbf{C}_i(t_n) \tag{4.19}$$

The sensors at each node are ordered from smallest to largest variance. The minimum internodal period, T_{min}, was set to $1s$. The internodal rate and sensor resolutions for all nodes are shown in plots (b) and (c), respectively. At sample $n = 21$, the desired update and prediction variances are increased which cause the sensor rate and resolution to both decrease. The peaks in the sensor resolution are due to nodes using sensor combinations that achieve the highest resolution during the respective time periods.

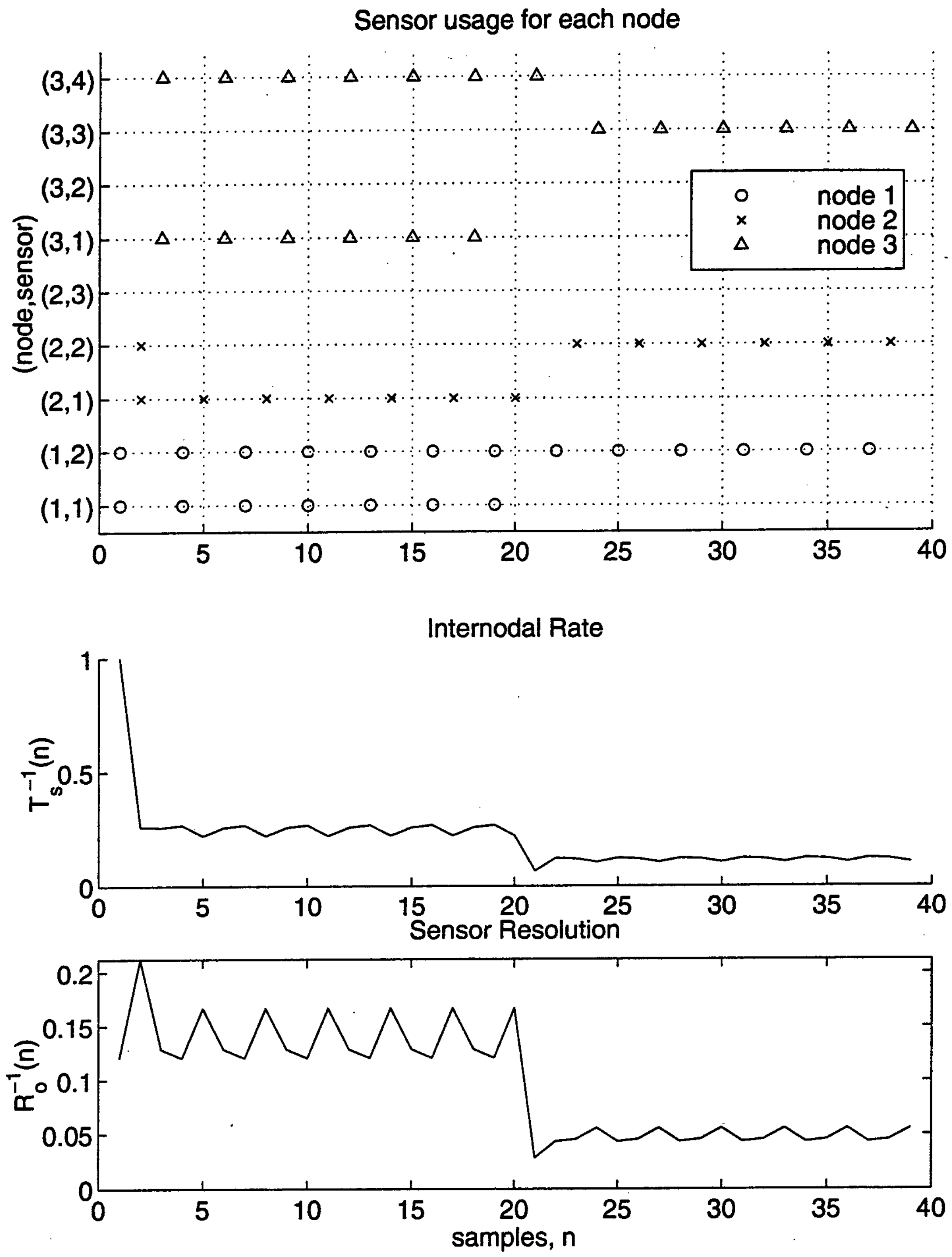


Figure 18 Ordered Nodes Algorithm results: (a) nodal sensor usage, (b) internodal rate, and (c) sensor resolution (simulation 1)

Each of these peaks in the resolution have a corresponding reduction in the computed internodal rate. For instance, when node 2 uses its 1*st* sensor at $n = 5$, the resolution peaks and the rate (computed by node 3) decreases. These two plots illustrate the interplay between rate and resolution in maintaining a desired variance.

Figure 19 plots the error variance and the desired update and prediction variances. The *peaks* and *troughs* of the sawtooth waveform correspond to the prediction and update variances, respectively. Between samples the error variance increases linearly because a Wiener process is used to model the target motion. The slope of the line is determined by the white noise variance, σ^2. The error variance is plotted versus time, and illustrates how the sample period increases during the second part of the

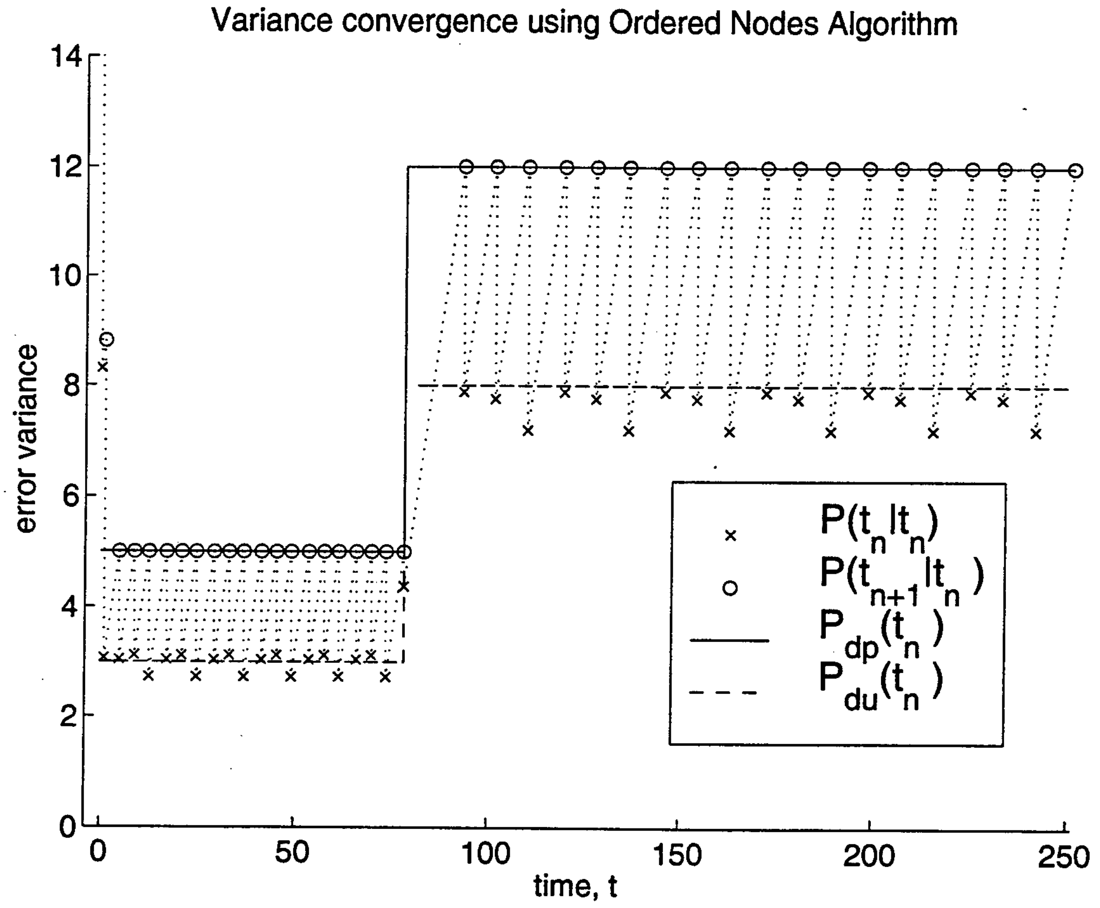


Figure 19 Variance convergence in Ordered Nodes Algorithm (simulation 1).

simulation. When each node sequentially uses the same sensor(s) and the same internodal rate, the steady-state error covariance can be modeled with a discrete *periodic*

Riccati equation (DPRE) [5]. The properties of the coefficients of a DPRE are

$$\begin{aligned}\mathbf{A}(t_n) &= \mathbf{A}(t_n + T)\\ \mathbf{Q}(t_n) &= \mathbf{Q}(t_n + T)\\ \mathbf{C}(t_n) &= \mathbf{C}(t_n + T)\\ \mathbf{R}(t_n) &= \mathbf{R}(t_n + T)\end{aligned} \tag{4.20}$$

where $T = \sum_{i=1}^{M} T_i(t_n)$ is the intranodal sample period. With a scalar state the ordered nodes algorithm will always result in a DPRE because the prediction variance given by the o's in Figure 19 are always the same for computing the best sensor combination.

