State Estimation for Distributed and Hybrid Systems

Alriksson, Peter

2008

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Alriksson, P. (2008). *State Estimation for Distributed and Hybrid Systems*. [Doctoral Thesis (compilation), Department of Automatic Control]. Department of Automatic Control, Lund Institute of Technology, Lund University.

Total number of authors:
1

State Estimation for Distributed and Hybrid Systems

State Estimation for Distributed and Hybrid Systems

Peter Alriksson

Department of Automatic Control
Lund University
Lund, 2008

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

ISSN 0280–5316
ISRN LUTFD2/TFRT--1084--SE

Abstract

This thesis deals with two aspects of recursive state estimation: distributed estimation and estimation for hybrid systems.

In the first part, an approximate distributed Kalman filter is developed. Nodes update their state estimates by linearly combining local measurements and estimates from their neighbors. This scheme allows nodes to save energy, thus prolonging their lifetime, compared to centralized information processing. The algorithm is evaluated experimentally as part of an ultrasound based positioning system.

The first part also contains an example of a sensor-actuator network, where a mobile robot navigates using both local sensors and information from a sensor network. This system was implemented using a component-based framework.

The second part develops, a recursive joint maximum a posteriori state estimation scheme for Markov jump linear systems. The estimation problem is reformulated as dynamic programming and then approximated using so called relaxed dynamic programming. This allows the otherwise exponential complexity to be kept at manageable levels.

Approximate dynamic programming is also used to develop a sensor scheduling algorithm for linear systems. The algorithm produces an off-line schedule that when used together with a Kalman filter minimizes the estimation error covariance.

Acknowledgments

First of all I would like to thank my supervisor Anders Rantzer. Anders has always given me the freedom to work on problems of my own choice. His intuition and positive spirit have served as big sources of inspiration. I remember numerous occasions when I got stuck on a problem, but after a short visit to Anders office, I was back on track again.

Having Anders as supervisor allowed me to go to California Institute of Technology during my second year at the department. Therefore I would also like to thank Professor Richard Murray at the Department of Control and Dynamical Systems and the people in his group for introducing me to some of the problems that I later ended up studying.

I would also like to thank Karl-Erik Årzén with whom I have traveled across Europe with a small robot, annoying security personnel at at least five different airports. Working with Karl-Erik has taught me how to make things work in practice and to focus on what is important.

Much of the more experimental work in this thesis has been possible only thanks to our research engineer Rolf Brown, who has designed most of the hardware. Rolf has always been very helpful, even though I think he doesn't always want to admit it.

Many thanks goes to the department backbone: the secretaries Eva, Agneta and Britt-Marie and the computer engineers Anders and Leif. Without you my time here would not have gone by so smoothly.

I would also like to mention all my great friends among the fellow PhD students. Thanks for all the great lunches, coffee breaks and conference trips. Special thanks goes to Erik Johannesson and Toivo Henningsson for all your valuable input to this thesis and to Martin Ohlin for all our Linux and C discussions. I would also like to mention Pontus Nordfeldt, with whom I shared room when I first started.

The work in this thesis has been made possible by financial support from the Swedish Research Council (VR) and the European Union through the projects "IST-Computation and Control" and "IST-RUNES".

My final thanks goes to my family and friends for reminding me about the world outside the M-building.

Contents

Preface

Contributions of the Thesis

The thesis consists of two introductory chapters and six papers. This section describes the contents of the introductory chapters and the contribution of each paper.

Chapter 1 – Recursive State Estimation

In this thesis a number of different methods based on probabilistic state estimation are both applied to specific problems and extended in various directions. This chapter aims at presenting these methods in a unified way. In particular, the relation between joint maximum a posteriori estimation, which is the basis of Paper VI, and other estimation techniques is discussed in more detail.

Chapter 2 – Sensor and Sensor-Actuator Networks

The first part of this chapter gives a brief overview of different data aggregation methods aimed at sensor networks. In particular, distributed state estimation, which is the subject of papers I, II and III, is discussed in more detail.

In the second part, a few reflections on software related issues in sensor networks are given. Specifically the importance of good simulation tools and resource management are emphasized.

Paper I

Alriksson, P. and A. Rantzer (2008): “Model based information fusion in sensor networks.” In *Proceedings of the 17th IFAC World Congress*. Seoul, South Korea.

Presents a model based information fusion algorithm for sensor networks and evaluates its performance through numerical simulations.

Contributions. The algorithm extends the common Kalman filter with one step where nodes exchange estimates of the aggregated quantity with their neighbors. Under stationary conditions, an iterative parameter selection procedure that aims at minimizing the stationary error covariance of the estimated quantity is developed.

The performance of the algorithm is evaluated through numerical simulations. These simulations indicate that the algorithm performs almost as good as a non-aggregating algorithm which thus uses more bandwidth. This paper improves on the results presented in:

Alriksson, P. and A. Rantzer (2006): “Distributed Kalman filtering using weighted averaging.” In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*. Kyoto, Japan.

Paper II

Alriksson, P. and A. Rantzer (2008): “Distributed Kalman filtering: Theory and experiments.” *Submitted to IEEE Transactions on Control Systems Technology*.

Extends the results of Paper I to compensate for effects of packet loss and develops a lightweight synchronization protocol.

Contributions. The information fusion algorithm presented in Paper I is extended to account for packet loss. It is also shown how Markov chain models of packet loss can be used to analyse performance.

A lightweight synchronization protocol that only makes use of information already transmitted by the information fusion algorithm is presented. This protocol is analysed using formal verification tools based on theory for timed automata.

Timing simulations of the protocol are also compared to experimental data. From both the simulated and experimental data the conclusion is drawn that timing related issues can contribute considerably to packet loss.

Paper III

Alriksson, P. and A. Rantzer (2007): "Experimental evaluation of a distributed Kalman filter algorithm." In *Proceedings of the 46th IEEE Conference on Decision and Control*. New Orleans, LA.

Presents an experimental evaluation of a distributed Kalman filter algorithm in terms of an ultrasound based localization system.

Contributions. In this paper an ultrasound based localization system using seven sensor nodes is developed. Distance measurements are combined using trilateration and then fused using the algorithm presented in:

Alriksson, P. and A. Rantzer (2006): "Distributed Kalman filtering using weighted averaging." In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*. Kyoto, Japan.

The distributed algorithm performs almost as good as a centralized solution. Two different strategies for handling packet loss are evaluated. One is proven far superior to the other.

Paper IV

Alriksson, P. and K.-E. Årzén (2008): "A component-based approach to ultrasonic self localization in sensor networks." Technical Report ISRN LUTFD2/TFRT--7619--SE. Department of Automatic Control, Lund University, Sweden.

A component based software architecture is used to develop a distributed ultrasound based localization system for mobile robots.

Contributions. In this report a component based middleware is used to develop an ultrasound based self localization system for mobile robots. The mobile robot navigates autonomously using both localization information provided by the network and local sensors. The localization problem is formulated as a state estimation problem which is then approximated using an extended Kalman filter. This system was part of a bigger software platform described in:

Alriksson, P., J. Nordh, K.-E. Årzén, A. Bicchi, A. Danesi, R. Sciadi, and L. Pallottino (2007): "A component-based approach to localization and collision avoidance for mobile multi-agent systems." In *Proceedings of the European Control Conference*. Kos, Greece.

The localization system was also simulated in TrueTime:

Årzén, K.-E., M. Ohlin, A. Cervin, P. Alriksson, and D. Henriksson (2007): "Holistic simulation of mobile robot and sensor network applications using TrueTime." In *Proceedings of the European Control Conference*. Kos, Greece.

Paper V

Alriksson, P. and A. Rantzer (2005): "Sub-optimal sensor scheduling with error bounds." In *Proceedings of the 16th IFAC World Congress*. Prague, Czech Republic.

Presents a sensor scheduling strategy based on approximate dynamic programming.

Contributions. This paper presents an algorithm based on approximate dynamic programming for computing a sequence of measurements that will yield a minimum estimation error covariance. The algorithm is demonstrated on a sixth order model of a fixed mounted helicopter. In this example the optimal sequence is periodic which allows it to be easily implemented.

Paper VI

Alriksson, P. and A. Rantzer (2008): "State estimation for Markov jump linear systems using approximate dynamic programming." *Submitted to IEEE Transactions on Automatic Control.*

Presents a general strategy based on approximate dynamic programming for constructing recursive state estimators. This general procedure is then applied to a model class referred to as Markov jump linear systems.

Contributions. This paper revisits the formulation of recursive joint maximum a posteriori state estimation as dynamic programming. This, in general very computationally intensive formulation, is approximated using methods from relaxed dynamic programming. The proposed estimation scheme is then applied to Markov jump linear models. In the case of a switching system, that is when the discrete mode changes arbitrarily, we prove that the proposed estimation scheme yields a stable estimator for a noise free system. Preliminary results where published in:

Alriksson, P. and A. Rantzer (2006): "Observer synthesis for switched discrete-time linear systems using relaxed dynamic programming." In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*. Kyoto, Japan.

Other Publications

Cervin, A. and P. Alriksson (2006): “Optimal on-line scheduling of multiple control tasks: A case study.” In *Proceedings of the 18th Euromicro Conference on Real-Time Systems*. Dresden, Germany.

This paper has not been included in the thesis because its main focus is not estimation. However, also in this paper relaxed dynamic programming is the key tool.

1

Recursive State Estimation

By “state estimation” we refer to the task of finding out “what a system is doing” based on what we observe. The word recursive in this context means that we wish to process observations as they are made and not have to redo all computations for every new observation.

The problem of recursive state estimation can be approached from a number of different directions. Throughout this thesis, we will assume that the system is subject to stochastic disturbances and thus the state of the system is also a stochastic variable. Therefore our treatment will be based on probability theory.

Recursive state estimation problems can often be formulated in a very compact way. However, solving the problem is often very difficult. Therefore much of the literature is devoted to finding approximations, some tailored at specific applications and others more general.

In this thesis a number of different methods from the field of state estimation are both applied to specific problems and extended in various directions. This chapter aims at briefly presenting these methods in a unified way. In particular, the relation between joint maximum a posteriori estimation, which is the basis of Paper VI, and other estimation techniques are discussed in more detail.

1.1 Problem Formulation

The most general model considered here is a discrete time hidden Markov model written in terms of three probability distributions:

$$\begin{cases} p(x_{k+1}|x_k) \\ p(x_0) \\ p(y_k|x_k) \end{cases} \tag{1.1}$$

Here $x_k \in \mathbf{R}^n$ denotes the state we want to estimate and $y_k \in \mathbf{R}^p$ are the observations made. The possible existence of known inputs has been suppressed for brevity. The first distribution $p(x_{k+1}|x_k)$ describes how the system evolves with time. The second distribution models how an observation y_k is related to the state of the system and the third represents prior knowledge about the initial state.

Example 1.1—Linear System with Additive Noise
Consider a linear time invariant system with additive independent noise distributed as $p_w(w_k)$ and $p_v(v_k)$:

$$\begin{aligned} x_{k+1} &= Ax_k + w_k \\ y_k &= Cx_k + v_k \end{aligned} \tag{1.2}$$

In this setup, the distributions (1.1) become

$$\begin{aligned} p(x_{k+1}|x_k) &= p_w(x_{k+1} - Ax_k) \\ p(y_k|x_k) &= p_v(y_k - Cx_k) \end{aligned} \tag{1.3}$$

If $p_w(w_k)$ is Gaussian with zero mean and covariance R_w, $p(x_{k+1}|x_k)$ also becomes Gaussian with the same covariance, but with mean Ax_k. The same holds for the measurement distribution $p(y_k|x_k)$. □

Filtering, Prediction and Smoothing. Recall that our objective is to determine what the system is doing, that is to estimate the state of the system using a sequence of measurements. More specifically we wish to estimate the state at time k given measurements $Y_l = \{y_0, \ldots, y_l\}$ up to and including time l. Depending on the relation between k and l this problem can be divided into three cases:

Prediction	$k > l$
Filtering	$k = l$
Smoothing	$k < l$

For the sake of simplicity, only the filtering problem will be considered from now on.

Bayesian- and Point Estimates. One possible solution to the recursive state estimation problem would be to construct an algorithm that only produces an estimate $\hat{x}_k$ of x_k using the sequence of measurements Y_k. This type of estimate is referred to as a point estimate of x_k.

An alternative, and more general, approach would be to construct an algorithm that not only computes a point estimate of x_k, but the full conditional probability distribution $p(x_k|Y_k)$ of x_k given all measurements Y_k. Different point estimates can then be computed from this distribution.

Next, we will present a general algorithm for recursively computing the probability density $p(x_k|Y_k)$. But first, a few tools from probability theory will be introduced.

Probability Theory

First define the conditional distribution $p(x|y)$ in terms of the joint distribution $p(x,y)$ and the marginal distribution $p(y)$ as

$$p(x,y) = p(x|y)p(y) \tag{1.4}$$

The marginal distribution $p(y)$ can be computed from the joint distribution through integration over x as

$$p(y) = \int p(x,y)dx \tag{1.5}$$

Now using (1.4) we can derive a central tool in stochastic state estimation, namely Bayes' theorem, which states that

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{1.6}$$

When writing the system model in terms of the conditional distribution $p(x_{k+1}|x_k)$ we have implicitly assumed that if x_k is known, knowledge about $x_{k-1},\dots,x_0$ does not change the distribution, that is

$$p(x_{k+1}|x_k,\dots,x_0) = p(x_{k+1}|x_k) \tag{1.7}$$

This type of model is known as a Markov model.

1.2 Conceptual Solution

In this section, recursive equations for the conditional distribution $p(x_k|Y_k)$ will be derived. The presentation is heavily inspired by the one found in [Schön, 2006]. First using Bayes' theorem (1.6) write

$$p(x_k|Y_k) = \frac{p(y_k|x_k,Y_{k-1})p(x_k|Y_{k-1})}{p(y_k|Y_{k-1})} \tag{1.8}$$

Next using the Markov property write

$$p(x_k|Y_k) = \frac{p(y_k|x_k)p(x_k|Y_{k-1})}{p(y_k|Y_{k-1})} \tag{1.9}$$

We now need to find an expression for the probability distribution $p(x_k|Y_{k-1})$ in terms of $p(x_{k-1}|Y_{k-1})$. First note that

$$\begin{aligned} p(x_k, x_{k-1}|Y_{k-1}) &= p(x_k|x_{k-1}, Y_{k-1})p(x_{k-1}|Y_{k-1}) \\ &= p(x_k|x_{k-1})p(x_{k-1}|Y_{k-1}) \end{aligned} \tag{1.10}$$

Now integrate with respect to x_{k-1}

$$p(x_k|Y_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|Y_{k-1})dx_{k-1} \tag{1.11}$$

The distribution $p(y_k|Y_{k-1})$ is normalization constant, independent of the state, that in general can be neglected. It can, however, be computed as

$$p(y_k|Y_{k-1}) = \int p(y_k|x_k)p(x_k|Y_{k-1})dx_k \tag{1.12}$$

The measurement update equation (1.9) and time update equation (1.11) (possibly together with (1.12)) constitute a recursive algorithm for general state estimation. The solution presented above is, however, somewhat illusive, as explicit expressions for the integrals are only available in very special cases.

The conceptual solution described above recursively computes both the filtering distribution $p(x_k|Y_k)$ and the one-step ahead prediction distribution $p(x_k|Y_{k-1})$. These are only special cases of the general problem of computing $p(x_k|Y_l)$. Distributions for both the smoothing and prediction problems can be derived using the machinery described above, see for example [Schön, 2006].

The Kalman Filter

If all distributions (1.1) are Gaussian, $p(x_k|Y_k)$ will also be Gaussian. Since a Gaussian distribution can be fully described by its mean and covariance, it is sufficient to derive recursive equations for these. The Kalman filter equations have been derived in numerous papers and books as a special case of the conceptual solution above, see for example [Ho and Lee, 1964] or [Schön, 2006].

It is, however, worth noticing, that in Kalman's original paper [Kalman, 1960] the filter was derived using orthogonal projection by minimizing the expected loss

$$\mathbf{E}\, L(x_k - \hat{x}_k)$$

for some reasonable function $L(\cdot)$. Kalman showed that the optimal estimator is an orthogonal projection on the space spanned by the observations in any of the two cases:

1. If the distributions (1.1) are Gaussian and $L(\cdot)$ belongs to a large class of functions described in for example [Jazwinski, 1970].
2. If the estimator is restricted to be linear and $L(\tilde{x}) = \tilde{x}^T\tilde{x}$.

The orthogonal projection is a function of the mean and covariance of a number of quantities. In the first case, the familiar Kalman filter equations can be used to recursively compute these.

Because the second case does not assume that the distributions (1.1) are Gaussian, it turns out to be useful when trying to find approximations of the conceptual solution.

Approximations of the Conceptual Solution

As mentioned above, deriving an analytical expression for the integral (1.11) is only possible in very special cases. In the general case one must almost always resort to approximations. To be able to discuss approximations of the conceptual solution, we first define explicit expressions for the system dynamics and measurement relationship as

$$\begin{aligned} x_{k+1} &= f(x_k, k) + w_k \\ y_k &= h(x_k, k) + v_k \end{aligned} \tag{1.13}$$

Here $w(k) \in \mathbf{R}^n$ and $v(k) \in \mathbf{R}^p$ are independent Gaussian zero mean variables with covariance R_w and R_v respectively. Next we will present three common approximation methods.

The Extended Kalman Filter. By far, the most common approximation technique is the Extended Kalman Filter [Jazwinski, 1970]. The extended Kalman filter is based on the first case under which the Kalman filter is optimal. In the extended Kalman filter the system (1.13) is approximated by a linear system through linearization around the latest estimate. Given a linear system with additive Gaussian noise, the distributions (1.1) become Gaussian and the Kalman filter can thus be applied. The problem with this approach is that, instead of approximating $p(x_k|Y_k)$ in a good way, the extended Kalman filter approximates the model. This can lead to bad performance or even divergence. However, the extended Kalman filter has proven to work well in many applications, one of which is presented in Paper IV.

The Unscented Kalman Filter. The unscented Kalman filter [Julier and Uhlmann, 1997] is instead based on the second case under which the Kalman filter is optimal, namely the best linear estimator in terms of mean square error. It is important to understand that the Kalman filter only uses information about the mean and covariance of various quantities, not their full probability distributions. If these means and covariances can be approximated well enough, the Kalman filter equations can be used and performance should be close to what a linear estimator can achieve.

The unscented Kalman filter uses a deterministic sampling approach where the mean and covariance is represented using a minimal set of carefully chosen sample points. When propagated through the true nonlinear system, these points capture the mean and covariance accurately to a 3rd order Taylor series expansion of the nonlinearity. See [Julier and Uhlmann, 2004] for a good introduction.

Particle Filters. In both the extended and unscented Kalman filters $p(x_k|Y_k)$ is approximated by its mean and covariance. If a more general distribution is needed, the use of Sequential Monte Carlo Methods such as the so called particle filter [Doucet *et al.*, 2001] is a common approach. The name particle filter comes from the fact that $p(x_k|Y_k)$ is approximated as a weighted sum of particles

$$p(x_k|Y_k) \approx \sum_{i=1}^{M} \tilde{q}_k^{(i)} \delta\left(x_k - x_{k|k}^{(i)}\right) \tag{1.14}$$

where $\delta(\cdot)$ is the Dirac delta function and $\tilde{q}_k^{(i)}$ denotes the weight associated with particle $x_{k|k}^{(i)}$. The problem with this approach is that if the size of the state space is very large, we potentially need an enormous amount of particles to get a good approximation. If, however, there is a linear substructure present in the dynamical model, the marginalized particle filter [Schön, 2006] can be used. In the marginalized particle filter, the state vector is partitioned into one part that is estimated using a Kalman filter and one part that is estimated using a particle filter. This allows for fewer particles to be used.

1.3 Point Estimates

So far we have only, possibly with the exception of the Kalman filter, discussed methods for approximating $p(x_k|Y_k)$. Now we will present a number of point estimates, some of which can be computed using these approximations. The two most obvious candidates are:

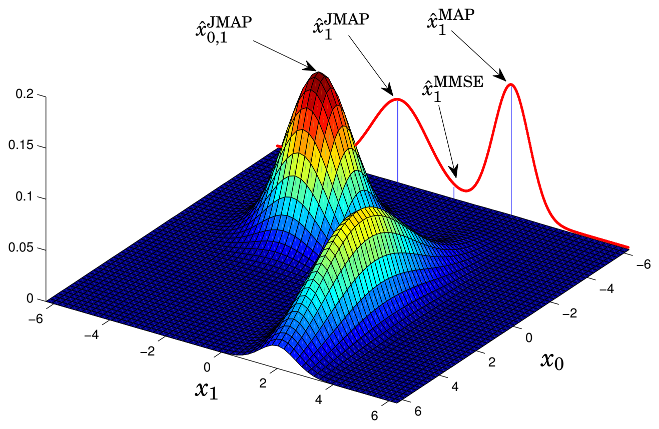


Figure 1.1 Two-dimensional distribution $p(x_0, x_1)$ together with its marginalization $p(x_1)$, with all three point estimates indicated. Note that the last element $\hat{x}_k^{\text{JMAP}}$ of the sequence $\hat{x}_{0,k}^{\text{JMAP}}$ in general does not coincide with $\hat{x}_k^{\text{MAP}}$.

1. Conditional mean

$$\hat{x}_k^{\text{MMSE}} = \int x_k p(x_k|Y_k) dx_k \tag{1.15}$$

2. Maximum a posteriori

$$\hat{x}_k^{\text{MAP}} = \arg\max_{x_k} p(x_k|Y_k) \tag{1.16}$$

The name MMSE is due to the fact that it can be shown [Jazwinski, 1970] that (1.15) *minimizes* the *mean square error* for all distributions $p(x_k|Y_k)$. Next we will introduce a point estimate that can not be computed from $p(x_k|Y_k)$, but will be the basis for the discussion in the next section.

3. Joint maximum a posteriori

$$\hat{x}_{0,k}^{\text{JMAP}} = \arg\max_{x_0,\ldots,x_k} p(x_0, \ldots, x_k|Y_k) \tag{1.17}$$

To compute the JMAP estimate, it first seems like a maximization problem that grows with time needs to be solved. Next we will, however, show that this problem can be solved recursively using dynamic programing.

In Figure 1.1 all three point estimates are shown for a two-dimensional distribution $p(x_0, x_1)$ together with its marginalization $p(x_1)$. Note that the last element $\hat{x}_1^{\text{JMAP}}$ of the sequence $\hat{x}_{0,1}^{\text{JMAP}}$ does not in general coincide with $\hat{x}_1^{\text{MAP}}$. We can thus draw the conclusion that, in general, all three point estimates presented above may differ. However, in the case of a Gaussian distribution, like in a Kalman filter, all three estimates coincide.

In a practical application, choosing a proper point estimate can be far from trivial. The MMSE estimate might be located in an area with low probability, thus making it a poor estimate, as in the case shown in Figure 1.1. When using a particle filter, computing the MAP estimate might prove difficult due to the discretization introduced by the delta function approximation.

1.4 Recursive JMAP State Estimation

The recursive solution to the JMAP problem was developed in the "mid-sixties" in a number of papers: In [Cox, 1964] a solution was presented under the assumption of Gaussian noise. These ideas were then generalized to arbitrary noise distributions in [Larson and Peschon, 1966]. The presentation here follows the latter. Recall that the problem we wish to solve is

$$\hat{X}_k = \arg\max_{X_k} p(X_k|Y_k) \tag{1.18}$$

where the superscript JMAP has been dropped and X_k denotes the sequence or trajectory $\{x_0, \ldots, x_k\}$. To develop a recursive solution, first introduce the function

$$I_k(x_k) = \max_{X_{k-1}} p(X_k|Y_k) \tag{1.19}$$

This function can be interpreted as the probability of the most probable trajectory terminating in x_k. To get a recursive algorithm we next express $I_k(x_k)$ in terms of $I_{k-1}(x_{k-1})$. Using Bayes' theorem and the Markov property, the following recursive relationship can be established:

$$I_k(x_k) = \max_{x_{k-1}} \frac{p(y_k|x_k)p(x_k|x_{k-1})}{p(y_k|Y_{k-1})} I_{k-1}(x_{k-1}) \tag{1.20}$$

For a detailed derivation, see [Larson and Peschon, 1966] or Paper VI where the derivation is done for a Markov jump linear system. The denominator of (1.20) is a normalization constant independent of the state, thus the recursive equation

$$\overline{I}_k(x_k) = \max_{x_{k-1}} p(y_k|x_k)p(x_k|x_{k-1})\overline{I}_{k-1}(x_{k-1}) \tag{1.21}$$

can be used instead. The iteration is initiated with

$$\overline{I}_0(x_0) = p(y_0|x_0)p(x_0) \tag{1.22}$$

We have now transformed the task of maximizing $p(X_k|Y_k)$ with respect to X_k into a recursive problem. Once again, note that maximizing $I(x_k|Y_k)$ and $p(x_k|Y_k)$ will in general not give the same estimate, as illustrated in Figure 1.1. Solving (1.21) for general distributions can of course be as hard as computing the integral (1.11) associated with the conceptual solution. The difference being that we now need to solve an optimization problem instead of an integration problem.

Dynamic Programming Formulation. The recursive equation (1.21) can be transformed into a dynamic programming problem by introducing the value function

$$V_k(x_k) = -\log \overline{I}_k(x_k) \tag{1.23}$$

The maximization problem (1.21) can now be written as a minimization problem on the form

$$V_k(x_k) = \min_{x_{k-1}} \{V_{k-1}(x_{k-1}) + L_k(x_k, x_{k-1})\} \tag{1.24}$$

$$L_k(x_k, x_{k-1}) = -\log p(y_k|x_k) - \log p(x_k|x_{k-1}) \tag{1.25}$$

Solving this value iteration problem for general distributions $p(y_k|x_k)$ and $p(x_k|x_{k-1})$ is often very difficult and one must almost always resort to approximations, two of which will be discussed next. But first note how the Kalman filter fits into this framework.

Relation to the Kalman Filter. As pointed out above, in the case of a Gaussian distribution, the last element of the JMAP estimate coincides with the MMSE and MAP estimates. This implies that the recursive algorithms (1.21) or (1.24) should both reproduce the Kalman filter equations. This is indeed also the case as demonstrated in [Larson and Peschon, 1966] or [Cox, 1964]. When the problem is formulated as in (1.24), the step cost L_k becomes a quadratic function, and thus a quadratic function on the form

$$V_k(x_k) = \begin{bmatrix} x_k \\ 1 \end{bmatrix}^T \pi_k \begin{bmatrix} x_k \\ 1 \end{bmatrix} \tag{1.26}$$

serves as value function.

Approximations of the Recursive JMAP Estimator

The general value iteration problem (1.24) can only be solved in special cases, like the Kalman filter discussed above. Next we will present two approximation techniques recently proposed in the literature.

Relaxed Dynamic Programming. Relaxed dynamic programming [Lincoln and Rantzer, 2006] was recently proposed as a method for conquering the complexity explosion usually associated with dynamic programming. The basic idea is to replace the equality (1.24) with two inequalities

$$\underline{V}_k(x_k) \le V_k^{\text{approx}}(x_k) \le \overline{V}_k(x_k) \tag{1.27}$$

where the upper and lower bounds are defined as

$$\begin{aligned} \overline{V}_k(x_k) &= \min_{x_{k-1}} \left\{ V_{k-1}^{\text{approx}}(x_{k-1}) + \overline{\alpha} L_k(x_k, x_{k-1}) \right\} \\ \underline{V}_k(x_k) &= \min_{x_{k-1}} \left\{ V_{k-1}^{\text{approx}}(x_{k-1}) + \underline{\alpha} L_k(x_k, x_{k-1}) \right\} \end{aligned} \tag{1.28}$$

Here the scalars $\overline{\alpha} > 1$ and $\underline{\alpha} < 1$ are slack parameters that can be chosen to trade off optimality against complexity. By the introduction of inequalities instead of equalities it is in principle possible to fit a simpler value function between the upper and lower bounds. If the approximate value function is computed as above, it will satisfy

$$\underline{\alpha} V_k(x_k) \le V_k^{\text{approx}}(x_k) \le \overline{\alpha} V_k(x_k) \tag{1.29}$$

which gives guarantees on how far from optimal the approximate solution is.

The approximate value function can be parametrized in many ways, as long as there exist methods for computing the upper and lower bounds and finding a V_k^{approx} satisfying (1.27). How to choose a good parametrization of the relaxed value function for a state feedback control setup has recently been studied in [Wernrud, 2008].

In Paper VI this approximation technique was applied to a state estimation problem for Markov jump linear systems. In that case, the value function was parametrized as a minimum of quadratic functions. That example will now be used to illustrate the approximation procedure.

Consider the illustration in Figure 1.2. The dashed curve shows the approximate value function V_{k-1}^{approx} computed at the previous time step $k-1$. At time k, first the upper $\overline{V}_k$ and lower $\underline{V}_k$ bounds are computed according to (1.28). The piecewise quadratic shape is due to the discrete nature of the Markov jump linear system dynamics. Next, using the flexibility introduced by the slack parameters, a simpler approximate value function V_k^{approx} can be fitted between the upper an lower bounds. Choosing a larger $\overline{\alpha}$ and/or smaller $\underline{\alpha}$ will increase the gap between the upper and lower bounds, thus making it easier to fit the new approximate value function.

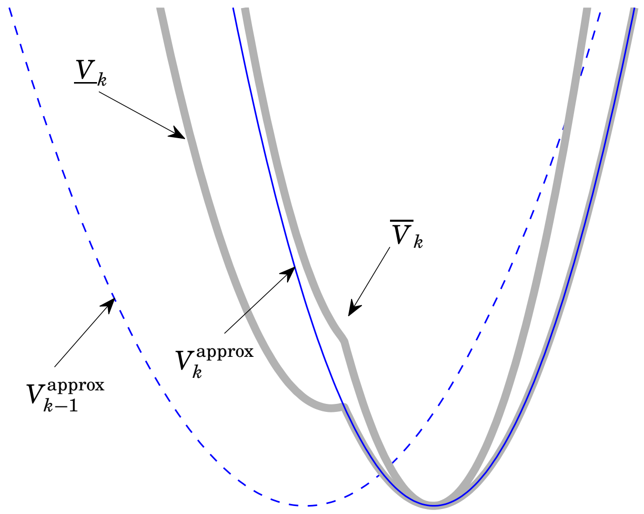


Figure 1.2 1-D illustration of relaxed dynamic programming. The dashed curve shows the approximate value function V_{k-1}^{approx} computed at the previous time step $k-1$. At time k, first the upper $\overline{V}_k$ and lower $\underline{V}_k$ bounds are computed according to (1.28). The piecewise quadratic shape is due to the discrete nature of the Markov jump linear system dynamics. Next, using the flexibility introduced by the slack parameters, a simpler approximate value function V_k^{approx} can be fitted between the upper an lower bounds.

Moving Horizon Estimation. Instead of approximating the recursive solution directly, moving horizon estimation takes the original problem

$$\arg\max_{X_k} p(X_k|Y_k) \tag{1.30}$$

as its starting point. Taking logarithms and using Bayes' theorem together with the Markov property allows us to rewrite (1.30) as

$$\arg\min_{X_k} \left\{ -\sum_{i=0}^{k} \log p(y_i|x_i) - \sum_{i=0}^{k-1} \log p(x_{i+1}|x_i) - \log p(x_0) \right\} \tag{1.31}$$

The main problem here is that this optimization problem grows with time. In the control community, infinite horizon problems have successfully been approximated with finite horizon approximations in a receding horizon fashion, resulting in the model predictive control framework. Using similar ideas the moving horizon estimation method only considers data in a finite time window of length N that moves forward with time,

$$\arg\min_{x_{k-N},\ldots,x_k} \left\{ -\sum_{i=k-N+1}^{k} \log p(y_i|x_i) - \sum_{i=k-N}^{k-1} \log p(x_{i+1}|x_i) + \Phi_k(x_{k-N}) \right\} \qquad (1.32)$$

The function $\Phi(x_{k-N})$, commonly referred to as arrival cost, is used to summarize previous information. In the literature, there is very little support on how to choose this function [Rawlings and Bakshi, 2006]. However, in [Rao, 2000] it was shown that if this function is chosen sufficiently small, the moving horizon estimation scheme is stable.

One big advantage with moving horizon estimation is that it is straight forward to include constraints into the minimization problem (1.32). Care must, however, be taken as state constraints can lead to models that do not fulfill the Markov property. Some of these issues are discussed in [Rao, 2000]. In [Ko and Bitmead, 2005] this problem is studied for systems with linear dynamics and linear equality constraints.

One interesting observation is that for $N = 1$ the moving horizon estimation scheme reduces to the dynamic programming equation (1.24) with $V_k(x_k) = \Phi_k(x_k)$. Thus one can view “dynamic programming estimation” as moving horizon estimation with a horizon of length one and a very elaborate arrival cost.

The field of moving horizon estimation has received considerable attention in literature during the last ten years. On the application side, the method has proven very useful for linear systems with linear constraints. Much of this success is due to the fact the minimization problem in this case becomes a quadratic program, which can be solved efficiently. For a review of the present situation [Rawlings and Bakshi, 2006] is a good starting point.

2

Sensor and Sensor-Actuator Networks

The advances in micro electronics have during the last three decades made embedded systems an integral part of our every day lives. In the last decade or so, the Internet has transformed the way we use desktop PCs from word processors to information portals. The next natural step is to also allow embedded systems to interact through the use of networking technology. Sometimes the names Pervasive Computing or Ubiquitous Computing are used to describe this evolution.

This chapter by no means claims to give a full description of the huge field of networked embedded systems. Instead it points out some areas which have inspired the work presented in this thesis. It also highlights a few practical problems that one may encounter when networked embedded systems are designed or even more so debugged.

Here we choose to divide the field into Sensor Networks and Sensor-Actuator Networks. The reason for this is that the latter introduces additional constraints inherent to all implementations of closed loop control systems.

2.1 Sensor Networks

Traditionally the main focus of many research efforts and applications have been networks of passive sensors, often referred to as “wireless sensor networks” or WSN for short. In these networks wireless transceivers are attached to a large number of sensors and the sensor readings are processed at a central server. This “sense- and-send” paradigm works well for low-frequency applications where high-latency is not an issue.

However, when the number of sensors grow and/or they have to report values back at a higher rate the wireless channels might get saturated. Also, as most WSN use multi-hop communication, nodes near the central server will drain their energy resources unnecessarily fast.