Example 2:

The second simulation using the ordered nodes algorithm uses three nodes each having three sensors. For each node, the sensors are ordered from "best" to "worst". The level curves of each sensor covariance is shown in Figure 20. After the 50*th* sample the elliptical annulus determined by the two desired information matrices is reduced. Each

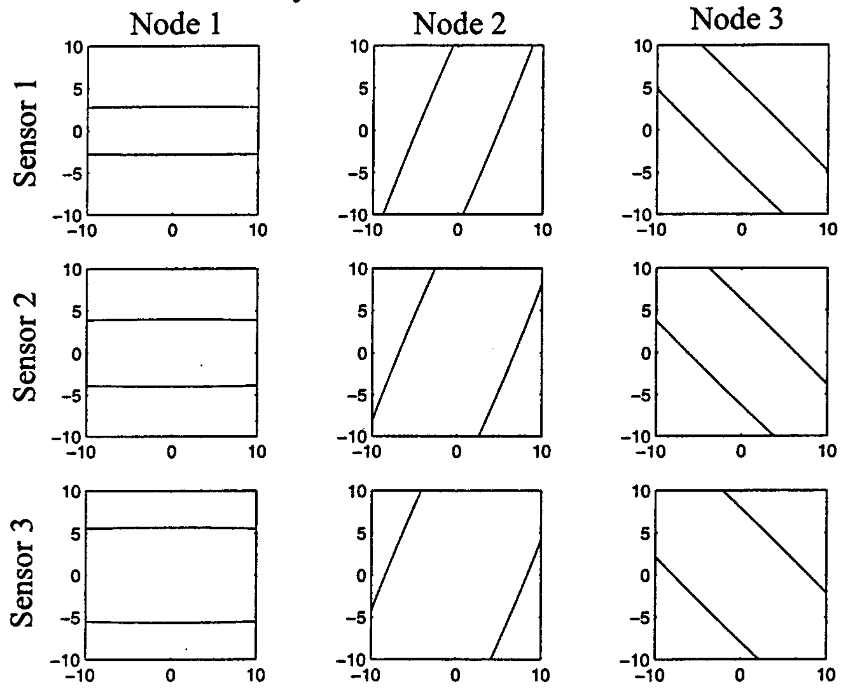


Figure 20 Covariance ellipses for each node's sensors

node's sensor manager responds by reducing the rate and resolution. The maximum internodal rate was specified to be 2Hz. Notice that during the first 50 samples each node uses all its available sensors. After the desired information is reduced, there is an

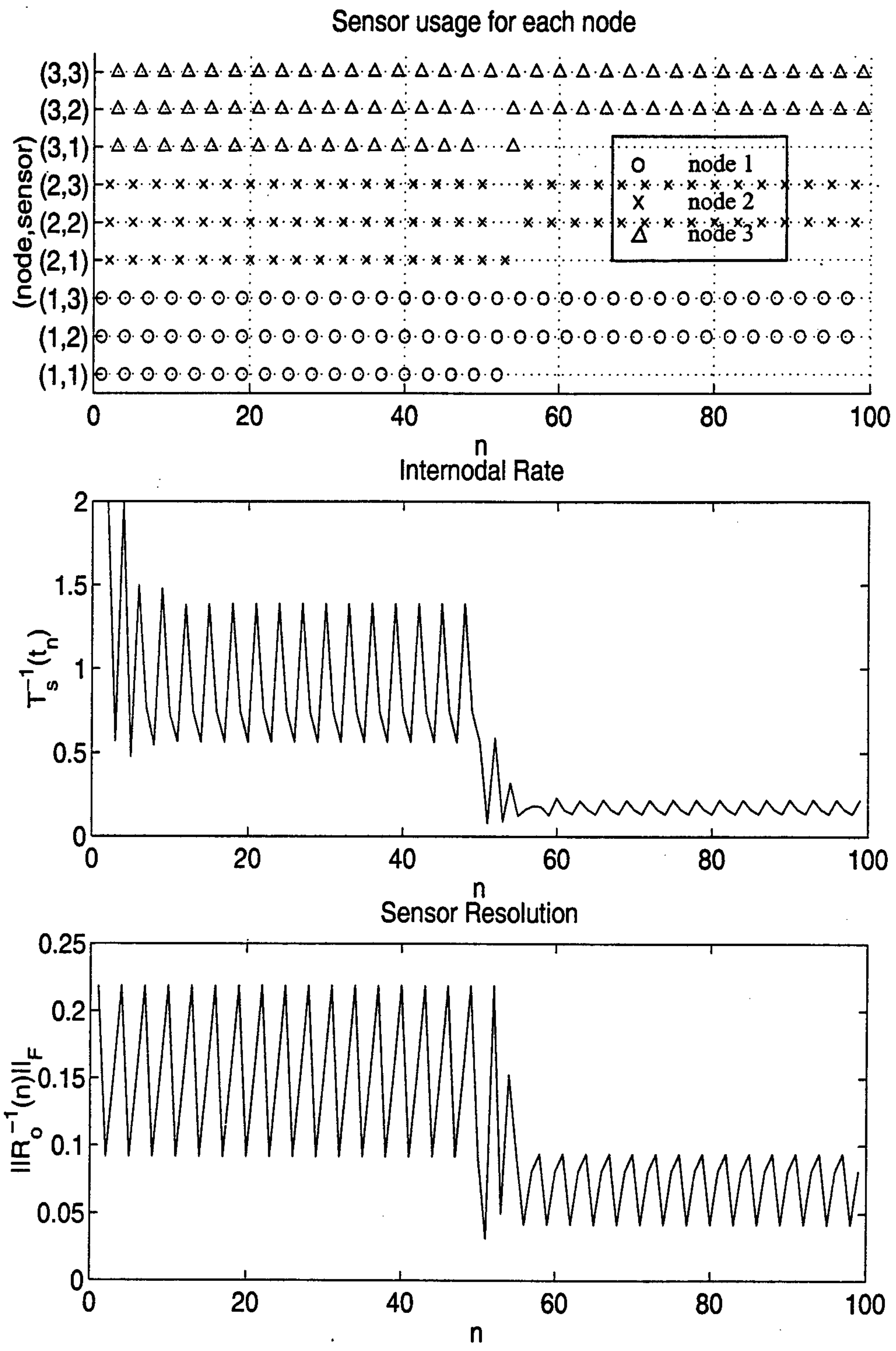


Figure 21 Ordered Nodes Algorithm results: (a) nodal sensor usage, (b) internodal rate, and (c) sensor resolution (simulation 2)

initial transient period after which the nodes always choose the same sensors.

The metric used to find the best set of sensors to bring the update covariance "close" to the desired update covariance was the Frobenius norm. This metric was used along with checking the condition $\lambda_i(\mathbf{P}_{\Gamma}^{-1}(t_n|t_n) - \mathbf{P}_{du}^{-1}) > 0$, which insured that the actual sensor information was greater than the desired. The subset of combinations of sensors that achieve this condition were considered and the one that achieved the smallest Frobenius norm was chosen. When no sensor combinations achieved the desired information, then the sensor set that minimized the Frobenius norm was chosen.

The prediction covariance is allowed to grow until it is tangent to the desired prediction covariance. The determinant is the function used to determine the optimal sample period that achieves making the ellipse of the prediction covariance tangent to the ellipse of the desired prediction covariance.

Figure 22 shows the prediction and update covariances during each half of the simulation. The desired covariances are indicated by *thick* solid ellipses. Notice that during the first part of the simulation the update covariance is never inside the desired update covariance. Even though each node is using all its sensors the upper bound is not achieved. During the second half of the simulation, however, some update covariances are completely within the desired update covariance.

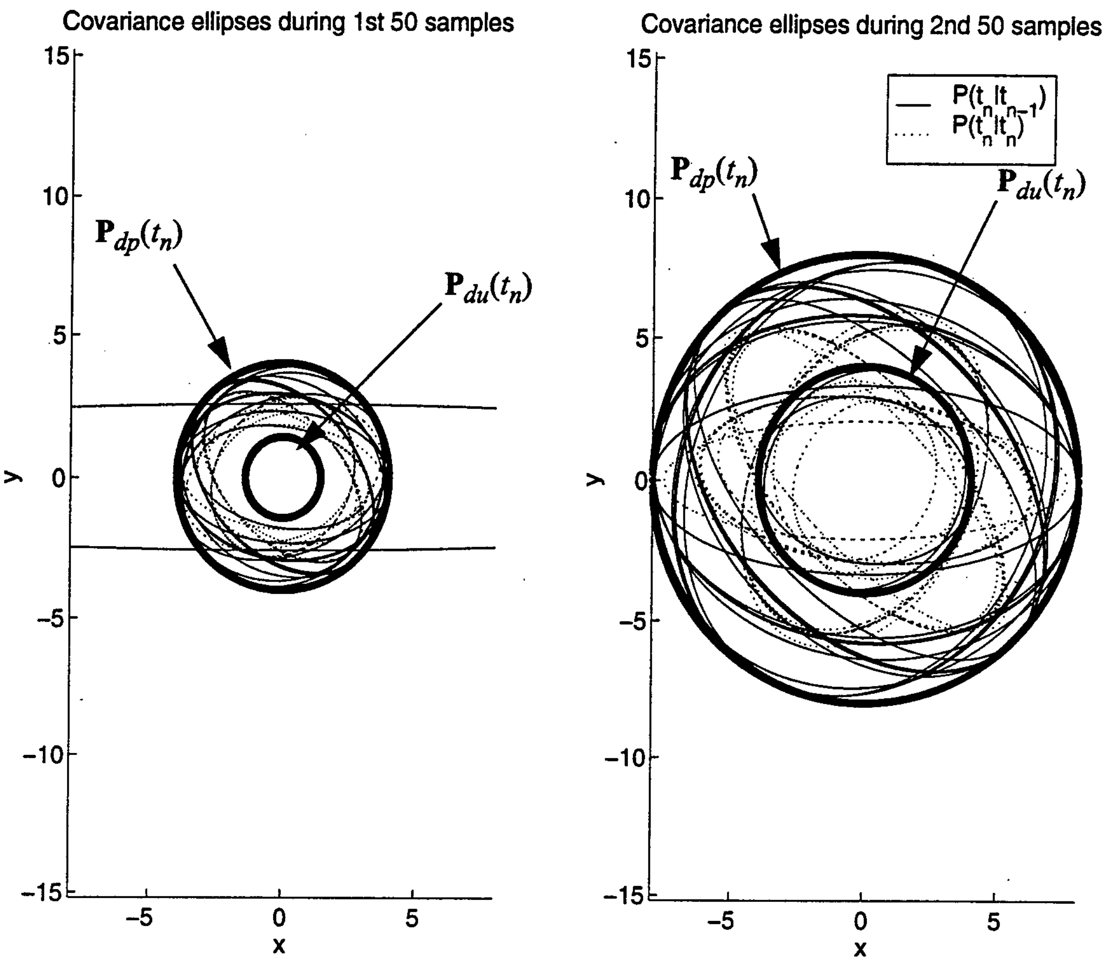


Figure 22 Covariance convergence in Ordered Nodes Algorithm

4.2.5 Extended Ordered Nodes Algorithm

The *extended ordered nodes* algorithm also imposes a random ordering on the networked processors. This ordering determines the sensing order and when the last node in the order finishes a sensing task, the first node resumes the process by performing another sensing task. The extended ordered nodes algorithm works as follows:

Algorithm: Extended Ordered Nodes

Variables: $i, j \in \{1, ..., M\}$, $\mathbf{P}_{dp}(t_n)$, $\mathbf{P}_{du}(t_n)$, T_{min}

Start:

I. A global random nodal ordering is selected for all nodes.

II. The *i*th node acquires a target, providing the first measurements and measurement covariances for all remaining nodes.