Design Challenges

When designing a WSN one has to consider a number of things [Akyildiz *et al.*, 2002] such as radio frequencies, frequency hopping, power tradeoffs, latency, interference, network protocols, routing, security and so on. In sense-and-send networks the major concern is often power consumption. The reason being that nodes are expected to operate for long periods of time using only battery power.

Contrary to what is often assumed in the control community, one large source of energy consumption in many commercially available sensor nodes is listening for packets. In for example [Dunkels *et al.*, 2007] the power consumption was estimated for a typical sense-and-send application. There it was reported that more than 80% of the total energy was consumed in idle listening mode. Similar results are also presented in Paper II, where it was noted that the power consumption when transmitting was only about 5% higher compared to idle listening.

This issue has been recognized by the more computer science oriented part of the sensor network community, resulting in duty-cycled MAC protocols like X-MAC or B-MAC, see [Buettner *et al.*, 2006]. The problem with duty-cycled MAC protocols is that longer latencies from transmission to reception are often the result.

Data Generation

Before we continue discussing more elaborate techniques to save energy and avoid saturating communication links, a number of ways that data might be generated will be specified [Akkaya and Younis, 2005]:

Event Based data generation refers to a scenario where nodes generate data based on some external event. This event could for example be generated if a measured quantity exceeds a specified threshold or if an object is detected in the vicinity of the sensor.

Periodic data generation refers to a situation where all sensors in the network generate data periodically. The distributed Kalman filter algorithm in papers I, II and III is an example of this scheme.

Query Based data generation refers to a setup where one or many users query the network for information. The sensor scheduling algorithm in Paper V is an example of this scheme.

In the case of event driven and periodic data generation one may also differentiate between the case where all nodes require information about the measured quantity or only a small set of so called sink nodes.

Data Aggregation

As the computational and memory resources of sensor nodes increase more elaborate algorithms can be used. Local filtering, data analysis and classification can be performed prior to transmission. This increased flexibility can be used to reduce the rate at which nodes have to communicate and also remove the need to route every single piece of data through the network. It is, however, worth noticing that using more complicated algorithms might introduce requirements on for example more accurate synchronization. Also, because of the increased complexity, many design flaws can only be detected after deployment. This leads to systems where the ability to dynamically change software becomes important.

Data aggregation in WSN is a huge field spanning from in depth mathematical treatments based on for example information theory and consensus algorithms to more practical aspects such as routing and the design of various communication protocols. Several surveys have been published on the subject of which [Akkaya and Younis, 2005], [Rajagopalan and Varshney, 2006] and [Luo *et al.*, 2007] are among the more recent. One way to classify data aggregation techniques is as in [Luo *et al.*, 2007] where the three categories: routing-driven, coding-driven and fusion-driven were used.

Routing-driven. In routing-driven approaches, the focus is on routing data to the sink node in an energy efficient way. Data aggregation only takes place opportunistically, that is only if packets that can be aggregated happen to meet on their way to the sink. One example of a routing-driven protocol is the query based Directed Diffusion algorithm [Intanagonwiwat *et al.*, 2003].

Coding-driven. Coding-driven algorithms focus on compressing raw sensor data locally to reduce redundancies, thus reducing the need for further aggregation. This problem is often approached from an information theoretic point of view. The compressed data can then be routed using for example a purely routing-driven approach. In for example [Cristescu *et al.*, 2005] messages are routed along the shortest-path tree and node transmission rates are chosen based on entropy measures.

Fusion-driven. Fusion-driven routing is the name for a large set of algorithms where the basic idea is not only to allow every node to aggregate data, but to route this aggregated data in such way that further aggregation is possible. The total energy required for information to reach the sink node(s) is thus minimized. Unfortunately solving this problem for arbitrary node placements and a general communication topology has proven to be NP-hard, see for example [Yen *et al.*, 2005].

2.2 Distributed State Estimation

Most of the techniques discussed in the previous section are directed at scenarios where information flows from the sensor network to one or a few sink nodes. An alternative approach is to let every node in the network have full, or at least partial, knowledge of the aggregated quantity. In distributed estimation problems this quantity is represented as the state x_k of a dynamical system. Each node updates its belief about the state using local measurements and a dynamical model much like what was described in Chapter 1. Nodes then exchange beliefs with their neighbors.

Often it is not only the directly measurable quantity, but some underlying phenomena that is of interest. In for example Paper III, sensor nodes measure their distance to a mobile robot, but it is really the position of the robot that is of interest. This scenario fits well within the distributed estimation framework. Much of the work on distributed state estimation, or distributed data fusion as it is sometimes called, has been done in the target tracking community. There the term track-to-track fusion is often used. For a recent survey directed at target tracking applications, but also of general interest, see [Smith and Singh, 2006].

Because nodes only use information generated by themselves and their neighbors, no routing/forwarding of packets is necessary. If these types of algorithms are implemented in such a way that the radio can be turned off during long periods, the energy consumption can be greatly reduced.

When using a dynamic model, the relationship between different measurements, both in time and among different nodes, is made explicit. If this assumed relationship is violated, for example by bad clock synchronization or sampling-period jitter, performance may degrade considerably.

Conceptual Solution

In a distributed estimation application, each node has its own belief about the aggregated quantity. This belief is then exchanged with neighboring nodes and thus the collective knowledge is increased.

Similar to what was done in Chapter 1, the distributed estimation problem can be formulated using probability distributions. However, combining distributions from two nodes is far from trivial. The problem is that one node's belief can be based on the same data as the belief of another node. To combine these, common data must be accounted for. Before explaining how to combine two distributions, a notation similar to the one in [Liggins *et al.*, 1997] is introduced:

Y_k^A	Information available in node A at time k.
$Y_k^{A\cup B}$	Information available in node A or B at time k.
$Y_k^{A\cap B}$	Information available in node A and B at time k.
$Y_k^{A\setminus B}$	Information available in node A and not B at time k.

Next we will derive a relation that can be used when combining two conditional distributions that contain common information. This relation was derived in for example [Chong *et al.*, 1982] and is summarized as a theorem below.

THEOREM 2.1
If the exclusive information in node A and B is conditionally independent given the state x_k, that is

$$p(Y_k^{A\setminus B}, Y_k^{B\setminus A}|x_k) = p(Y_k^{A\setminus B}|x_k)p(Y_k^{B\setminus A}|x_k) \tag{2.1}$$

the combined belief $p(x_k|Y_k^{A\cup B})$ is related to the two individual distributions $p(x_k|Y_k^A)$ and $p(x_k|Y_k^B)$ as

$$p(x_k|Y_k^{A\cup B}) \propto \frac{p(x_k|Y_k^A)p(x_k|Y_k^B)}{p(x_k|Y_k^{A\cap B})} \tag{2.2}$$

where $\propto$ denotes proportional to.

Proof. First note that

$$p(x_k|Y_k^{A\cup B}) = p(x_k|Y_k^{A\setminus B}, Y_k^{B\setminus A}, Y_k^{A\cap B}) \tag{2.3}$$

Now using Bayes' theorem write

$$p(x_k|Y_k^{A\setminus B}, Y_k^{B\setminus A}, Y_k^{A\cap B}) \propto p(Y_k^{A\setminus B}, Y_k^{B\setminus A}|x_k, Y_k^{A\cap B})p(x_k|Y_k^{A\cap B}) \tag{2.4}$$

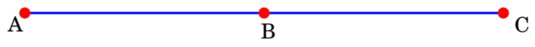
The conditional independence assumption implies that

$$p(Y_k^{A\setminus B}, Y_k^{B\setminus A}|x_k, Y_k^{A\cap B}) = p(Y_k^{A\setminus B}|x_k, Y_k^{A\cap B})p(Y_k^{B\setminus A}|x_k, Y_k^{A\cap B}) \tag{2.5}$$

Using Bayes' theorem on the two factors in (2.5) together with (2.3) and (2.4) gives the desired relation. □

When more than two nodes are involved this formula can be used repeatedly, combining distributions one at the time. To illustrate this, consider the following example where the time index has been dropped for brevity.

Example 2.1
Consider the following simple communication graph:



Node B can first combine information from A using (2.2) as

$$p(x|Y^{A\cup B}) \propto \frac{p(x|Y^A)p(x|Y^B)}{p(x|Y^{A\cap B})} \tag{2.6}$$

and then from C as

$$p(x|Y^{A\cup B\cup C}) \propto \frac{p(x|Y^{A\cup B})p(x|Y^C)}{p\left(x|Y^{(A\cup B)\cap C}\right)} \tag{2.7}$$

The problem with this approach is how to compute the common information in the denominators. Because these nodes communicate according to a tree topology, A and C can not have any common information that B is not aware of, or formally $Y^{(A\cup B)\cap C} = Y^{B\cap C}$. The conditional distributions can thus be combined as

$$p(x|Y^{A\cup B\cup C}) \propto \frac{p(x|Y^A)p(x|Y^B)p(x|Y^C)}{p(x|Y^{A\cap B})p(x|Y^{B\cap C})} \tag{2.8}$$

How to compute the common information for a general communication topology was for example studied in [Liggins *et al.*, 1997]. □

Unfortunately the conditional independence assumption (2.1) is rather restrictive. It is satisfied in the following two cases:

1. The state evolution is deterministic, that is $x_{k+1} = f_k(x_k)$ for some known function $f_k(\cdot)$.
2. Before taking a new measurement, nodes keep exchanging beliefs until they all possess the same information.

If these assumptions are not satisfied the problem becomes more involved. The problem can be solved by expanding the belief representation $p(x_k|Y_k)$ to the distribution $p(x_0,\dots,x_k|Y_k)$ of the full trajectory. With this representation, (2.1) is fulfilled as long as the measurement noise in different nodes is independent. Unfortunately the size of $p(x_0,\dots,x_k|Y_k)$ grows with time. If, however, an upper bound τ on the maximum delay between when a measurement is collected and when it has been used in all nodes is available, this information can be used to reduce the size of the belief distribution to $p(x_{k-\tau},\dots,x_k|Y_k)$. This approach was used in [Rosencrantz *et al.*, 2003] where a distributed particle filter was developed.

The next section is devoted to a number of approximate methods for the special case of linear dynamics with Gaussian disturbances.

Approximate Distributed Kalman Filters

In this section we will assume that the measured quantity y_k^i in node $i \in \{1, \dots, N\}$ can be modelled as the output of a linear system subject to white Gaussian noise w_k and v_k^i

$$\begin{aligned} x_{k+1} &= Ax_k + w_k \\ y_k^i &= C^i x_k + v_k^i \end{aligned} \tag{2.9}$$

The process noise w_k and measurement noise v_k^i are assumed independent with covariance matrices R_w and R_v^i respectively. Under these assumptions, distributed state estimation algorithms are often referred to as distributed Kalman filters or DKFs for short. In a DKF, all probability distributions are Gaussian and thus only means and covariances need to be considered.

One way to approximate the optimal solution presented above is to use the merge equation (2.2) even if the conditional independence assumption (2.1) is not satisfied. This approach was used in for example [Grime *et al.*, 1992] and in a series of papers by the same authors.

An alternative approximation technique is to combine estimates using a weighted linear combination. The weights are optimized to yield a minimal error covariance matrix of the combined estimate. This scheme is sometimes referred to as the Bar-Shalom-Campo algorithm [Bar-Shalom and Campo, 1986]. Papers I, II and III are also based on this idea.

To find optimal weights, the cross covariance between estimates must be known. If this is not the case, the so called covariance intersection algorithm [Julier and Uhlmann, 1997b] can be used. The covariance intersection algorithm produces a consistent linear combination without knowledge of the error cross covariance between the estimates. Note that the resulting covariance is always larger or equal to the optimal one.

Performance for various approximation schemes was investigated in [Mori *et al.*, 2002], which also serves as a good survey on algorithms based on the two approximation approaches discussed above.

Consensus Based Kalman Filtering. Recently there has been a large interest in so called consensus algorithms. As defined in [Olfati-Saber *et al.*, 2007] consensus, in the context of networks of agents, means "to reach an agreement regarding a certain quantity of interest that depends on the state of all agents". To reach this agreement a so called consensus algorithm is used. A consensus algorithm is "an interaction rule that specifies the information exchange between an agent and all of its neighbors on the network". These algorithms can be designed in a number of different ways, for example to maximize the rate at which consensus is reached. This was done in for example [Xiao and Boyd, 2004].

The first consensus-based distributed Kalman filters [Olfati-Saber, 2005] used the information form of a centralized Kalman filter as their starting point. The Kalman filter on information form can be written using the information matrix

$$Z_k^{-1} = \mathbf{E}\ (x - \hat{x}_k)(x - \hat{x}_k)^T \tag{2.10}$$

and information vector

$$z_k = Z_k \hat{x}_k \tag{2.11}$$

where $\hat{x}_k$ denotes the state estimate. When it is necessary to specify on which information an estimate is based, the notation $z_{k|k}$ $(Z_{k|k})$ and $z_{k|k-1}$ $(Z_{k|k-1})$ is used to denote filtering and one-step ahead prediction respectively.

Writing the Kalman filter in information form has the advantage that conditionally independent measurements can be incorporated additively:

$$\begin{aligned} z_{k|k} &= z_{k|k-1} + \sum_{i=1}^{N} (C^i)^T (R_v^i)^{-1} y_k^i \\ Z_{k|k} &= Z_{k|k-1} + \sum_{i=1}^{N} (C^i)^T (R_v^i)^{-1} C^i \end{aligned} \tag{2.12}$$

The prediction step,

$$\begin{aligned} z_{k|k-1} &= Z_{k|k-1} A Z_{k-1|k-1}^{-1} z_{k-1|k-1} \\ Z_{k|k-1} &= (A Z_{k-1|k-1}^{-1} A^T + R_w)^{-1} \end{aligned} \tag{2.13}$$

however, becomes somewhat more involved. If both sums in (2.12) are available in all N nodes, for example by all-to-all communication, each node can run the filter described above. When only neighbor-to-neighbor communication is possible, consensus filters are used to compute these two sums. Note that even if the system dynamics are time-invariant the first sum is time-varying because it includes the actual measurements y_k^i. This fact has the important implication that unless the consensus algorithm is run at a much higher rate than the filter, also this approach is only approximate.

Recently, in [Olfati-Saber, 2007] a number of consensus-based algorithms, where also neighboring state estimates are used, were developed. These algorithms improve upon the approach described above, but are still only approximations.

2.3 Sensor-Actuator Networks

In a sensor-actuator network the WSN ideas are taken one step further. Here sensors are not only required to measure, aggregate and transmit data, but also to act on it. In these type of networks the sense-and-send paradigm does not work very well, because of the long latency introduced when decisions are made by a central server. Sensor-actuator networks can be viewed as an application of distributed decision making. Distributed decision making has been an active research area for several decades, see for example [Gattami, 2007] for references and a recent mathematical treatment.

In a network designed to only collect information, it is usually not critical when this information reaches its destination, as long as it does and does so in an energy efficient way. However, if a network not only collects information from the physical world, but also acts on it, it makes a big difference if this information is delayed. This has implications on the tradeoffs made when designing for example data aggregation algorithms. In the case of distributed estimation, usually filtering, that is estimating the current state based on information up to and including the current time, is considered. This fits well with the requirements imposed by closed loop control.

The low quality wireless links often used further complicates the problem. Even if lost packets are retransmitted, there can still be situations where no or very little information gets through. In these situations, it is important that a node can make a reasonable decision based solely on local information. In Paper IV a mobile robot uses a sensor network to navigate. However, when no data is available from the network, the robot navigates using only local sensors. This situation can be captured by the general state estimation framework presented in Chapter 1.

2.4 Software Platforms

Real-time critical systems have traditionally executed in real-time operating systems, or at least on computational platforms where timing issues can be kept under control. However, the tradeoff between cost and functionality has forced also real-time critical applications to run on severely resource-constrained platforms. Recently these issues have been studied in for example [Henriksson, 2006].

In sensor and sensor-actuator networks the hardware platforms available are usually even more resource constrained. In addition to this, the available operating systems are often focused on traditional WSN applications where delay and accurate timing is of less importance.

Recently there has been a trend towards modular and component-based software development in these type of systems. In for example the European integrated project Reconfigurable Ubiquitous Networked Embedded Systems [RUNES, 2007] a component-based middleware was developed. When using a component-based design methodology it is of great importance to achieve a situation where component properties do not change as a result of interactions with other components. If this is not the case, the benefits of a component-based design methodology are to a large extent only illusive.

In the networked embedded systems we consider here, the four main resources are: memory, CPU time, network bandwidth and energy. To achieve the situation described above, a component must make explicit its uses and requirements of all these resources. This is, however, not enough. The way components are combined into software systems must make sure that the utilization of all resources does not exceed their maximum capacity. Ideally a component should be able to recognize that it does not get the resources it requires and adapt accordingly. Of course designing such systems still remains a great challenge.

Sensor-actuator networks typically consist of a number of different hardware platforms ranging from low-end sensor nodes through gateways all the way to powerful central servers. The usage of a common network protocol, such as IP, makes it possible to add new platforms without having to write adaptor functions. Recent developments in network technology such as the μIP stack [Dunkels, 2003] has made it possible to seamlessly integrate severely resource-constrained platforms with high-end central servers.

2.5 Development Tools

When developing distributed control and/or estimation algorithms operating on severely resource-constrained platforms the lack of distributed debugging and monitoring tools becomes evident. Normally straight forward tasks such as logging of measured signals become a problem. Using wired logging might not be possible if the network covers a large geographical area and wireless logging will consume bandwidth and CPU time, thus effecting the system under study. Because many issues in these type of distributed applications can only be observed after deployment, these problems can consume a considerable amount of time during the development process.

One way to resolve the situation is through simulation. In the literature there are numerous simulators for wireless sensor networks, some of which are listed below:

TOSSIM [Levis *et al.*, 2003] is simulator for TinyOS [TinyOS Open Technology Alliance, 2008] that compiles directly from TinyOS code.

ns-2 [ns-2, 2008] is a general purpose network simulator that supports simulation of TCP, routing and multi-cast protocols over wired and wireless networks.

Mannasim [Mannasim, 2008] is a module for WSN simulation based on the network simulator (ns-2). Mannasim extends ns-2 introducing new modules for design, development and analysis of different WSN applications.

OMNeT++ [Varga, 2001] is a discrete event simulation environment. Its primary application area is simulation of communication networks.

J-Sim [J-sim, 2008] is a component-based, compositional simulation environment written in Java.

COOJA [Österlind *et al.*, 2007] is a Contiki [Dunkels *et al.*, 2004] simulator that allows cross-level simulation of both network and operating system.

The simulation tools presented above are mainly focused on network simulation. When introducing more complex data aggregation algorithms, such as the distributed Kalman filter, one also need to consider how timing-effects influence performance. This calls for so called co-simulation tools, where embedded systems, networks and physical systems can be simulated all within the same tool. In Paper II the Matlab/Simulink based tool TrueTime [Andersson *et al.*, 2005] was used to investigate how synchronization effected packet loss in a distributed Kalman filter algorithm.

3

Future Work

Both state estimation for hybrid and distributed systems and sensor and sensor-actuator networks are very active research areas. The work presented in this thesis can be extended in several directions.

Dynamic Programming Estimation

The approximate dynamic programming techniques used here, might also prove fruitful for other problem classes. The key issue is, however, how to parametrize the value function in an efficient way.

On the theoretical side, it could be interesting to quantify the relationship between the relaxation parameters and estimation performance.

Recently, there has been increased interest in using relaxed dynamic programming for estimating the degree of suboptimality in receding horizon control schemes. See for example [Grüne and Pannek, 2008]. Similar ideas might be used to develop the connection between moving horizon estimation and “dynamic programming estimation”. Perhaps relaxed dynamic programming can be used as a systematic tool for constructing the arrival cost.

System Theoretic Tools in Sensor(-Actuator) Networks

Currently there is a trend towards consensus based algorithms in the control community. These algorithms could prove useful for many, but far from all, problems in distributed estimation and decision making. Looking at these problems from a general state estimation perspective might also prove insightful.

Perhaps the main drawback with the distributed Kalman filter algorithm in papers I, II and III is that it requires global knowledge of the communication topology at the deployment phase. One way to relax this assumption could be to make use of the covariance intersection algorithm [Julier and Uhlmann, 1997b].

Software Platforms

When working with more practical aspects of sensor-actuator networks one gets painfully aware how much time simple tasks as updating software on a deployed network or logging of various quantities can take. Inherent in the distributed nature of the application is that many effects can only be studied after deployment. Recently, some very promising Java based platforms have been released. Perhaps these together with good simulation tools can resolve some of these issues.

When introducing model-based system-theoretical tools, the requirements of accurate timing and synchronization increases. For this to work well with a component-based software architecture, these requirements must be made explicit in the component description. This is especially important when software modules can be loaded dynamically. Reservation-based scheduling is a promising concept that might resolve some of these problems. However, how to fit a sophisticated run-time kernel, a network stack and still have room for applications on a low-end platform still remains an open question.

References

Akkaya, K. and M. Younis (2005): “A survey on routing protocols for wireless sensor networks.” *Ad Hoc Networks*, **3:3**, pp. 325–349.

Akyildiz, I., W. Su, Y. Sankarasubramaniam, and E. Cayirci (2002): “A survey on sensor networks.” *Communications Magazine, IEEE*, **40:8**, pp. 102–114.

Andersson, M., D. Henriksson, A. Cervin, and K.-E. Årzén (2005): “Simulation of wireless networked control systems.” In *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*. Seville, Spain.

Bar-Shalom, Y. and L. Campo (1986): “The effect of the common process noise on the two-sensor fused-track covariance.” *IEEE Transactions on Aerospace and Electronic Systems*, **AES-22:6**, pp. 803–805.

Buettner, M., G. Yee, E. Anderson, and R. Han (2006): “X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks.” In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, pp. 307– 320. Boulder, Colorado, USA.

Chong, C. Y., S. Mori, E. Tse, and R. P. Wishner (1982): “Distributed estimation in distributed sensor networks.” *American Control Conference*, **19**, pp. 820–826.

Cox, H. (1964): “On the estimation of state variables and parameters for noisy dynamic systems.” *IEEE Trans. Automat. Contr.*, **9**, January, pp. 5–12.

Cristescu, R., B. Beferull-Lozano, and M. Vetterli (2005): “Networked slepian-wolf: theory, algorithms, and scaling laws.” *IEEE Transactions on Information Theory*, **51:12**, pp. 4057–4073.

Doucet, A., N. de Freitas, and N. Gordon (2001): *Sequential Monte Carlo methods in practice*. Statistics for engineering and information science. Springer.

Dunkels, A. (2003): “Full TCP/IP for 8 Bit Architectures.” In *Proceedings of the First ACM/ Usenix International Conference on Mobile Systems, Applications and Services (MobiSys 2003)*. USENIX, San Francisco.

Dunkels, A., B. Grönvall, and T. Voigt (2004): "Contiki - a lightweight and flexible operating system for tiny networked sensors." In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*. Tampa, Florida, USA.

Dunkels, A., F. Österlind, N. Tsiftes, and Z. He (2007): "Software-based on-line energy estimation for sensor nodes." In *Proceedings of the Fourth Workshop on Embedded Networked Sensors (Emnets IV)*. Cork, Ireland.

Gattami, A. (2007): *Optimal Decisions with Limited Information*. PhD thesis ISRN LUTFD2/TFRT--1079--SE, Department of Automatic Control, Lund University, Sweden.

Grime, S., H. F. Durrant-Whyte, and P. Ho (1992): "Communication in decentralized data-fusion systems." In *In Proc. American Control Conference*, pp. 3299–3303.

Grüne, L. and J. Pannek (2008): "Trajectory based suboptimality estimates for receding horizon controllers." In *Proceedings of the 18th International Symposium on Mathematical Theory of Networks and Systems MTNS2008*. Blacksburg, Virginia.

Henriksson, D. (2006): *Resource-Constrained Embedded Control and Computing Systems*. PhD thesis ISRN LUTFD2/TFRT--1074--SE, Department of Automatic Control, Lund Institute of Technology, Sweden.

Ho, Y. C. and R. C. K. Lee (1964): "A Bayesian approach to problems in stochastic estimation and control." *IEEE Trans. Automat. Contr.*, **9**, October, pp. 333–339.

Intanagonwiwat, C., R. Govindan, D. Estrin, J. Heidemann, and F. Silva (2003): "Directed diffusion for wireless sensor networking." *IEEE/ACM Transactions on Networking*, **11:1**, pp. 2–16.

J-sim (2008): `http://www.j-sim.org`.

Jazwinski, A. H. (1970): *Stochastic Processes and Filtering Theory*. Academic Press.

Julier, S. and J. Uhlmann (1997a): "A new extension of the Kalman filter to nonlinear systems." In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*.

Julier, S. and J. Uhlmann (1997b): "A non-divergent estimation algorithm in the presence of unknown correlations." *Proceedings of the American Control Conference*, **4**, pp. 2369–2373.

Julier, S. and J. Uhlmann (2004): “Unscented filtering and nonlinear estimation.” *Proceedings of the IEEE*, **92:3**, pp. 401–422.

Kalman, R. E. (1960): “A new approach to linear filtering and prediction problems.” *Transactions of the ASME–Journal of Basic Engineering*, **82:Series D**, pp. 35–45.

Ko, S. and R. R. Bitmead (2005): “State estimation of linear systems with state equality constraints.” In *Proccedings of the 16th IFAC World Congress*.

Larson, R. E. and J. Peschon (1966): “A dynamic programming approach to trajectory estimation.” *IEEE Trans. Automat. Contr.*, **11**, July, pp. 537–540.

Levis, P., N. Lee, M. Welsh, and D. Culler (2003): “TOSSIM: accurate and scalable simulation of entire TinyOS applications.” In *SenSys ’03: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pp. 126–137. ACM, New York, NY, USA.

Liggins, M., C.-Y. Chong, I. Kadar, M. Alford, V. Vannicola, and S. Thomopoulos (1997): “Distributed fusion architectures and algorithms for target tracking.” *Proceedings of the IEEE*, **85:1**, pp. 95–107.

Lincoln, B. and A. Rantzer (2006): “Relaxing dynamic programming.” *IEEE Transactions on Automatic Control*, **51:8**, pp. 1249–1260.

Luo, H., Y. Liu, and S. Das (2007): “Routing correlated data in wireless sensor networks: A survey.” *IEEE Network*, **21:6**, pp. 40–47.

Mannasim (2008): `http://www.mannasim.dcc.ufmg.br`.

Mori, S., W. H. Barker, C.-Y. Chong, and K.-C. Chang (2002): “Track association and track fusion with nondeterministic target dynamics.” *IEEE Transactions on Aerospace and Electronic Systems*, **38:2**, pp. 659–668.

ns-2 (2008): `http://www.isi.edu/nsnam/ns/`.

Olfati-Saber, R. (2005): “Distributed Kalman filter with embedded consensus filters.” In *Proceedings of the 44th IEEE Conference on Decision and Control, and European Control Conference*.

Olfati-Saber, R. (2007): “Distributed Kalman filtering for sensor networks.” In *Proceedings of the 46th Conference on Decision and Control*, pp. 5492–5498. New Orleans, LA, USA.

Olfati-Saber, R., J. A. Fax, and R. M. Murray (2007): “Consensus and cooperation in networked multi-agent systems.” *Proceedings of the IEEE*, **95:1**, pp. 215–233.

Rajagopalan, R. and P. Varshney (2006): “Data-aggregation techniques in sensor networks: A survey.” *IEEE Communications Surveys and Tutorials*, **8:4**, pp. 48–63.

Rao, C. V. (2000): *Moving Horizon Strategies for the Constrained Monitoring and Control of Nonlinear Discrete-Time Systems*. PhD thesis, University of Wisconsin-Madison.

Rawlings, J. and B. Bakshi (2006): “Particle filtering and moving horizon estimation.” *Computers and Chemical Engineering*, **30:10-12**, pp. 1529–1541.

Rosencrantz, M., G. Gordon, and S. Thrun (2003): “Decentralized sensor fusion with distributed particle filters.” In *Proc. Conf. Uncertainty in Artificial Intelligence, Acapulco, Mexico*.

RUNES (2007): “Reconfigurable ubiquitous networked embedded systems.” `http://www.ist-runes.org`.

Schön, T. B. (2006): *Estimation of Nonlinear Dynamic Systems : Theory and Applications*. PhD thesis, Linköping Univ.

Smith, D. and S. Singh (2006): “Approaches to multisensor data fusion in target tracking: A survey.” *IEEE Transactions on Knowledge and Data Engineering*, **18:12**, pp. 1696–1711.

TinyOS Open Technology Alliance (2008): *TinyOS*. `http://www.tinyos.net`.

Varga, A. (2001): “The OMNeT++ discrete event simulation system.” In *In the Proceedings of the European Simulation Multiconference (ESM’2001)*. Prague, Czech Republic.

Wernrud, A. (2008): *Approximate Dynamic Programming with Applications*. PhD thesis ISRN LUTFD2/TFRT--1082--SE, Department of Automatic Control, Lund University, Sweden.

Xiao, L. and S. Boyd (2004): “Fast linear iterations for distributed averaging.” *Systems and Control Letters*, **53:1**, pp. 65–78.

Yen, H.-H., F. Y.-S. Lin, and S.-P. Lin (2005): “Energy-Efficient Data-Centric Routing in Wireless Sensor Networks.” *IEICE Trans. Commun.*, **E88-B:12**, pp. 4470–4480.

Österlind, F., A. Dunkels, J. Eriksson, N. Finne, and T. Voigt (2007): “Cross-level simulation in cooja.” In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*. Delft, The Netherlands.

Paper I

Model Based Information Fusion in Sensor Networks

Peter Alriksson and Anders Rantzer

Abstract

In this paper, a model based sensor fusion algorithm for sensor networks is presented. The algorithm, referred to as distributed Kalman filtering is based on a previously presented algorithm with the same name. The weight selection process has been improved yielding performance improvements of several times for the examples studied. Also, solutions to both optimization problems involved in the iterative offline weight selection process are given as closed form expressions. The algorithm is also demonstrated on a typical signal tracking application.

1. Introduction

In recent years the increases in battery and processing power of sensor nodes has made a wide range of sensing applications possible. However as the number of sensors in a network increase the need for efficient data aggregation becomes more and more evident. For a small sensor network routing measurements to a central node using for example Ad hoc On Demand Distance Vector (AODV) routing might be feasible, see [Perkins *et al.*, 2003]. However as the network grows, the computational- and network load both in the central node and in bottleneck nodes throughout the network will be a major problem. Also these nodes will drain their energy resources unnecessarily fast.

There are numerous data fusion techniques in the sensor network literature, but most fall into two categories: data driven and model based. An example of a data driven technique would for example be finding the maximum temperature in an area. Each node compares its temperature with its neighbors and only the maximum is transmitted. In this paper we will focus on a model based approach. One simple example would be to estimate the mean temperature in an area. The temperature could then be modeled as a constant quantity that is observed through a number of noisy sensors. In the model based approach the quantity of interest is not required to be directly measurable but can be estimated from previous measurements using a model.

2. Previous Work

The technique used in this paper is often referred to as distributed Kalman filtering. In a distributed Kalman filter, nodes exchange estimates of the quantity of interest possibly together with the their local measurements.

An early reference is [Durrant-Whyte *et al.*, 1990] where a decentralized Kalman filter was proposed. However, this algorithm requires every node to be able to communicate with every other node, which is not possible in the setup studied here.

One common technique is to apply consensus filters, see [Olfati-Saber *et al.*, 2007], on various quantities such as the measurements, covariances and/or state estimates. These consensus filters usually operate at a faster rate than the sampling rate, thus allowing the network to reach an agreement before the state estimate is updated. Under this assumption the choice of Kalman gain can be treated in the same way as a centralized Kalman filter. Recent papers in this area include [Olfati-Saber, 2007], [Spanos *et al.*, 2005] and [Xiao *et al.*, 2005]. In [Carli *et al.*, 2007] it was noted that if the assumption of agreement is not fulfilled the optimal Kalman gain for a centralized filter does not coincide with that of a distributed. This issue was also addressed in [Schizas *et al.*, 2007]. In [Speranzon *et al.*, 2006] the scalar case was studied under the assumption that nodes communicate only once between each measurement.

In [Alriksson and Rantzer, 2006] a two step procedure for distributed Kalman filtering was developed. This algorithm consists of one part that is done online and one offline part where parameters for the online part are selected. This paper aims at improving the parameter selection step of that paper.

This paper is organized as follows. In Section 3 we present the mathematical problem studied and give necessary assumptions. In Section 4 the online part of the algorithm is given for clarity. Section 5 presents the improved offline parameter selection process and in Section 6 three numerical examples are given.

3. Problem Formulation

Consider the following discrete-time linear system

$$x(k+1) = Ax(k) + w(k) \tag{1}$$

where $x(k) \in \mathbf{R}^n$ is the state of the system and $w(k) \in \mathbf{R}^n$ is a stochastic disturbance. The disturbance is assumed to be a white zero mean Gaussian process with covariance defined in (3).

The process is observed by N agents each with some processing and communication capability. The agents are labeled $i = 1, 2, \dots, N$ and form the set $\mathcal{V}$. The communication topology is modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the edge (i, j) is in $\mathcal{E}$ if and only if node i and node j can exchange messages. The nodes to which node i communicates are called neighbors and are contained in the set N_i. Note that node i is also included in the set N_i.

Each node observes the process (1) by a measurement $y_i(k) \in \mathbf{R}^{m_i}$ of the form

$$y_i(k) = C_i x(k) + e_i(k) \tag{2}$$

where $e_i(k) \in \mathbf{R}^{m_i}$ is a white zero mean Gaussian process. The process- and measurement disturbances are correlated according to

$$\mathbf{E}\begin{bmatrix} w(k) \\ e_1(k) \\ \vdots \\ e_N(k) \end{bmatrix}\begin{bmatrix} w(l) \\ e_1(l) \\ \vdots \\ e_N(l) \end{bmatrix}^T = \begin{bmatrix} R_w & 0 & \dots & 0 \\ 0 & R_{e11} & \dots & R_{e1N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & R_{eN1} & \dots & R_{eNN} \end{bmatrix}\delta_{kl} \tag{3}$$

where $\delta_{kl} = 1$ only if $k = l$. Note that this is a heterogeneous setup where each agent is allowed to to take measurements of arbitrary size and precision. Further the disturbances acting on the measurements are allowed to be correlated.

Each node is only allowed to communicate *estimates* with its neighbors and only once between each measurement. Further the only assumption made on the graph structure is that it has to be connected, other assumptions such as requiring it to be loop free are not necessary. No node is superior to any other and thus no central processing is allowed after deployment. This setup is somewhat different from the setup used in for example distributed control problems where each node in the graph also has dynamics associated with it. The reader should think of the problem studied here as for example a network of sensors trying to estimate the position of an external object they observe.

The goal is to make sure that every node in the network has a good estimate $\hat{x}_i(k)$ of the state $x(k)$.

4. Online Computations

The algorithm consists of the two traditional estimation steps measurement update and prediction, together with an additional step where the nodes communicate and merge estimates. We will refer to an estimate after measurement update as local and after the communication step as regional.

1. **Measurement update**
   The local estimate $\hat{x}_i^{local}(k|k)$ is formed by the predicted regional estimate $\hat{x}_i^{reg}(k|k-1)$ and the local measurement $y_i(k)$

   $$\hat{x}_i^{local}(k|k) = \hat{x}_i^{reg}(k|k-1) + K_i[y_i(k) - C_i\hat{x}_i^{reg}(k|k-1)] \tag{4}$$

   where K_i is computed off-line. The predicted estimate at time zero is defined as $\hat{x}_i^{reg}(0|-1) = \hat{x}_0$ where $\hat{x}_0$ is the initial estimate of $x(0)$.

2. **Merging**
   First the agents exchange their estimates over the communication channel. This communication is assumed to be error and delay free. The merged estimate $\hat{x}_i^{reg}(k|k)$ in node i is defined as a linear combination of the estimates in the neighboring nodes N_i.

   $$\hat{x}_i^{reg}(k|k) = \sum_{j\in N_i} W_{ij}\hat{x}_j^{local}(k|k) \tag{5}$$

The weighting matrices W_{ij} are computed off-line by the procedure described in Section 5.

3. **Prediction**
Because the measurement- and process noises are independent the prediction step only includes

$$\hat{x}_i^{reg}(k+1|k) = A\hat{x}_i^{reg}(k|k) \tag{6}$$

5. Offline Parameter Selection

As the best estimate will be available after the merging step we will focus on minimizing the estimation error covariance after this step. First, let the estimation error in node i be denoted

$$\tilde{x}_i(k|k) = x(k) - \hat{x}_i^{reg}(k|k) \tag{7}$$

and its (cross)covariance

$$P_{ij}^{reg}(k|k) = E\tilde{x}_i(k|k)\tilde{x}_j^T(k|k) \tag{8}$$

Next introduce the stacked estimation error

$$\tilde{x}(k|k) = \begin{bmatrix} \tilde{x}_1(k|k) \\ \vdots \\ \tilde{x}_N(k|k) \end{bmatrix} \tag{9}$$

with covariance $P^{reg}(k|k)$. Using (5) and requiring that

$$W_{ij} = 0 \quad \text{if } (i,j) \notin \mathcal{E} \tag{10}$$

the covariance after step 2) can be written as

$$P^{reg}(k|k) = WP^{local}(k|k)W^T \tag{11}$$

To keep the estimate unbiased we also need to require that

$$\sum_{j\in N_i} W_{ij} = I \qquad \forall i \in \mathcal{V} \tag{12}$$

The covariance $P^{local}(k|k)$ after step 1) can be expressed as

$$P^{local}(k|k) = [\,I \quad \tilde{K}\,]\,F\,[\,I \quad \tilde{K}\,]^T \tag{13}$$

where

$$F = \begin{bmatrix} I \\ -\tilde{C} \end{bmatrix} P^{reg}(k|k-1) \begin{bmatrix} I \\ -\tilde{C} \end{bmatrix}^T + \begin{bmatrix} 0 & 0 \\ 0 & R_e \end{bmatrix} \tag{14}$$

and

$$\tilde{K} = \begin{bmatrix} K_1 & & \\ & \ddots & \\ & & K_N \end{bmatrix} \quad \tilde{C} = \begin{bmatrix} C_1 & & \\ & \ddots & \\ & & C_N \end{bmatrix} \tag{15}$$

After step 3) each block of the covariance matrix is updated as

$$P_{ij}^{reg}(k|k-1) = AP_{ij}^{reg}(k-1|k-1)A^T + R_w \tag{16}$$

with $P_{ij}^{reg}(0|-1) = P_0$ where P_0 is the initial estimation error covariance. Equations (11), (13) and (16) form an iterative procedure for computing the steady state covariance as time approaches infinity for given values of W and $\tilde{K}$.

Ideally, we would like to find values W and $\tilde{K}$ that minimizes the steady state value of $\operatorname{tr} P^{reg}(k|k)$ as $k \to \infty$ subject to the constraints (10) and (12). This non-convex problem will be approximated in two steps. Instead of minimizing the steady state covariance directly, an approximate iterative procedure in analogy with the standard Kalman filter will be used.

Combining (13) and (11) the minimization problem to be solved in each iteration can be written as

$$\begin{array}{ll} \min\limits_{\tilde{K},W} & \operatorname{tr} W \, [\, I \quad \tilde{K} \,] \, F \, [\, I \quad \tilde{K} \,]^T \, W^T \\ \text{s.t} & \text{(10) and (12)} \end{array} \tag{17}$$

This problem will be solved using an alternating minimization type method.

The algorithm is divided into three steps. The first two steps correspond to the minimization problem (17) and in the third step the covariance is updated. The procedure is then iterated until convergence.

1. **K-step**

$$\tilde{K}(k) = \arg\min_{\tilde{K}} \operatorname{tr} W(k-1) \, [\, I \quad \tilde{K} \,] \, F \, [\, I \quad \tilde{K} \,]^T \, W^T(k-1)$$

2. **W-step**

For all $i \in \mathcal{V}$

$$\begin{array}{lll} W_{i\cdot}(k) = & \arg\min\limits_{W_{i\cdot}} & \operatorname{tr} W_{i\cdot} \, [\, I \quad \tilde{K}(k) \,] \, F \, [\, I \quad \tilde{K}(k) \,]^T \, W_{i\cdot}^T \\ & \text{s.t} & \text{(10) and (12)} \end{array}$$

3. **P-step**

$$\begin{aligned} P^{reg}(k|k) &= W(k)\,[\,I \quad \tilde{K}(k)\,]\,F\,[\,I \quad \tilde{K}(k)\,]^T\,W^T(k) \\ P^{reg}_{ij}(k+1|k) &= AP^{reg}_{ij}(k|k)A^T + R_w \qquad \forall i,j \in \mathcal{V} \end{aligned}$$

The algorithm is initialized with $P^{\text{reg}}_{ij}(0|-1) = P_0$ and $W(0) = I$. Note that in step 2) the minimization with respect to $W_{i\cdot}$, that is block row i, for each $i \in \mathcal{V}$ is equivalent to minimization with respect to the full matrix W.

Compared to the algorithm proposed in [Alriksson and Rantzer, 2006] the K-step now takes into account the fact that the estimates will be merged. Both the K- and W-step are quadratic minimization problems with explicit solutions which allows the algorithm to be applied to large scale systems.

5.1 The K-step

In this section the optimization problem from the K-step will be studied. To simplify notation time indices are dropped:

$$\min_{\tilde{K}} \operatorname{tr} W\,[\,I \quad \tilde{K}\,]\,F\,[\,I \quad \tilde{K}\,]^T\,W^T \tag{18}$$

In this section linear conditions that the optimal $\tilde{K}$ have to fulfill will be derived. First partition F as in

$$[\,I \quad \tilde{K}\,]\begin{bmatrix} F_{11} & F_{12} \\ F_{12}^T & F_{22} \end{bmatrix}[\,I \quad \tilde{K}\,]^T \tag{19}$$

To isolate the free parameters in $\tilde{K}$ it is expressed as a sum

$$\tilde{K} = \sum_{i=1}^{N} U_i^T K_i V_i \tag{20}$$

where

$$\begin{aligned} U_i &= [\,0_{n\times n(i-1)} \quad I_{n\times n} \quad 0_{n\times n(N-i)}\,] \\ V_i &= [\,0_{m_i\times l_i} \quad I_{m_i\times m_i} \quad 0_{m_i\times \tilde{l}_i}\,] \end{aligned} \tag{21}$$

$$l_i = \textstyle\sum_{j=1}^{i-1} m_j \quad \text{and} \quad \tilde{l}_i = \sum_{j=i+1}^{N} m_j \tag{22}$$

Using the decomposition (20) of $\tilde{K}$, conditions for optimality of (18) are given by

$$U_i W^T W\,[\,I \quad \tilde{K}\,]\begin{bmatrix} F_{12} \\ F_{22} \end{bmatrix} V_i^T = 0 \qquad \forall i \in \mathcal{V} \tag{23}$$

To simplify notation first introduce

$$G_{ij} = U_i W^T W U_j^T \quad \text{and} \quad H_{ij} = V_j F_{22} V_i^T \tag{24}$$

$$Q_i = U_i W^T W F_{12} V_i^T \tag{25}$$

Now (23) can be rewritten as

$$\textstyle\sum_{j=1}^{N} G_{ij} K_j H_{ij} = -Q_i \quad , \quad \forall i \in \mathcal{V} \tag{26}$$

To solve this set of matrix equations vectorization of the matrices will be used:

$$\sum_{j=1}^{N} (H_{ij}^T \otimes G_{ij}) \bar{K}_j = -\bar{Q}_i \tag{27}$$

where

$$\bar{K}_j = \text{vec}(K_j) \quad \text{and} \quad \bar{Q}_i = \text{vec}(Q_i) \tag{28}$$

This can be written in matrix form as

$$\begin{bmatrix} H_{11}^T \otimes G_{11} & \cdots & H_{1N}^T \otimes G_{1N} \\ \vdots & \ddots & \vdots \\ H_{N1}^T \otimes G_{N1} & \cdots & H_{NN}^T \otimes G_{NN} \end{bmatrix} \begin{bmatrix} \bar{K}_1 \\ \vdots \\ \bar{K}_N \end{bmatrix} = - \begin{bmatrix} \bar{Q}_1 \\ \vdots \\ \bar{Q}_N \end{bmatrix} \tag{29}$$

Thus we have derived linear equations for the optimal $\bar{K}_i$ which gives the optimal $\tilde{K}$.

5.2 W-Step

Introducing the sparsity constraint (10) is equivalent to removing rows and columns corresponding to weights that are required to be zero. Thus for each i the optimization problem can be written as

$$\begin{aligned} \min_{\tilde{W}} \quad & \text{tr}\, \tilde{W} \tilde{P} \tilde{W}^T \\ \text{s.t.} \quad & \tilde{W} e = I_n \end{aligned} \tag{30}$$

where $e = [\, I_n \dots I_n \,]^T$. Here $\tilde{W}$ contains the non-zero blocks of $W_{i\cdot}$ and $\tilde{P}$ the corresponding elements of the matrix $[\, I \quad \tilde{K}(k) \,]\, F \,[\, I \quad \tilde{K}(k) \,]^T$. Using Lagrange multipliers it can be shown, see [Sun and Deng, 2004], that conditions for optimality are

$$\underbrace{\begin{bmatrix} \tilde{P} & e \\ e^T & 0 \end{bmatrix}}_{G} \begin{bmatrix} \tilde{W}^T \\ \Lambda \end{bmatrix} = \begin{bmatrix} 0 \\ I_n \end{bmatrix} \tag{31}$$

The equation system (31) is in general underdetermined, so to get a unique solution the following minimization problem is introduced

$$\begin{aligned} \min_{\tilde{W}} \quad & \operatorname{tr} \tilde{W}\tilde{W}^T \\ \text{s.t.} \quad & (31) \end{aligned} \tag{32}$$

All solutions satisfying (31) can be parametrized in terms of V as

$$\begin{bmatrix} \tilde{W}^T \\ \Lambda \end{bmatrix} = \underbrace{G^\dagger \begin{bmatrix} 0 \\ I_n \end{bmatrix}}_{d} + G^0 V \tag{33}$$

where $G^\dagger$ denotes the Moore-Penrose pseudo inverse and G^0 a matrix of vectors spanning the null space of G. Now (32) can be rewritten as

$$\min_V \operatorname{tr} \begin{bmatrix} V \\ I \end{bmatrix}^T \begin{bmatrix} I & (G_1^0)^T d_1 \\ d_1^T G_1^0 & d_1^T d_1 \end{bmatrix} \begin{bmatrix} V \\ I \end{bmatrix} \tag{34}$$

where d_1 and G_1^0 are the parts corresponding to $\tilde{W}^T$. The solution to this unconstrained quadratic minimization problem is given by $V = -(G_1^0)^T d_1$. Thus the optimal $\tilde{W}$ is given by

$$\tilde{W} = d_1^T (I - G_1^0 (G_1^0)^T) \tag{35}$$

6. Numerical Examples

In this section three numerical examples will be studied. The first example is chosen to illustrate the performance improvement gained by modifying the K-step compared to the algorithm presented in [Alriksson and Rantzer, 2006]. The second example illustrates how varying the communication topology influences achieved performance. The third example illustrates an application where a scalar time varying signal, such as for example the temperature in an area, is measured by a sensor network.

6.1 Performance Comparison

Ideally, when comparing two suboptimal algorithms one would like to compare them to the results of the optimal solution. However in this case the $\tilde{K}$ and W yielding the optimal covariance P^{opt} can only be computed for very simple systems with special structure.

As a comparison we will use an observer scheme that relies on communication of measurements rather than estimates, but respects the imposed communication topology. For all nodes to be able to maintain an optimal estimate, they must have access to all measurements. The amount of communication required to achieve this greatly exceeds the required communication for the scheme in Section 4 and in most cases is practically impossible. This scheme will however yield a lower estimation error covariance P^{meas} than P^{opt} because having access to all measurements is clearly at least as good as having access to estimates generated by these measurements. Thus we have

$$P^{\text{meas}} \leq P^{\text{opt}} \leq P \tag{36}$$

where P refers to the scheme presented in Section 4.

As communication is only allowed to occur once every sample, the communication topology will impose a delay equal to the graph distance to a particular node. If measurement noise in different nodes is assumed independent, delayed measurements can be incorporated in the current estimate by extending the state space.

Note that the covariance will be different in different nodes due to the imposed communication topology. To evaluate overall performance the mean over all nodes in the network will be used.

In Figure 1 the relative performance $\frac{\operatorname{tr} P - \operatorname{tr} P^{\text{meas}}}{\operatorname{tr} P^{\text{meas}}}$ is plotted as a function of the process noise R_w for both weight selection algorithms. The performance was evaluated for 843 randomly generated second order systems with a communication topology described by graphs with 10 nodes and 5.95 neighbors on average. The shaded regions are 95% confidence intervals for the mean over all 843 systems. The measurement noise covariance matrix R_e was chosen as the identity matrix.

Because estimates are used as information carriers and communication is only allowed to take place once every sampling interval the process noise parameter R_w determines the effective distance from which a node collects information. Therefore one would expect the suboptimal solution to deteriorate as R_w is decreases, this is also confirmed by the results in Figure 1.

6.2 Connectivity Dependencies

The effects on estimation performance of 1620 randomly generated communication topologies with 20 nodes was studied for a system with integrator dynamics. As a measure of connectivity the average number of neighbors was used. An alternative measure would be the algebraic connectivity of the associated graph. Both these measures give similar results but the average number of neighbors is more intuitive.

As mentioned in Section 6.1 the effective radius from which information is used increases as R_w decreases. Therefore, choosing a small value,

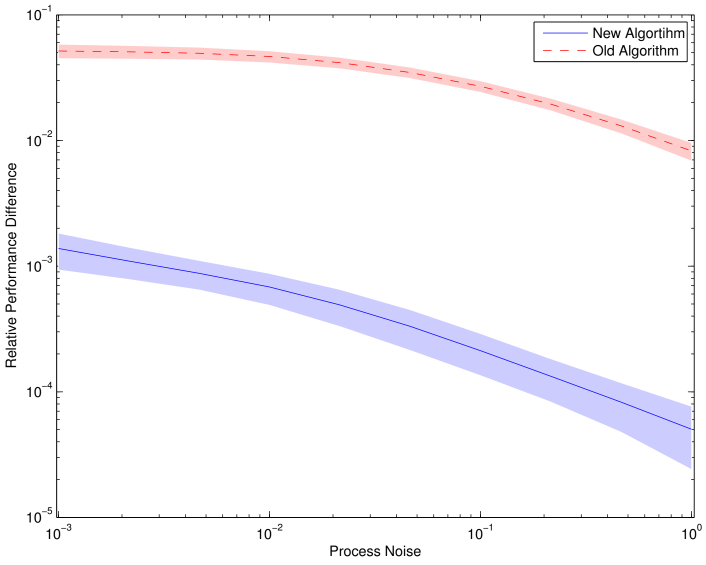


Figure 1. Comparison of the relative difference $\frac{\operatorname{tr} P - \operatorname{tr} P^{\text{meas}}}{\operatorname{tr} P^{\text{meas}}}$ for the algorithm presented in [Alriksson and Rantzer, 2006] and the one presented here for 843 randomly generated second order systems on a graph with 10 nodes with 5.95 neighbors on average. The shaded regions are 95% confidence intervals for the mean over all 843 systems.

such as $R_w = 0.001$, of the process noise parameter will make effects caused by different communication topologies more evident.

In Figure 2 the variance is plotted as a function of the average number of neighbors for each of the 1620 topologies using the algorithm presented in [Alriksson and Rantzer, 2006], the one presented in Section 5 and the scheme with delayed measurements presented in Section 6.1. The improvement compared to the previous weight selection algorithm is more evident for strongly connected graphs. However even for very sparse graphs the improvement is more than 50%.

6.3 Signal Tracking

This example aims at demonstrating how the proposed estimation scheme can be used in a situation where a sensor network is used to estimate the mean of a time varying signal in an area. Here 50 sensors are used to measure a signal described by

$$x(k) = \sin\left(\frac{2\pi}{100}k\right) + \sin\left(\frac{4\pi}{100}k\right)$$

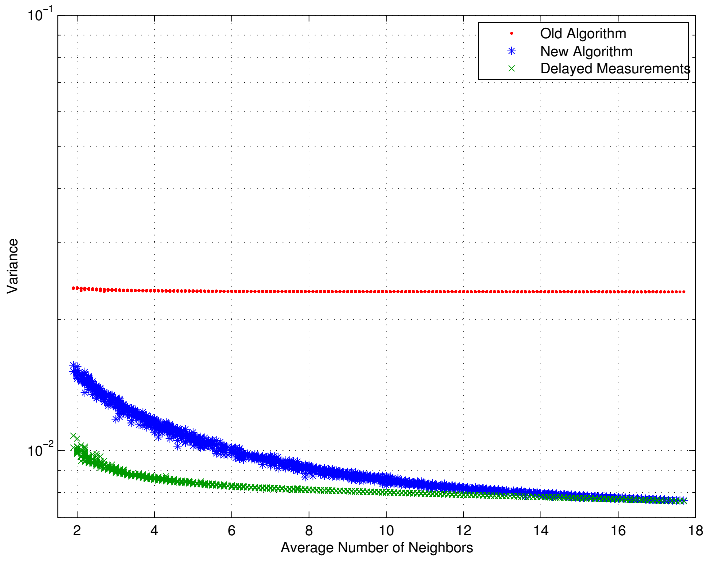


Figure 2. Variance plotted as a function of the average number of neighbors for 1620 randomly generated graphs of size 20 using the algorithm presented in [Alriksson and Rantzer, 2006], the one presented in Section 5 and the scheme with delayed measurements presented in Section 6.1.

Each node measures $x(k)$ corrupted by Gaussian white noise with unit variance. Further, the noise is assumed independent between nodes.

Two different signal models will be used: an integrator and a double integrator. The reason for not using a fourth order model capable of fully describing $x(k)$ is that in general, an exact model of the signal studied is hardly ever available.

Four different estimation schemes will be compared:

Centralized refers to a scenario where measurements are fused in a central node without any communication delay.

Delayed Measurements refers to the scenario described in Section 6.1.

Distributed refers to the scheme described in Section 4 with the weight selection procedure of Section 5.

Local refers to a scenario where no communication is used. Here each node runs a Kalman filter based on local information only.

As both the integrator and double integrator model differs from the true model describing $x(k)$ the choice of process noise covariance R_v is

Table 1. Optimal configuration for the four schemes.

	Signal Model	$\hat{R}_v$
Centralized	Double Integrator	$\begin{bmatrix} 0 & 0 \\ 0 & 0.001 \end{bmatrix}$
Delayed Measurements	Integrator	0.01
Distributed	Double Integrator	$\begin{bmatrix} 0 & 0 \\ 0 & 0.002 \end{bmatrix}$
Local	Integrator	0.09

crucial for the performance. Here two different ways of choosing R_v will be used. The first involves making a maximum likelihood estimate of the process noise covariance R_v for the centralized case and then using that estimate as the true value. In the case of a double integrator model the optimal ML-estimate is

$$\hat{R}_v = \begin{bmatrix} 0 & 0 \\ 0 & 0.001 \end{bmatrix}$$

and for the case of an integrator model $\hat{R}_v = 0.03$.

The second approach aims at making a fair comparison between the four schemes. To this end, both the process noise covariance and the model structure will be optimized to yield the best performance (measured as the root mean square (RMS) of the estimation error). The optimal configurations are summarized in Table 1.

The performance, measured as RMS of the estimation error, for the four different schemes in the three different model setups is presented in Figure 3. In the two middle cases the estimation performance will vary depending on which node is studied, this is represented as shaded boxes. As expected, using a more complex model generally improves performance except for the case with delayed measurements. A possible explanation for this is that to make use of old measurements the model must be used heavily, thus making the scheme very sensitive to modeling errors.

In the distributed case the double integrator model improves performance significantly. This shows the importance of allowing a more complex model structure than a simple first order model that is often assumed.

In Figure 4 typical trajectories are shown for the four different estimation schemes together with the true value of $x(k)$.

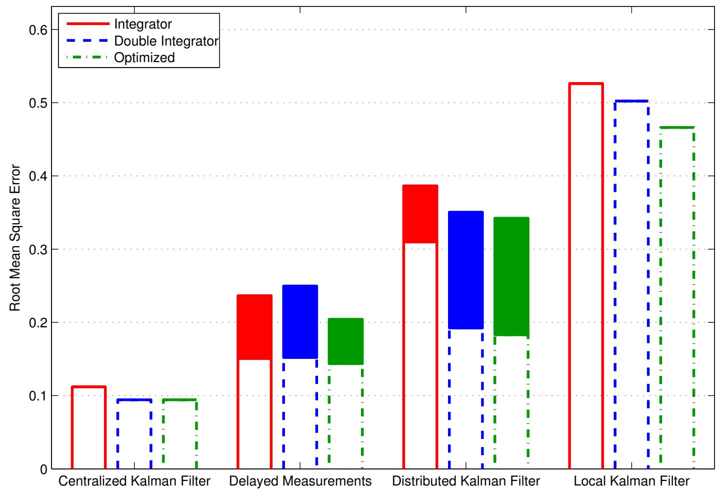


Figure 3. RMS of the estimation error, for the four different schemes in three different setups. In the cases referred to as integrator and double integrator, the process noise parameter was chosen as the ML-estimate without time delays. In the optimized case, both the model structure and parameters were optimized for the specific estimation scheme. The shaded regions represent max- and minimum values among all nodes.

7. Conclusions

In this paper an enhanced weight selection algorithm for the distributed Kalman filter algorithm presented in [Alriksson and Rantzer, 2006] has been presented. Improvements in terms of covariance reduction of several times have been noticed for the examples studied. The algorithm relies on the assumption that both the dynamics and communication topology are time-invariant and known at deployment. Slow variations in the communication topology and dynamics can be handled by recomputing the parameters on a regular basis. Fast variations in the communication topology can be treated as packet loss, against which the algorithm has proved robust.

Ideally both the weights for neighboring estimates, W, and local measurements, $\tilde{K}$, should be optimized jointly. However this is a non convex problem in general. Instead of a joint optimization in W and $\tilde{K}$, W is held constant equal to the value from the previous iteration when $\tilde{K}$ is optimized and $\tilde{K}$ is held constant while W is optimized. This reduces both optimization problems to quadratic optimization problems for which expressions in closed form are derived. Compared to the previous algorithm, fewer but bigger optimization problems are now solved.

The second contribution of this paper is to evaluate performance of the

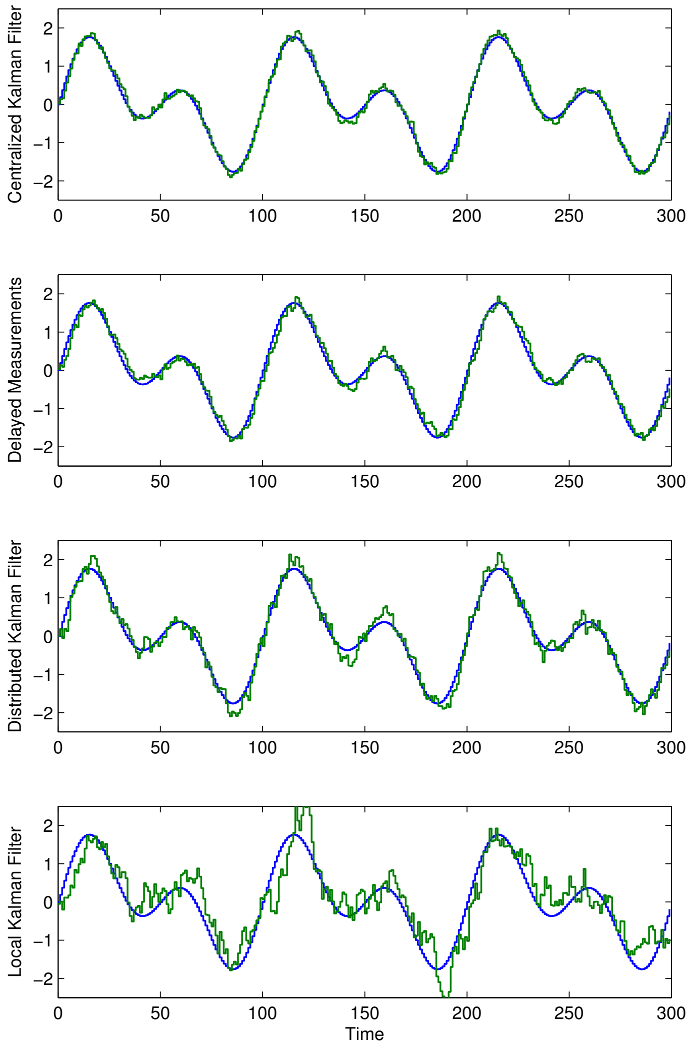


Figure 4. Typical trajectories for the four different estimation schemes together with the true value of $x(k)$.

estimation algorithm on a number of numerical examples. The first two numerical Monte Carlo studies conclude that a significant performance improvement has been gained through the new weight selection algorithm. In the third numerical example the importance of allowing a more complex signal model than for example the commonly used integrator model is highlighted.

Acknowledgement

This work was partially funded by the RUNES Integrated Project contract IST-2004-004536.

References

Alriksson, P. and A. Rantzer (2006): "Distributed Kalman filtering using weighted averaging." In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*. Kyoto, Japan.

Carli, R., A. Chiuso, L. Schenato, and S. Zampieri (2007): "Distributed Kalman filtering using consensus strategies." In *Proceedings of the 46th Conference on Decision and Control*, pp. 5486–5491. New Orleans, LA, USA.

Durrant-Whyte, H., B. Rao, and H. Hu (1990): "Toward a fully decentralized architecture for multi-sensor data fusion." *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1331–1336 vol.2.

Olfati-Saber, R. (2007): "Distributed Kalman filtering for sensor networks." In *Proceedings of the 46th Conference on Decision and Control*, pp. 5492–5498. New Orleans, LA, USA.

Olfati-Saber, R., J. A. Fax, and R. M. Murray (2007): "Consensus and cooperation in networked multi-agent systems." *Proceedings of the IEEE*, **95:1**, pp. 215–233.

Perkins, C., E. Belding-Royer, and S. Das (2003): "Ad hoc on-demand distance vector (AODV) routing."

Schizas, I. D., G. B. Giannakis, S. I. Roumeliotis, and A. Ribeiro (2007): "Anytime optimal distributed Kalman filtering and smoothing." *IEEE/SP 14th Workshop on Statistical Signal Processing, 2007. SSP '07*, pp. 368–372.

Spanos, D. P., R. Olfati-Saber, and R. M. Murray (2005): "Distributed sensor fusion using dynamic consensus." In *Proccedings of the 16th IFAC World Congress*.

Speranzon, A., C. Fischione, and K. H. Johansson (2006): "Distributed and collaborative estimation over wireless sensor networks." In *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 1025–1030. San Diego.

Sun, S.-L. and Z.-L. Deng (2004): “Multi-sensor optimal information fusion Kalman filter.” *Automatica*, **40:6**, pp. 1017–1023.

Xiao, L., S. Boyd, and S. Lall (2005): “A scheme for robust distributed sensor fusion based on average consensus.” In *Proceedings of the Information Processing for Sensor Networks (IPSN’05)*.

Paper II

Distributed Kalman Filtering: Theory and Experiments

Peter Alriksson and Anders Rantzer

Abstract

In this paper, a model based sensor fusion algorithm for sensor networks is presented. The algorithm extends the common Kalman filter with one step where nodes exchange estimates of the aggregated quantity with their neighbors, thus enhancing estimation performance. Under stationary conditions, an iterative parameter selection procedure is developed to minimize the stationary error covariance matrix of the estimate. A number of implementational aspects such as synchronization and packet loss are both formally analysed and investigated through experiments.

Submitted to IEEE Transactions on Control Systems Technology, 2008.

1. Introduction

The recent increase in battery and processing power of sensor nodes has made a wide range of sensing applications possible. However, as the number of sensors in a network increase, the need for efficient data aggregation becomes more and more evident. For a small sensor network, routing measurements to a central node using for example Ad hoc On Demand Distance Vector (AODV) routing [Perkins and Royer, 1999] might be feasible. However, as the network grows, the computational and network load both in the central node and in bottleneck nodes throughout the network will be a major problem. Also these nodes will drain their energy resources unnecessarily fast.

One way to approach this problem is through energy efficient routing algorithms, see for example [Akkaya and Younis, 2005] and [Rajagopalan and Varshney, 2006]. In addition to minimizing the required energy, a routing protocol can also try to route data in such a way that fusible data have a high probability to pass through the same node. These data pieces can then be fused, thus reducing the amount of data that need to be transmitted [Luo *et al.*, 2007].

Energy efficient routing algorithms often focus on problems where data flows from the network to one or a few so called sink nodes. When nodes are not only required to collect data but also to act on it, having a few sink nodes make all decisions might introduce long delays. In these situations it can be favourable to let all nodes have at least partial knowledge of the quantities upon which decisions are based. From now on we will refer to these quantities as the state. This setup is sometimes called distributed state estimation and is the topic of the rest of this paper.

1.1 Distributed State Estimation

The most obvious solution is of course to route all measurements to all nodes. Each node can then compute an optimal estimate of the state. However, in many situations this approach will use to much communication bandwidth and energy resources.

A different approach is to not route measurements through the network but to use consensus algorithms [Olfati-Saber *et al.*, 2007] to propagate measurement information. The first consensus-based distributed Kalman filters [Olfati-Saber, 2005] were based on the information form of a centralized Kalman filter. In these algorithms, consensus filters were used to distribute sensor and covariance data to all nodes and estimates were not exchanged. As was pointed out in [Carli *et al.*, 2007], unless the consensus filters run at a much higher rate than at which measurements are taken, this approach is only approximate. In [Olfati-Saber, 2007] an improved scheme where nodes use estimates from their neighbors was proposed.

The estimates are combined in such a way that the scheme is stable, but performance is not considered. The distributed state estimation problem has also been studied under the assumption that the state is constant [Xiao *et al.*, 2005] or slowly varying [Speranzon *et al.*, 2008].

Using state estimates as information carriers is not a new idea. Much of the early work on distributed state estimation has been done in the target tracking community where the term track-to-track fusion is often used. For a recent survey see [Smith and Singh, 2006]. Distributed state estimation in its most general form can be formulated in terms of probability distributions. Each node keeps a distribution representing its belief about the current state. This distribution is then updated using local measurements and the distributions kept in other nodes. Because nodes share information, the belief in two nodes will be partly based on the same measurements. To combine two distributions one must take the common information into account. This problem was solved in [Chong *et al.*, 1982] under the important assumption that "new" information in two nodes conditioned on the current state is independent. This assumption is fulfilled if either the state evolution is deterministic or if nodes keep exchanging distributions until all nodes have the same information before a new measurement is added. However, even if these assumptions are satisfied, computing the common information for a general communication topology with loops is very involved, see for example [Liggins *et al.*, 1997]. Another way to fulfill the assumptions in [Chong *et al.*, 1982] is to let every node not only have a belief about the current state, but the trajectory of previous states. This approach was used in [Rosencrantz *et al.*, 2003] where a distributed particle filter was developed.

As discussed above, the optimal distributed state estimation problem is in general very involved, thus much of the literature, including this paper, deals with approximate solutions. One way to approximate the problem is to use the solution from [Chong *et al.*, 1982] even if the assumptions under which is was derived are not fulfilled. This approach was used in for example [Grime *et al.*, 1992] and is expected to work well if the process noise is small and thus the state evolution is approximately deterministic.

An alternative approximation technique, which is the basis of this paper, is to combine estimates using a weighted linear combination. The weights are then optimized to yield a minimal error covariance matrix of the combined estimate. A solution in the case of two nodes was given in [Bar-Shalom and Campo, 1986] and later generalized to an arbitrary number of nodes in for example [Kim, 1994]. Note, however, that these papers only treat the problem of combining estimates, not how to generate estimates that when combined yield a small estimation error covariance.

1.2 Contributions

Contrary to most of the strategies described above, where all computations are distributed, the proposed scheme is divided into one deployment phase and one online phase.

In the online phase, nodes update their state estimates using a model, local measurements and estimates from their neighbors. How this is done is specified with a set of parameters for each node.

During the deployment phase, these parameters are optimized by a central node with global communication topology knowledge, thus reducing the amount of communication needed during the online phase. The parameter selection procedure not only addresses the problem of combining estimates, but also how to generate estimates that when combined yield a small estimation error. Optimizing parameters offline of course limits the proposed approach to situations where the communication topology and model parameters are constant or only slowly varying.

A preliminary version of the algorithm was published in [Alriksson and Rantzer, 2006] and later improved upon in [Alriksson and Rantzer, 2008]. The contributions in relation to those conference papers are twofold: it is shown how to incorporate knowledge about unreliable links in the parameter optimization problem and a number of implementational considerations such as node synchronization and timing-related packet loss are investigated.