1. *Computation of internodal sample period*: Based on the update covariance $\mathbf{P}_i(t_n|t_n)$ at the *i*th node and the desired prediction covariance $\mathbf{P}_{dp}(t_n)$, the *j*th node $(j = mod(i, M) + 1)$ computes the internodal sample period $T_j(t_n)$ and the associated prediction information $\mathbf{P}_j^{-1}(t_n + T_j|t_n)$. If $T_j < T_{min}$ then let $T_j = T_{min}$.
2. *Search for sensor combinations*: The *j*th node uses $\mathbf{P}_j^{-1}(t_n + T_j|t_n)$, $\mathbf{P}_{du}^{-1}(t_n)$, and Φ_j to compute the optimal sensor set Φ_{oj} and $\mathbf{P}_j^{-1}(t_{n+1}|t_{n+1})$. The *j*th node computes $\lambda_{min}(\mathbf{M})$ where $\mathbf{M} = \mathbf{P}_{du}^{-1}(t_n) - \mathbf{P}_j^{-1}(t_{n+1}|t_{n+1})$.
3. *Intersample internodal communication*: If $\lambda_{min}(\mathbf{M}) > 0$, then request additional sensing resources from the *k*th node $(k = mod(j, M) + 1)$, communicate $\sum_{i \in \Phi_{oj}} \mathbf{C}^{\mathrm{T}}\mathbf{R}_i^{-1}\mathbf{C}$, $T_j(t_n)$, and let $j = k$. Goto step 2. If $\lambda_{min}(\mathbf{M}) < 0$, goto step 4.
4. *Measurement*: When $t = t_n + T_j(t_n)$, then each node participating in the joint optimization samples using the optimal set(s) of sensors Φ_{oj}.
5. *Communication*: Each node taking measurements communicates measurements and measurement covariances to all other nodes.
6. Goto step 1 and repeat process at the next node in order.

End.

This algorithm provides improved covariance control. There are two communication phases per internodal sample period, unless a node does not request additional sensing resources. The first communication period is for finding the optimal set of sensors. The second communication period is after each node has chosen what sensors to use and communicates the measurements and measurement covariances.

If an optimal search is performed between two nodes, this could cause one nodes sensing resources to not be used at all. Besides this problem there are several other problems associated with doing an optimal search in this algorithm. When a down stream node computes a new combination for previous nodes sensors, communication would have to go in both directions after each additional node computes the next optimal combination of sensors. A suboptimal search that groups previous nodes' sensors would allow for a simplex transmission during the internodal intersample optimization. This grouping reduces the size of the internodal search, allows each node to control the use of its own suite of sensors, and requires each node to participate in the sensing process.

In some tracking scenarios, better estimates are required in some directions than in other directions. For example, when locking onto a target to fire, a certain minimum estimation error must be achieved before the weapon may be fired. In this situation the estimates of the angles to the target may be more important than the range. Desired covariances can be chosen to simulate this type of requirement.

The following example shows how the extended ordered nodes algorithm performs. For this simulation the desired covariance matrices were given disparate eigenvalues and the desired update and desired prediction matrices are related by a positive scale factor.

Example 3:

Five nodes track one target in the x-y plane. Each node has three sensors. The desired covariances are decreased after the $25th$ sample. The minimum internodal sample period is 0.5 seconds. After the 25*th* sample, the rate and resolution both

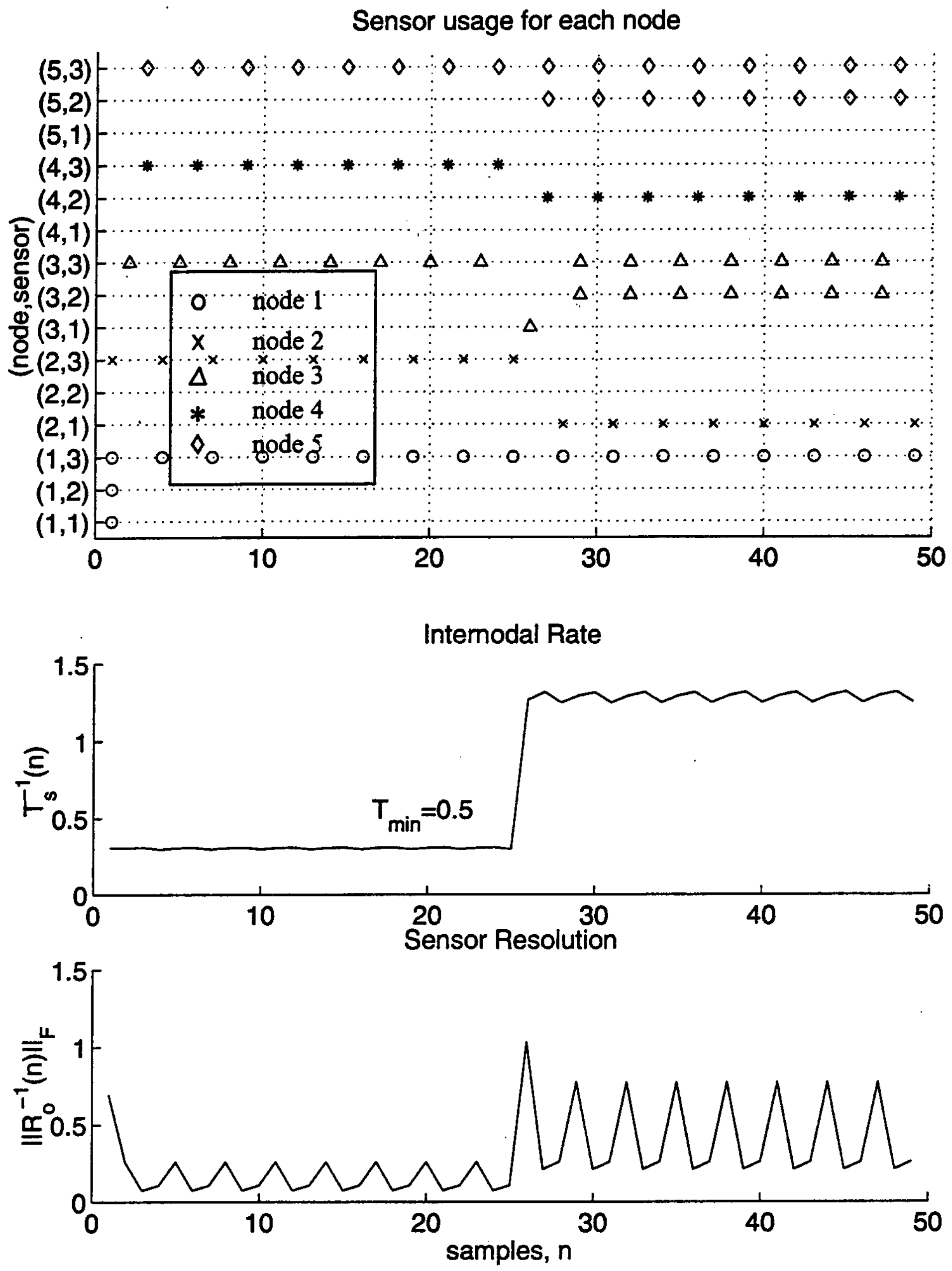


Figure 23 Extended Ordered Nodes Algorithm results: (a) nodal sensor usage, (b) internodal rate, and (c) sensor resolution

increase in response to the increase in desired information. Notice also that internodal sampling occurs between different nodes.

The next plot shows the desired covariances during the first half and second half of the simulation. The update and prediction covariances are also shown during each part of the simulation. The desired covariance goals are treated as upper bounds on the update and prediction covariances. The determinant is used for computing the optimal rate and the Frobenius norm is used for computing the optimal resolutions.

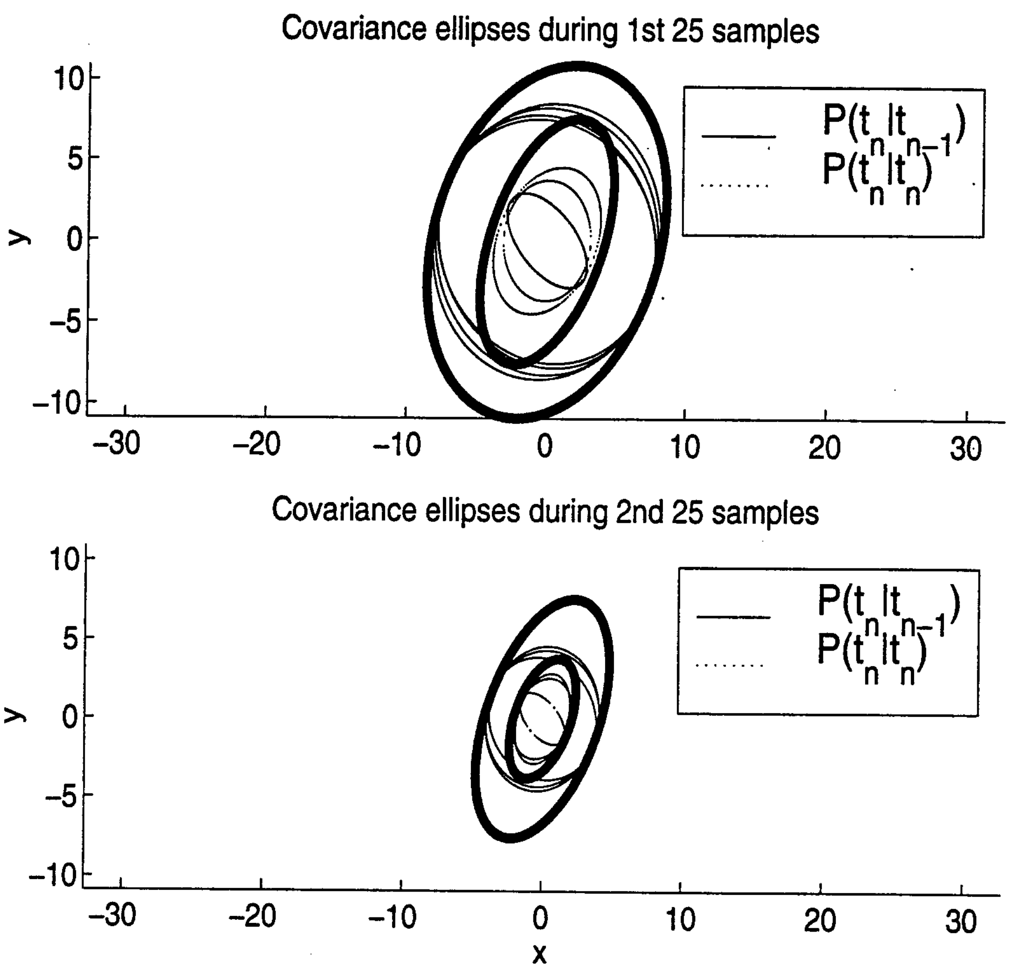


Figure 24 Covariance control in the Extended Ordered Nodes algorithm.