1.3 Organization

The paper is organized as follows: In section 2 the problem is formally stated and some notation is introduced. Section 3 presents the online part of the proposed scheme. In Section 4 it is shown how the estimation error covariance evolves for a given set of parameters. These parameters are then optimized for both the case of ideal communication and lossy links. In Section 5 a simple communication protocol is developed, analyzed and evaluated through experiments. It is also shown that simulations of timing related packet losses coincide with experimentally observed behavior. Section 6 presents a signal tracking case study to illustrate the important performance vs power consumption tradeoff. Finally some concluding remarks are given.

2. Problem Formulation

Consider the following discrete-time linear system

$$x(k+1) = Ax(k) + w(k) \tag{1}$$

where $x(k) \in \mathbf{R}^n$ is the state of the system and $w(k) \in \mathbf{R}^n$ is a stochastic disturbance. The disturbance is assumed to be a white zero mean Gaussian process. Further, the initial state $x(0)$ is assumed Gaussian with mean $\check{x}_0$ and covariance P_0.

The process is observed by N agents, each with some processing and communication capability. The agents are labeled $i = 1, 2, \ldots, N$ and form the set $\mathcal{V}$. The communication topology is modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the edge (i, j) is in $\mathcal{E}$ if and only if node i and node j can exchange messages. The nodes to which node i communicates are called neighbors and are contained in the set N_i. Note that node i is also included in the set N_i.

Each node observes the process (1) by a measurement $y_i(k) \in \mathbf{R}^{m_i}$ of the form

$$y_i(k) = C_i x(k) + e_i(k) \tag{2}$$

where $e_i(k) \in \mathbf{R}^{m_i}$ is a white zero mean Gaussian process. The process- and measurement disturbances are correlated according to

$$\mathbf{E}\begin{bmatrix} w(k) \\ e_1(k) \\ \vdots \\ e_N(k) \end{bmatrix}\begin{bmatrix} w(l) \\ e_1(l) \\ \vdots \\ e_N(l) \end{bmatrix}^T = \mathbf{E}\begin{bmatrix} w(k) \\ e(k) \end{bmatrix}\begin{bmatrix} w(l) \\ e(l) \end{bmatrix}^T = \begin{bmatrix} R_w & 0 \\ 0 & R_e \end{bmatrix}\delta_{kl} \tag{3}$$

where $\delta_{kl} = 1$ only if $k = l$. Note that this is a heterogeneous setup where each agent is allowed to take measurements of arbitrary size and precision. Further, the disturbances acting on the measurements are allowed to be correlated.

Each node is only allowed to communicate *estimates* with its neighbors and only once between each measurement. No node is superior to any other and thus no central processing is allowed after deployment. The goal is to make sure that every node in the network has a good estimate $\hat{x}_i(k)$ of the state $x(k)$.

2.1 Notation

Throughout this paper we will make frequent use of matrices with equal size partitions on the form

$$W = \begin{bmatrix} W_{11} & \dots & W_{1N} \\ \vdots & \ddots & \vdots \\ W_{N1} & \dots & W_{NN} \end{bmatrix} = \begin{bmatrix} W_{1\cdot} \\ \vdots \\ W_{N\cdot} \end{bmatrix}$$

and block-diagonal matrices

$$\tilde{C} = \text{diag}(C_1, \dots, C_N) = \begin{bmatrix} C_1 & & \\ & \ddots & \\ & & C_N \end{bmatrix}$$

3. Online Computations

As mentioned in the introduction, performing optimal distributed state estimation in general requires nodes to not only keep a belief about the present state, but also about past states. How long this history has to be depends on the maximum delay between when a measurement is taken and when it has been fused in all nodes. Even in the linear Gaussian setting studied here, this would require nodes to exchange a potentially large amount of estimates and covariance information. All this communication will consume bandwidth and energy, perhaps more so than if all measurements would have been routed to all nodes.

Here we will instead, as stated in the problem formulation, only let nodes exchange estimates of the current state. Nodes will then form an estimate based on their neighbors' estimates using a weighted linear combination. Note that this procedure only requires the cross covariance between estimation errors in the involved nodes to be known. There is no need to keep track of common information which, in case of a general communication topology, can be very involved.

The proposed algorithm consists of the two traditional estimation steps measurement update and prediction, together with an additional step where the nodes communicate and merge estimates. We will refer to an estimate after measurement update as local and after the communication step as regional.

ALGORITHM 1—ONLINE COMPUTATIONS

1. **Measurement update**
   The local estimate $\hat{x}_i^{\text{local}}(k|k)$ is formed by the predicted regional estimate $\hat{x}_i^{\text{reg}}(k|k-1)$ and the local measurement $y_i(k)$

$$\hat{x}_i^{\text{local}}(k|k) = \hat{x}_i^{\text{reg}}(k|k-1) + K_i[y_i(k) - C_i\hat{x}_i^{\text{reg}}(k|k-1)] \tag{4}$$

   where $K_i \in \mathbf{R}^{n\times m_i}$ is computed offline using Algorithm 2. The predicted estimate at time zero is defined as $\hat{x}_i^{\text{reg}}(0|-1) = \check{x}_0$ where $\check{x}_0$ is the mean of $x(0)$.

2. **Merging**
   First nodes exchange estimates over the communication channel. The merged estimate $\hat{x}_i^{\text{reg}}(k|k)$ in node i is then formed as a linear combination of the estimates in the neighboring nodes N_i

$$\hat{x}_i^{\text{reg}}(k|k) = \sum_{j\in N_i} W_{ij}\hat{x}_j^{\text{local}}(k|k) \tag{5}$$

   The weighting matrices $W_{ij} \in \mathbf{R}^{n\times n}$ are computed offline using Algorithm 2.

3. **Prediction**
   Because the measurement- and process noises are uncorrelated the prediction step only includes

$$\hat{x}_i^{\text{reg}}(k+1|k) = A\hat{x}_i^{\text{reg}}(k|k) \tag{6}$$

□

Next the the problem of choosing the parameters K_i and W_{ij} will be addressed.

4. Offline Parameter Selection

Throughout this section will assume that the central node selecting the parameters K_i and W_{ij} have knowledge of the global communication topology and measurement models used in all nodes.

4.1 Covariance Evolution

Before describing the proposed parameter selection algorithm we need to establish how the estimation error covariance evolves when Algorithm 1 is executed. First introduce the joint estimation error

$$\tilde{x}^{\text{local}}(k|k) = \begin{bmatrix} I_n \\ \vdots \\ I_n \end{bmatrix} x(k) - \begin{bmatrix} \hat{x}_1^{\text{local}}(k|k) \\ \vdots \\ \hat{x}_N^{\text{local}}(k|k) \end{bmatrix}$$

with covariance $P^{\text{local}}(k|k) \in \mathbf{R}^{nN\times nN}$. Next define $\tilde{x}^{\text{reg}}(k|k)$, $P^{\text{reg}}(k|k)$, $\tilde{x}^{\text{reg}}(k+1|k)$ and $P^{\text{reg}}(k+1|k)$ in the same way. The evolution of these covariance matrices are summarized in the following lemma:

Lemma 1
If the weight matrix W fulfills

$$W_{ij} = 0 \quad \text{if } (i,j) \notin \mathcal{E} \tag{7}$$

and

$$\sum_{j\in N_i} W_{ij} = I_n \quad \forall i \in \mathcal{V} \tag{8}$$

executing Algorithm 1 results in a covariance of the joint estimation error that evolves as

$$P^{local}(k|k) = [\,I \quad \tilde{K}\,]\,F\,[\,I \quad \tilde{K}\,]^T \tag{9}$$

$$P^{reg}(k|k) = WP^{local}(k|k)W^T \tag{10}$$

$$P^{reg}_{ij}(k+1|k) = AP^{reg}_{ij}(k|k)A^T + R_w \quad \forall i,j \in \mathcal{V} \tag{11}$$

$$P^{reg}_{ij}(0|-1) = P_0 \quad \forall i,j \in \mathcal{V} \tag{12}$$

where

$$F = \begin{bmatrix} I \\ -\tilde{C} \end{bmatrix} P^{reg}(k|k-1) \begin{bmatrix} I \\ -\tilde{C} \end{bmatrix}^T + \begin{bmatrix} 0 & 0 \\ 0 & R_e \end{bmatrix}$$
$$\tilde{K} = \text{diag}(K_1,\dots,K_N)$$
$$\tilde{C} = \text{diag}(C_1,\dots,C_N)$$

Proof. See [Alriksson and Rantzer, 2008]. □

Note that even though Algorithm 1 is only an approximate solution, Lemma 1 describes the evolution of the true estimation error covariance using Algorithm 1.

The Kalman Filter. Towards an iterative parameter optimization algorithm first repeat how the covariance evolves in the case of a fully centralized Kalman filter:

$$P^{\text{central}}(k|k) = [\, I_n \quad K^{\text{central}} \,]\, F^{\text{central}}\, [\, I_n \quad K^{\text{central}} \,]^T \tag{13}$$

$$P^{\text{central}}(k+1|k) = AP^{\text{central}}(k|k)A^T + R_w \tag{14}$$

with

$$F^{\text{central}} = \begin{bmatrix} I_n \\ -C_1 \\ \vdots \\ -C_N \end{bmatrix} P^{\text{central}}(k|k-1) \begin{bmatrix} I_n \\ -C_1 \\ \vdots \\ -C_N \end{bmatrix}^T + \begin{bmatrix} 0 & 0 \\ 0 & R_e \end{bmatrix} \tag{15}$$

If the system is observable, the optimal K^{central} can be found by sequentially minimizing $P^{\text{central}}(k|k)$ while updating the covariance using (13)-(15). The key to this procedure is that the optimization can be done by completion of squares and thus the optimal $P^{\text{central}*}(k|k)$ is optimal in semi-definite sense $P^{\text{central}*}(k|k) \preceq P^{\text{central}}(k|k)$.

Inspired by the close resemblance between the Kalman filter iteration and Lemma 1 we now present an iterative approximate optimization procedure for determining the parameters W and $\tilde{K}$.

4.2 Optimization

Compared to the Kalman filter iteration (13)-(15) the iteration in Lemma 1 differs on a few key points. First of all there are two sets of parameters W and $\tilde{K}$. Because $\tilde{K}$ is block diagonal, $P^{\text{local}}(k|k)$ can not be minimized by completing squares and thus a scalar criteria needs to be introduced. This has the important implication that the optimal $P^{\text{local}}(k|k)$ will in general not be optimal in semi-definite sense. This in turn implies that $P^{\text{reg}}(k|k)$ should be optimized with respect to both W and $\tilde{K}$. Unfortunately the same holds for $P^{\text{reg}}(k+1|k)$ and consequently all future covariance matrices.

However, for many systems, sequential minimization of $\operatorname{tr} P^{\text{reg}}(k|k)$ with respect to both W and $\tilde{K}$ yields close to optimal performance. In [Alriksson and Rantzer, 2008] performance was evaluated for a large number of systems and communication topologies.

Performing the joint optimization with respect to W and $\tilde{K}$ for networks with many nodes is however intractable. Here we will instead use a heuristic based on the observation that W will not change too much between two iterations and thus the previous value can be used when minimizing with respect to $\tilde{K}$. This heuristic makes both optimization problems quadratic which allows them be solved efficiently.

The algorithm is divided into three steps: The first two steps correspond to minimizing $\operatorname{tr} P^{\text{reg}}(k|k)$ and in the third step the covariance is updated. The procedure is then iterated until convergence.

ALGORITHM 2—PARAMETER SELECTION

1. K**-step**

$$\tilde{K}(k) = \underset{\tilde{K}}{\arg\min} \quad \operatorname{tr} W(k-1)\,[\,I \quad \tilde{K}\,]\,F\,[\,I \quad \tilde{K}\,]^T\,W^T(k-1)$$

2. W**-step**
   For all $i \in \mathcal{V}$:

$$W_{i\cdot}(k) = \begin{cases} \underset{W_{i\cdot}}{\arg\min} & \operatorname{tr} W_{i\cdot}\,[\,I \quad \tilde{K}(k)\,]\,F\,[\,I \quad \tilde{K}(k)\,]^T\,W_{i\cdot}^T \\ \text{s.t} & \text{(7) and (8)} \end{cases}$$

3. P**-step**

$$\begin{aligned} P^{reg}(k|k) &= W(k)\,[\,I \quad \tilde{K}(k)\,]\,F\,[\,I \quad \tilde{K}(k)\,]^T\,W^T(k) \\ P^{reg}_{ij}(k+1|k) &= AP^{reg}_{ij}(k|k)A^T + R_w \qquad \forall i,j \in \mathcal{V} \end{aligned}$$

The algorithm is initialized with $P^{\text{reg}}_{ij}(0|-1) = P_0$ and $W(0) = I_{nN}$. □

Note that in step 2) the minimization with respect to $W_{i\cdot}$, that is block row i of W, for each $i \in \mathcal{V}$ is equivalent to minimization with respect to the full matrix W.

Both the K- and W-step are quadratic minimization problems with explicit solutions which allows the algorithm to be applied to large scale systems. Solution procedures for both the K- and W-steps are given in the appendix.

4.3 Packet Loss

When dealing with wireless networks, one must always keep in mind that wireless links are unreliable and packets are frequently lost. In the proposed scheme, lost packets has the consequence that estimates from all neighboring nodes might not be available in the merge step of Algorithm 1. In [Alriksson and Rantzer, 2007] experiments indicated that using a node's current estimate as a replacement is favorable compared to using the last received estimate. Therefore that strategy will also be used here.

First Lemma 1 will be modified to account for packet loss under the assumption that losses can be modeled as a sequence of independent events. It will then be shown how to adapt Algorithm 2 to this situation. Because the parameters W and $\tilde{K}$ are optimized offline, the optimization is restricted to the expected covariance with respect to all possible packet loss sequences. That is, a node is not allowed to change its parameters based on the observed packet loss sequence. We also demonstrate how performance can be analyzed using a more general Markov-chain packet-loss model.

Covariance Evolution. Here we will use a stochastic packet loss model where the probability of node i loosing a packet from node j is denoted Γ_{ij}. Further it is assumed that packet losses can be modelled as independent events, both in time and among nodes.

First let $\Omega_i = 2^{N_i \setminus i}$ denote the power set of all neighbors (excluding itself) to node i. For each element $\omega \in \Omega_i$, representing from which neighbors a packet was lost, define the replacement matrix

$$M_{kl}(\omega, i) = \begin{cases} I & (k \notin \omega \text{ and } k = l) \text{ or } (k \in \omega \text{ and } l = i) \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

Example 1
To illustrate the concepts described above consider the graph in Figure 1. For node 2 we for example have $N_2 = \{1,2,3\}$, $\Omega_2 = \{\{\},\{1\},\{3\},\{1,3\}\}$,

$$M(\{3\},2) = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & I & 0 \end{bmatrix} \quad \text{and} \quad M(\{1,3\},2) = \begin{bmatrix} 0 & I & 0 \\ 0 & I & 0 \\ 0 & I & 0 \end{bmatrix}$$

□

Because of the stochastic packet loss model, expectations are now taken also with respect to all possible packet loss sequences. The regional covariance can thus be written as

$$P_{ij}^{\text{reg}}(k|k) = \mathbf{E}\, P_{ij|\omega_1,\omega_2}^{\text{reg}} \tag{17}$$

where $P_{ij|\omega_1,\omega_2}^{\text{reg}}$ denotes the conditional regional covariance given that nodes i and j lost packets from the nodes in $\omega_1 \in \Omega_i$ and $\omega_2 \in \Omega_j$ respectively. Now using the replacement matrix $M(\omega, i)$ the conditional regional covariance can be expressed as

$$P_{ij|\omega_1,\omega_2}^{\text{reg}} = W_{i\cdot} M(\omega_1, i) P^{local}(k|k) M^T(\omega_2, j) W_{j\cdot}^T \tag{18}$$

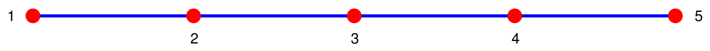


Figure 1. Graph used in Example 1 and Section 4.4.

Using the packet loss model, the expectation in (17) can be computed as

$$P_{ij}^{\text{reg}}(k|k) = \sum_{\omega_1 \in \Omega_i} \sum_{\omega_2 \in \Omega_j} P_{ij|\omega_1,\omega_2}^{\text{reg}} \alpha(\omega_1, \omega_2, i, j) \tag{19}$$

where $\alpha(\omega_1, \omega_2, i, j)$ denotes the probability of node i and j to only loose packets from the nodes in ω_1 and ω_2 respectively. This probability can be computed as

$$\begin{aligned} \alpha(\omega_1, \omega_2, i, j) &= \begin{cases} \tilde{\alpha}(\omega_1, i) & i = j \text{ and } \omega_1 = \omega_2 \\ 0 & i = j \text{ and } \omega_1 \neq \omega_2 \\ \tilde{\alpha}(\omega_1, i)\tilde{\alpha}(\omega_2, j) & i \neq j \end{cases} \\ \tilde{\alpha}(\omega, i) &= \prod_{k \in \omega} \Gamma_{ik} \prod_{k \in N_i \setminus i \setminus \omega} (1 - \Gamma_{ik}) \end{aligned} \tag{20}$$

Note that $\prod_{k \in \omega} \Gamma_{ik} = 1$ if ω is empty. Replacing (10) with (19) is thus sufficient to account for the proposed strategy used if packets are lost.

Optimization. When choosing $\tilde{K}$ and W it is desirable to take effects of packet loss into account. Because the merge equation (10) is the basis of both the K- and W-step in Algorithm 2, both these steps must be modified. The W-step is easily modified using (19) and requiring that $W_{i\cdot}$ does not depend on ω_1

$$W_{i\cdot} = \arg\min_{W_{i\cdot}} \operatorname{tr} W_{i\cdot} \left(\sum_{\omega_1 \in \Omega_i} \tilde{\alpha}(\omega_1, i) M(\omega_1, i) P^{local}(k|k) M^T(\omega_1, i) \right) W_{i\cdot}^T \tag{21}$$

Thus the problem is on the same form as the W-step in Algorithm 2, for which a solution procedure is given in Appendix A.2. In the K-step, the trace of the regional covariance matrix is minimized with respect to $\tilde{K}$ or equivalently

$$\min_{\tilde{K}} \sum_{i \in \mathcal{V}} \operatorname{tr} P_{ii}^{reg}(k|k) \tag{22}$$

Now, instead of using (10) to compute $P_{ii}^{reg}(k|k)$, it is replaced by (19). This minimization problem is also quadratic in $\tilde{K}$ and can be solved using a very similar procedure as the one presented in Appendix A.1.

Markov Chain Model. The assumption that packet loss can be modeled as a sequence of independent events allowed us to modify the weight selection procedure in Section 4. If a more detailed packet loss model such as a Markov chain is available, analyzing performance for a given set of weights is possible using Markov jump linear system theory as follows.

First let $\theta(k)$ denote the discrete state of a Markov chain representing which links that loose a packet at time k. Now the stacked error dynamics can be written as

$$\tilde{x}^{\text{reg}}(k+1|k) = A_{cl}(\theta(k))\tilde{x}^{\text{reg}}(k|k-1) + B_{cl}(\theta(k)) \begin{bmatrix} e(k) \\ w(k) \end{bmatrix} \tag{23}$$

where

$$\begin{aligned} A_{cl}(\theta) &= \tilde{A}\overline{W}(\theta)\left(I - \tilde{K}\tilde{C}\right) \\ B_{cl}(\theta) &= \begin{bmatrix} \tilde{A}\overline{W}(\theta)\tilde{K} & \begin{bmatrix} I_n \\ \vdots \\ I_n \end{bmatrix} \end{bmatrix} \\ \tilde{A} &= \text{diag}(A, \dots, A) \in \mathbf{R}^{nN \times nN} \end{aligned} \tag{24}$$

The effective weighting matrix $\overline{W}(\theta)$ can be constructed from W as

$$\overline{W}_{ij}(\theta) = \begin{cases} W_{ii} + \sum\limits_{l \in L(\theta,i)} W_{il} & i == j \\ 0 & j \in L(\theta, i) \\ W_{ij} & \text{otherwise} \end{cases} \tag{25}$$

where $L(\theta, i)$ is the set of nodes from which node i looses a packet when the Markov chain is in state θ.

The Markov chain model allows behaviors such as bursty and correlated losses between nodes to be modelled. The stationary covariance for a Markov jump linear system can be computed by solving a system of linear matrix equations, see for example [Costa *et al.*, 2005].

4.4 Numerical Example

In this section, a simple numerical example will be given to illustrate the proposed algorithm. The example consists of a small network with five nodes communicating according to the topology in Figure 1. The process parameters were chosen as $A = 1$ and $C_i = 1$. To illustrate how the weights depend on the communication topology, consider a situation where the

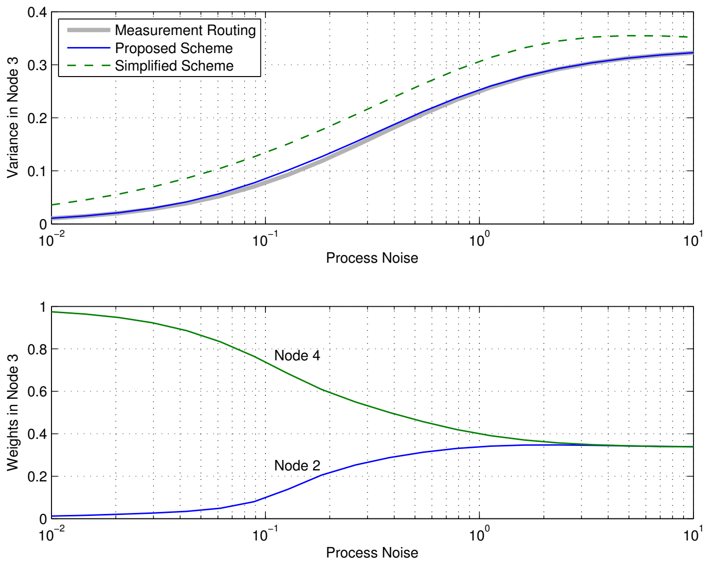


Figure 2. Upper: Variance for the proposed scheme, measurement routing and a simplified scheme that does not require global knowledge at the deployment phase. Lower: Weights W_{32} and W_{34} in Node 3 for the proposed scheme. When the process noise is small, estimates in Node 4 are favoured due to its proximity to Node 5, which as superior measurement accuracy.

measurement precision in one node, in this case Node 5, is superior to all other nodes. Here we for example set

$$R_e = \begin{bmatrix} I_4 & 0 \\ 0 & 0.001 \end{bmatrix}$$

In the upper part of Figure 2 the stationary estimation error variance $P_{33}^{\text{reg}}(k|k)$ in Node 3 is shown as a function of the process noise. As a comparison the variance for two other schemes are also plotted. In the first case, referred to as Measurement Routing, nodes run standard Kalman filters with measurements from all nodes as input. The measurements are, however, delayed according to the communication topology. This scheme serves as a lower bound on the achievable variance. In the second case, referred to as Simplified Scheme, the weights are chosen as

$$W_{ij} = \frac{1}{\|N_i\|} I_n$$

and K_i is chosen as the optimal K_i for a truly decentralized non-communicating Kalman filter. Note that this scheme does not require global knowledge

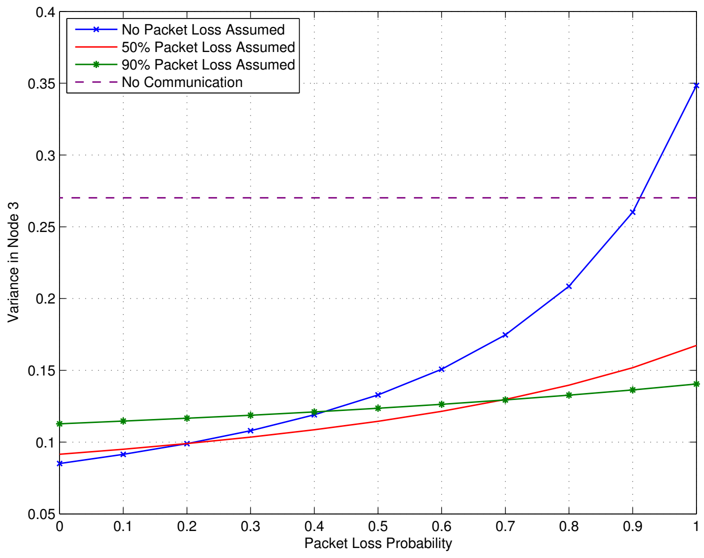


Figure 3. Variance in Node 3 as function of packet loss probability when parameters are optimized for three different packet-loss levels. The variance of a non-communicating estimator is plotted as a comparison.

about either the communication topology or measurement models during the deployment phase.

In the lower part of Figure 2 the weights W_{32} and W_{34} in Node 3 are shown as a function of the process noise variance R_w. If the process noise is small, Node 3 will favour estimates from Node 4 because its proximity to Node 5. If, however, the process noise is large, information originating in Node 5 will be too old before it reaches Node 3 and thus both W_{32} and W_{34} will be equal. This shows that Algorithm 2 indeed takes advantage of global communication topology and measurement model knowledge.

To illustrate how packet loss influences performance, the packet loss probability in the link between Node 4 and Node 5 was varied. Figure 3 shows the estimation error variance $P_{33}^{\text{reg}}(k|k)$ in Node 3 for various assumed packet loss probabilities. The process noise variance was chosen as $R_w = 0.1$ and all other links were left unaffected. Note that if perfect communication is assumed, performance deteriorates below what can be achieved without any communication. If, however, 50% packet loss is assumed, the performance degradation, both for high and low packet loss probabilities, is quite modest.

Table 1. Measured power consumption of a TMote Sky Mote. Note the small difference between listening and transmitting.

Mode	Power Consumption
Radio Off	9mW
Listening	60mW
Transmitting	62mW

5. Implementational Considerations

When designing algorithms for any sensor network, power consumption, timing and packet loss play central roles. Previous implementations of the proposed algorithm [Alriksson and Rantzer, 2007] have shown that estimation performance to a large extent can be evaluated offline using logged data as long as packet loss and timing effects are taken into account. Therefore the aim of this section is to first develop a simple communication protocol and then compare packet loss and timing simulations using this protocol to experimental data. All simulations are performed in TrueTime [Andersson *et al.*, 2005] which is a Matlab/Simulink based simulator that supports both real time kernels and different network protocols.

The cheap hardware often used in sensor networks usually has limited clock accuracy. Relative clock drifts among nodes of the same type of several milliseconds per hour is not uncommon. Also, when nodes are replaced, a restart of the entire network is not desirable, thus some kind of clock synchronization is needed. In the literature there are numerous clock synchronization protocols available (see e.g. [Carli *et al.*, 2008] and [Sadler and Swami, 2006]), however most of them rely on additional communication of for example clock differences. These protocols typically achieve perfect synchronization, which might not be necessary.

Contrary to what is often assumed, the power consumption of a sensor node that is listening for packets is of the same magnitude as when it is transmitting. For example, the approximate power consumption for a TMote Sky Mote [Sentilla Corporation, 2007] equipped with a IEEE 802.15.4 compliant transceiver running Contiki [Dunkels *et al.*, 2004] is presented in Table 1. This shows that when designing a communication protocol, minimizing the time a node spends listening for packets is of great importance.

In the literature there are numerous so called duty-cycled MAC protocols [Buettner *et al.*, 2006] that try to minimize power consumption, but often at the cost of increased latency. Here we will instead develop a communication protocol that handles duty-cycling at the application level, thus avoiding unknown latencies.

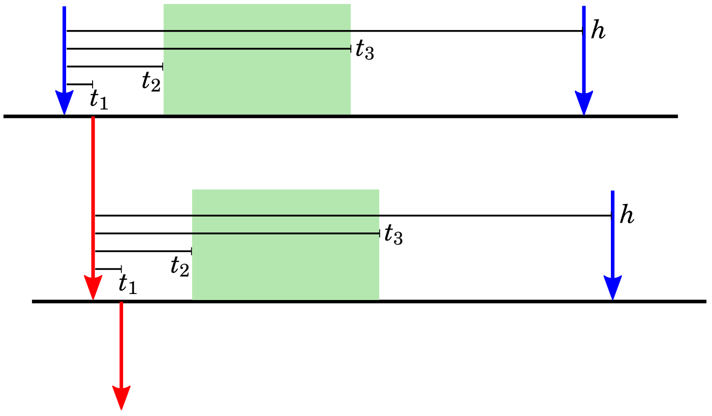


Figure 4. Illustration of the communication protocol in Section 5.1. Note how the second node is triggered by the first node. The shaded region illustrates the time period when the radio is turned off.

5.1 Communication Protocol

The aim of this section is to develop and evaluate a simple communication protocol that ensures synchronized sampling while trying to minimize packet loss and power consumption. The protocol is described in pseudo code below:

```
waitUntil(packetReceived())
loop
  t=time
  sampleProcess()
  updateEstimateWithMeasurement()
  waitUntil(time>t+t1)
  broadcastEstimate()
  waitUntil(time>t+t2)
  radioOff()
  mergeEstimate()
  predictEstimate()
  waitUntil(time>t+t3)
  radioOn()
  waitUntil(packetReceived() or time>t+h)
end
```

It is assumed that radio packets are received, processed and saved by a high priority process, like an interrupt handler, during all times when the radio is on. The network is initialized by inserting an external packet to trigger the first node, which then triggers its neighbors and so on.

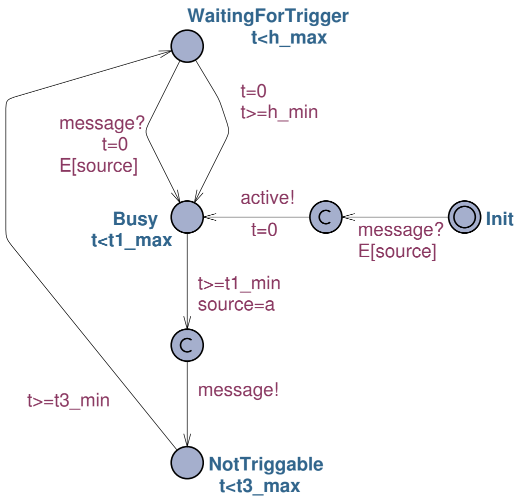


Figure 5. UPPAAL description of the possible behavior regarding synchronization for one node.

The protocol has three parameters t_1, t_2, t_3 together with the sampling interval h. The first parameter t_1 is the time a node waits before broadcasting its estimate. The second parameter t_2 is the time a node spends collecting estimates from its neighbors before turning the radio off. This time should be chosen greater than $2t_1$ to ensure that a node does not miss any packets from its neighbors. The third parameter t_3 determines how long the radio is turned off. In Figure 4 the protocol is illustrated for the case of two nodes, where the first node triggers the second.

One simple observation is that the sampling spread, that is the time difference between when the first and last node samples the process, is upper bounded by t_1 times the diameter of the graph (the maximum graph distance between any two nodes) or the sampling interval h, whichever is smallest. However, in typical sensor network applications the sampling period is very long compared to t_1, thus the sampling spread will in general be small relative to the sampling interval even for graphs with a large diameter.

Synchronization Analysis. Verifying properties for all possible situations and communication topologies is an overwhelming task. However formal verification tools like UPPAAL [Behrmann *et al.*, 2004] makes it possible to detect many design flaws. The UPPAAL description language is a non-deterministic guarded command language with data types. Figure 5 shows a simplified UPPAAL description of the *possible* behavior of *one* node.

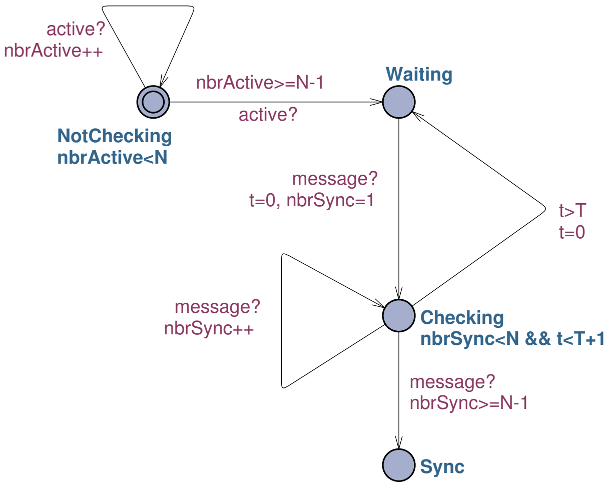


Figure 6. UPPAAL description of the property that all nodes must sample the process within T time units from when the first node samples the process.

This description is then instantiated N times with the node identifier a replaced with consecutive numbers $1, \ldots, N$.

Note that only aspects relevant to synchronization issues are modelled. For example as long as t_2 is smaller than t_3 it does not effect the synchronizing behavior of the network, thus it is excluded from the description. Also note that the timing parameters are specified in terms of their upper and lower limits which allows for non-determinism. The incidence matrix `E` is used to represent which nodes that can hear when a node broadcasts a message. An edge marked for example `message?` will only fire if the corresponding edge marked `message!` fires. A condition such as `t<h_max` associated with a location must be true for the automaton to be allowed to remain in that location. Conditions associated with edges force them to only fire if their conditions are true.

To verify a property, a desired behavior must also be specified. When for example the automaton in Figure 6 is in the `Sync`-state, N nodes have sampled the process within T time units. Using UPPAAL it is then possible to verify that all possible executions of the N copies of the automaton in Figure 5 will drive the automaton in Figure 6 to the `Sync`-state

Note however that the verification is only valid for one particular communication topology and parameter set. Yet useful insight of the communication protocol can be gained by examining various topologies and parameter sets. For example if $t_{3max} < h_{min} - c_1 t_{1min} - c_2 t_{1max} - c_3$ the network will synchronize for all possible communication topologies involv-

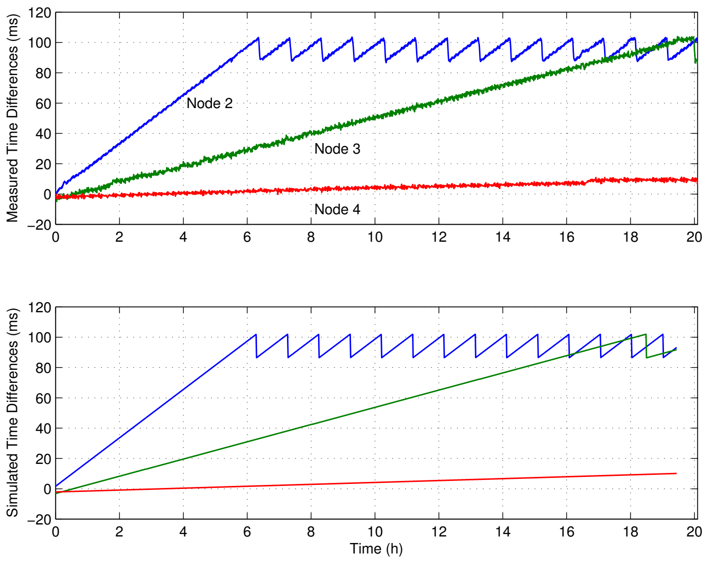


Figure 7. Measured and simulated difference between when nodes sample the process. Each line represents the difference between when nodes 2, 3 and 4 sample the process compared to Node 1. Clock drifts range from 0 to 16ms/h. The sawtooth behavior is due to time quantization.

ing 4 nodes. The constants c_1, c_2 and c_3 depend on the communication topology, but are small compared to h. To avoid that a node is triggered more than once within the sampling interval, t_{3min} must be chosen large enough. The exact limit depends on the communication topology, but if $t_{3min} > Nt_{1max}$ this situation can be avoided.

5.2 Experimental Validation

To investigate the behavior of the proposed communication protocol a small experimental testbed with four nodes, all within communication range, was used. The testbed consisted of four TMote Sky Motes [Sentilla Corporation, 2007] equipped with IEEE 802.15.4 compliant transceivers. The relative clock drifts among the nodes ranged from 0 to 16ms/h.

Timing Behavior. In Figure 7 both the simulated and measured difference between when different nodes sample the process are shown. The three lines represent the difference between when nodes 2, 3 and 4 sample the process compared to Node 1. Initially all four nodes sample the process within a few milliseconds from each other. However, due to the relative clock drifts synchronization is gradually lost. In this experiment t_1 was

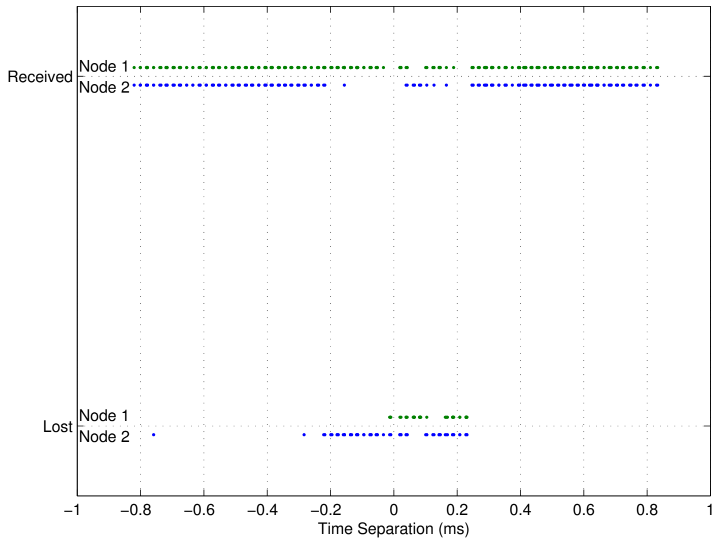


Figure 8. Received and lost packets in Node 3 as function of the transmission time separation between Node 1 and Node 2. A time separation higher than approximately 0.25ms gives virtually no packet loss, whereas the opposite results in a very unpredictable behavior.

chosen to 100ms which has the effect that when two nodes sample the process more than 100ms apart, for example after approximately 6 hours, the slowest one is triggered by a packet from the fastest one. However, due to a time quantization of 15.625ms the next sampling interval of the slowest node will be slightly shorter and thus it will not be triggered by the fastest node the next time. This results in the sawtooth behavior that can be seen in Figure 7.

Timing-Related Packet Loss. The packet loss probability in real world applications depend on many things, among which effects of multipath propagation, external disturbances and timing of packet transmissions dominate. Because the first two are out of control when designing a communication protocol, we will focus on the latter.

The IEEE 802.15.4 protocol specifies the use of CSMA/CA (Carrier sense multiple access with collision avoidance), so as long as two nodes can hear each other, the application layer in these two nodes should be able to request a transmission at the same time without the risk of collision. However, experiments show that the packet loss probability increases significantly if packet transmission times are not separated enough. In Figure 8 the packet reception in one node is shown when two other nodes

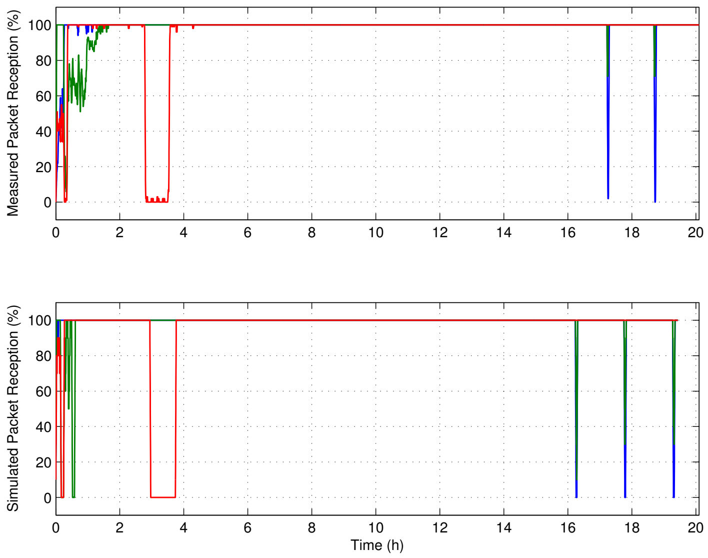


Figure 9. Measured and simulated packet reception in Node 1. Note the periods of almost no packet reception when nodes transmission times are not separated enough. For example between 3 and 4 hours nodes 1 and 4 interfere and after 16 hours nodes 2 and 3 interfere a number of times. The simulation and measurement mismatch is mostly due to differences in clock drifts and starting times.

are transmitting a 14 byte packet once a second. The packet reception in the third node becomes very unpredictable if transmissions times are not separated by more than approximately 0.25ms.

Taking the CSMA/CA behavior into account, Figure 9 shows both measured and simulated packet reception in Node 1 for the same time interval as as in Figure 7. Note that when packet transmission times are not separated enough the packet reception probability drops to almost zero. For example between 3 and 4 hours nodes 1 and 4 interfere and after 16 hours nodes 2 and 3 interfere a number of times. The length of these intervals depend on the relative clock drifts among the nodes.

Both figures 7 and 9 show that in some situations a timing based tool like TrueTime can be used to quite accurately predict the behavior of a real system.

Having long periods of poor packet reception is if course not desirable. One simple way of avoiding this is to randomize the parameter t_1, that is the delay between sampling the process and transmitting the estimate.

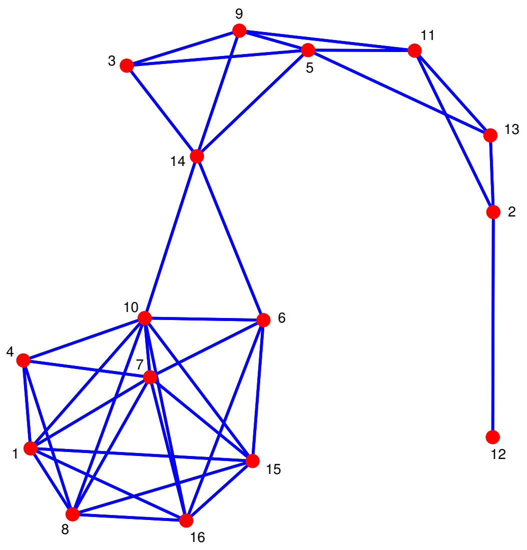


Figure 10. Communication topology used in the case study.

6. Case Study

In this section it will be demonstrated how the proposed estimation scheme can be used in a situation where a sensor network is used to estimate the spatial mean of a time varying signal in an area. Here 16 sensors are used to measure a signal described by

$$x(t) = \sin\left(\frac{2\pi}{100}t\right) + \sin\left(\frac{4\pi}{100}t\right)$$

Each node measures $x(t)$ corrupted by Gaussian white noise with unit variance. Further, the noise is assumed independent between nodes. The nodes are placed randomly using a uniform distribution and can communicate if they are are closer than a maximum distance. Minimizing this distance while keeping the graph connected results in the communication topology shown in Figure 10.

Two different signal models will be used: an integrator and a double integrator. The reason for not using a fourth order model capable of fully describing $x(t)$ is that in general, an exact model of the signal studied is hardly ever available.

Because none of the two models can describe the signal perfectly, the process noise covariance parameter R_w for the two models was estimated

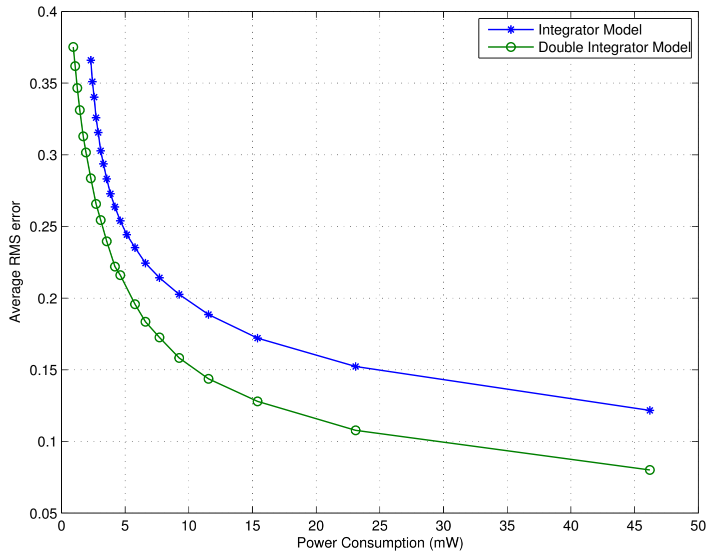


Figure 11. Tradeoff between power consumption and performance. The power consumption is changed by changing the sampling interval h.

from data using the maximum likelihood method. Note that the optimal values of R_w will depend on the sampling interval.

In sensor networks one of the most important tradeoffs is the one between performance and power consumption. As mentioned in Section 5 the dominating state in terms of energy consumption is when a node is listening for radio packets. When using the communication protocol described in Section 5.1 the radio can be turned off during long periods, except in the beginning and end of a sampling interval. As a result, the average power consumption can be approximated by the following simple model

$$P_{\text{average}} \approx P_{\text{off}} + \frac{(P_{\text{on}} - P_{\text{off}})(t_2 + t_3')}{h} \quad h \geq t_2 + t_3' \tag{26}$$

Here P_{off} is the idle power consumption, P_{on} is the average power consumption when the radio is turned on, $t_3' = h - t_3$ is the time the radio is turned on at the end of a sampling interval and t_2 is the time the radio is on at the beginning of each sampling interval. Note that t_2 and t_3' do not depend on h.

In Figure 11 the performance of the distributed algorithm, measured as the mean RMS error among all 16 nodes, is plotted as function of P_{average} for both signal models. The different power levels where achieved by varying the sampling interval in a range from 1s to 50s. Numerical values for the constants in (26) are P_{on}=62mW, P_{off}=8.7mW, t_2=125ms and t'_3=625ms, which represent typical values from a TMote Sky [Sentilla Corporation, 2007] running Contiki [Dunkels *et al.*, 2004]. Note that the qualitative behavior does not depend on the particular values of these constants. The benefit of using a more complex signal model is apparent for all power levels.

7. Conclusions

In this paper a data aggregation algorithm for sensor networks was presented. The algorithm extends the common Kalman filter with one step where nodes exchange *estimates* of the aggregated quantity with their neighbors. Exchanging estimates instead of measurements can potentially reduce the required number of messages exchanged per unit time, thus prolonging battery life.

Under stationary conditions, an approximate iterative parameter selection procedure that aims at minimizing the stationary covariance matrix of the estimated quantity was developed. The procedure will in general not find the global optimum, but numerical experiments show that it in general performs well.

A number of implementational aspects such as synchronization and packet loss were both formally analyzed and investigated through experiments. Finally, a signal tracking case study was used to illustrate the important tradeoff between performance and power consumption.

Acknowledgement

This work was partially funded by the RUNES Integrated Project contract IST-2004-004536 and the Swedish research council.

A. Appendix

A.1 The K-step

In this section the optimization problem from the K-step will be studied. To simplify notation, time indices are dropped:

$$\min_{\tilde{K}} \quad \operatorname{tr} W [\, I \quad \tilde{K} \,] F [\, I \quad \tilde{K} \,]^T W^T \tag{27}$$

First partition F as in

$$[\, I \quad \tilde{K} \,] \begin{bmatrix} F_{11} & F_{12} \\ F_{12}^T & F_{22} \end{bmatrix} [\, I \quad \tilde{K} \,]^T \tag{28}$$

To isolate the free parameters in $\tilde{K}$ it is expressed as a sum

$$\tilde{K} = \sum_{i=1}^{N} U_i^T K_i V_i \tag{29}$$

where

$$\begin{aligned} U_i &= [\, 0_{n \times n(i-1)} \quad I_{n \times n} \quad 0_{n \times n(N-i)} \,] \\ V_i &= [\, 0_{m_i \times l_i} \quad I_{m_i \times m_i} \quad 0_{m_i \times \tilde{l}_i} \,] \end{aligned} \tag{30}$$

$$l_i = \textstyle\sum_{j=1}^{i-1} m_j \quad \text{and} \quad \tilde{l}_i = \textstyle\sum_{j=i+1}^{N} m_j \tag{31}$$

Using the decomposition (29) of $\tilde{K}$, conditions for optimality of (27) are given by

$$U_i W^T W [\, I \quad \tilde{K} \,] \begin{bmatrix} F_{12} \\ F_{22} \end{bmatrix} V_i^T = 0 \qquad \forall i \in \mathcal{V} \tag{32}$$

To simplify notation first introduce

$$G_{ij} = U_i W^T W U_j^T \quad \text{and} \quad H_{ij} = V_j F_{22} V_i^T \tag{33}$$

$$Q_i = U_i W^T W F_{12} V_i^T \tag{34}$$

Now (32) can be rewritten as

$$\textstyle\sum_{j=1}^{N} G_{ij} K_j H_{ij} = -Q_i \quad , \quad \forall i \in \mathcal{V} \tag{35}$$

To solve this set of matrix equations vectorization of the matrices will be used.

$$\sum_{j=1}^{N} (H_{ij}^T \otimes G_{ij}) \bar{K}_j = -\bar{Q}_i \tag{36}$$

where

$$\bar{K}_j = \text{vec}(K_j) \quad \text{and} \quad \bar{Q}_i = \text{vec}(Q_i) \tag{37}$$

This can be written in matrix form as

$$\begin{bmatrix} H_{11}^T \otimes G_{11} & \cdots & H_{1N}^T \otimes G_{1N} \\ \vdots & \ddots & \vdots \\ H_{N1}^T \otimes G_{N1} & \cdots & H_{NN}^T \otimes G_{NN} \end{bmatrix} \begin{bmatrix} \bar{K}_1 \\ \vdots \\ \bar{K}_N \end{bmatrix} = - \begin{bmatrix} \bar{Q}_1 \\ \vdots \\ \bar{Q}_N \end{bmatrix} \tag{38}$$

Thus we have derived linear equations for the optimal $\bar{K}_i$ which gives the optimal $\tilde{K}$.

A.2 W-Step

Introducing the sparsity constraint (7) is equivalent to removing rows and columns corresponding to weights that are required to be zero. Thus for each i the optimization problem can be written as

$$\begin{aligned} \min_{\tilde{W}} \quad & \text{tr } \tilde{W}\tilde{P}\tilde{W}^T \\ \text{s.t.} \quad & \tilde{W}e = I_n \end{aligned} \tag{39}$$

where $e = [\, I_n \dots I_n \,]^T$. Here $\tilde{W}$ contains the non-zero blocks of $W_{i\cdot}$ and $\tilde{P}$ the corresponding elements of the matrix $[\, I \quad \tilde{K}(k) \,]\, F\, [\, I \quad \tilde{K}(k) \,]^T$. Using Lagrange multipliers it can be shown, see [Sun and Deng, 2004], that conditions for optimality are

$$\underbrace{\begin{bmatrix} \tilde{P} & e \\ e^T & 0 \end{bmatrix}}_{G} \begin{bmatrix} \tilde{W}^T \\ \Lambda \end{bmatrix} = \begin{bmatrix} 0 \\ I_n \end{bmatrix} \tag{40}$$

The equation system (40) is in general underdetermined, so to get a unique solution the following minimization problem is introduced

$$\begin{aligned} \min_{\tilde{W}} \quad & \text{tr } \tilde{W}\tilde{W}^T \\ \text{s.t.} \quad & (40) \end{aligned} \tag{41}$$

All solutions satisfying (40) can be parametrized in terms of V as

$$\begin{bmatrix} \tilde{W}^T \\ \Lambda \end{bmatrix} = \underbrace{G^\dagger \begin{bmatrix} 0 \\ I_n \end{bmatrix}}_{d} + G^0 V \tag{42}$$

where $G^{\dagger}$ denotes the Moore-Penrose pseudo inverse and G^0 a matrix of vectors spanning the null space of G. Now (41) can be rewritten as

$$\min_{V} \operatorname{tr} \begin{bmatrix} V \\ I \end{bmatrix}^T \begin{bmatrix} I & (G_1^0)^T d_1 \\ d_1^T G_1^0 & d_1^T d_1 \end{bmatrix} \begin{bmatrix} V \\ I \end{bmatrix} \tag{43}$$

where d_1 and G_1^0 are the parts corresponding to $\tilde{W}^T$. The solution to this unconstrained quadratic minimization problem is given by $V = -(G_1^0)^T d_1$. Thus the optimal $\tilde{W}$ is given by

$$\tilde{W} = d_1^T (I - G_1^0 (G_1^0)^T) \tag{44}$$

References

Akkaya, K. and M. Younis (2005): "A survey on routing protocols for wireless sensor networks." *Ad Hoc Networks*, **3:3**, pp. 325–349.

Alriksson, P. and A. Rantzer (2006): "Distributed Kalman filtering using weighted averaging." In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*. Kyoto, Japan.

Alriksson, P. and A. Rantzer (2007): "Experimental evaluation of a distributed Kalman filter algorithm." In *Proceedings of the 46th IEEE Conference on Decision and Control*. New Orleans, LA.

Alriksson, P. and A. Rantzer (2008): "Model based information fusion in sensor networks." In *Proceedings of the 17th IFAC World Congress*. Seoul, Korea.

Andersson, M., D. Henriksson, A. Cervin, and K.-E. Årzén (2005): "Simulation of wireless networked control systems." In *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*. Seville, Spain.

Bar-Shalom, Y. and L. Campo (1986): "The effect of the common process noise on the two-sensor fused-track covariance." *IEEE Transactions on Aerospace and Electronic Systems*, **AES-22:6**, pp. 803–805.

Behrmann, G., A. David, and K. G. Larsen (2004): "A tutorial on UPPAAL." In Bernardo and Corradini, Eds., *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, number 3185 in LNCS, pp. 200–236. Springer–Verlag.

Buettner, M., G. Yee, E. Anderson, and R. Han (2006): “X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks.” In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, pp. 307– 320. Boulder, Colorado, USA.

Carli, R., A. Chiuso, L. Schenato, and S. Zampieri (2007): “Distributed Kalman filtering using consensus strategies.” In *Proceedings of the 46th Conference on Decision and Control*, pp. 5486–5491. New Orleans, LA, USA.

Carli, R., A. Chiuso, L. Schenato, and S. Zampieri (2008): “A PI consensus controller for networked clocks synchronization.” In *Proceedings of the 17th IFAC World Congress*.

Chong, C. Y., S. Mori, E. Tse, and R. P. Wishner (1982): “Distributed estimation in distributed sensor networks.” *American Control Conference*, **19**, pp. 820–826.

Costa, O. L. V., M. Fragoso, and R. Marques (2005): *Discrete-time Markov jump linear systems*. Probability and its applications (New York). Springer, cop.

Dunkels, A., B. Grönvall, and T. Voigt (2004): “Contiki - a lightweight and flexible operating system for tiny networked sensors.” In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*. Tampa, Florida, USA.

Grime, S., H. F. Durrant-Whyte, and P. Ho (1992): “Communication in decentralized data-fusion systems.” In *In Proc. American Control Conference*, pp. 3299–3303.

Kim, K. H. (1994): “Development of track to track fusion algorithms.” In *Proceedings of the American Control Conferance*, pp. 1037–1041.

Liggins, M., C.-Y. Chong, I. Kadar, M. Alford, V. Vannicola, and S. Thomopoulos (1997): “Distributed fusion architectures and algorithms for target tracking.” *Proceedings of the IEEE*, **85:1**, pp. 95–107.

Luo, H., Y. Liu, and S. Das (2007): “Routing correlated data in wireless sensor networks: A survey.” *IEEE Network*, **21:6**, pp. 40–47.

Olfati-Saber, R. (2005): “Distributed Kalman filter with embedded consensus filters.” In *Proceedings of the 44th IEEE Conference on Decision and Control, and European Control Conference*.

Olfati-Saber, R. (2007): “Distributed Kalman filtering for sensor networks.” In *Proceedings of the 46th Conference on Decision and Control*, pp. 5492–5498. New Orleans, LA, USA.

Olfati-Saber, R., J. A. Fax, and R. M. Murray (2007): "Consensus and cooperation in networked multi-agent systems." *Proceedings of the IEEE*, **95:1**, pp. 215–233.

Perkins, C. and E. Royer (1999): "Ad-hoc on-demand distance vector routing." *Proceedings of the second IEEE workshop on Mobile Computing Systems and Applications, WMCSA '99.*, Feb, pp. 90–100.

Rajagopalan, R. and P. Varshney (2006): "Data-aggregation techniques in sensor networks: A survey." *IEEE Communications Surveys and Tutorials*, **8:4**, pp. 48–63.

Rosencrantz, M., G. Gordon, and S. Thrun (2003): "Decentralized sensor fusion with distributed particle filters." In *Proc. Conf. Uncertainty in Artificial Intelligence, Acapulco, Mexico.*

Sadler, B. M. and A. Swami (2006): "Synchronization in sensor networks: An overview." In *IEEE MILCOM.*

Sentilla Corporation (2007): *TMote Sky*. `http://www.sentilla.com`.

Smith, D. and S. Singh (2006): "Approaches to multisensor data fusion in target tracking: A survey." *IEEE Transactions on Knowledge and Data Engineering*, **18:12**, pp. 1696–1711.

Speranzon, A., C. Fischione, K. Johansson, and A. Sangiovanni-Vincentelli (2008): "A distributed minimum variance estimator for sensor networks." *IEEE Journal on Selected Areas in Communications*, **26:4**, pp. 609–621.

Sun, S.-L. and Z.-L. Deng (2004): "Multi-sensor optimal information fusion Kalman filter." *Automatica*, **40:6**, pp. 1017–1023.

Xiao, L., S. Boyd, and S. Lall (2005): "A scheme for robust distributed sensor fusion based on average consensus." In *Proceedings of the Information Processing for Sensor Networks (IPSN'05).*

Paper III

Experimental Evaluation of a Distributed Kalman Filter Algorithm

Peter Alriksson and Anders Rantzer

Abstract

This paper evaluates the performance of a distributed Kalman filter applied to an ultrasound based positioning application with seven sensor nodes. By distributed we mean that all nodes in the network desires an estimate of the full state of the observed system and there is no centralized computation center after deployment. Communication only takes place between neighbors and only once each sampling interval. The problem is solved by communicating estimates between neighbors and then forming a weighted average as the new estimate. The weights are optimized to yield a small estimation error covariance in stationarity. The minimization can be done off line thus allowing only estimates to be communicated. In the experimental setup the distributed solution performs almost as good as a centralized solution. The proposed algorithm also proved very robust against packet loss.

1. Introduction

As battery and processing power of nodes in sensor networks increases the possibility of more intelligent estimation schemes become more and more important. The use of sensor networks was first driven by military applications, but with cheaper technology many other areas could make use of sensor networks, see for example [Hall and Llinas, 1997] and [Viswanathan and Varshney, 1997]. Advantages with wireless sensor networks typically include increased flexibility and more robustness, as more than one unit is performing the same task.

However these advantages come with a cost: When communicating over wireless channels packet loss becomes a major problem and decentralized algorithms tend to be more complex than centralized solutions.

To illustrate the pros and cons a target tracking application will be considered. In simple target tracking applications, the task is to estimate the position of an external object. In some situations measurements are taken from spatially separated locations and an estimate is needed at each location. Ideally all measurements should be used at all locations, however this may require high bandwidth communication channels between all nodes.

To reduce the required bandwidth a distributed solution where only the position estimate is communicated among neighbors will be considered.

The problem where estimates are communicated has been given great attention in the literature. In for example [Durrant-Whyte *et al.*, 1990] a decentralized Kalman filter was proposed. However, this algorithm requires every node to be able to communicate with every other node, which might not be possible. An alternative approach is to only allow nodes to communicate with their neighbors. As opposed to the case where measurements are communicated no routing is required when estimates are used as information carriers.

Without direct communication between all nodes a new problem is introduced, namely how to combine estimates from just neighboring nodes. To optimally combine two estimates one has to know the mutual information between them. Computing this quantity for a general communication graph is a difficult task, that requires global knowledge of the topology. In the case of a loop-free graph the problem was solved in [Grime *et al.*, 1992] by introduction of a channel filter This approach was used in a coordinated search strategy application, see [Bourgault and Durrant-Whyte, 2004]. The problem has also been studied intensively in the dynamic consensus literature, see for example [Olfati-Saber *et al.*, 2007] and the references therein. In [Speranzon *et al.*, 2006] a similar problem was studied, but for scalar systems.

This paper is organized as follows. Section 2 and 3 gives a brief overview of the theory used in the rest if the paper. The main part of the paper constitutes of sections 4, 5 and 6 where the experimental setup together with the results are presented.

1.1 Target Tracking

In real target tracking applications sophisticated radar systems are used to take measurements of the position of a target moving in three dimensions. Here a simplified setup is used where the target to be tracked is a mobile robot moving in two dimensions.

Localization of mobile robots can be performed with a number of techniques. In laboratory experiments it is common to use vision, e.g., a ceiling-mounted camera combined with an image-processing system. Another possibility is dead-reckoning using a high-precision inertial measurement unit on board the robot. A problem with dead reckoning-based approaches, however, is that they do not use feedback and thus unmeasurable disturbances will cause position errors that cannot be compensated for. In an outdoor environment GPS would be another possibility.

The localization approach chosen here is based on ultrasound. The basic idea is to transmit a wireless radio packet simultaneously with an ultrasound pulse from each sender node. The receiver nodes measure the difference in time of arrival between the radio packet and the ultrasound pulse and can in this way calculate their distance to the sender node. By combining, or fusing, several distance measurements an estimate of the position can be obtained.

Two main approaches exist, [Smith *et al.*, 2004]. In an *active mobile* system the infrastructure, in this case the sensor network, has receivers at known locations, which estimate distances to a mobile device based on active transmissions from the device. These distances are then reported to a central node for processing. Examples of this approach are the Active Badge [Want *et al.*, 1992], and the Ubisense [Cadman, 2003] systems. In a *passive mobile* system, instead, the infrastructure has active beacons at known positions that periodically transmits signals to a passive mobile device. The mobile device then use these signals to compute its current location. The most famous example of this is the Cricket system [Priyantha *et al.*, 2000].

An advantage of the active approach is that it is more likely to perform accurate tracking than the passive approach. The passive approach, on the other hand, scales better with the number of mobile devices. Since the main objective here is for the sensor network to handle the localization, an approach similar to the active one was chosen. However here no central computation center is used.

As the robot has a practically unlimited power supply compared to the nodes in the sensor network, it is reasonable to assume that the robot can reach all nodes in the network with high probability. Thus all nodes can measure the distance to the robot at each sampling time and the robot can transmit its expected movement to the sensor network. The nodes however operate under severe power restrictions, thus only neighbor to neighbor communication is possible.

2. Mathematical Formulation

The motion of the robot is described by a discrete-time linear model of the following form, derived in [Alriksson, 2007].

$$x(k+1) = Ax(k) + Bu(k) + v(k) \tag{1}$$

Here $x(k) \in \mathbf{R}^n$ denotes the state of the system, $u(k) \in \mathbf{R}^m$ a known input and $v(k) \in \mathbf{R}^n$ a stochastic disturbance. The disturbance is assumed to be a white zero mean Gaussian process with covariance defined below. Note that in this simplified setup, it is assumed that the external input $u(k)$ is known to all nodes. As mentioned in Section 1.1 this assumption is satisfied in the experimental setup.

The process is observed by N agents each with some processing and communication capability. The agents are labeled $i = 1, 2, \ldots, N$ and form the set V. The communication topology is modeled as a graph $G = (V, E)$, where the edge (i, j) is in E if and only if node i and node j can exchange messages. The nodes to which a node communicates are called neighbors and are contained in the set N_i. Note that node i is also included in the set N_i.

Each node observes the process (1) by a measurement $y_i(k) \in \mathbf{R}^{p_i}$ of the following form

$$y_i(k) = C_i x(k) + e_i(k) \tag{2}$$

where $e_i(k) \in \mathbf{R}^{p_i}$ is a white zero mean Gaussian process. The measurement- and process disturbances are correlated according to

$$E \begin{bmatrix} v(k) \\ e_1(k) \\ \vdots \\ e_N(k) \end{bmatrix} \begin{bmatrix} v(l) \\ e_1(l) \\ \vdots \\ e_N(l) \end{bmatrix}^T = \begin{bmatrix} R_v & 0 & \ldots & 0 \\ 0 & R_{e11} & \ldots & R_{e1N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & R_{eN1} & \ldots & R_{eNN} \end{bmatrix} \delta_{kl} \tag{3}$$

where $\delta_{kl} = 1$ only if $k = l$. Note that this is a heterogeneous setup where each agent is allowed to to take measurements of arbitrary size and precision. Further the disturbances acting on the measurements are allowed to be correlated.

Each node is only allowed to communicate with its neighbors and only once between each measurement. Further the only assumption made on the graph structure is that it is connected, other assumptions such as requiring it to be loop free are not necessary. No node is superior to any other and thus after deployment no central processing is allowed.

The goal is to make sure that every node in the network has a good estimate $\hat{x}_i(k)$ of the state $x(k)$.

3. Distributed Kalman Filter

When constructing an algorithm based on estimates instead of measurements care must be taken on how to combine estimates in a good way. The problem is that estimates in different nodes are not independent, as they contain the same process noise, and possibly also the same measurement information. To optimally combine two estimates the mutual information must be subtracted.

For a graph with loops, two nodes can not compute the mutual information by just using local information. Information can for example travel from node A to node C and then to node B. When node A and B are to compute their mutual information they may not be aware of the coupling through C.

To solve the problem for a general communication topology neighboring estimates are weighted so that the error covariance of the merged estimate is minimized. This approach will not give the optimal solution in general, but is applicable to graphs with loops. Weighted averaging can be seen as a generalization of the two-sensor track-fusion algorithm presented in [Bar-Shalom and Campo, 1986]. There is great freedom when choosing both how to weigh local measurements and neighboring estimates. In [Alriksson and Rantzer, 2006] a procedure aimed at minimizing the covariance of the estimation error is presented, but other objectives such as minimizing the amount of communication for a given accuracy could also be considered.

3.1 On-Line Computations

The algorithm consists of the two traditional estimation steps, measurement update and prediction together with an additional step where the nodes communicate and merge estimates. We will refer to an estimate after measurement update as local and after the communication step as regional.

1. **Measurement update**
   The local estimate $\hat{x}_i^{local}(k|k)$ is formed by the predicted regional estimate $\hat{x}_i^{reg}(k|k-1)$ and the local measurement $y_i(k)$

   $$\hat{x}_i^{local}(k|k) = \hat{x}_i^{reg}(k|k-1) + K_i[y_i(k) - C_i\hat{x}_i^{reg}(k|k-1)]. \tag{4}$$

   where K_i is computed off-line using for example the procedure presented in [Alriksson and Rantzer, 2006]. The predicted estimate at time zero is defined as $\hat{x}_i^{reg}(0|-1) = \hat{x}_0$ where $\hat{x}_0$ is the initial estimate of $x(0)$.

2. **Merging**
   First the agents exchange their estimates over the communication channel. This communication is assumed to be error and delay free. The merged estimate $\hat{x}_i^{reg}(k|k)$ in node i is defined as a linear combination of the estimates in the neighboring nodes N_i.

   $$\hat{x}_i^{reg}(k|k) = \sum_{j\in N_i} W_{ij}\hat{x}_j^{local}(k|k) \tag{5}$$

   The weighting matrices W_{ij} could for example be chosen using the procedure described in [Alriksson and Rantzer, 2006].

3. **Prediction**
   Because the measurement- and process noises are independent the prediction step only includes

   $$\hat{x}_i^{reg}(k+1|k) = A\hat{x}_i^{reg}(k|k) + Bu(k) \tag{6}$$

4. Experimental Setup

To generate measurements, corresponding to (2), an ultrasound based system together with trilateration will be used.

The stationary sensor nodes are each equipped with an ultrasound receiver and the mobile robot is equipped with an ultrasound transmitter. The stationary sensor nodes are implemented as Tmote Sky sensor network nodes together with a small ultrasound receiver circuit interfaced to the node via an AD converter, see Figure 1.

Both the ultrasound transmitters and receivers are designed to be isometric, i.e., to transmit and receive in the full 360^0 degree plane.

The robot used in the experiments is a dual-drive robot developed in Lund. It is equipped with three Atmel AVR Mega16 processors and one TMote Sky node, see Figure 1. Two AVR processors are used to control the

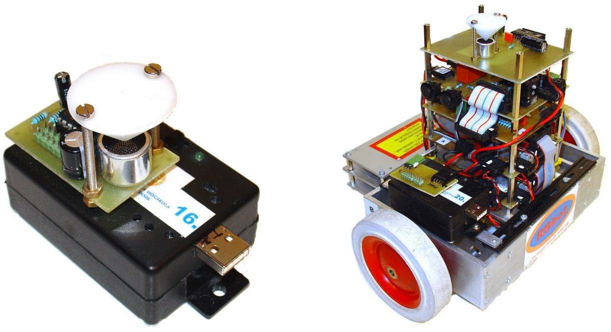


Figure 1. Stationary sensor network node with ultrasound receiver circuit and robot with ultrasound sender. The nodes are packaged in a plastic box to reduce wear.

wheel speeds using PI-controllers. The remaining AVR is used to generate ultrasound. For a detailed description of the hardware see [Årzén *et al.*, 2007]. Throughout all experiments the robot is controlled using a wireless joystick.

4.1 Ultrasound Based Localization

The implemented localization method works according to the following principles. At the beginning of each measurement cycle, the robot transmits a broadcast radio message to alert the nodes of the incoming ultrasound pulse. After a fixed time the robot then emits an ultrasound pulse. When the radio message reaches the node, it starts to sample the ultrasound microphone. Then the stationary nodes detect the beginning of the pulse using a moving median filter of length three.