Notice how the update covariance matrices are always less than the desired update covariance matrices, i.e. $\mathbf{P}_i(t_n|t_n) < \mathbf{P}_{du}(t_n)$. This illustrates how the extended ordered nodes algorithm is more flexible at achieving desired covariance goals.

4.3 Periodic Riccati Difference Equation

The ordered nodes algorithm in example 1 showed that for a single state the algorithm always results in a discrete periodic Riccati equation (DPRE). Experimentally, using simulations of higher order systems, the ordered and extended ordered nodes algorithms appear to always result in a periodic sequence for the optimal rate and resolution. This seems to indicate that each algorithm may also result in a DPRE for higher order systems. To accurately compute the steady state performance of a Kalman filter estimator whose rate and resolution is varying periodically in time requires solving the DPRE.

Using an analysis similar to that reviewed for the constant coefficient DMRE, we here review a solution for the DPRE. This technique changes the *Nth* order periodically time-varying nonlinear matrix difference equation into a 2*Nth* order constant coefficient linear matrix difference equation [5]. The time-varying Riccati equation is

$$\begin{aligned}\mathbf{P}(n+1) = {} & \mathbf{Q}(n) + \mathbf{A}(n)\mathbf{P}(n)\mathbf{A}^{\mathrm{T}}(n) - \\ & \mathbf{A}(n)\mathbf{P}(n)\mathbf{C}^{\mathrm{T}}(n)(\mathbf{R}(n) + \mathbf{C}(n)\mathbf{P}(n)\mathbf{C}^{\mathrm{T}}(n))^{-1}\mathbf{C}(n)\mathbf{P}(n)\mathbf{A}^{\mathrm{T}}(n)\end{aligned} \tag{4.21}$$

where the coefficients have the properties $\mathbf{A}(n) = \mathbf{A}(n+\Delta)$, $\mathbf{C}(n) = \mathbf{C}(n+\Delta)$, $\mathbf{Q}(n) = \mathbf{Q}(n+\Delta)$, and $\mathbf{R}(n) = \mathbf{R}(n+\Delta)$ with Δ an integer. The periodic symplectic matrix is

$$\mathbf{M}(n) = \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{Q}(n) & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{A}(n)^{-\mathrm{T}} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}(n)\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{C}^{\mathrm{T}}(n)\mathbf{R}(n)^{-1}\mathbf{C}(n)\\ \mathbf{0} & \mathbf{I}\end{bmatrix} \tag{4.22}$$

Consider the periodic linear homogeneous matrix difference equation

$$\mathbf{Y}(n+1) = \mathbf{M}(n)\mathbf{Y}(n) \tag{4.23}$$

where $\mathbf{Y}(n) = \begin{bmatrix}\mathbf{U}(n)\\ \mathbf{V}(n)\end{bmatrix}$, $\mathbf{P}(n) = \mathbf{V}(n)\mathbf{U}(n)^{-1}$ and $\mathbf{U}(n), \mathbf{V}(n) \in \Re^{N\times N}$. This equation can be changed into a time invariant equation as follows

$$\mathbf{Y}(n+1) = \mathbf{M}(n)\mathbf{Y}(n)$$

$$\mathbf{Y}(n+2) = \mathbf{M}(n+1)\mathbf{Y}(n+1) = \mathbf{M}(n+1)\mathbf{M}(n)\mathbf{Y}(n)$$

$$\cdots$$

$\mathbf{Y}(n+\Delta) = \mathbf{M}(n+\Delta-1)\mathbf{Y}(n+\Delta-1) = \mathbf{M}(n+\Delta-1)\ldots\mathbf{M}(n)\mathbf{Y}(n)$

This last equation can be written as $\tilde{\mathbf{Y}}(m+1) = \tilde{\mathbf{M}}\tilde{\mathbf{Y}}(m)$ where

$$\tilde{\mathbf{M}} = \prod_{i=n}^{n+\Delta-1} \mathbf{M}(i) \tag{4.24}$$

with $\tilde{\mathbf{M}}$ now time-invariant. Since each of the $\mathbf{M}(i)$ are symplectic, the product $\tilde{\mathbf{M}}$ is also symplectic (see Appendix B). With initial conditions $\mathbf{U}(0) = \mathbf{I}$ and $\mathbf{V}(0) = \mathbf{P}_0$, the solution of $\tilde{\mathbf{Y}}(m)$ and $\tilde{\mathbf{P}}(m)$ is $\tilde{\mathbf{Y}}(m) = \tilde{\mathbf{M}}^m\tilde{\mathbf{Y}}(0)$ and $\tilde{\mathbf{P}}(n) = \tilde{\mathbf{V}}(n)\tilde{\mathbf{U}}(n)^{-1}$, respectively. This solution for the covariance is a down sampled version of the original covariance. The two covariance matrices $\mathbf{P}(m)$ and $\tilde{\mathbf{P}}(m)$ are related by $\tilde{\mathbf{P}}(m) = \mathbf{P}(m\Delta + m_o)$. This concludes the review of one solution of the DPRE.

Chapter 5

CONCLUSION

Target tracking models were reviewed in chapter 2. The Kalman filter was reviewed for both centralized networks and decentralized networks. Solutions to the scalar and matrix Riccati difference equations were researched and these solutions reviewed. Due to the development of several new distributed sensor management algorithms that resulted in periodic behavior of the system coefficients, we also researched and reviewed a solution to the discrete periodic Riccati equation (DPRE). While these solutions to the Riccati equations were not used in the sensor managers, an understanding of the solution allows us to better predict the steady state performance of the Kalman filter so that we can make better choices of the desired covariance.

Using a covariance control approach we developed a novel sensor management scheme based upon the choice of two desired covariances. The desired prediction covariance was used to control the prediction covariance through the choice of sample rate. The desired update covariance was used for controlling the update covariance through the choice of sensor combinations. Analysis of different functions were given for measuring the "distance" between two positive definite covariance matrices. Two new metrics were developed based upon using the singular value decomposition (SVD) of the desired covariance matrix and the prediction or update covariance matrix. These covariance control techniques were used to develop two algorithms for distributed sensor management. Analysis of these algorithms was done by evaluating a number of examples, three of which were presented in detail in this thesis. These examples illustrated the techniques and how the algorithms performed.

Future work consists of applying the above sensor management techniques to multitarget tracking scenarios. This may require development of new metrics due to the additional issues involved with tracking multiple targets. Some of these issues are 1) better description of sensors in terms of agile and non-agile sensing resources and sensor capabilities, 2) crossing or interacting targets and a desire to keep them separated, and 3) addressing cluttered measurements in the development of better filtering algorithms and its consequent effects on the sensor manger.

REFERENCES

[1] Ablowitz, M. J. and Fokas, S. A., *Complex Variables: Introduction and Applications,* Cambridge University Press, Cambridge, United Kingdom, 1997.

[2] Bar-Shalom, Y. and Fortmann, T., *Tracking and Data Association,* Academic Press, Inc., San Diego, 1988.

[3] Bar-Shalom, Y. and Li, X., *Estimation and Tracking: Principles, Techniques, and Software,* Artech House, Boston, 1993.

[4] Berg, T. M. and Durrant-Whyte, H. F., "Model Distribution in Decentralized Multi-Sensor Data Fusion", *Proc. American Control Conference*, vol. 3, pp. 2292-3, 1991.

[5] Bittanti, S., Laub, A. J. and Willems, J.C., *The Riccati Equation,* Springer-Verlag, Berlin, 1991.

[6] Blair, W. D. and Watson G. A., *Benchmark Problem for RADAR resource allocation and tracking maneuvering targets in the presence of ECM.* Dahlgren Division, Naval Surface Warfare Center, Systems Research and Technology Department, Sept. 1996.

[7] Hong, L., Ding Z., and Wood R. A., "Development of multirate model and multirate interacting multiple model algorithm for multiplatform multisensor tracking," *SPIE Optical Engineering,* vol. 37, no. 2, pp. 453-67, Feb. 1998.

[8] Kalandros, M. and Pao, L. Y., "Controlling Target Estimate Covariance in Centralized Multisensor Systems," *Proceedings of the American Control Conference*, Philadelphia, PA, pp. 2749-2753, June 1998.

[9] Kocic, V. L. and Ladas, G., "Global Behavior of Nonlinear Difference Equations of Higher Order with Applications," *Mathematics and Its Applications*, vol. 256, pp 177-188, Kluwer Academic Publishers, Dordrecht, Netherlands, 1993.

[10] Laub, J. A. and Meyer, K., "Canonical Forms for Symplectic and Hamiltonian Matrices," *Celestial Mechanics*, pp 213-238, D. Reidel Publishing Company, Dordrecht, Holland, 1974.

[11] Musick S. and Malhotra R., "Chasing the Elusive Sensor Manager," *Proceedings of the IEEE 1994 NAECON,* Vol. 1, pp. 606-613, Dayton, OH, IEEE: New York, NY, May 1994.

[12] Mullis, C.T., Advanced Linear Systems Class Notes, University of Colorado, 1998.

[13] Nash, J., "Optimal Allocation of Tracking Resources," *Proceedings of the 1977 IEEE Conference on Decision and Control*, Vol. 1, pp 1177-1180, December 1977, New Orleans, LA, IEEE: New York, NY.

[14] Pao, L. Y. and Baltz, N. T., "Control of Sensor Information in Distributed Multisensor Systems," *Proceedings of the American Control Conference*, San Diego, CA, to appear, June 1999.

[15] Pao, L. Y., Kalandros, M. and Thomas, J., "Controlling Target Estimate Covariance in Centralized Multisensor Systems", *Colorado Advanced Software Institute (CASI) Final Report*, 1997.

[16] Pao, L. Y. and Kalandros, M., "The Effects of Delayed Sensor Requests on Sensor Manager Systems," *AIAA Guidance Navigation and Control Conference*, Boston, MA, August 1998.

[17] Popoli, R., "The Sensor Management Imperative," *Multitarget-Multisensor Tracking: Applications and Advances* Vol. 2, pp. 325-392, Artech House, Boston, 1992.

[18] Rao, B. S. Y., Durrant-Whyte, H. F. and Sheen, J. A., "A Fully Decentralized Multi-Sensor System For Tracking and Surveillance", *The International Journal of Robotics Research,* Vol. 12. No. 1, February 1993.

[19] Schmaedeke, W., "Information-based Sensor Management," *SPIE Proceedings*, Vol. 1955, April 1993.