The sample index where the pulse was detected is proportional to the distance between the stationary nodes and the robot when the pulse was emitted. If the speed of sound, the sampling interval in the nodes and the fixed delay between ultrasound- and radio transmission are known the actual distance can be computed. The position can then be computed using trilateration.

4.2 Trilateration

Trilateration is a method to find the position of an object based on distance measurements to three objects with known positions. In three dimensions the problem has two solutions, however the correct one can usually be determined from physical considerations. The basic problem is to find a solution $[p_x \quad p_y \quad p_z]^T$ to the following three nonlinear equations

$$\begin{aligned}
(p_x - p_{x1})^2 + (p_y - p_{y1})^2 + (p_z - p_{z1})^2 &= d_1^2 \\
(p_x - p_{x2})^2 + (p_y - p_{y2})^2 + (p_z - p_{z2})^2 &= d_2^2 \\
(p_x - p_{x3})^2 + (p_y - p_{y3})^2 + (p_z - p_{z3})^2 &= d_3^2.
\end{aligned}$$

where p_{xi}, p_{yi} and p_{zi} are known positions of the nodes and d_i is the distance from node i to the object to be positioned. The problem can be transformed to a system of two linear equations and one quadratic equation by e.g subtracting the second and third equation from the first, see [Manolakis, 1996] for a detailed analysis.

An alternative more geometric approach was taken in [Thomas and Ros, 2005] where the problem is solved using Cayley-Menger determinants. This approach has the benefit of a geometric interpretation of the solution in terms of volumes, areas and distances. Also the error analysis with respect to e.g distance errors is simplified.

As the robot is assumed to only move in the xy-plane, the problem can be reduced to a set of two linear equations. The two linear equations will always have a solution unless all three known points are positioned on a line. The two linear equations define two lines, see Figure 2, which can be represented as

$$a_0 y = a_1 + a_2 x \tag{7}$$

$$b_0 y = b_1 + b_2 x \tag{8}$$

where

$$\begin{aligned}
a_0 &= 2(p_{y2} - p_{y1}) \\
a_1 &= d_1^2 - d_2^2 + p_{y2}^2 - p_{y1}^2 + p_{x2}^2 - p_{x1}^2 - 2p_z(p_{z2} - p_{z1}) \\
a_2 &= 2(p_{x1} - p_{x2}) \\
b_0 &= 2(p_{y3} - p_{y1}) \\
b_1 &= d_1^2 - d_3^2 + p_{y3}^2 - p_{y1}^2 + p_{x3}^2 - p_{x1}^2 - 2p_z(p_{z3} - p_{z1}) \\
b_2 &= 2(p_{x1} - p_{x3}).
\end{aligned}$$

Note that the z-coordinate p_z of the robot is assumed to be known, as it is only moving in the xy-plane. The intersection point of these two lines constitute the trilaterated position $[p_x^{\text{tri}} \quad p_y^{\text{tri}}]^T$. Even though the three circles do not intersect in one point, the algorithm still provides a reasonable result. For a detailed discussion on how errors both in distance measurements and node positions influence the trilateration result, see [Manolakis, 1996] and [Thomas and Ros, 2005].

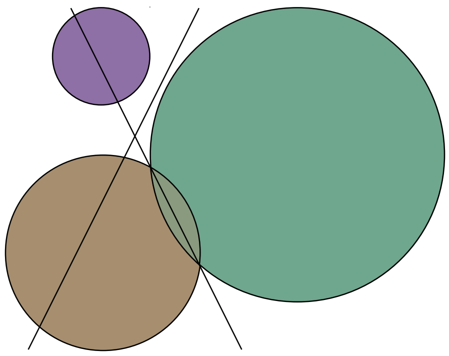


Figure 2. Two lines defining the solution to the trilateration problem. Each line corresponds to one pair of circles. For clarity only two of the three possible lines are shown. This setup shows both overlapping and non-overlapping circles. Sensor nodes are located at the center of each circle.

4.3 Camera Based Localization

To evaluate the distributed ultrasound based localization system an independent localization system is needed. In the experimental setup a camera based system was used. The robot was equipped with two markers to aid the vision system. The camera system consists of one fixed mounted camera with a resolution of 640×480 pixels. Each marker is located in the image using an algorithm based on a Harris corner detector, see [Harris and Stephens, 1988]. If the robot is assumed to move in a plane, an image coordinate can be transformed to a point p^{cam} in the plane using a linear transformation. The heading can then be computed using the two markers. In the experimental setup used, the vision based localization system had an accuracy of approximately 1cm.

4.4 Choice of Noise Covariance Matrices

The choice of measurement- and process covariance matrices (R_v and R_e) are crucial to the performance of the algorithm. The process noise covariance matrix determines the confidence in the model. Here R_v will be used as a tuning parameter to trade off between noise rejection and trust in the model.

The measurement noise covariance matrix R_e in the experimental setup is a symmetric 14×14 matrix, thus it has 105 free parameters. The somewhat standard choice of a diagonal matrix does not apply here as the trilaterated position measurements are highly correlated. Instead, using a test trajectory, R_e was estimated as a normalized version of

$$(\hat{R}_e)_{ij} = \frac{1}{T}\sum_{k=0}^{T-1}(p_i^{tri}(k) - p^{cam}(k))(p_j^{tri}(k) - p^{cam}(k))^T \tag{9}$$

where $p_i^{tri}(k)$ is the trilaterated position in node i at time k and $p^{cam}(k)$ is the position generated from the vision system. The diagonal elements of $\hat{R}_e$ are thus the squared RMS-value of the trilateration error. The advantage of using squared RMS-values instead of for example the variance is that systematic errors are reflected in the RMS-value. As trilateration is a nonlinear operation, the error is dependent of the position. Thus the result of (9) is dependent on the specific trajectory the the robot has followed. Ideally one would want to allow R_e to vary with time, however this would make the weights W time varying thus making it more complicated to compute them off-line.

To illustrate the correlation pattern let us define the RMS-correlation matrix as

$$\rho_{ij} = \frac{(\hat{R}_e)_{ij}}{\sqrt{(\hat{R}_e)_{ii}(\hat{R}_e)_{jj}}} \tag{10}$$

In Figure 3 elements associated with the x-position for a typical trajectory are shown for both measurements (left) and simulations (right). The correlation pattern caused by trilateration is clearly visible both in the experimental and simulated data.

5. Communication Protocol

As discussed in Section 1.1 it is assumed that the robot can reach all nodes in the network with radio packets. Thus the robot constitutes a global clock which simplifies the communication protocol.

As a single node can only measure the distance to the robot, nodes need to form groups of three to be able to perform trilateration. One node in each group collects distance measurements from the other two and then computes the position of the robot using trilateration. To reduce utilized bandwidth, distance measurements and position estimates are transmitted in the same package. The protocol implemented is illustrated by the schedule in Figure 4.

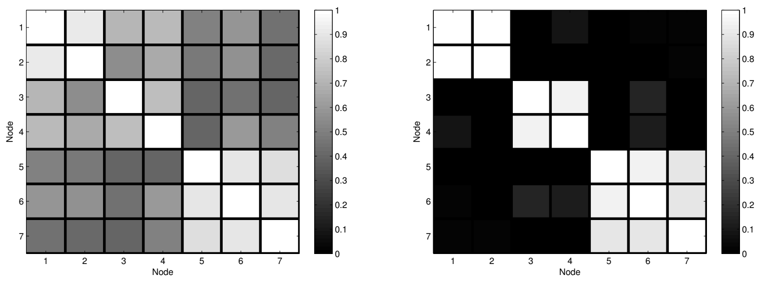


Figure 3. Elements of the RMS-correlation matrix ρ associated with the x-position for measurements (left) and simulations (right). The correlation pattern introduced by trilateration is clearly visible.

1. At the beginning of each period the robot asks the wheel controllers for the current wheel velocities.
2. The robot sends a broadcast message to all nodes with its expected movement, that is previous heading estimate and wheel velocities. This corresponds to the term $Bu(k)$ in (1).
3. After sending the packet the robot updates its heading estimate based on wheel speeds and position estimates received from the network at step 7.
4. When the packet transmitted at step 2 reaches a node, it starts to sample the incoming ultrasound pulse. The sampling is interrupted when the edge of the pulse is reached.
5. The nodes compute their new estimate based on information received during the previous time interval.
6. Each node logs data using a wired network to reduce interference.
7. After a specified time based on its identity number the nodes broadcast their new position estimate together with the distance measurement taken at step 4. These messages are then received by neighboring nodes, including the robot if it is in range.
8. Finally the robot receives commands from a joystick used to control it.

In the implementation, data is transmitted after the prediction step instead of between the update- and prediction step as described in Section 3. However it is straightforward to modify the algorithm for this scenario.

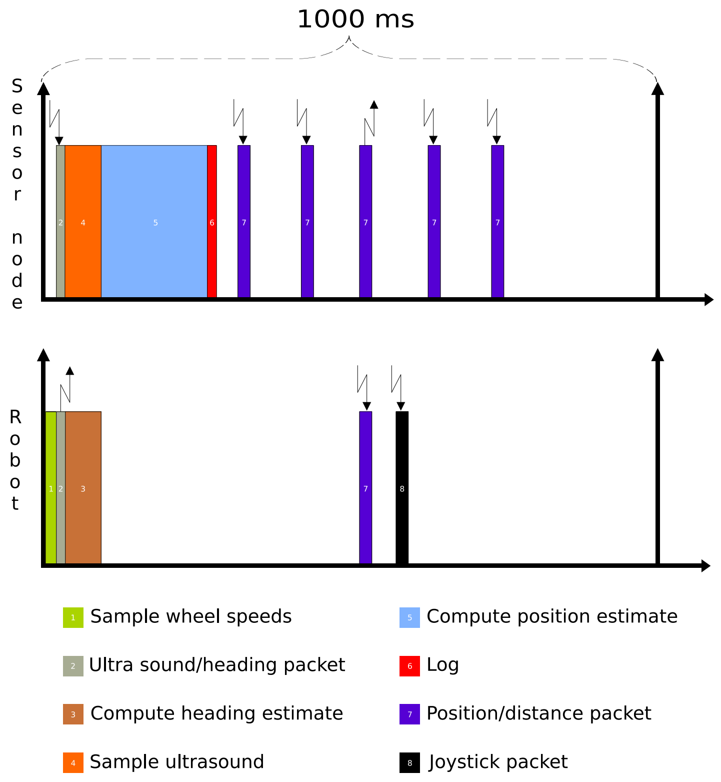


Figure 4. Communication schedule used in experiments.

6. Experimental Results

To evaluate performance, the root mean square (RMS) of the difference to the position estimated by the vision system p^{cam} will be used.

When evaluating the impact of different parameters and design choices it is crucial that experiments are repeatable. However, as the radio environment where the experiments were performed is highly non-stationary, repeatability was a big problem. This issue was resolved by studying estimates generated in Matlab using logged trilateration-, wheel speed- and packet arrival data as input. The average RMS difference for estimates generated in Matlab compared to real experiments is approximately 1 cm. This error is mostly due to quantization effects in the logging of wheel speeds.

The main results of the experiments are summarized in Figure 5 where the RMS error for different types of estimation schemes are shown. Global

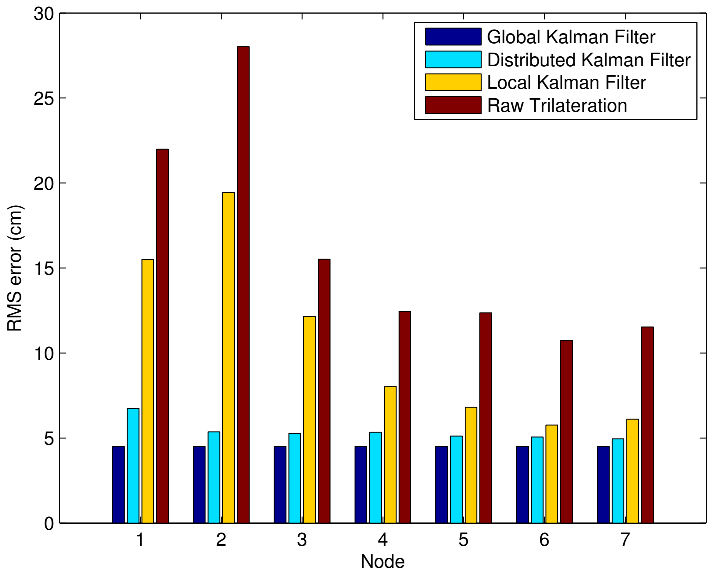


Figure 5. RMS error $(\hat{p} - p^{cam})$ for different types of estimation schemes. Global refers to a Kalman filter that has access to all information without any delay. Distributed is the proposed distributed Kalman filter and Local refers to a Kalman filter that only uses local trilateration information. As a comparison raw trilateration is also plotted. The big difference in trilateration accuracy among nodes is due to the relative position of a node compared to the robot trajectory.

Kalman filter means that the filter has access to trilateration information from all nodes without any delay. Distributed denotes the proposed algorithm, whereas in the local case only local trilateration information is used. As a comparison the raw trilaterated estimate is also plotted. Note that even in the local case a node needs to communicate with two of its neighbors to be able to perform trilateration. The big difference in trilateration accuracy among nodes is due to the relative position of a node compared to the robot trajectory.

For both the global- and distributed Kalman filter to perform well it is crucial that the relation between the diagonal elements of R_e is correct. To achieve this, $\hat{R}_e$ was computed based on the same trajectory as the RMS errors in Figure 5.

Examining the results, we can draw the conclusion that the distributed algorithm performs almost as good as a global solution. One can also note that the performance of the local estimator is very close to that of both the global and distributed schemes in e.g nodes 6 and 7 where the measurement accuracy is high. Using a permutation test [Moore and McCabe, 2006] differences between the global- and distributed Kalman filter could only be verified at a 95% confidence level in node 1. Using the same test, differences between the local- and distributed Kalman filter could not be verified in nodes 6 and 7. The permutation test used here is somewhat conservative as it does not utilize the correlation between different estimates generated from the same data.

One critical issue for an algorithm that utilizes a wireless communication channel is sensitivity to packet loss. In the results presented in Figure 5 the average packet loss probability was approximately 10%. To further investigate how sensitive the proposed algorithm is to packet loss, a number of simulations were performed. In this study two different ways of handling lost packets were investigated: to use the last received packet and to use the local estimate. In Figure 6 the average RMS error among the seven nodes is plotted as a function of packet loss for the two different methods. As a comparison the average RMS error for raw trilateration and local estimation are also shown. To isolate the errors caused by the distributed Kalman filter algorithm, packets used in the trilateration computations were left unaffected by the increased packet loss probability. Therefore, the errors for raw trilateration and local estimation are unaffected by the packet loss. From Figure 6 it is apparent that using the local estimate when a packet is lost is preferable. If this approach is used the performance of the distributed solution approaches the local solution as the packet loss approaches one for this example.

7. Conclusions

In this paper an optimization based algorithm for distributed estimation is evaluated experimentally. The algorithm is based on standard Kalman filtering results and then extended with one step where nodes merge their estimates. The estimates are merged by a weighted average approach.

The algorithm applies to a broad category of communication topologies, including graphs with loops. The weights are optimized off-line allowing only estimates to be communicated among the nodes. All communication is restricted to neighboring nodes, which allows the algorithm to scale.

An experimental evaluation was done to demonstrate how the proposed algorithm performs in an uncertain environment where, for example packets are lost and different nodes are not perfectly synchronized in time. The scenario chosen is one where seven nodes in a sensor network estimate the position of a mobile robot using ultrasound. It was concluded that the performance in RMS sense of the proposed algorithm was very close to the performance of a optimal global solution. Also the distributed Kalman filter proved to be very insensitive to packet loss, which is of great impor-

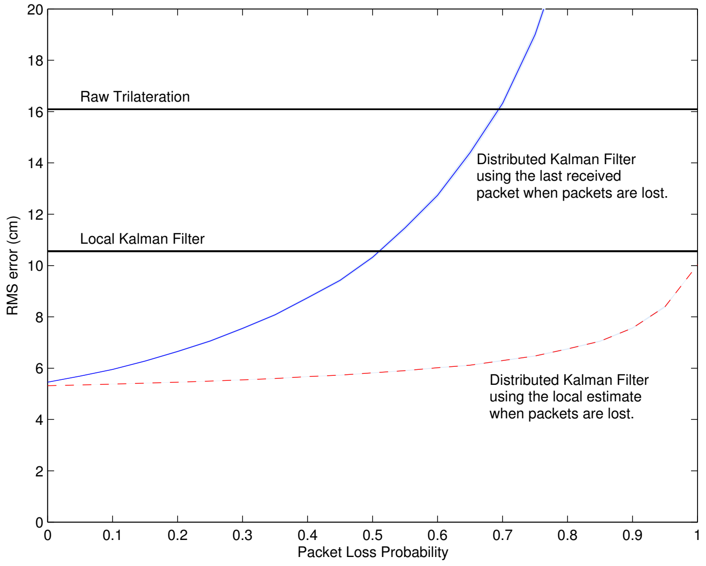


Figure 6. Average RMS error $(\hat{p} - p^{cam})$ as a function of packet loss for the distributed Kalman filter when the last received packet is used when a packet is lost (solid) and when the local estimate is used (dashed). As a comparison the average RMS error for raw trilateration and local estimation is also shown.

tance when dealing with wireless communication links. As presented in Figure 6 the performance degradation at for example 10% and 50% packet loss are only 1.1% and 9.4% respectively.

Acknowledgment

This work was partially funded by the RUNES Integrated Project contract IST-2004-004536.

References

Alriksson, P. (2007): "Distributed Kalman filtering using weighted averaging - theory and experiments." unpublished report, `http://www.control.lth.se/publications/`.

Alriksson, P. and A. Rantzer (2006): "Distributed Kalman filtering using weighted averaging." In *Proceedings of the 17th International*

Symposium on Mathematical Theory of Networks and Systems. Kyoto, Japan.

Årzén, K.-E., M. Ohlin, A. Cervin, P. Alriksson, and D. Henriksson (2007): "Holistic simulation of mobile robot and sensor network applications using TrueTime." In *Proceedings of the European Control Conference*. Kos, Greece.

Bar-Shalom, Y. and L. Campo (1986): "The effect of common process noise on two-sensor fused-track covariance." *IEEE Transactions on Aerospace and Electronic Systems*, November, pp. 803–805.

Bourgault, F. and H. F. Durrant-Whyte (2004): "Communication in general decentralized filters and the coordinated search strategy." In *The 7th International Conference on Information Fusion*.

Cadman, J. (2003): "Deploying commercial location-aware systems." In *Proc. Fifth International Conference on Ubiquitous Computing*.

Durrant-Whyte, H., B. Rao, and H. Hu (1990): "Toward a fully decentralized architecture for multi-sensor data fusion." *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1331–1336 vol.2.

Grime, S., H. F. Durrant-Whyte, and P. Ho (1992): "Communication in decentralized data-fusion systems." In *In Proc. American Control Conference*, pp. 3299–3303.

Hall, D. and J. Llinas (1997): "An introduction to multisensor data fusion." *Proceedings of the IEEE*, **85:1**, pp. 6–23.

Harris, C. and M. Stephens (1988): "A combined corner and edge detector." In *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151.

Manolakis, D. E. (1996): "Efficient solution and performance analysis of 3-d position estimation by trilateration." *IEEE Transactions on Aerospace and Electronic Systems*, **32:4**, pp. 1239–1249.

Moore, D. S. and G. P. McCabe (2006): *Introduction to the Practice of Statistics*, 5th edition. W H Freeman.

Olfati-Saber, R., J. A. Fax, and R. M. Murray (2007): "Consensus and cooperation in networked multi-agent systems." *Proceedings of the IEEE*, **95:1**, pp. 215–233.

Priyantha, N., A. Chakraborty, and H. Balakrishnan (2000): "The Cricket location-support system." In *Proc. Sixth ACM MOBICOM Conf.* Boston, MA.

Smith, A., H. Balakrishnan, M. Goraczko, and N. B. Priyantha (2004): "Tracking Moving Devices with the Cricket Location System." In *2nd International Conference on Mobile Systems, Applications and Services (Mobisys 2004)*. Boston, MA.

Speranzon, A., C. Fischione, and K. H. Johansson (2006): "Distributed and collaborative estimation over wireless sensor networks." In *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 1025–1030. San Diego.

Thomas, F. and L. Ros (2005): "Revisiting trilateration for robot localization." *IEEE Transactions on Robotics*, **21:1**, pp. 93–102.

Viswanathan, R. and P. Varshney (1997): "Distributed detection with multiple sensors part i. fundamentals." *Proceedings of the IEEE*, **85:1**, pp. 54–63.

Want, R., A. Hopper, V. Falcao, and J. Gibbons (1992): "The Active Badge Location System." *ACM Transactions on Information Systems*, **10:1**, pp. 91–102.

Paper IV

A Component-Based Approach to Ultrasonic Self Localization in Sensor Networks

Peter Alriksson and Karl-Erik Årzén

Abstract

This report describes the development of an ultrasound based self localization system for mobile robots. The robot navigates using a combination of internal wheel encoders and absolute distance information from a sensor network. The sensor fusion problem is formulated as a nonlinear state estimation problem which is approximated using an extended Kalman filter.

Using a general state estimation formulation has the advantage that if no information can be provided by the network, due to for example heavy network load, the robot can still navigate using only internal information.

The software was developed using a component-based middleware.

Technical report ISRN LUTFD2/TFRT–7619–SE, Department of Automatic Control, Lund University, Sweden, May 2008

1. Introduction

Networked embedded systems play an increasingly important role and affect many aspects of our lives. By enabling embedded systems to communicate, new applications are being developed in areas such as health-care, industrial automation, power distribution, rescue operations and smart buildings. Many of these applications will result in a more efficient, accurate and cost effective solution than previous ones. The European integrated project Reconfigurable Ubiquitous Networked Embedded Systems [RUNES, 2007] brings together 21 industrial and academic teams in an attempt to enable the creation of large scale, widely distributed, heterogeneous networked embedded systems that inter-operate and adapt to their environments. The inherent complexity of such systems must be simplified if the full potential for networked embedded systems is to be realized. The RUNES project aims to develop technologies (system architecture, middleware, networking, control etc.) to assist in this direction, primarily from a software and communications standpoint.

Networked control systems impose additional requirements that arise from the need to manipulate the environment in which the networked systems are embedded. Timing and predictability constraints inherent in control applications are difficult to meet in general, due to the variations and uncertainties introduced by the communication system: delays, jitter, data rate limitations, packet losses etc. For example, if a control loop is closed over a wireless link, it should tolerate lost packets and be able to run in open loop over periods of time. Resource limitations of wireless networks also have important implications for the control design process, since limitations such as energy constraints for network nodes need to be integrated into the design specifications. The added complexity and need for re-usability in the design of control over wireless networks suggest a modular design framework.

In this report, we propose a component-based approach to handle the software complexity of networked control systems. A general framework is presented and it is shown how it can be instantiated in the specific problem of robot self localization and control. Section 1.1 briefly presents the RUNES middleware and component architecture. In the RUNES project a motivating scenario was developed, this scenario is described in Section 1.2. The different components and their connections are presented in Section 2 and in sections 3 and 4 the algorithms used for self localization and robot control are discussed.

1.1 Middleware and Components

The RUNES middleware [Mascolo *et al.*, 2005] is illustrated in Figure 1. The middleware acts as a glue between the sensor, actuator, gateway and

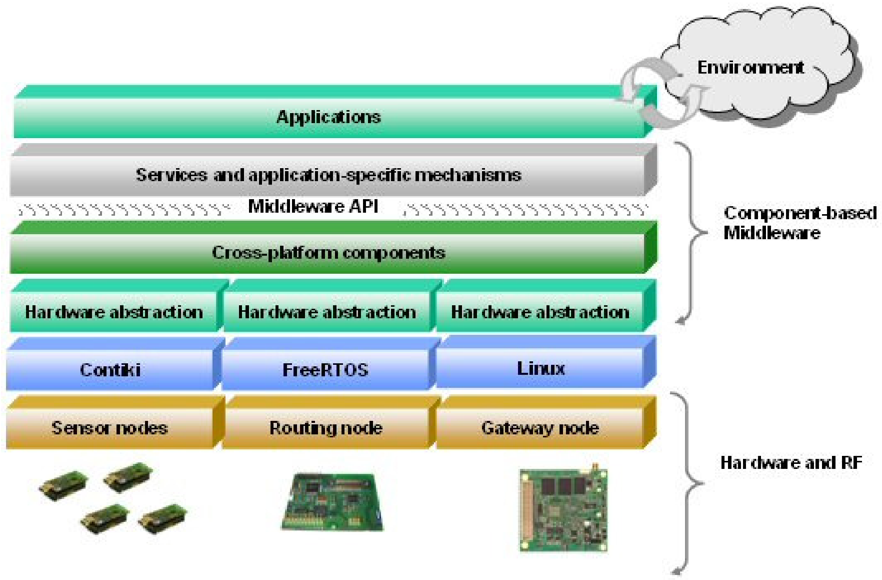


Figure 1. Overview of the RUNES middleware platform. The component-based middleware resides between the application and the operating systems of the individual network nodes.

routing devices, operating systems, network stacks, and applications. It defines standards for implementing software interfaces and functionalities that allow the development of well-defined and reusable software. The basic building block of the middleware developed in RUNES is a software component. From an abstract point of view, a component is an autonomous software module with well defined functionalities that can interact with other components only through interfaces and receptacles. Interfaces are sets of functions, variables and associated data types that are accessible by other components. Receptacles are required interfaces by a component and make explicit the inter-component dependencies. A graphical representation of a RUNES component is shown in Figure 2.

The connection of two components occurs between a single interface and a single receptacle. Such association is called binding and is shown in more detail in Figure 3. Part of the RUNES middleware has been demonstrated to work well together with the operating system Contiki [Dunkels *et al.*, 2004], which was developed for low memory low-computation devices. The implementation of the component model for Contiki is known as the component runtime kernel (CRTK). This component framework provides for instance dynamic run-time bindings of components, i.e., during

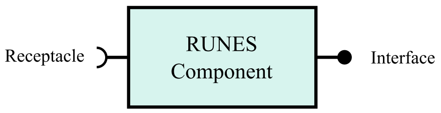


Figure 2. Graphical representation of a RUNES component. Interfaces are sets of functions, variables and associated data types that are accessible by other components. Receptacles are required interfaces by a component.

execution it allows components to be substituted with other components with the same interface.

1.2 Motivating Scenario

One of the major aims of the RUNES project is to create a component-based middleware that is capable of reducing the complexity of application construction for networked embedded systems of all types. Versions of the component runtime kernel, which forms the basis of the middleware, are available for a range of different hardware platforms. However, the task is a complex one, since the plausible set of sensing modalities, environmental conditions, and interaction patterns is very rich. To illustrate one potential application in greater detail, the project selected a disaster relief scenario, in which a fire occurs within a tunnel, much as happened in the Mont Blanc tunnel in 1999. In this, the rescue services require information about the developing scenario both before arrival and during rescue operations, and such information is provided by a network of sensors, placed within the tunnel, on robots, and on rescue personnel themselves. We explore the scenario in more detail below, but it should be noted this is intended to be representative of a class of applications in which system robustness is important and the provision of timely information is crucial. So, for example, much the same considerations apply in the prevention of, or response to, Chemical, Biological, Radiological, Nuclear or Explosive (CBRNE) attacks; likewise, search and rescue operations, and even industrial automation systems form application domains with similar requirements for predictability of response given challenging external conditions.

In the RUNES scenario, we project what might happen in a similar situation if the vision of the US Department of Homeland Security's SAFECOM programme becomes a reality. The scenario is based around a storyline that sets out a sequence of events and the desired response of the system, part of which is as follows. Initially, traffic flows normally through the road tunnel; then an accident results in a fire. This is detected by a wired system, which is part of the tunnel infrastructure, and is reported back to the Tunnel Control Room. The emergency services are summoned by Tunnel Control Room personnel. As a result of the fire, the wired infrastructure is damaged and the link is lost between fire detection nodes (much as happened in Mont Blanc). However, using wireless communication as a backup, information from (for example) fire and smoke sensors continues to be delivered to the Tunnel Control Room seamlessly. The first response team arrives from the fire brigade and rapidly deploys search and rescue robots, following on foot behind. Each robot and firefighter carries a wireless communication gateway node, sensors for environmental temperature, chemical and smoke monitoring, and the robots carry light detectors that help them identify the seat of the blaze.

The role of the robots in this scenario is twofold: to help identify hazards and people that need attention, without exposing the firefighters to danger; and to augment the communications infrastructure to ensure that both tunnel sensor nodes and those on firefighters remain in contact with the command and control systems that the situation commander uses to make informed decisions about how best to respond. To accomplish this, the robots are moving autonomously in the tunnel taking into account information from tunnel sensors about the state of the environment, from a human controller about overall mission objectives, and from received signal strength measurements from the wireless systems of various nodes about the communication quality. The robots coordinate their activity with each other through communication over wireless links. Local backup controllers allow the robots to behave reasonably in the event that communication is lost.

2. Software Components

This section presents an overview of the components and their bindings used for navigation, self localization and control of a single autonomous robot. Components aimed at for example restoring or improving network connectivity, radio power control, coordination of multiple robots, security issues and so on are described in [Årzén *et al.*, 2007] and its companion papers.

A component-based middleware has, at least in theory, the big advantage that components can be developed and tested independently. However, when dealing with components that interact with the environment, one must be aware of how the interconnection of components effect real-time performance. Thus some of the advantages with a component-based architecture can be illusive, especially in embedded systems with limited resources. Techniques like reservation-based scheduling are promising, but it is still an active research area, see for example [Lindberg, 2007] for a recent survey.

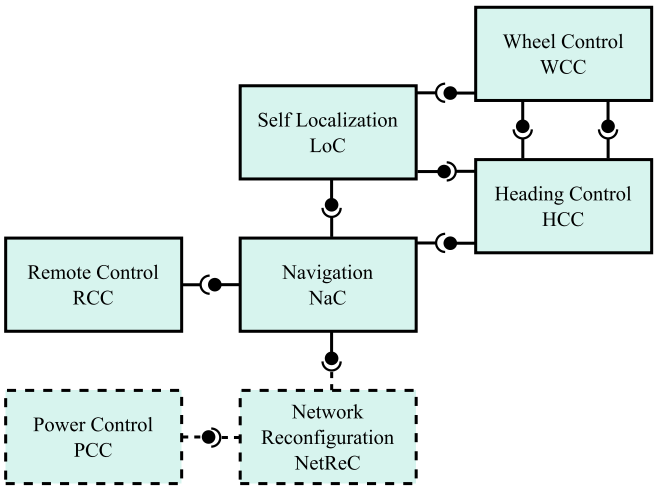


Figure 3. Overview of the component architecture used for navigation and deployment of a single autonomous robot.

In Figure 3 an overview of the component architecture is shown. The system also contains a network communication component which provides UDP and TCP services together with Ad hoc On-Demand Vector Routing.

The interface-receptacle framework implies that components can be added in an hierarchical fashion. For example, the wheel control component (WCC), which does not require a connection to any other component and thus have no receptacles, serves as the base of the hierarchy. The self localization component (LoC) then binds to the WCC and so on.

As depicted in Figure 1 one component can reside in multiple devices, possible running different operating systems. In case of the LoC and RCC this is inherent to their function, whereas in the case of the HCC, WCC and NaC it is an implementational choice. All components except the PC part of the RCC was implemented in C code.

Next the different components together with their interfaces and receptacles will be described.

2.1 Wheel Control Component

The wheel control component is the most rudimentary component discussed here. It provides low level control of the movement of the robot. Two interfaces are provided by the WCC: one containing the current angular velocity of the two wheels and one providing a function to set the reference velocities.

2.2 Self Localization Component

The self localization component resides both in the robot and in a number of stationary anchor nodes. To reduce the computational time of the algorithm, some computations are also distributed among two microprocessors within the robot.

The LoC provides one interface containing estimates of the robots position and heading in a predefined coordinate system. The interface also contains the covariance matrix of the estimation error which acts as a quality measure of the estimates. To enhance accuracy the LoC uses information about the movement of the robot. This is represented as a receptacle requesting wheel velocities.

2.3 Heading Control Component

The heading control component provides one interface containing functions for changing the desired heading together with the reference velocity of the robot. Measurements of the heading and wheel velocities are requested through two receptacles. The HCC also requires a way of changing the wheel reference velocities, this is represented as a receptacle requesting an interface with this functionality.

2.4 Navigation Component

The navigation component provides one interface containing a function to change the desired location of the robot. Measurements of the robots location and heading are requested through one receptacle. The NaC requires a way of changing the heading and velocity references, this requirement is also represented as a receptacle. The algorithms used for navigation are described in [Nordh, 2007] and [Edhner, 2007].

2.5 Remote Control Component

Naturally, also the remote control component is distributed between the controlling computer and the robot. The RCC has only one receptacle, representing a requirement of an interface that provides a function for changing the desired location of the robot. The remote control component generates this value through interaction with a remote user.

3. Self Localization

A prerequisite for navigation is self localization, i.e., the robots must know their current position and heading. Since the tunnel is assumed to be well-known, automatic map building is not considered. Instead it is assumed that the overall layout of the tunnel is known, with the exception of the position of a number of stationary obstacles, modeling, e.g., stalled vehicles.

Self localization of mobile robots can be performed with a number of techniques. In laboratory experiments it is common to use vision, e.g., a ceiling-mounted camera combined with an image-processing system. In the tunnel scenario this is not a realistic approach due to, e.g., problems with light and smoke. Another possibility is to use dead-reckoning using a high-precision inertial measurement sensor unit on-board the robot. A problem with dead reckoning-based approaches, however, is that they are open loop and that unmeasurable disturbances will cause position errors that cannot be compensated for. In an outdoor environment GPS would have been another possibility, but inside a tunnel this is less realistic.

The self localization approach chosen in the RUNES project is based on measuring the distance to a number of objects with known positions, known as anchors, and then computing the position of the unknown object, in this case the robot, using these measurements. In a real road tunnel, the means of acquiring these distance measurements, would be dependent on the environment in which the system must operate. As the aim of this project was not to develop a positioning system capable of operating in the harsh environment of a burning road tunnel, but to demonstrate the benefits of a component-based design approach, a simple ultrasound based solution was chosen.