[20] Schmaedeke, W. and Kastella K., "Information Based Sensor Management and IMMKF," *SPIE Proceedings*, Vol. 3373, pp. 390-401, April 1998.

[21] Singer, R. A., "Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets," *IEEE Trans. on Aerospace Electronic Systems*, Vol. AES-5, pp. 473-483, July 1970.

[22] Van Keuk, G., "Software Structure and Sampling Strategy for Automatic Target Tracking with a Phased Array Radar," *AGARD Conf. Proc. No. 252, Strategies for Automatic Track Initiation*, Monterey, CA, pp. 11-1 to 11-13, October 1978.

[23] Weisstein, E. W., *CRC Concise Encyclopedia of Mathematics*, CRC Press, New York, 1999.

Appendix A

Rate Optimization Polynomials

The desired prediction covariance, $\mathbf{P}_{dp}(t_n)$, and the prediction covariance, $\mathbf{P}(t_n|t_{n-1})$, are used to develop polynomials for computing the optimum sample period. The prediction covariance is

$$\mathbf{P}(t_n|t_{n-1}) = \mathbf{A}(T)\mathbf{P}(t_{n-1}|t_{n-1})\mathbf{A}(T)^{\mathrm{T}} + \mathbf{Q}(T) \qquad \text{(A.1)}$$

The difference between these two matrices is expressed as

$$\mathbf{M}_N(T) = \mathbf{P}_{dp}(t_n) - \mathbf{P}(t_n|t_{n-1}) \qquad \text{(A.2)}$$

where $\mathbf{M}_N(T) \in \Re^{N \times N}$. The characteristic equation for $\mathbf{M}_N(T)$ is

$$f(s) = det(s\mathbf{I} - \mathbf{M}_N(T)) = \sum_{i=0}^{N} a_i(T)s^i \qquad \text{(A.3)}$$

where each of the coefficients are functions of the sample period T. Using the coefficients of the characteristic equation we can develop metrics based upon the determinant and the trace.

The following sections in this appendix develop the polynomial coefficients of $f(s)$ for different models. We specifically look at the discretized continuous models and the direct discrete models for different length state vectors. Let each element of the desired prediction covariance be $\mathbf{P}_{dp}(t_n)_{ij} = d_{ij}$ and the update covariance be

$\mathbf{P}(t_{n-1}|t_{n-1})_{ij} = p_{ij}$.

1.1 Discretized Continuous Model

First Order:

The first order model has the form

$$\mathbf{M}_1(T) = d_{11} - a^2 p_{11} - T\sigma^2 \tag{A.4}$$

The optimal sample period for this Wiener process is

$$T = \frac{d_{11} - a^2 p_{11}}{\sigma^2} \tag{A.5}$$

This model would describe a white noise velocity model making the position a Wiener process.

Second Order:

The second order model has the form

$$\mathbf{M}_2(T) = \begin{bmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{bmatrix} - \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ T & 1 \end{bmatrix} - \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix} \sigma^2 \tag{A.6}$$

The characteristic equation for the above matrix has polynomial coefficients that are a function of the sample period.

$$f(s) = det(s\mathbf{I} - \mathbf{M}_2(T)) = a_2(T)s^2 + a_1(T)s + a_0(T) \tag{A.7}$$

$$\begin{aligned} a_2(T) &= 1 \\ a_1(T) &= b_3T^3 + b_2T^2 + b_1T + b_0 \\ a_0(T) &= c_4T^4 + c_3T^3 + c_2T^2 + c_1T + c_0 \end{aligned} \tag{A.8}$$

The coefficients of the third order polynomial $a_1(T)$ are

$$\begin{aligned} b_3 &= \frac{1}{3}\sigma^2 \\ b_2 &= p_{22} \\ b_1 &= 2p_{12}+\sigma^2 \\ b_0 &= p_{11}+p_{22}-d_{11}-d_{22} \end{aligned} \tag{A.9a}$$

The coefficients of the fourth order polynomial $a_0(T)$ are

$$\begin{aligned} c_4 &= \frac{1}{12}\sigma^4 \\ c_3 &= \frac{1}{3}p_{22}\sigma^2-\frac{1}{3}d_{22}\sigma^2 \\ c_2 &= d_{12}\sigma^2+p_{12}\sigma^2-p_{22}d_{22} \\ c_1 &= 2d_{12}p_{22}-2p_{12}d_{22}+p_{11}\sigma^2-d_{11}\sigma^2 \\ c_0 &= d_{11}d_{22}-p_{12}^2-d_{12}^2+2d_{12}p_{12}-p_{11}d_{22}-d_{11}p_{22}+p_{11}p_{22} \end{aligned} \tag{A.9b}$$

This model describes a white noise acceleration model also called a Wiener process velocity model.

Third Order:

The third order model has the form

$$\begin{aligned} \mathbf{M}_3(T) &= \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{12} & d_{22} & d_{23} \\ d_{13} & d_{23} & d_{33} \end{bmatrix} - \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{12} & p_{22} & p_{23} \\ p_{13} & p_{23} & p_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ T & 1 & 0 \\ T^2/2 & T & 1 \end{bmatrix} \\ &\quad -\sigma^2 \begin{bmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{bmatrix} \end{aligned} \tag{A.10}$$

The characteristic equation is a third order polynomial in s

$$f(s) = det(s\mathbf{I}-\mathbf{M}_3(T)) = m_3(T)s^3+m_2(T)s^2+m_1(T)s+m_0(T) \tag{A.11}$$

$$
\begin{aligned}
m_3(T) &= 1 \\
m_2(T) &= \textstyle\sum_{i=0}^{5} a_i T^i \\
m_1(T) &= \textstyle\sum_{i=0}^{8} b_i T^i \\
m_0(T) &= \textstyle\sum_{i=0}^{9} c_i T^i
\end{aligned}
\tag{A.12}
$$

The coefficients of the 5th order polynomial $m_2(T)$ are

$$a_5 = \sigma^2/20 \tag{A.13a}$$

$$a_4 = p_{33}/4 \tag{A.13b}$$

$$a_3 = p_{23} + \sigma^2/3 \tag{A.13c}$$

$$a_2 = p_{13} + p_{22} + p_{33} \tag{A.13d}$$

$$a_1 = 2p_{12} + 2p_{23} + \sigma^2 \tag{A.13e}$$

$$a_0 = p_{11} + p_{22} + p_{33} - d_{11} - d_{22} - d_{33} \tag{A.13f}$$

the coefficients of the 8th order polynomial $m_1(T)$ are

$$b_8 = \sigma^4/960 \tag{A.14a}$$

$$b_7 = p_{33}\sigma^2/120 \tag{A.14b}$$

$$b_6 = \frac{1}{45}\sigma^4 + \frac{7}{120}p_{23}\sigma^2 \tag{A.14c}$$

$$b_5 = \sigma^2\frac{1}{12}p_{13} - \sigma^2\frac{1}{20}(d_{22} + d_{33}) + \sigma^2\frac{2}{15}(p_{22} + p_{33}) \tag{A.14d}$$

$$b_4 = \frac{1}{12}\sigma^4 + \frac{2}{3}p_{23}\sigma^2 + \frac{5}{12}p_{12}\sigma^2 + \frac{1}{4}(d_{12}\sigma^2 + p_{33}(p_{22} - d_{22} - d_{33}) - p_{23}^2) \tag{A.14e}$$

$$
\begin{aligned}
b_3 = {} & p_{12}p_{33} - p_{23}d_{22} - p_{23}d_{33} + d_{12}p_{33} - p_{13}p_{23} + \\
& \frac{1}{3}\sigma^2(p_{11} + 3p_{22} + d_{13} - d_{33} + p_{33} + 2p_{13} - d_{11})
\end{aligned}
\tag{A.14f}
$$

$$b_2 = 3d_{12}p_{23} - p_{13}^2 - p_{22}d_{22} + p_{12}p_{23} - p_{13}d_{33} - p_{13}p_{22} - p_{23}^2 - p_{13}d_{22} - p_{22}d_{33} + p_{33}(d_{13} + p_{22} + p_{11} - d_{11} - d_{33}) + \sigma^2(2p_{12} + d_{23} + p_{23}) \quad \text{(A.14g)}$$

$$b_1 = \sigma^2(p_{22} + p_{11} - d_{22} - d_{11}) + 2(p_{12}p_{33} + d_{23}p_{33} + d_{12}p_{22} - p_{13}p_{23}) + 2(d_{12}p_{13} + d_{13}p_{23} - p_{12}d_{22} - p_{12}p_{13} - p_{23}d_{33} - d_{11}p_{23} - p_{12}d_{33} + p_{11}p_{23}) \quad \text{(A.14h)}$$

$$b_0 = -p_{23}^2 - d_{13}^2 + (d_{11} - p_{11})(d_{33} + d_{22} - p_{33} - p_{22}) - d_{12}^2 - p_{12}^2 - d_{22}p_{33} - p_{22}d_{33} + p_{22}p_{33} + d_{22}d_{33} - d_{23}^2 + 2(d_{13}p_{13} + d_{23}p_{23} + d_{12}p_{12}) - p_{13}^2 \quad \text{(A.14i)}$$

and the coefficients of the 9th order polynomial $m_0(T)$ are

$$c_9 = \sigma^6/8640 \quad \text{(A.15a)}$$

$$c_8 = \sigma^4(p_{33} - d_{33})/960 \quad \text{(A.15b)}$$

$$c_7 = \frac{1}{120}(\sigma^4 p_{23} - \sigma^2 p_{33}d_{33} + \sigma^4 d_{23}) \quad \text{(A.15c)}$$

$$c_6 = \frac{1}{45}\sigma^4(p_{22} - d_{22}) + \frac{1}{72}\sigma^4(p_{13} - d_{13}) + \frac{7}{120}\sigma^2(p_{33}d_{23} - p_{23}d_{33}) \quad \text{(A.15d)}$$

$$c_5 = \frac{1}{12}\sigma^2(d_{12}\sigma^2 - p_{33}d_{13} - p_{13}d_{33} + p_{12}\sigma^2) - \frac{2}{15}\sigma^2(p_{33}d_{22} + p_{22}d_{33}) + \frac{7}{20}p_{23}d_{23}\sigma^2 + \frac{1}{20}(p_{22}p_{33} + d_{22}d_{33}\sigma^2 - p_{23}^2\sigma^2 - d_{23}^2\sigma^2) \quad \text{(A.15e)}$$