3.1 Ultrasound Distance Measurements

The basic idea is to transmit a wireless radio packet simultaneously with an ultrasound pulse from each sender node. The receiver nodes measure the difference in time of arrival between the radio packet and the ultrasound pulse and can in this way calculate their distance to the sender node.

Two main approaches exists, [Smith *et al.*, 2004]. In an *active mobile* system the infrastructure, in this case the tunnel, has receivers at known locations, which estimate distances to a mobile device based on active transmissions from the device. Examples of this approach are the Active Badge [Want *et al.*, 1992], and the Ubisense [Cadman, 2003] systems. In a *passive mobile* system, instead, the infrastructure has active beacons at known positions that periodically transmits signals to a passive mobile device. The most famous example of this is the Cricket system [Priyantha *et al.*, 2000].

An advantage of the active approach is that it is more likely to perform accurate tracking than the passive approach. The passive approach, on the other hand, scales better with the number of mobile devices. Since in the tunnel scenario good tracking is important and the number of mobile robots is small, the active approach was chosen. The stationary sensor

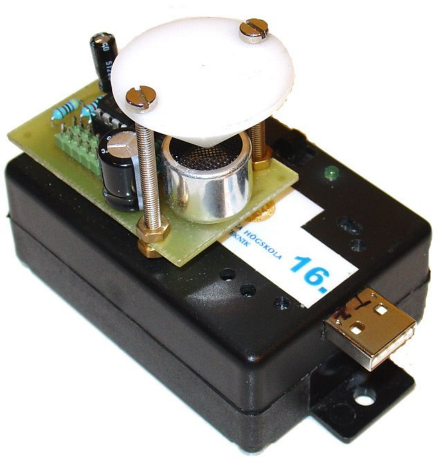


Figure 4. Stationary sensor network nodes with ultrasound receiver circuit. The nodes are packaged in a plastic box to reduce wear.

nodes in the tunnel are each equipped with an ultrasound receiver and each mobile robot is equipped with an ultrasound transmitter. The stationary sensor nodes are implemented as Tmote Sky sensor network “motes” together with a small ultrasound receiver circuit interfaced to the mote via the AD converter, see Figure 4. The mobile robots are equipped with an ultrasound transmitter circuit. Both the ultrasound transmitters and receivers are designed to be isometric, i.e., to transmit and receive in the full 360° degree plane.

A second reason for choosing ultrasound-based self localization is that it involves the use of the sensor network in closed loop. One of the objectives of the RUNES project was to investigate the possibilities and problems associated with networked control over sensor networks. In wireless networks the lack of worst-case latency guarantees and risk of losing radio packets creates extra challenges for control. The ultrasound based location method provides a possibility for evaluating this.

3.2 State Estimation

Inferring the position (or any other quantity) of an unknown object from measurements of the distance to a number of objects with a priori known positions can be done using a variety of methods. In general the distance measurements are corrupted by noise, so any method used should involve some kind of filtering. If additional measurements such as wheel velocities and heading information is available, this information should also be incorporated in the estimate of the position. Both these requirements suggest the use of a dynamic model of the object to be positioned together with some statistical inference technique. The problem can thus be formulated as a general nonlinear state estimation problem on the form

$$\begin{aligned} x(k+1) &= f(x(k),k) + w(k) \\ y(k) &= h(x(k),k) + v(k) \end{aligned} \tag{1}$$

where $x(k) \in \mathbf{R}^n$ is the state vector to be estimated, $y(k) \in \mathbf{R}^p$ is the measured output, $w(k) \in \mathbf{R}^n$ and $v(k) \in \mathbf{R}^p$ are unknown disturbances. Note that a known input can be seen as a time dependent $f(\cdot)$.

When solving a general nonlinear state estimation problem one must almost always resort to approximations of some sort. The by far most common approach is the Extended Kalman filter [Jazwinski, 1970] where the probability distribution of $x(k)$ given all previous information is approximated using a Gaussian distribution. A Gaussian distribution is fully described by its mean and covariance, thus it is sufficient to find an approximate way of updating these quantities after a new measurement is taken and as time progresses. In the Extended Kalman Filter, or EKF for short, this is done through linearization of $f(\cdot)$ and $h(\cdot)$.

An alternative way of updating the mean and covariance is through the use of the so called unscented transform, which results in the Unscented Kalman Filter (UKF) [Julier and Uhlmann, 1997]. The UKF uses a deterministic sampling approach where the mean and covariance is represented using a minimal set of carefully chosen sample points. When propagated through the true nonlinear system, these points capture the mean and covariance accurately to the 3rd order Taylor series expansion of the nonlinearity.

In both the EKF and UKF the the probability density of $x(k)$ is approximated by a Gaussian distribution. If a more general distribution is needed, the use of sequential Monte Carlo methods such as the so called particle filter [Doucet *et al.*, 2001] is a common approach.

All the methods presented so far, aim at approximating the full *probability distribution* of $x(k)$ given all previous information. A different approach is the joint maximum a posteriori- or trajectory estimation method where a point estimate of the full trajectory $x(0) \ldots x(k)$ is generated. Using dynamic programming a recursive procedure for generating a point estimate of the last value $x(k)$ can be derived in principal. However, in general the complexity of this recursive scheme grows with k making it impossible to implement. One common method that finds an approximation of the joint maximum a posteriori estimate is moving horizon estimation (MHE) [Rao, 2000]. In MHE a point estimate of $x(k-N) \ldots x(k)$ for some fixed value N is generated by solving an optimization problem online.

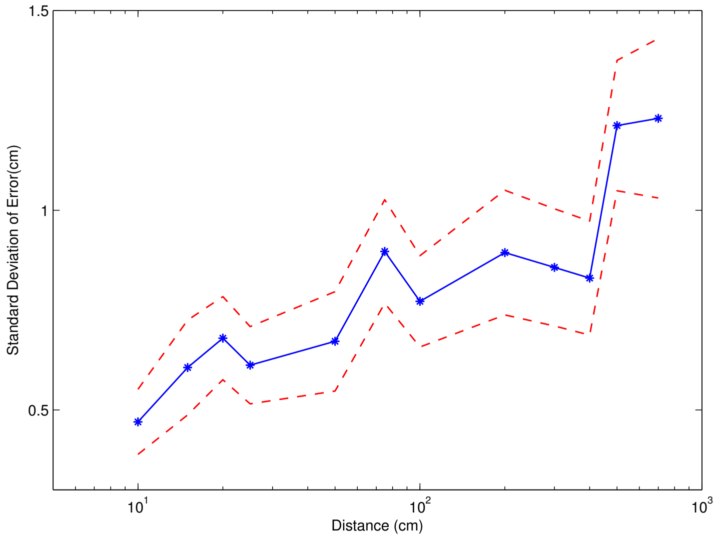


Figure 5. Standard deviation of the distance measurement error as a function of distance together with 95% confidence intervals.

3.3 Measurement Models

The measurement function $h(\cdot)$ is determined by if the distance measurements are preprocessed or not. If no preprocessing is done it will simply be the Euclidean distance between the object and the anchors. This setup will be referred to as the direct measurement model. In this case the unknown disturbance $v(k)$ can, at least approximately, be modeled as white Gaussian noise independent of the state. As can be seen in Figure 5 the implemented system gives an error of about 1 cm for distances up to 7 m. Also, the accuracy is fairly independent of the distance.

Trilateration. One common way of preprocessing the distance measurements is called trilateration. Trilateration is a process, where three (in the plane) distance measurements together with the known positions of the anchor nodes, produces an estimate of the position of the unknown object. In this case $h(\cdot)$ reduces to an identity map for the position coordinates, but at the cost of making $v(k)$ highly dependent of $x(k)$. This dependence is in general difficult to model, due to the nonlinear characteristics of the trilateration procedure.

The basic problem is to find a solution $[\, p_x \quad p_y \quad p_z \,]^T$ to the following three nonlinear equations

$$\begin{aligned}(p_x - p_{x1})^2 + (p_y - p_{y1})^2 + (p_z - p_{z1})^2 &= d_1^2\\ (p_x - p_{x2})^2 + (p_y - p_{y2})^2 + (p_z - p_{z2})^2 &= d_2^2\\ (p_x - p_{x3})^2 + (p_y - p_{y3})^2 + (p_z - p_{z3})^2 &= d_3^2.\end{aligned}$$

where p_{xi}, p_{yi} and p_{zi} are known positions of the anchors and d_i is the distance from anchor i to the object to be positioned. The problem can be transformed to a system of two linear equations and one quadratic equation by e.g subtracting the second and third equation from the first, see [Manolakis, 1996] for a detailed analysis.

An alternative more geometric approach was taken in [Thomas and Ros, 2005] where the problem is solved using Cayley-Menger determinants. This approach as the benefit of a geometric interpretation of the solution in terms of volumes, areas and distances. Also the error analysis with respect to e.g distance errors is simplified.

As the robot is assumed to only move in the xy-plane, the problem can be reduced to a set of two linear equations as discussed above. The two linear equations will always have a solution unless all three known points are positioned on a line. The two linear equations defines two lines, see Figure 6, which can be represented as

$$a_0 y = a_1 + a_2 x \qquad (2)$$

$$b_0 y = b_1 + b_2 x \qquad (3)$$

where

$$\begin{aligned}a_0 &= 2(p_{y2} - p_{y1})\\ a_1 &= d_1^2 - d_2^2 + p_{y2}^2 - p_{y1}^2 + p_{x2}^2 - p_{x1}^2 - 2p_z(p_{z2} - p_{z1})\\ a_2 &= 2(p_{x1} - p_{x2})\\ b_0 &= 2(p_{y3} - p_{y1})\\ b_1 &= d_1^2 - d_3^2 + p_{y3}^2 - p_{y1}^2 + p_{x3}^2 - p_{x1}^2 - 2p_z(p_{z3} - p_{z1})\\ b_2 &= 2(p_{x1} - p_{x3}).\end{aligned}$$

Note that the z-coordinate p_z of the robot is assumed to be known, as it is only moving in the xy-plane. The intersection point of these two lines constitute the trilaterated position $[\,p_x^{\text{tri}} \quad p_y^{\text{tri}}\,]^T$. Even though the three circles do not intersect in one point, the algorithm provides a reasonable result. However, for nearly singular situations like the one shown in Figure 7, the result produced need not to be very accurate. For a detailed

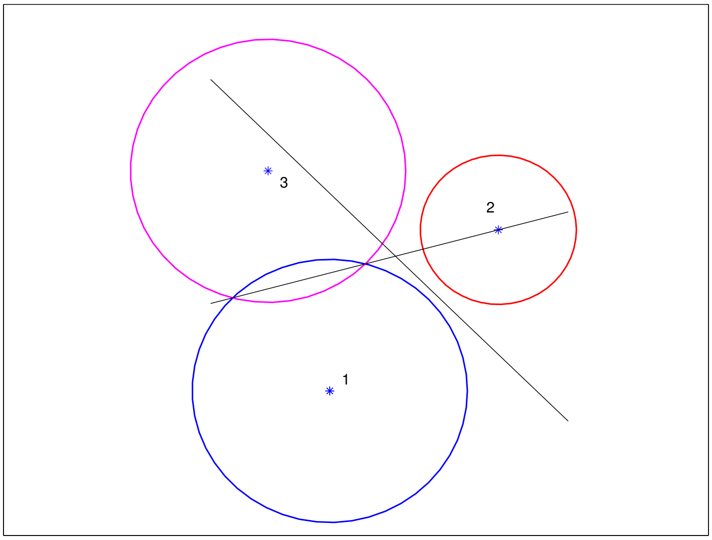


Figure 6. The intersection of two lines defining the solution to the trilateration problem. The three circles correspond to distance measurements to three anchor nodes with known positions. When measurement noise is present the circles may as shown overlap or not intersect. However, the trilateration procedure results in a reasonable result in both these cases.

discussion on how errors both in distance measurements and node positions influence the trilateration result, see [Thomas and Ros, 2005] and [Manolakis, 1996].

3.4 Choice of measurement model

Using the direct measurement model has two major advantages over trilateration: First of all, the direct approach can make use of only one or two measurements, whereas to perform trilateration three measurements are necessary. In a system relying on wireless communication, this is an important advantage. Secondly, modeling the state dependent noise in the trilateration case is very difficult. This results in that the algorithm is unaware of how good a measurement really is. This information is available to the direct approach as it uses a nonlinear measurement function $h(\cdot)$.

The trilateration approach has the advantage that it is conceptually simpler, as after a trilateration computation is done, an estimate of the position is directly available. In the direct measurement model, the position estimate is the result of an iterative algorithm involving other known signals and/or unknown states.

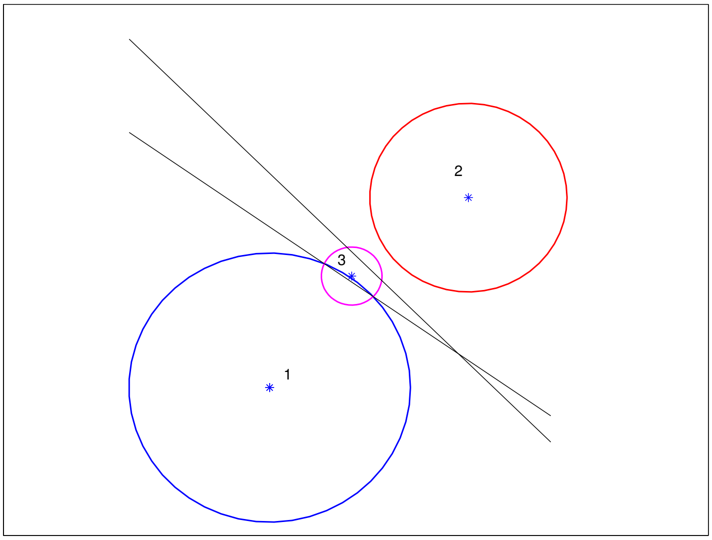


Figure 7. Nearly singular configuration where the trilateration result (intersection of the two lines) need not be very accurate. The distance measurements suggest that the object is located in the vicinity of anchor 3, but the trilateration procedure gives a result far away.

3.5 Dynamical Model

The choice of dynamical model $f(\cdot)$ should reflect both the available knowledge and which quantities that are of interest. The robot used in the RUNES project is a two wheeled dual drive robot with an unactuated support. Using this type of robot has the big advantage that when turning, the actuated wheels are not slipping, thus a third order kinematic model can easily be derived.

The robot together with coordinate definitions is shown in Figure 8. Using the two position variables p_x and p_y together with the heading θ as state variables the model can be written as

$$\begin{cases} \dot{p}_x = \dfrac{R}{2}(\omega_1 + \omega_2)\cos(\theta) \\ \dot{p}_y = \dfrac{R}{2}(\omega_1 + \omega_2)\sin(\theta) \\ \dot{\theta} = \dfrac{R}{D}(\omega_2 - \omega_1) \end{cases} \tag{4}$$

where R is the radius of the wheels and D is the distance between them. Inputs to the system are the angular velocities ω_1 and ω_2 of the two

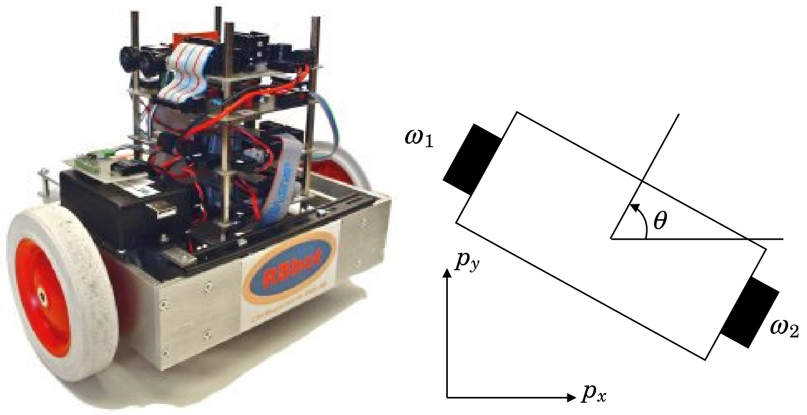


Figure 8. Definition of coordinates with respect to the robot.

wheels. In this model, these angular velocities are assumed to be completely known as they are controlled by two PI-controllers, see Section 4.2.

To get a model on the form (1) the continuous time model (4) must be discretized. The most common choice of discretization scheme is the simple forward Euler scheme,

$$x(k+1) = x(k) + T\dot{x}(k) = f(x(k), k) \tag{5}$$

However, due to the limited computational resources in the hardware platform, the sampling interval T must be kept rather large. For large sampling intervals the forward Euler scheme can perform very badly. This motivates the use of a higher order scheme such as a second order Adams-Moulton method,

$$x(k+1) = x(k) + \frac{T}{2}\left(\dot{x}(k) + \dot{x}(k+1)\right) \tag{6}$$

In general the Adams-Mouton method is implicit, that is, we have to solve for $x(k+1)$ using some numerical method. However, for the continuous time model (4) an explicit solution can be obtained. Note that $f(x(k), k)$ will depend on the wheel velocities at time $k+1$, but this poses no problem if the model is not used for prediction.

3.6 Extended Kalman Filter

This section presents the extended Kalman filter and points out some implementational issues. The presentation is made with the direct measurement model in mind, but the same procedure applies for trilateration.

Care must however be taken with how to compute the measurement noise covariance matrix R_v in the case of trilateration.

The EKF consists of two steps: time update and measurement update. In the time update step the mean $\hat{x}$ and covariance P of the Gaussian approximation of the probability density of $x(k)$ given all information up to k is updated as

$$\begin{aligned} \hat{x}(k+1|k) &= f(\hat{x}(k|k), k) \\ P(k+1|k) &= F(k)P(k|k)F^T(k) + R_w \end{aligned} \tag{7}$$

where R_w is the covariance of the process disturbance $w(k)$. The matrix $F(k)$ is the Jacobian of $f(\cdot)$ with respect to $x(k)$ at $\hat{x}(k|k)$,

$$F(k) = \frac{\partial}{\partial x} f(x,k) \bigg|_{x=\hat{x}(k|k)} \tag{8}$$

When a measurement is available the mean and covariance is updated as

$$\begin{aligned} \hat{x}(k|k) &= \hat{x}(k|k-1) + K(k)\left(y(k) - h(\hat{x}(k|k-1), k)\right) \\ P(k|k) &= (I - K(k)H(k))P(k|k-1) \end{aligned} \tag{9}$$

where

$$\begin{aligned} K(k) &= P(k|k-1)H^T(k)S^{-1}(k) \\ S(k) &= H(k)P(k|k-1)H(k)^T + R_v \\ H(k) &= \frac{\partial}{\partial x} h(x,k) \bigg|_{x=\hat{x}(k|k-1)} \end{aligned} \tag{10}$$

and R_v denotes the covariance of the measurement disturbance $v(k)$.

Note that if the Adams-Mouton approximation is used the time update step has to be computed at the beginning of the sampling interval due to the dependence in $f(x,k)$ of $\omega_1(k+1)$ and $\omega_2(k+1)$.

Because the number of received distance measurements may vary with time, the size of $H(k)$, $S(k)$ and $K(k)$ will also change over time. In the extreme, when no measurements are received, the measurement update step is simply ignored and the estimates are updated using the dynamical model only. This is often referred to as dead reckoning.

Perhaps both the most numerically sensitive and computationally demanding part of the above computations is the inversion of the output prediction error covariance matrix $S(k)$. This square matrix has the same dimension as the number of received measurements. For a linear Kalman filter, processing measurements with independent noise one at a time, thus reducing the matrix inversion to division with a scalar, is equivalent to processing them all at once. For a description, see for example [Jazwinski, 1970] or [Mendel, 1971] where a detailed computational analysis is done. For the EKF a similar procedure can be used, however sequential and non-sequential processing are in general not equivalent due to the nonlinear update equation. To be consistent with the structure of (9) $H(k)$ should be re-linearized after the addition of each new measurement.

3.7 Experimental Validation

To validate the accuracy of a self localization system using an EKF with the direct measurement model, a reference camera system was used. The EKF with sequential measurement processing was implemented in C code using 32 bit software emulated floating point arithmetic.

In Figure 9 the estimated position and heading is shown together with the measurements generated by the camera system. The camera system had a accuracy of about 1 cm and 6 degrees. The estimates were generated using seven anchor nodes distributed to form equilateral triangles covering the working area.

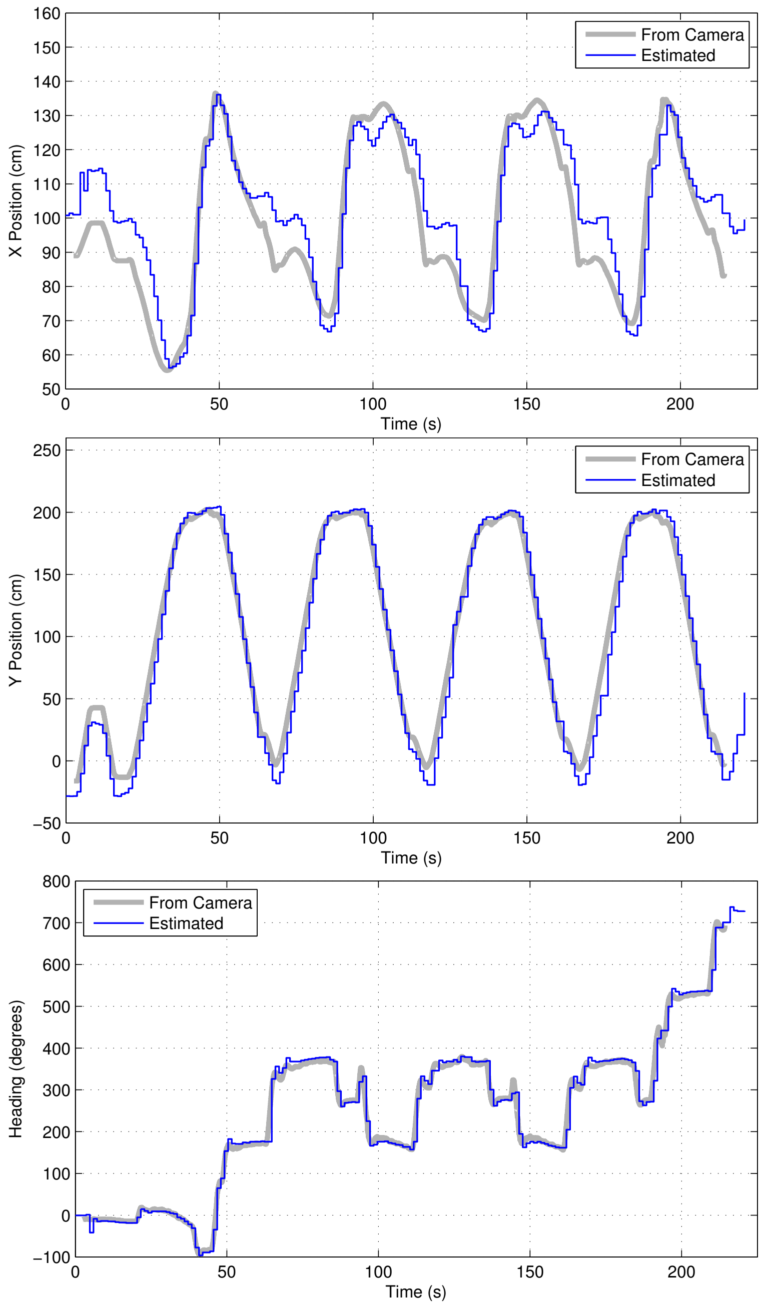


Figure 9. Estimates generated by a self localization system using the EKF together with the direct measurement model together with measurements generated by a reference camera system.

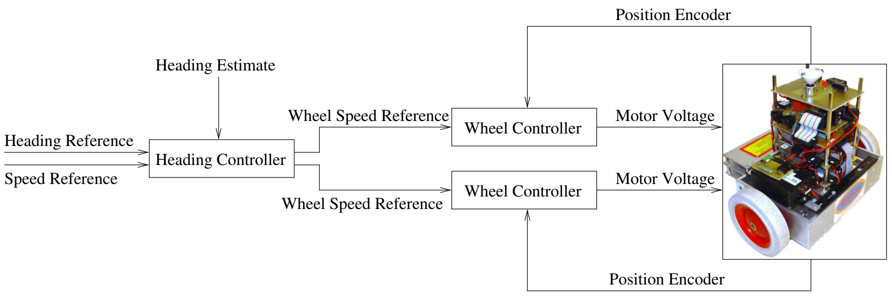


Figure 10. Hierarchical control system used to transform heading- and speed references into actual control signals for the two motors.

4. Robot Control

Many robot navigation algorithms produce heading- and speed references as their output. These reference values must be turned into actual control signals for the two motors driving the robot. In Figure 10 a hierarchical control system for thus purpose is shown. First the heading- and speed references are transformed into wheel speed references. The two wheels are then controlled individually to these references.

4.1 Heading Control

Using the same dynamical model of the heading θ as in the self localization algorithm and assuming that the wheel speeds are well controlled, a model suitable for control can be written as

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \Delta\omega\end{bmatrix} = \begin{bmatrix}0 & \frac{R}{D}\\ 0 & -\frac{1}{T_\omega}\end{bmatrix}\begin{bmatrix}\theta\\ \Delta\omega\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{T_\omega}\end{bmatrix}\Delta\omega_{\text{ref}} \tag{11}$$

Here $\Delta\omega = \omega_2 - \omega_1$ and $T_\omega \approx 0.4$ s is the time constant of the closed loop wheel control systems. Sampling the system with a sampling interval of 400 ms and designing an LQ-controller that minimizes

$$\sum_{k=0}^{\infty}(\theta(k) - \theta_{\text{ref}}(k))^2 + R^2\Delta\omega^2(k) + 10R^2\Delta\omega_{\text{ref}}^2(k) \tag{12}$$

results in a controller on the form

$$\Delta\omega_{\text{ref}} = -L_1(\theta - \theta_{\text{ref}}) - L_2\Delta\omega \tag{13}$$

Because $\dot{\theta}$ is proportional to $\Delta\omega$, the control law can be interpreted as a PD-controller. The two wheel speed references are then computed as

$$\begin{aligned}\omega_{\text{ref},1} &= \frac{v_{\text{ref}}}{R} - \frac{1}{2}\Delta\omega_{\text{ref}} \\ \omega_{\text{ref},2} &= \frac{v_{\text{ref}}}{R} + \frac{1}{2}\Delta\omega_{\text{ref}}\end{aligned} \tag{14}$$

where v_{ref} is the speed reference of the robot in m/s.

4.2 Wheel Speed Control

As mentioned in Section 3.5 the angular velocity of the two wheels are controlled by two PI-controllers operating at 100 Hz. The angular velocity is estimated from position encoders by differentiating the position signal and then using a low pass filter with a time constant of 100 ms.

In the heading control model it was assumed that the closed loop wheel speed control system has a time constant of $T_w \approx 0.4$ s. This is achieved by prefiltering the reference signal with a first order low pass filter. The time constant T_w had to be chosen large enough to prevent the wheels from slipping, which would violate the assumptions made when deriving the kinematic model (4).

5. Conclusions

The self localization and robot control system was part of a large scale demo involving a variety of hardware platforms, wireless radio technologies and operating systems. This situation very much captures what might be expected in a real world scenario. In the light of the experience draw from the demo, a number of areas, some where further development is needed, can be pointed out:

Co-existence of different wireless radio technologies need to be further investigated. In the RUNES project IEEE 802.15.4, IEEE 802.15.1 (Bluetooth) and IEEE 802.15.11 (WLAN), all operating in virtually the same frequency band, where used simultaneously. This together with the complicated indoor radio environment created by multipath propagation and the presence of people, gave rise to time variations which were virtually impossible to predict. How to develop closed loop control systems capable of handling this environment still remains a very challenging task.

When constructing distributed control/estimation systems operating on severely resource-constrained platforms the lack of distributed debugging and monitoring tools becomes evident. Normally straight forward tasks such as logging of measured signals become a problem. Using wired logging is not possible if the network covers a large geographical area and wireless logging will consume bandwidth and CPU time, thus effecting the system under study. Debugging also becomes cumbersome as many problems will only be detected after deployment. Thus, there is great potential for development of tools resolving these issues. Another option is of course simulation, but simulating wireless radios, dynamical systems and software all at the same time is no simple task.

In heterogeneous environments such as the one described above, the use of a common network technology is very important. In the RUNES project a IP based solution was used. This allowed for example sensor nodes to communicate with desktop PCs without the use of adaptor functions, that would have been necessary if for example ZigBee would have been used. The key technology that allowed this is the μIP stack [Dunkels, 2003] which is capable of running on systems with severe resource constraints.

Acknowledgement

Many thanks to Jerker Nord and Jan Edhner for their efforts in implementing, testing and debugging the software. We also thank Rolf Braun for developing the hardware. This work was supported by RUNES Integrated Project contract IST-2004-004536 and the Swedish research council.

References

Årzén, K.-E., A. Bicchi, G. Dini, S. Hailes, K. H. Johansson, J. Lygeros, and A. Tzes (2007): "A component-based approach to the design of networked control systems." *European Journal of Control*, **13:2-3**.

Cadman, J. (2003): "Deploying commercial location-aware systems." In *Proc. Fifth International Conference on Ubiquitous Computing*.

Doucet, A., N. de Freitas, and N. Gordon (2001): *Sequential Monte Carlo methods in practice*. Statistics for engineering and information science. Springer.

Dunkels, A. (2003): "Full TCP/IP for 8 Bit Architectures." In *Proceedings of the First ACM/ Usenix International Conference on Mobile Systems, Applications and Services (MobiSys 2003)*. USENIX, San Francisco.

Dunkels, A., B. Grönvall, and T. Voigt (2004): "Contiki - a lightweight and flexible operating system for tiny networked sensors." In *Proceedings of* *the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*. Tampa, Florida, USA.

Edhner, J. (2007): "Obstacle avoidance for mobile robots." Master's Thesis ISRN LUTFD2/TFRT--5803--SE. Department of Automatic Control, Lund University, Sweden.

Jazwinski, A. H. (1970): *Stochastic Processes and Filtering Theory*. Academic Press.

Julier, S. and J. Uhlmann (1997): "A new extension of the Kalman filter to nonlinear systems." In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*.

Lindberg, M. (2007): "A survey of reservation-based scheduling." Technical Report ISRN LUTFD2/TFRT--7618--SE. Department of Automatic Control, Lund University, Sweden.

Manolakis, D. E. (1996): "Efficient solution and performance analysis of 3-d position estimation by trilateration." *IEEE Transactions on Aerospace and Electronic Systems*, **32:4**, pp. 1239–1249.

Mascolo, C., S. Zachariadis, G. P. Picco, P. Costa, G. Blair, N. Bencomo, G. Coulson, P. Okanda, and T. Sivaharan (2005): "D5.2 RUNES middleware architecture." RUNES deliverable. `http://www.ist-runes.org/publicdeliverables.html`.

Mendel, J. (1971): "Computational requirements for a discrete Kalman filter." *IEEE Transactions on Automatic Control*, **16:6**, pp. 748–758.

Nordh, J. (2007): "Ultrasound-based navigation for mobile robots." Master's Thesis ISRN LUTFD2/TFRT--5789--SE. Department of Automatic Control, Lund University, Sweden.

Priyantha, N., A. Chakraborty, and H. Balakrishnan (2000): "The Cricket location-support system." In *Proc. Sixth ACM MOBICOM Conf.* Boston, MA.

Rao, C. V. (2000): *Moving Horizon Strategies for the Constrained Monitoring and Control of Nonlinear Discrete-Time Systems*. PhD thesis, University of Wisconsin-Madison.

RUNES (2007): "Reconfigurable ubiquitous networked embedded systems." `http://www.ist-runes.org`.

Smith, A., H. Balakrishnan, M. Goraczko, and N. B. Priyantha (2004): "Tracking Moving Devices with the Cricket Location System." In *2nd International Conference on Mobile Systems, Applications and Services (Mobisys 2004)*. Boston, MA.

Thomas, F. and L. Ros (2005): “Revisiting trilateration for robot localization.” *IEEE Transactions on Robotics*, **21:1**, pp. 93–102.

Want, R., A. Hopper, V. Falcao, and J. Gibbons (1992): “The Active Badge Location System.” *ACM Transactions on Information Systems*, **10:1**, pp. 91–102.

Paper V

Sub-Optimal Sensor Scheduling with Error Bounds

Peter Alriksson and Anders Rantzer

Abstract

In this paper the problem of sub-optimal sensor scheduling with a guaranteed distance to optimality is considered. Optimal in the sense that the sequence that minimizes the estimation error covariance matrix for a given time horizon is found. The search is done using relaxed dynamic programming. The algorithm is then applied to one simple second order system and one sixth order model of a fixed mounted model lab helicopter.

1. Introduction

Recently a great deal of attention has been given to the subject of wireless sensor networks. As the number of sensors in an area increases, the communication limitations imposed by bandwidth constraints will be more and more evident. Thus not allowing all sensors to communicate their measurements at each sampling interval could be very useful. Also, sensors might be battery powered and thus saving power is an essential factor. To save power a sensor can be put in stand-by mode and then woken by the estimator at certain points in time. There could also be situations where it is impossible to use two sensors at the same time due to the nature of the sensors, ultra sonic sensors is one example.

All these problems urge for algorithms that not only decide how to weight different sensors at different points in time, but also which sensors to use. How to choose which sensor or sensors to use at a specific moment is a nontrivial task studied by many others. In [Meier III *et al.*, 1967] the problem of discrete time sensor scheduling is solved by enumeration of all possible sensor schedules. The combinatorial explosion limits this approach to very short sensor schedules. A local gradient search is also proposed, but this approach doesn't guarantee that the global optimum is found. It is also shown that if the state estimates are to be used for state feedback, the plant control policy can be determined separately from the measurement schedule. The optimal sensor schedule on the other hand depends on the plant control policy. In [Chung *et al.*, 2004] an effort is made to prune the search tree by the use of a sliding window algorithm and a thresholding algorithm.

The sensor scheduling problem has also been approached from a continuous time direction. In [Athans, 1972] it is shown that the sensor scheduling problem can be transformed to a discrete-valued optimal control problem. This problem is then solved using a gradient search algorithm. In [Lee *et al.*, 2001] the discrete-valued optimal control problem of sensor scheduling is transformed into a continuous-valued optimal control problem by the use of a control parameterization enhancing transform (CPET). A method for robust sensor scheduling is developed in [Savkin *et al.*, 2001]. Here the problem with growing complexity is tackled in a model predictive way.

Another related problem studied by many others is that of choosing the time distribution of measurements with one sensor given a measurement budget. This problem is studied for discrete time systems in [Shakeri *et al.*, 1995] and for continuous time systems in [Skafidas and Nerode, 1998].