$$c_4 = \frac{5}{12}\sigma^2(p_{13}d_{23} - p_{23}d_{13} + d_{12}p_{33} - p_{12}d_{33}) - \frac{2}{3}p_{23}d_{22}\sigma^2 + \frac{2}{3}p_{22}d_{23} + \frac{1}{4}\sigma^2(p_{12}p_{33} + d_{13}d_{23} - p_{13}p_{23} - d_{12}d_{33}) + \frac{1}{12}\sigma^4(p_{11} - d_{11}) + \frac{1}{4}(p_{23}^2 d_{33} - p_{22}p_{33}d_{33} - p_{33}d_{23}^2 + p_{33}d_{22}d_{33}) \quad \text{(A.15f)}$$

$$\begin{aligned}
c_3 = {} & p_{33}d_{13}d_{23}-p_{12}p_{33}d_{33}-p_{23}^2d_{23}-p_{23}d_{23}^2-d_{12}p_{33}d_{33}+p_{22}d_{23}p_{33} \\
& +\frac{5}{3}d_{12}p_{23}\sigma^2+\frac{5}{3}p_{12}d_{23}\sigma^2-p_{22}d_{22}\sigma^2+p_{23}d_{22}d_{33}+p_{13}p_{23}d_{33} \\
& +\frac{1}{3}p_{12}p_{23}\sigma^2-\frac{2}{3}p_{13}d_{22}\sigma^2-\frac{2}{3}p_{22}d_{13}\sigma^2-\frac{1}{3}p_{13}^2\sigma^2-\frac{1}{3}d_{11}p_{33}\sigma^2 \\
& +\frac{1}{3}\sigma^2(p_{11}p_{33}-p_{13}p_{22}-d_{13}d_{22}-d_{13}^2-p_{11}d_{33}+d_{11}d_{33}+d_{12}d_{23}-p_{13}d_{13})
\end{aligned} \tag{A.15g}$$

$$\begin{aligned}
c_2 = {} & p_{23}^2d_{22}+p_{23}^2d_{13}+p_{13}^2d_{33}-d_{13}^2p_{33}-p_{13}d_{23}^2 \\
& +d_{11}p_{33}d_{33}+p_{13}d_{22}d_{33}-p_{11}p_{33}d_{33}+d_{12}p_{33}d_{23}-p_{12}p_{23}d_{33}-d_{13}p_{33}d_{22} \\
& +2\sigma^2(d_{12}p_{22}-p_{12}d_{22})+p_{22}(d_{22}d_{33}-d_{22}p_{33}-d_{13}p_{33}-d_{23}^2+p_{13}d_{33}) \\
& +\sigma^2(p_{11}p_{23}-d_{11}d_{23}-d_{11}p_{23}+d_{12}d_{13}+p_{11}d_{23}-p_{12}p_{13}-p_{12}d_{13}+d_{12}p_{13}) \\
& -3d_{12}p_{23}d_{33}+3d_{23}(p_{23}d_{13}-p_{13}p_{23}+p_{12}p_{33})
\end{aligned} \tag{A.15h}$$

$$\begin{aligned}
c_1 = {} & 2(p_{22}d_{13}d_{23}-d_{11}d_{23}p_{33}-p_{12}d_{23}^2-d_{13}^2p_{23}-d_{12}p_{23}^2-p_{22}p_{13}d_{23}) \\
& +2(p_{13}d_{13}d_{23}-p_{13}^2d_{23}+p_{12}d_{23}p_{23}+p_{12}d_{22}d_{33}-p_{12}d_{22}p_{33}+p_{12}p_{13}d_{33}) \\
& +\sigma^2(p_{11}p_{22}+d_{11}d_{22}+2d_{12}p_{12}-d_{11}p_{22}-p_{11}d_{22}-p_{12}^2-d_{12}^2) \\
& +2(p_{11}d_{23}p_{33}+d_{11}p_{23}d_{33}+d_{12}p_{23}d_{23}+p_{13}p_{23}d_{22}-p_{11}p_{23}d_{33}-p_{12}d_{13}p_{33}) \\
& +2(p_{13}d_{13}p_{23}-d_{13}p_{23}d_{22}+d_{12}d_{13}p_{33}-d_{12}p_{22}d_{33}-d_{12}p_{13}d_{33}+d_{12}p_{22}p_{33})
\end{aligned} \tag{A.15i}$$

$$\begin{aligned}
c_0 = {} & (d_{23}^2+p_{23}^2)(d_{11}-p_{11})+(d_{13}^2+p_{13}^2)(d_{22}-p_{22})+(d_{12}^2+p_{12}^2)(d_{33}-p_{33}) \\
& +2p_{23}(d_{12}d_{13}-d_{11}d_{23}-d_{12}p_{13}-d_{12}p_{13}-d_{12}d_{13}-p_{12}d_{13}) \\
& +2d_{12}(p_{13}d_{23}-p_{12}d_{33}+p_{12}d_{33}+p_{12}p_{33}-d_{13}d_{23}+p_{13}d_{23}) \\
& +2(p_{12}d_{13}d_{23}-p_{12}p_{13}d_{23}+p_{12}p_{13}p_{23}+d_{13}p_{13}p_{22}+p_{11}d_{23}p_{23}-d_{13}p_{13}d_{22}) \\
& -(d_{11}-p_{11})(d_{22}-p_{22})(d_{33}-p_{33})
\end{aligned} \tag{A.15j}$$

This model describes a white noise jerk model also called a Wiener process acceleration model.

1.2 Direct Discretized Model

In these models the only difference is in how the process noise covariance is approximated.

First Order:

With the first order model we can solve for the optimal sample rate directly. The first order direct discretized model is

$$\mathbf{M}_1(T) = d_{11} - a^2 p_{11} - \sigma^2 T^2 \qquad \text{(A.16)}$$

Letting $\mathbf{M}_1(T) = 0$, the optimal sample period is $T = \left(\frac{d_{11} - a^2 p_{11}}{\sigma^2}\right)^{1/2}$, where the negative solution has been ignored. When $d_{11} > a^2 p_{11}$ then there will always exist a positive value for the sample period.

Comparing this solution of the sample period to that found using the discretized continuous model we see that the two sample periods are related by the square root.

Second Order:

The equation for this second order model is

$$\mathbf{M}_2(T) = \begin{bmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{bmatrix} - \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ T & 1 \end{bmatrix} - \sigma^2 \begin{bmatrix} T^4/4 & T^3/2 \\ T^3/2 & T^2 \end{bmatrix} \qquad \text{(A.17)}$$

The characteristic equation is a second order polynomial in the complex variable s

$$f(s) = det(s\mathbf{I} - \mathbf{M}_2(T)) = m_2(T)s^2 + m_1(T)s + m_0(T) \qquad \text{(A.18)}$$

where the polynomial coefficients in T are

$$\begin{aligned} m_2(T) &= 1 \\ m_1(T) &= a_4 T^4 + a_3 T^3 + a_2 T^2 + a_1 T + a_0 \\ m_0(T) &= b_4 T^4 + b_3 T^3 + b_2 T^2 + b_1 T + b_0 \end{aligned} \qquad \text{(A.19)}$$

The coefficients of the 4th order polynomial $m_1(T)$ are

$$a_0 = p_{22} + p_{11} - d_{22} - d_{11} \tag{A.20a}$$

$$a_1 = 2p_{12} \tag{A.20b}$$

$$a_2 = p_{22} + \sigma^2 \tag{A.20c}$$

$$a_3 = 0 \tag{A.20d}$$

$$a_4 = \frac{1}{4}\sigma^2 \tag{A.20e}$$

and the coefficients of the 4th order polynomial $m_0(T)$ are

$$b_0 = d_{11}d_{22} - p_{12}^2 - d_{12}^2 + 2d_{12}p_{12} - p_{11}d_{22} - d_{11}p_{22} + p_{11}p_{22} \tag{A.21a}$$

$$b_1 = 2d_{12}p_{22} - 2p_{12}d_{22} \tag{A.21b}$$

$$b_2 = \sigma^2(p_{11} - d_{11}) - p_{22}d_{22} \tag{A.21c}$$

$$b_3 = \sigma^2(d_{12} + p_{12}) \tag{A.21d}$$

$$b_4 = \frac{1}{4}\sigma^2(p_{22} - d_{22}) \tag{A.21e}$$

When the update covariance changes at each sample interval the coefficients of these polynomials must be recomputed.

Third Order:

The equation for this third order model is

$$\begin{aligned}\mathbf{M}_3(T) = &\begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{12} & d_{22} & d_{23} \\ d_{13} & d_{23} & d_{33} \end{bmatrix} - \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{12} & p_{22} & p_{23} \\ p_{13} & p_{23} & p_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ T & 1 & 0 \\ T^2/2 & T & 1 \end{bmatrix} \\ &-\sigma^2 \begin{bmatrix} T^6/36 & T^5/12 & T^4/6 \\ T^5/12 & T^4/4 & T^3/2 \\ T^4/6 & T^3/2 & T^2 \end{bmatrix}\end{aligned} \tag{A.22}$$

The characteristic equation is a third order polynomial in s

$$f(s) = det(s\mathbf{I} - \mathbf{M}_3(T)) = m_3(T)s^3 + m_2(T)s^2 + m_1(T)s + m_0(T) \qquad \text{(A.23)}$$

$$\begin{aligned} m_3(T) &= 1 \\ m_2(T) &= \textstyle\sum_{i=0}^{6} a_i T^i \\ m_1(T) &= \textstyle\sum_{i=0}^{8} b_i T^i \\ m_0(T) &= \textstyle\sum_{i=0}^{8} c_i T^i \end{aligned} \qquad \text{(A.24)}$$

The coefficients of 6th order polynomial $m_2(T)$ are

$$a_6 = \sigma^2/36 \qquad \text{(A.25a)}$$

$$a_5 = 0 \qquad \text{(A.25b)}$$

$$a_4 = (\sigma^2 + p_{33})/4 \qquad \text{(A.25c)}$$

$$a_3 = p_{23} \qquad \text{(A.25d)}$$

$$a_2 = \sigma^2 + p_{13} + p_{33} + p_{22} \qquad \text{(A.25e)}$$

$$a_1 = 2p_{23} + 2p_{12} \qquad \text{(A.25f)}$$

$$a_0 = p_{11} + p_{22} + p_{33} - d_{11} - d_{22} - d_{33} \qquad \text{(A.25g)}$$

the coefficients of the 8th order polynomial $m_1(T)$ are

$$b_8 = \frac{1}{144} p_{33}\sigma^2 \qquad \text{(A.26a)}$$

$$b_7 = \frac{1}{18} p_{23}\sigma^2 \qquad \text{(A.26b)}$$

$$b_6 = \sigma^2\left(\frac{1}{9}p_{33} + \frac{1}{9}p_{22} + \frac{1}{12}p_{13} - \frac{1}{36}d_{22} - \frac{1}{36}d_{33}\right) \qquad \text{(A.26c)}$$

$$b_5 = \sigma^2\left(\frac{2}{3}p_{23} + \frac{1}{3}p_{12} + \frac{1}{6}d_{12}\right) \qquad \text{(A.26d)}$$

$$b_4 = -\frac{1}{4}d_{22}p_{33} + \frac{1}{4}p_{33}\sigma^2 + \frac{1}{3}d_{13}\sigma^2 + \frac{2}{3}p_{13}\sigma^2 - \frac{1}{4}p_{33}d_{33} - \frac{1}{4}p_{23}^2$$
$$-\frac{1}{4}d_{33}\sigma^2 + \frac{1}{4}p_{22}p_{33} + p_{22}\sigma^2 + \frac{1}{4}p_{11}\sigma^2 - \frac{1}{4}d_{11}\sigma^2 \quad \text{(A.26e)}$$

$$b_3 = -p_{23}d_{22} + p_{23}\sigma^2 + d_{23}\sigma^2 + p_{12}p_{33}$$
$$-p_{23}d_{33} + d_{12}p_{33} - p_{13}p_{23} + 2p_{12}\sigma^2 \quad \text{(A.26f)}$$

$$b_2 = \sigma^2(p_{11} + p_{22} - d_{22} - d_{11}) + p_{12}p_{23} - p_{13}d_{22} + p_{33}(p_{22} - d_{33})$$
$$-p_{22}d_{22} - p_{23}^2 + d_{13}p_{33} + 3d_{12}p_{23} - p_{13}^2 + p_{11}p_{33} - p_{22}d_{33} - p_{13}d_{33} - p_{13}p_{22} \quad \text{(A.26g)}$$

$$b_1 = 2d_{23}p_{33} - 2p_{12}d_{22} - 2p_{13}p_{23} + 2p_{11}p_{23} + 2d_{13}p_{23} - 2d_{11}p_{23}$$
$$-2p_{23}d_{33} + 2d_{12}p_{22} + 2p_{12}p_{33} - 2p_{12}p_{13} - 2p_{12}d_{33} + 2d_{12}p_{13} \quad \text{(A.26h)}$$

$$b_0 = 2d_{12}p_{12} - d_{13}^2 - d_{11}p_{33} - p_{13}^2 - p_{12}^2 - p_{11}d_{33} - d_{12}^2 + d_{11}d_{33} - p_{11}d_{22}$$
$$+2d_{13}p_{13} + d_{22}d_{33} - d_{22}p_{33} - p_{22}d_{33} + p_{22}p_{33} - d_{23}^2 + 2d_{23}p_{23} - p_{23}^2 \quad \text{(A.26i)}$$
$$-d_{11}p_{22} + p_{11}p_{33} + d_{11}d_{22} + p_{11}p_{22}$$

and the coefficients of the 8th order polynomial $m_0(T)$ are

$$c_8 = -\frac{1}{144}\sigma^2 p_{33}d_{33} \quad \text{(A.27a)}$$

$$c_7 = \frac{1}{18}\sigma^2(p_{33}d_{23} - p_{23}d_{33}) \quad \text{(A.27b)}$$

$$c_6 = \sigma^2\frac{7}{18}p_{23}d_{23} + \sigma^2\frac{1}{36}(p_{22}p_{33} + d_{22}d_{33} - d_{23}^2 - p_{23}^2)$$
$$-\sigma^2\frac{1}{12}(p_{13}d_{33} + p_{33}d_{13}) - \sigma^2\frac{1}{9}(p_{22}d_{33} + p_{33}d_{22}) \quad \text{(A.27c)}$$

$$c_5 = \sigma^2\left(\frac{1}{6}p_{12}p_{33} - \frac{2}{3}p_{23}d_{22} + \frac{1}{2}p_{13}d_{23} - \frac{1}{6}d_{12}d_{33} - \frac{1}{2}p_{23}d_{13}\right)$$
$$+\sigma^2\left(\frac{1}{3}d_{12}p_{33} + \frac{2}{3}p_{22}d_{23} + \frac{1}{6}d_{13}d_{23} - \frac{1}{3}p_{12}d_{33} - \frac{1}{6}p_{13}p_{23}\right) \quad \text{(A.27d)}$$

$$\begin{aligned}c_4 = {} & \frac{1}{3}p_{12}p_{23}\sigma^2 - \frac{2}{3}p_{13}d_{22}\sigma^2 - \frac{2}{3}p_{22}d_{13}\sigma^2 - p_{22}d_{22}\sigma^2 - \frac{1}{2}p_{13}d_{13}\sigma^2 \\ & + \frac{1}{4}\sigma^2(d_{33}(d_{11} - p_{11}) - d_{13}^2 - d_{11}p_{33} - p_{13}^2) \\ & + \frac{5}{3}d_{12}p_{23}\sigma^2 + \frac{5}{3}p_{12}d_{23}\sigma^2 + \frac{1}{3}d_{12}d_{23}\sigma^2 - \frac{1}{3}d_{13}d_{22}\sigma^2 - \frac{1}{3}p_{13}p_{22}\sigma^2 \\ & + \frac{1}{4}(p_{23}^2 d_{33} - p_{33}d_{23}^2 + p_{33}d_{33}(d_{22} - p_{22}))\end{aligned} \tag{A.27e}$$

$$\begin{aligned}c_3 = {} & -p_{23}d_{23}^2 - p_{23}^2 d_{23} + p_{11}d_{23}\sigma^2 - d_{12}p_{33}d_{33} + d_{12}p_{13}\sigma^2 + p_{22}d_{23}p_{33} \\ & - p_{12}p_{33}d_{33} - 2p_{12}d_{22}\sigma^2 - d_{11}d_{23}\sigma^2 + p_{33}d_{13}d_{23} + 2d_{12}p_{22}\sigma^2 + p_{13}p_{23}d_{33} \\ & + p_{11}p_{23}\sigma^2 - p_{12}d_{13}\sigma^2 + d_{12}d_{13}\sigma^2 - p_{12}p_{13}\sigma^2 + p_{23}d_{22}d_{33} - d_{11}p_{23}\sigma^2\end{aligned} \tag{A.27f}$$

$$\begin{aligned}c_2 = {} & p_{13}^2 d_{33} - p_{13}d_{23}^2 - p_{22}d_{23}^2 - p_{12}p_{23}d_{33} + 3p_{12}d_{23}p_{33} + 3p_{23}d_{13}d_{23} \\ & - p_{11}d_{22}\sigma^2 - p_{12}^2\sigma^2 + p_{11}p_{22}\sigma^2 + d_{11}d_{22}\sigma^2 - d_{11}p_{22}\sigma^2 - d_{12}^2\sigma^2 - d_{13}p_{33}d_{22} \\ & - 3d_{12}p_{23}d_{33} + d_{12}p_{33}d_{23} + p_{23}^2 d_{22} - d_{13}^2 p_{33} + p_{23}^2 d_{13} + 2d_{12}p_{12}\sigma^2 \\ & + p_{22}d_{22}d_{33} - p_{22}d_{22}p_{33} + p_{13}d_{22}d_{33} + p_{13}p_{22}d_{33} - 3p_{13}d_{23}p_{23} - p_{22}d_{13}p_{33} \\ & - p_{11}p_{33}d_{33} + d_{11}p_{33}d_{33}\end{aligned} \tag{A.27g}$$

$$\begin{aligned}c_1 = {} & -2p_{12}d_{22}p_{33} + 2p_{13}d_{13}p_{23} - 2p_{11}p_{23}d_{33} - 2d_{12}p_{13}d_{33} - 2p_{13}^2 d_{23} \\ & + 2p_{11}d_{23}p_{33} + 2d_{12}p_{22}p_{33} + 2p_{13}d_{13}d_{23} - 2p_{22}p_{13}d_{23} + 2d_{12}d_{13}p_{33} \\ & + 2p_{22}d_{13}d_{23} + 2d_{12}p_{23}d_{23} + 2d_{11}p_{23}d_{33} + 2p_{12}d_{23}p_{23} + 2p_{12}p_{13}d_{33} \\ & - 2p_{12}d_{13}p_{33} - 2d_{11}d_{23}p_{33} - 2d_{13}^2 p_{23} - 2d_{12}p_{22}d_{33} + 2p_{13}p_{23}d_{22} \\ & + 2p_{12}d_{22}d_{33} - 2d_{12}p_{23}^2 - 2p_{12}d_{23}^2 - 2d_{13}p_{23}d_{22}\end{aligned} \tag{A.27h}$$

$$\begin{aligned}c_0 = {} & -p_{11}p_{23}^2 - p_{11}d_{23}^2 + p_{12}^2 d_{33} - d_{12}^2 p_{33} + d_{12}^2 d_{33} - p_{12}^2 p_{33} - p_{13}^2 p_{22} \\ & + p_{13}^2 d_{22} - 2d_{11}d_{23}p_{23} - 2d_{12}d_{13}d_{23} + 2d_{12}d_{13}p_{23} + 2d_{12}p_{13}d_{23} - 2d_{12}p_{13}p_{23} \\ & + 2(d_{12}p_{12}p_{33} - d_{12}p_{12}d_{33} + p_{12}d_{13}d_{23} - p_{12}d_{13}p_{23}) \\ & + 2(p_{12}p_{13}p_{23} + d_{13}p_{13}p_{22} - d_{13}p_{13}d_{22} - p_{12}p_{13}d_{23} + p_{11}d_{23}p_{23}) \\ & - d_{11}p_{22}p_{33} + d_{11}d_{23}^2 + p_{11}p_{22}p_{33} + d_{11}p_{22}d_{33} - p_{11}p_{22}d_{33} - d_{13}^2 p_{22} \\ & - p_{11}d_{22}p_{33} + p_{11}d_{22}d_{33} + d_{11}p_{23}^2 + d_{11}d_{22}p_{33} - d_{11}d_{22}d_{33} + d_{13}^2 d_{22}\end{aligned} \tag{A.27i}$$

Appendix B

Lie Groups

The definition and properties of Lie groups are found in [23]. A matrix group G, is a set of nonsingular matrices that are closed under (matrix) multiplication and inversion and always contain the identity matrix. This says that if **S** and **T** are both elements of G, then $\mathbf{S}^{-1}$, $\mathbf{T}^{-1}$, **I**, **ST** and **TS** are also elements of G. Unitary matrices are one example of a Lie group. Unitary matrices are described by the set $U(N) = \{\mathbf{U}^*\mathbf{U} = \mathbf{I}, \mathbf{U} \in C^{N \times N}\}$ where C denotes the domain of complex numbers. Defining $\mathbf{U}, \mathbf{V} \in U(N)$, the Lie group properties are verified as follows

$$(\mathbf{I})^*(\mathbf{I}) = \mathbf{I} \tag{B.1}$$

$$(\mathbf{UV})^*(\mathbf{UV}) = \mathbf{V}^*\mathbf{U}^*\mathbf{UV} = \mathbf{V}^*(\mathbf{I})\mathbf{V} = \mathbf{I} \tag{B.2}$$

$$(\mathbf{U}^{-1})^*(\mathbf{U}^{-1}) = \mathbf{UU}^* = \mathbf{I} \tag{B.3}$$

2.1 Symplectic Matrices

Symplectic matrices form a Lie group. Symplectic matrices are described by the set $S(2N) = \{\mathbf{S}^T\mathbf{JS} = \mathbf{J}, \mathbf{S} \in \Re^{2N \times 2N}\}$ where $\mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix}$ $(\mathbf{J}^{-1} = -\mathbf{J} = \mathbf{J}^T)$.

Defining $\mathbf{S}, \mathbf{T} \in S(2N)$, the three properties of a Lie group are verified as follows

$$(\mathbf{I})^T\mathbf{J}(\mathbf{I}) = \mathbf{J} \tag{B.4}$$

$$(\mathbf{ST})^T\mathbf{J}(\mathbf{ST}) = \mathbf{T}^T(\mathbf{S}^T\mathbf{JS})\mathbf{T} = \mathbf{T}^T\mathbf{JT} = \mathbf{J} \tag{B.5}$$

$$(\mathbf{S}^{-1})^T\mathbf{J}(\mathbf{S}^{-1}) = (\mathbf{S}^{-1})^T\mathbf{S}^T\mathbf{JS}(\mathbf{S}^{-1}) = \mathbf{J} \tag{B.6}$$

Now let $\mathbf{M} \in S(2N)$ and $\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$ where $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D} \in \Re^{N \times N}$. Substituting this matrix into the definition we get

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{\mathrm{T}}\mathbf{C} - (\mathbf{A}^{\mathrm{T}}\mathbf{C})^{\mathrm{T}} & \mathbf{A}^{\mathrm{T}}\mathbf{D} - \mathbf{C}^{\mathrm{T}}\mathbf{B} \\ -(\mathbf{A}^{\mathrm{T}}\mathbf{D} - \mathbf{C}^{\mathrm{T}}\mathbf{B})^{\mathrm{T}} & \mathbf{B}^{\mathrm{T}}\mathbf{D} - (\mathbf{B}^{\mathrm{T}}\mathbf{D})^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} \qquad \text{(B.7)}$$

Equating subblocks in (B.7), we have three conditions that must be satisfied

$$\begin{aligned} \mathbf{A}^{\mathrm{T}}\mathbf{D} - \mathbf{C}^{\mathrm{T}}\mathbf{B} &= \mathbf{I} \\ \mathbf{A}^{\mathrm{T}}\mathbf{C} - (\mathbf{A}^{\mathrm{T}}\mathbf{C})^{\mathrm{T}} &= \mathbf{0} \\ \mathbf{B}^{\mathrm{T}}\mathbf{D} - (\mathbf{B}^{\mathrm{T}}\mathbf{D})^{\mathrm{T}} &= \mathbf{0} \end{aligned} \qquad \text{(B.8)}$$

Symplectic matrices are always invertible because

$$det(\mathbf{M}) = det(\mathbf{M})^2 det(\mathbf{J}) = det(\mathbf{M}^{\mathrm{T}}\mathbf{J}\mathbf{M}) = det(\mathbf{J}) = 1 \qquad \text{(B.9)}$$

where $det(\mathbf{M}) = \lambda_1\lambda_1^{-1}(\lambda_2\lambda_2^{-1})\ldots(\lambda_N\lambda_N^{-1}) = 1$, because the eigenvalues come in reciprocal pairs. We can also manipulate the definition as follows

$$\begin{aligned} \mathbf{M}^{\mathrm{T}}\mathbf{J}\mathbf{M} &= \mathbf{J} \\ \mathbf{M}\mathbf{J}^{-1}(\mathbf{M}^{\mathrm{T}}\mathbf{J}\mathbf{M} &= \mathbf{J})\mathbf{M}^{-1}\mathbf{J}^{-1} \\ \mathbf{M}\mathbf{J}^{-1}\mathbf{M}^{\mathrm{T}} &= \mathbf{J}^{-1} \\ \mathbf{M}\mathbf{J}\mathbf{M}^{\mathrm{T}} &= \mathbf{J} \end{aligned} \qquad \text{(B.10)}$$

Using the last equation in (B.10) and doing the same computation as in (B.7) we have

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \mathbf{A}\mathbf{B}^{\mathrm{T}} - (\mathbf{A}\mathbf{B}^{\mathrm{T}})^{\mathrm{T}} & \mathbf{A}\mathbf{D}^{\mathrm{T}} - \mathbf{B}\mathbf{C}^{\mathrm{T}} \\ -(\mathbf{A}\mathbf{D}^{\mathrm{T}} - \mathbf{B}\mathbf{C}^{\mathrm{T}})^{\mathrm{T}} & \mathbf{C}\mathbf{D}^{\mathrm{T}} - (\mathbf{C}\mathbf{D}^{\mathrm{T}})^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} \qquad \text{(B.11)}$$

Equating subblocks in (B.11), we have three more conditions that must also be satisfied

$$\begin{aligned} \mathbf{A}\mathbf{D}^{\mathrm{T}} - \mathbf{B}\mathbf{C}^{\mathrm{T}} &= \mathbf{I} \\ \mathbf{A}\mathbf{B}^{\mathrm{T}} - (\mathbf{A}\mathbf{B}^{\mathrm{T}})^{\mathrm{T}} &= \mathbf{0} \\ \mathbf{C}\mathbf{D}^{\mathrm{T}} - (\mathbf{C}\mathbf{D}^{\mathrm{T}})^{\mathrm{T}} &= \mathbf{0} \end{aligned} \tag{B.12}$$

From the connection imposed by (B.10) it is apparent that the equations in (B.8) imply the equations in (B.12).

It is easy to show that the eigenvalues of a symplectic matrix come in reciprocal pairs: We can write $\mathbf{M} = \mathbf{J}^{-1}\mathbf{M}^{-\mathrm{T}}\mathbf{J}$ and using this in the eigenvalue formula we have

$$\begin{aligned} \mathbf{M}\boldsymbol{v} &= \lambda\boldsymbol{v} \\ \mathbf{J}^{-1}\mathbf{M}^{-\mathrm{T}}\mathbf{J}\boldsymbol{v} &= \lambda\boldsymbol{v} \\ \mathbf{M}^{-\mathrm{T}}(\mathbf{J}\boldsymbol{v}) &= \lambda(\mathbf{J}\boldsymbol{v}) \\ \mathbf{M}^{\mathrm{T}}(\mathbf{J}\boldsymbol{v}) &= \lambda^{-1}(\mathbf{J}\boldsymbol{v}) \\ \mathbf{M}^{\mathrm{T}}\boldsymbol{u} &= \lambda^{-1}\boldsymbol{u} \end{aligned} \tag{B.13}$$

where $\boldsymbol{u} = \mathbf{J}\boldsymbol{v}$. This means that λ^{-1} is an eigenvalue of $\mathbf{M}^{\mathrm{T}}$ with eigenvector $\boldsymbol{u}$ and since $\lambda(\mathbf{M}) = \lambda(\mathbf{M}^{\mathrm{T}})$ then λ^{-1} must also be an eigenvalue of $\mathbf{M}$.

Using the definition of a symplectic matrix we can express the inverse as $\mathbf{M}^{-1} = \mathbf{J}^{-1}\mathbf{M}^{\mathrm{T}}\mathbf{J}$. Then expressing $\mathbf{M}$ in terms of subblocks we have

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{D}^{\mathrm{T}} & -\mathbf{B}^{\mathrm{T}} \\ -\mathbf{C}^{\mathrm{T}} & \mathbf{A}^{\mathrm{T}} \end{bmatrix} \tag{B.14}$$

Because products of symplectic matrices are also symplectic and inverses of symplectic matrices are also symplectic, a natural question to ask is whether there exists an eigendecomposition of a symplectic matrix **M**, i.e. $\mathbf{M} = \mathbf{T}\mathbf{D}\mathbf{T}^{-1}$, such that $\mathbf{D}$ and $\mathbf{T}$ are also symplectic. If we can find such a **D** and **T**, this would simplify the computation of $\mathbf{T}^{-1}$ because we can use the representation given in (B.14).

2.1.1Three subgroups of the symplectic group

The following are each subgroups of the symplectic group.

$$S_1(2N) = \left\{ \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{R} & \mathbf{I} \end{bmatrix}, \mathbf{R} = \mathbf{R}^{\mathrm{T}} \in \Re^{N \times N} \right\} \tag{B.15a}$$

$$S_2(2N) = \left\{ \begin{bmatrix} \mathbf{I} & \mathbf{Q} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \mathbf{Q} = \mathbf{Q}^{\mathrm{T}} \in \Re^{N \times N} \right\} \tag{B.15b}$$

$$S_3(2N) = \left\{ \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{\mathrm{T}} \end{bmatrix}, det(\mathbf{A}) \neq 0 \right\} \tag{B.15c}$$

This can be written as $S_1(2N) \subset S(2N)$, $S_2(2N) \subset S(2N)$, and $S_3(2N) \subset S(2N)$. Any products of these subgroups is also symplectic. One solution of the DMRE changes the $N \times N$ nonlinear matrix difference equation into a $2N \times 2N$ linear matrix difference equation with state transition matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{C}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{C} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{Q} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{B.16}$$

where $\mathbf{M}$ is symplectic.

Appendix C

List of Acronyms

CCD	Charge Coupled Device
CDMA	Code Division Multiple Access
DMRE	Discrete Matrix Riccati Equation
DPRE	Discrete Periodic Riccati Equation
DSP	Digital Signal Processor
ESA	Electronically Scanned Array (Electronically Steered Antenna, Phased Array)
FDMA	Frequency Division Multiple Access
GPS	Global Positioning System
RADAR	RAdio Detection And Ranging
RCS	RADAR Cross Section
SONAR	SOund Navigation And Ranging
SVD	Singular Value Decomposition
TDMA	Time Division Multiple Access