In this paper a method of choosing the sensor switching strategy as well as the Kalman estimator gain for a discrete time system is presented. The objective is to minimize a function of the estimation error covariance matrix at the final time step. The method finds a sub-optimal strategy within a pre-specified distance to optimality. The complexity of the algorithm typically increases rapidly in the first iterations, but then levels out below a constant level. The paper is organized as follows. In Section 2 the class of system to which the algorithm applies is presented and an estimator structure is proposed. In Section 3 the optimization algorithm is presented and the connection to the work by [Lincoln, 2003] is developed. Section 4 presents two examples and finally Section 5 talks about problems and future extensions.

2. Problem Formulation

Consider a discrete time system described by

$$\begin{cases} x(n+1) = Ax(n) + Bu(n) + v(n) \\ y_i(n) = C_i x(n) + e_i(n) \end{cases}$$

where $x(n) \in \mathcal{R}^n$ is the state of the process, $v(n) \in \mathcal{R}^n$ a white Gaussian stochastic process with zero mean. The system is observed through M sensor groups $i \in I = \{1 \dots M\}$ with outputs $y_i \in \mathcal{R}^{p_i}$ all disturbed by zero mean white Gaussian noise $e_i(n)$. The process noise and measurement noise has the following correlation matrix.

$$E \begin{bmatrix} v(n) \\ e_i(n) \end{bmatrix} \begin{bmatrix} v(n) \\ e_i(n) \end{bmatrix}^T = R_i$$

At each time instant the system can only be observed through one sensor group. To estimate the state of the system a Kalman filter of the following form will be used.

$$\begin{aligned} \hat{x}(n+1) &= A\hat{x}(n) + Bu(n) + K(n)\tilde{y}_{k(n)} \\ \tilde{y}_{k(n)} &= y_{k(n)} - C_{k(n)}\hat{x}(n) \end{aligned}$$

Here $k \in \kappa(n)$ denotes a specific sequence in the set $\kappa(n)$ of all possible sequences and $K \in \Lambda(n)$ a specific gain sequence in the set $\Lambda(n)$ of all possible gain sequences.. For a fixed sequence k and gain sequence K the estimation error covariance is given by equation (1) see [Åström and Wittenmark, 1997].

$$\begin{aligned} P(n+1,k,K) = (A - K(n)C_{k(n)})P(n,k)(A - K(n)C_{k(n)})^T \\ + \begin{bmatrix} I \\ -K^T(n) \end{bmatrix}^T R_{k(n)} \begin{bmatrix} I \\ -K^T(n) \end{bmatrix} \end{aligned} \quad (1)$$

The initial estimation error covariance $P(0) = P_0$ which should reflect the knowledge of the initial state. The aim is to find a switching sequence k and a gain sequence K such that the following function is minimized.

$$J(N) = \min_{\substack{k(0)\ldots k(N-1)\\ K(0)\ldots K(N-1)}} \operatorname{tr}(P(N)W) \tag{2}$$

The problem can be solved by iterating equation (1) and expanding the search tree for all possible sequences. The size of the search tree $\|\kappa(n)\|$ will however grow as M^n which makes this procedure impossible in practice.

3. Finding an α-optimal Sequence

To address the problem of increasing complexity a way of pruning the search tree has to be developed. In [Lincoln, 2003] a way of pruning the search three for the problem of choosing a switching control law is developed. As expected that problem turns out to be the dual of the problem addressed here and thus the algorithm could be used with only small modifications.

Let $\Pi(n)$ denote the set of all potentially α-optimal estimation error covariances at time step n. α-optimal means that the estimation error covariance matrix associated with the found sequence fulfills

$$\underline{\alpha} \min_{\underline{\pi}^* \in \underline{\Pi}^*} \underline{\pi}^* \leq \min_{\pi \in \Pi(n)} \pi \leq \overline{\alpha} \min_{\overline{\pi}^* \in \overline{\Pi}^*} \overline{\pi}^*$$

where $\pi^* \in \Pi^*(n)$ denotes an element in the optimal set. The initial set $\Pi(0)$ is equal to the initial estimation error covariance P_0. Further let $\kappa_{opt}(n-1)$ denote the set of corresponding sequences and $\Lambda_{opt}(n-1)$ the set of corresponding gain sequences.

To continue the iteration first define

$$\overline{P}_i(n+1) = (A - K(n)C_i)\pi(n)(A - K(n)C_i)^T + \begin{bmatrix} I \\ -K^T(n) \end{bmatrix}^T \overline{\alpha} R_i \begin{bmatrix} I \\ -K^T(n) \end{bmatrix}$$

and

$$\underline{P}_i(n+1) = (A - K(n)C_i)\pi(n)(A - K(n)C_i)^T + \begin{bmatrix} I \\ -K^T(n) \end{bmatrix}^T \underline{\alpha} R_i \begin{bmatrix} I \\ -K^T(n) \end{bmatrix}$$

where $\pi(n) \in \Pi(n)$. Then an upper $\overline{\Pi}$ and lower bound $\underline{\Pi}$ for $\Pi(n+1)$ is calculated as

$$\begin{aligned}\overline{\Pi} &= \left\{ \min_{K(n)} \overline{P}_i(n+1) \,|\, \pi(n) \in \Pi(n), i \in I \right\} \\ \underline{\Pi} &= \left\{ \min_{K(n)} \underline{P}_i(n+1) \,|\, \pi(n) \in \Pi(n), i \in I \right\}.\end{aligned} \tag{3}$$

That is for each $\pi \in \Pi$ and $i \in I$ the estimation error covariance matrix is computed by minimizing over $K(n)$. Next the set of possible sequences

$$\kappa_{cand} = \{[k \quad i] \,|\, k \in \kappa_{opt}(n-1), i \in I\}$$

is computed. The set of corresponding gain matrix sequences is then computed using Procedure 1.

```
Procedure 1
For each $\pi \in \Pi(n)$
  1. Pick the corresponding sequence $K \in \Lambda_{opt}(n-1)$
  2. For each $i \in I$
       • Calculate the minimizing
             $K(n) = \arg\min_{K(n)} \underline{P}_i(n+1)$
       • Add the concatenated sequence $[K \quad K(n)]$ to $\Lambda_{cand}$.
```

Now the objective is to find a set $\Pi(n+1)$ such that

$$\min_{\underline{\pi} \in \underline{\Pi}} \underline{\pi} \le \min_{\pi \in \Pi(n+1)} \pi \le \min_{\overline{\pi} \in \overline{\Pi}} \overline{\pi} \tag{4}$$

Together with the set $\Pi(n+1)$, a set of corresponding sequences $\kappa_{opt}(n)$ and a set of matrix gain sequences $\Lambda_{opt}(n)$ are also needed. This problem is solved by Procedure 2 which is a more detailed version of Procedure 3.2 in [Lincoln, 2003].

```
Procedure 2
  1. Sort $\overline{\Pi}$ so that
         $\mathrm{tr}\overline{\pi}_i \leq \mathrm{tr}\overline{\pi}_j \quad \forall i < j$
     $\Lambda_{cand}$ and $\kappa_{cand}$ are ordered in the same way.
  2. Let $\Pi(n+1) = \kappa_{opt}(n) = \Lambda_{opt}(n) = \oslash$
  3. Pick the first $\overline{\pi} \in \overline{\Pi}$ and remove it from $\overline{\Pi}$.
  4. If there exists x s.t.
         $x^T\overline{\pi}x < x^T\pi x \quad \forall \pi \in \Pi(n+1)$
     then
       • Pick the first $\underline{\pi} \in \underline{\Pi}$, $k \in \kappa_{cand}$ and $K \in \Lambda_{cand}$.
       • Add this $\underline{\pi}$ to $\Pi(n+1)$ and remove $\underline{\pi}$ from $\underline{\Pi}$.
       • Add this $k$ to $\kappa_{opt}$ and remove $k$ from $\kappa_{cand}$.
       • Add this $K$ to $\Lambda_{opt}$ and remove $K$ from $\Lambda_{cand}$.
       • Go to step 3
  5. Remove the first $\underline{\pi}$ from $\underline{\Pi}$.
     Remove the first $k$ from $\kappa_{cand}$.
     Remove the first $K$ from $\Lambda_{cand}$.
     If $\overline{\Pi} \neq \oslash$ go to step 3.
```

The new set of possibly α-optimal estimation error covariance matrices can now be used to compute upper and lower bounds for the next iteration. The iteration procedure can be ended when the old set fulfills (4) that is

$$\min_{\underline{\pi}\in\underline{\Pi}} \underline{\pi} \leq \min_{\pi\in\Pi(n)} \pi \leq \min_{\overline{\pi}\in\overline{\Pi}} \overline{\pi}$$

or when n=N.

The slack parameters $\overline{\alpha}$ and $\underline{\alpha}$ are used to control the tradeoff between complexity and accuracy. To find the α-optimal sequence it is enough to find the element in $\Pi(N)$ that minimizes (2) and pick the sequence associated with that estimation error covariance matrix.

4. Examples

In this section two examples will be given to illustrate the optimization algorithm presented in Section 3. First a very simple second order system will be used to show the basic principle. Then a sixth order fixed mounted helicopter from the lab at Lund Institute of Technology will be used to illustrate that the procedure is applicable to problems of higher order.

4.1 A Second Order System

Consider the following discrete second order system with two sensors

$$\begin{cases} x(n+1) = \begin{bmatrix} 0.9 & 0.009 \\ 0.009 & 0.9 \end{bmatrix} x(n) + v(n) \\ y_1(n) = \begin{bmatrix} 1 & 0 \end{bmatrix} x + e_1(n) \\ y_2(n) = \begin{bmatrix} 0 & 1 \end{bmatrix} x + e_2(n) \end{cases}$$

with noise covariance matrices

$$R_1 = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_2 = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & r \end{bmatrix} \quad P_0 = I\,.$$

The system consists of two weakly interconnected states, each observed through a separate sensor. The goal is to minimize (2) with $W = I$ and $N = 50$.

The α-optimal sequence was computed for three different values of the parameter $r = \{\,10\,, 50\,, 110\,\}$. These values correspond to sequence k_1, k_2 and k_3 in Figure 1. The slack parameters where chosen as $\overline{\alpha} = 1.01$ and $\underline{\alpha} = \overline{\alpha}^{-1}$, which gave a set $\Pi(50)$ of size 5.

As the variance of the measurement noise at sensor 2 increases the number of samples taken from sensor 2 also increases up to a certain point. For this particular example sensor 2 is not used at all for a value of $r > 104$. The singular values of the observability Gramian

$$\sigma\left(W_o\right) = \begin{bmatrix} 5.5534 \\ 0.0138 \end{bmatrix}$$

give an indication that x_2 has a low degree of observability from sensor 1. Thus the observer needs to use sensor 2 despite the large amount of measurement noise associated with it.

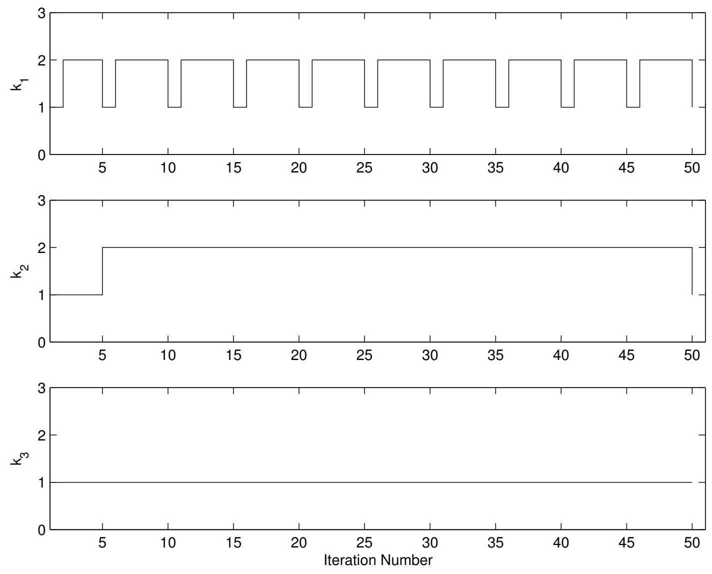


Figure 1. Sequences for measurement noise variance $r = \{ 10 , 50 , 110 \}$ at sensor two.

4.2 The Model Helicopter

In this section a sensor switching strategy for a fixed mounted model helicopter will be developed. The model helicopter is situated in the Automatic Control Lab at Lund Institute of Technology. For a detailed description of the helicopter see [Gäfvert, 2001]. The helicopter can be modeled by a sixth order system with the following state vector.

State	Description
w_1	Angular Velocity of Propeller 1
w_2	Angular Velocity of Propeller 2
ϕ	Yaw angle
$\dot{\phi}$	Yaw rate
θ	Pitch angle
$\dot{\theta}$	Pitch rate

A discrete time model with sample time $h = 0.05$ linearized around the forced equilibrium point $x^0 = [\,w_1^0 \quad w_2^0 \quad 0 \quad 0 \quad 0 \quad 0\,]^T$ is given by

$$x(n+1) = Ax + v(n).$$

The system is observed through one yaw position sensor, one pitch angle sensor and one angular velocity sensor for propeller 1. The three sensors denoted y_1, y_2 and y_3 are disturbed by white Gaussian noise.

$$\begin{bmatrix} y_1(n) \\ y_2(n) \\ y_3(n) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} x(n) + e(n)$$

The corresponding covariance matrices are

$$R_1 = R_2 = R_3 = P_0 = I.$$

The objective parameters where chosen as $N = 100$ and $W = I$. Because of the higher complexity of this problem the the slack parameters where chosen as $\overline{\alpha} = 1.5$ and $\underline{\alpha} = \overline{\alpha}^{-1}$. This resulted in a set $\Pi(100)$ of size 4.

The α-optimal sequence was computed and is given in part one of Figure 2. The sequence is periodic with the period $[\,1 \quad 2 \quad 1 \quad 2 \quad 3 \quad 2\,]$ except for the first 14 samples. The reason for this is the influence of the initial estimation error covariance P_0. The trace of the error covariance matrix $P(n)$ was also calculated for the α-optimal sequence and for a periodic sequence consisting of the triple $[\,1 \quad 2 \quad 3\,]$. As a comparison the trace of $P(n)$ was also computed for a Kalman filter which uses all three sensors at each sampling instant. These traces are found in part two and three of Figure 2.

To illustrate the pruning of sequences the size of the sequence candidate set κ is plotted in Figure 3. Here one can see the rapid increase in size the first 10 or so samples, but then instead of growing exponentially the size levels out. The size of $\kappa(n)$ never exceeds 7. If the slack parameters $\overline{\alpha}$ and $\underline{\alpha}$ are changed the shape of the complexity graph will remain the same, but the steady state level will be greater.

5. Conclusions and Future Work

In this paper, the problem of minimizing the estimation error covariance matrix at the final step for a time discrete linear system, when only one sensor group may be used at each sampling interval is considered. The problem was solved by slightly modifying the the procedure given in [Lincoln, 2003] for minimizing the quadratic cost with respect to a control

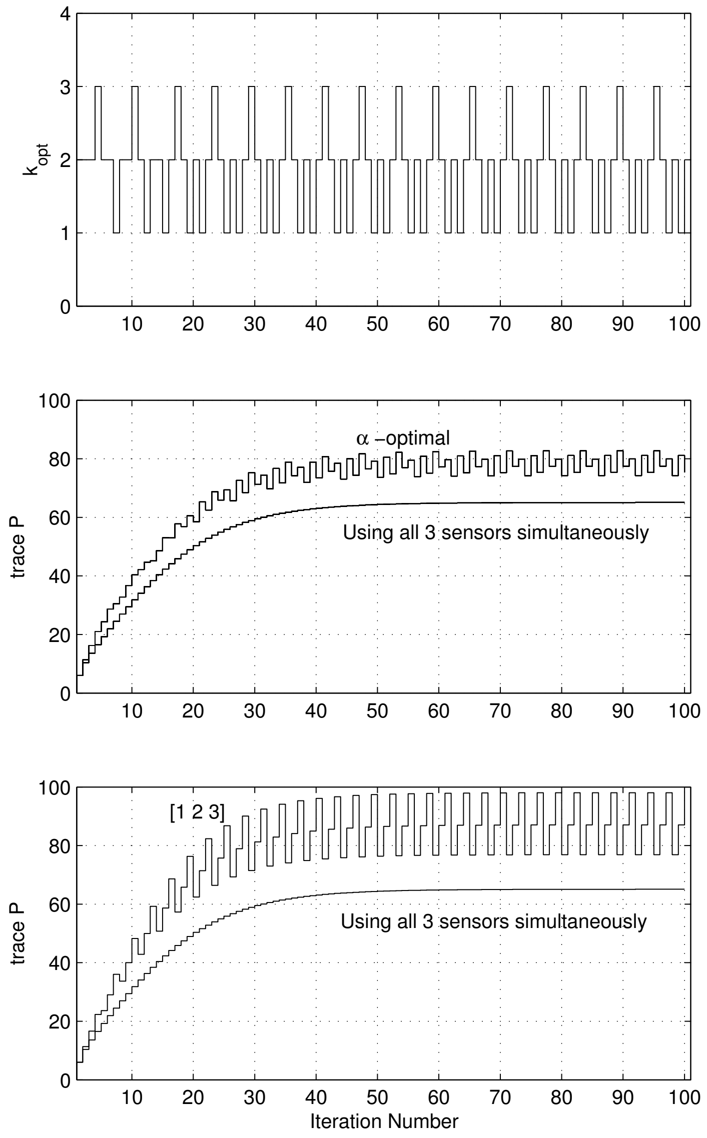


Figure 2. α-optimal sequence k_{opt} together with traces of $P(n)$ for k_{opt} and for the sequence $[1 \quad 2 \quad 3]$.

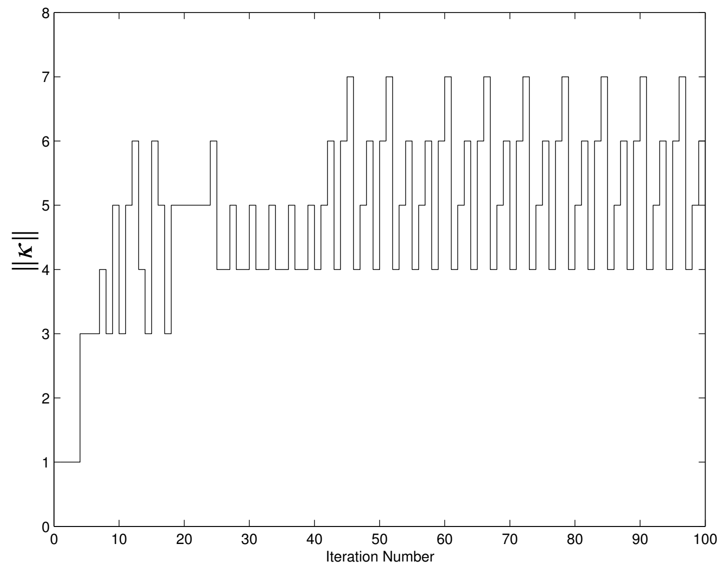


Figure 3. Complexity in terms of the size of the sequence candidate set $\kappa(n)$.

sequence. The procedure uses relaxed dynamic programming where the parameters $\overline{\alpha}$ and $\underline{\alpha}$ are used in the tradeoff between complexity and distance to optimality. The procedure was applied to two different examples, one simple second order example and one more complex sixth order example.

One drawback with the procedure presented here is that it minimizes the error covariance matrix at the final step only. Future work would include to extend the procedure to more general cost functions such as

$$J(N) = \min_{\substack{k(0)...k(N-1)\\K(0)...K(N-1)}} \sum_{0}^{N} \operatorname{tr}(P(n)W(n)).$$

This is possibly a more difficult task, because the choice of sensor and Kalman gain not only effects the covariance in the next sample, but the value of the cost function for all future samples.

Work is also going on to combine switching observers with switching controllers to achieve sub-optimal performance of control systems with limited communication abilities.

Acknowledgment

This work was supported by the European Union grant EU IST 2001-33520 – Control and Computation (CC).

References

Åström, K. J. and B. Wittenmark (1997): *Computer-Controlled Systems*. Prentice Hall.

Athans, M. (1972): “On the determination of optimal costly measurement strategies for linear stochastic systems.” *Automatica*, **8:4**, pp. 397–412.

Chung, T., V. Gupta, B. Hassibi, J. Burdick, and R. M. Murray (2004): “Scheduling for distributed sensor networks with single sensor measurement per time step.” In *2004 IEEE International Conference on Robotics and Automation, New Orleans, LA*.

Gäfvert, M. (2001): “Modelling of the eth helicopter laboratory process.” Technical Report ISRN LUTFD2/TFRT--7596--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Lee, H., K. Teo, and A. Lim (2001): “Sensor scheduling in continuous time.” *Automatica*, **37:12**, pp. 2017–2023.

Lincoln, B. (2003): *Dynamic Programming and Time-Varying Delay Systems*. PhD thesis ISRN LUTFD2/TFRT--1067--SE, Department of Automatic Control, Lund Institute of Technology, Sweden.

Meier III, L., J. Peschon, and R. M. Dressler (1967): “Optimal control of measurement subsystems.” *EEE Transactions Automatic Control*, **12**, Ocober, pp. 528–536.

Savkin, A., R. Evans, and E. Skafidas (2001): “The problem of optimal robust sensor scheduling.” *Systems and Control Letters*, **43:2**, pp. 149–157.

Shakeri, M., K. R. Pattipati, and D. L. Kleinman (1995): “Optimal measurement scheduling for state estimation.” *IEEE Transactions on Aerospace and Electronic Systems*, **31:2**, pp. 716–715.

Skafidas, E. and A. Nerode (1998): “Optimal measurement scheduling in linear quadratic gaussian control problems.” *Proceedings of the 1998 IEEE International Conference on Control Applications*, **2**, pp. 1225–1229.

Paper VI

State Estimation for Markov Jump Linear Systems using Approximate Dynamic Programming

Peter Alriksson and Anders Rantzer

Abstract

In this paper, recursive Joint Maximum a Posteriori (JMAP) state estimation for Markov jump linear systems is considered. The JMAP estimation problem is formulated as dynamic programming and then approximated using so called relaxed dynamic programming. The proposed estimator is demonstrated on the problem of estimating the state of a finite Markov chain observed in colored noise.

In the case of a switching system, that is when the mode changes in an arbitrary fashion, we also show that under suitable observability assumptions, the recursive state estimator produces an estimate that approaches the true state in the absence of noise.

Submitted to IEEE Transactions on Automatic Control, 2008. Preliminary results published in:

Alriksson, P. and A. Rantzer (2006): "Observer synthesis for switched discrete-time linear systems using relaxed dynamic programming." In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*. Kyoto, Japan.

1. Introduction

The subject of this paper is recursive state estimation for a class of discrete-time hybrid systems referred to as Markov Jump Linear Systems or MJLS for short.These systems have dynamics and measurements described by the equations:

$$\begin{aligned} x_{k+1} &= A(\theta_k)x_k + B(\theta_k)u_k + \text{noise} \\ y_k &= C(\theta_k)x_k + D(\theta_k)u_k + \text{noise} \end{aligned} \tag{1}$$

In a MJLS the discrete mode θ_k evolves according to a Markov chain. Generally the covariance of the Gaussian noise is allowed to depend on the mode. In this paper, the goal is to compute an estimate of both the continuous state x_k and the discrete mode θ_k given measurements y_k.

Markov jump linear models are used in a wide range of areas such as signal processing, target tracking, speech recognition and econometrics to name a few. A more comprehensive list of application areas are available in [Oh *et al.*, 2006]. Specifically, many signal processing problems such as de-interleaving of pulse trains, IIR channel equalisation and state estimation of a Markov chain in colored noise, can be formulated as MJLS estimation problems, see for example [Logothetis and Krishnamurthy, 1999] and the references therein. In target tracking [Mazor *et al.*, Jan 1998], constant velocity and acceleration models are often used to describe the movement of an object to be tracked.

The area of recursive state estimation for Markov jump linear systems has received considerable attention in the literature during the last three decades, starting with the paper by Ackerson and Fu [Ackerson and Fu, 1970]. However, the optimal filter derived in [Ackerson and Fu, 1970] requires infinite memory and thus much attention has been devoted to deriving finite memory approximations. Perhaps the most popular approach, at least for target tracking, is the Interacting Multiple Model (IMM) algorithm proposed in [Blom, 1984]. The problem has also been approached using simulation based methods such as particle filters [Doucet *et al.*, 2001]. In [Costa, 1994] the best (in mean square sense) linear filter was derived and it was shown to be of dimension nM, where n is the size of x_k and M is the number of modes.

In [Logothetis and Krishnamurthy, 1999] a non-recursive estimator based on the expectation maximization (EM) algorithm was proposed. It was also shown that many estimators based on Kalman filter banks plus mode decision logic can be interpreted as recursive implementations of the EM-algorithm.

1.1 Switching Systems

A closely related class of systems is what is often referred to as Switching Systems. In a switching system, the mode θ_k, is assumed to change in an arbitrary fashion. The switching systems literature has mainly focused on noise-free dynamics with stability of the estimation error as a central concept.

In [Alessandri and Coletta, 2001] the modes are assumed to be known and a classical Luenberger observer is designed for the switching system using an LMI formulation. The assumption of known modes is relaxed in [Babaali *et al.*, 2004] and [Babaali and Egerstedt, 2005], where observers are derived using linear algebra methods. In [Balluchi *et al.*, 2005] and [Balluchi *et al.*, 2002] the mode is estimated by comparing the residuals from a bank of Luenberger observers. The continuous state is then estimated using a Luenberger observer for the resulting time-varying linear system, where the uncertainty in the mode estimate is ignored.

The problem has also been approached from a receding horizon point of view. In [Bemporad *et al.*, 1999] the moving horizon estimation problem is solved using mixed integer quadratic program solvers. Recently, in [Alessandri *et al.*, 2005b] linear algebra methods where used to form a set of possible mode sequences, each for which a quadratic program was solved. The method was simplified further in [Alessandri *et al.*, 2005a].

When designing observers for switching systems the concept of observability plays a central role. The notion of observability for a switching system is , however, far more complex than for linear systems. The concept has been treated in numerous papers including [Babaali and Egerstedt, 2004] and [Vidal *et al.*, 2002].

1.2 Contributions and Organization

The main contribution of this paper is to revisit the formulation of joint maximum a posteriori estimation as dynamic programming and use recent advances in approximate dynamic programming to construct recursive state estimators. This general idea is then demonstrated on both Markov jump linear and switching systems.

In the case of switching systems, we also show that under suitable observability assumptions, the recursive state estimator produces an estimate $\hat{x}_k$ that approaches the true state x_k in the absence of noise. Preliminary results have been published in [Alriksson and Rantzer, 2006].

This paper is organized as follows. In Section 2 the problem is formally stated and some notation is introduced. Sections 3 and 4 give an introduction to recursive state estimation and present the main algorithm. In Section 5 a numerical example is given. Section 6 treats the special case of switching systems, for which a stability result is presented. Finally some concluding remarks are given.

2. Problem Statement

The system under consideration is

$$\begin{aligned} x_{k+1} &= A(\theta_k)x_k + B(\theta_k)u_k + w_k \\ y_k &= C(\theta_k)x_k + D(\theta_k)u_k + v_k \end{aligned} \tag{2}$$

where $x_k \in \mathbf{R}^n$ denotes the continuous state, $u_k \in \mathbf{R}^m$ a known input and $w_k \in \mathbf{R}^n$ a process disturbance. The mode of the system at time k is denoted θ_k and takes values from a finite set $I = \{1, \dots, M\}$. The system is observed through a continuous measurement $y_k \in \mathbf{R}^p$ which is corrupted by a measurement disturbance $v_k \in \mathbf{R}^p$.

The process and measurement disturbances are assumed to be white zero mean Gaussian stochastic processes with covariance $R_w(\theta_k) \in \mathbf{R}^{n\times n}$ and $R_v(\theta_k) \in \mathbf{R}^{p\times p}$ respectively. It is also assumed that w_k and v_k are independent. The initial state x_0 is assumed Gaussian with covariance $P_0 \in \mathbf{R}^{n\times n}$ and mean $\check{x}$. The mode variable θ_k is modeled by a finite state Markov chain with initial probabilities $p(\theta_0)$ and transition probabilities $p(\theta_{k+1}|\theta_k)$.

The goal of this work is to develop a recursive state estimator for θ_k and x_k given the sequence of measurements $\{y_0, \dots, y_l\}$. Here we will consider both filtering $k = l$ and smoothing $k < l$.

2.1 Notation

First let

$$X_{i:j} = \{x_i, \dots, x_j\} = \{x_k\}_{k=i}^{j}$$

denote the sequence of continuous states at time i up to time j. Further let $X_j = X_{0:j}$ and define Θ_j and Y_j in the same way. Throughout this paper we will make frequent use of quadratic forms

$$\|x\|_Q^2 = x^T Q x$$

where x^T denotes matrix transpose. When dealing with stability in the case of switching systems, it is convenient to first introduce the following notation

$$\tilde{A}(\Theta_{i:j}) = \begin{cases} A(\theta_j)A(\theta_{j-1})\dots A(\theta_i) & j \geq i \\ I & j < i \end{cases}$$

$$\mathcal{A}(\Theta_{N-2}) = \begin{bmatrix} \tilde{A}(\Theta_{-1}) \\ \vdots \\ \tilde{A}(\Theta_{N-2}) \end{bmatrix} = \begin{bmatrix} I \\ \vdots \\ A(\theta_{N-2})\dots A(\theta_0) \end{bmatrix}$$

$$\mathcal{C}(\Theta_{N-1}) = \begin{bmatrix} C(\theta_0) & & \\ & \ddots & \\ & & C(\theta_{N-1}) \end{bmatrix}$$

$$\mathcal{O}(\Theta_{N-1}) = \mathcal{C}(\Theta_{N-1})\mathcal{A}(\Theta_{N-2})$$

3. Recursive State Estimation

The recursive state estimation problem can be approached in many ways. Here we will adopt a statistical Bayesian point of view.

Perhaps the most common way to proceed is to develop recursive equations for the conditional probability $p(x_k, \theta_k | Y_k)$ of the state. Various estimates such as Minimum Mean Square Error (MMSE) or Maximum a Posteriori (MAP), together with measures of uncertainty can then be computed from this quantity. The recursive equations typically involve integration (summation) over the state variables at each time step. Solving these integrals for a general setup is often very difficult. Instead simulation based methods such as particle filters [Doucet *et al.*, 2001] are often used.

An alternative is to construct estimates using the joint probability $p(X_k, \Theta_k | Y_k)$ of the whole trajectory given all measurements. To get a recursive estimator we will restrict the problem to computing the most probable state sequence

$$\arg \max_{X_k, \Theta_k} p(X_k, \Theta_k | Y_k)$$

This approach will be referred to as recursive Joint Maximum a Posteriori (JMAP) state estimation. Using forward dynamic programming, this problem can in principle be solved recursively as demonstrated in Section 4. The recursive procedure encounters the same difficulties as general dynamic programming, namely that of an explosion of computational complexity. Recently this problem has been addressed in a receding horizon fashion, giving rise to the moving horizon estimation (MHE) framework, see for example [Rao, 2000] for a detailed treatment. In this paper we instead use relaxed dynamic programming [Lincoln and Rantzer, 2006] to handle the complexity explosion.

Note that, maximizing $p(x_k, \theta_k | Y_k)$ will in general not yield the same estimate as the x_k obtained from maximizing $p(X_k, \Theta_k | Y_k)$, see for example Figure 1 or Chapter 9 in [Goodwin *et al.*, 2005] for a thorough treatment.

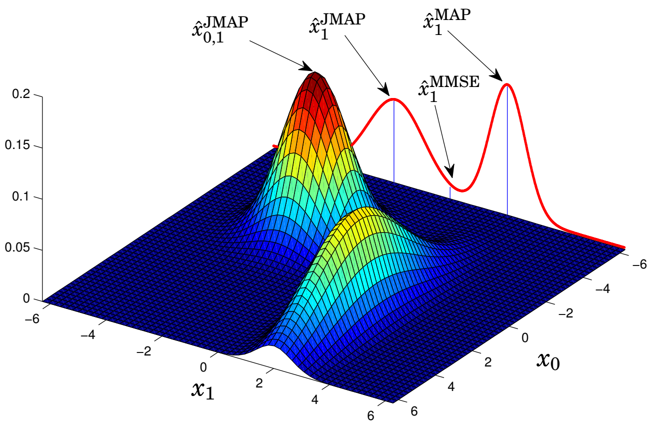


Figure 1. Two-dimensional distribution $p(x_0, x_1)$ together with its marginalization $p(x_1)$, with three common point estimates indicated. Note that the last element $\hat{x}_k^{\text{JMAP}}$ of the sequence $\hat{X}_k^{\text{JMAP}}$ in general does not coincide with $\hat{x}_k^{\text{MAP}}$.

However, for the special case of a linear system with Gaussian noise both approaches give the well known Kalman filter, see [Ho and Lee, 1964] and [Cox, 1964] for derivations.

4. Recursive JMAP State Estimation

In this section we will show how to transform the JMAP problem to a recursive state estimation problem using forward dynamic programming. The presentation follows the ideas in [Larson and Peschon, 1966]. Recall that the problem we want to solve is

$$\{\hat{X}_k, \hat{\Theta}_k\} = \arg\max_{X_k, \Theta_k} p(X_k, \Theta_k | Y_k) \tag{3}$$

First define

$$I_k(x_x, \theta_k) = \max_{X_{k-1}, \Theta_{k-1}} p(X_k, \Theta_k | Y_k) \tag{4}$$

To get a recursive formulation, Bayes' theorem is used to express I_k in terms of I_{k-1}. First rewrite the conditional probability using Bayes theorem as

$$p(X_k, \Theta_k | Y_k) = \frac{p(y_k | X_k, \Theta_k, Y_{k-1}) p(X_k, \Theta_k | Y_{k-1})}{p(y_k | Y_{k-1})} \tag{5}$$

Using the Markov property of the system, (5) can be written as

$$p(X_k,\Theta_k|Y_k) = \frac{p(y_k|x_k,\theta_k)p(x_k,\theta_k|x_{k-1},\theta_{k-1})p(X_{k-1},\Theta_{k-1}|Y_{k-1})}{p(y_k|Y_{k-1})} \tag{6}$$

Now using the Markov jump assumption we can write

$$\begin{aligned} p(x_k,\theta_k|x_{k-1},\theta_{k-1}) &= p(x_k|\theta_k,x_{k-1},\theta_{k-1})p(\theta_k|x_{k-1},\theta_{k-1}) \\ &= p(x_k|\theta_{k-1},x_{k-1})p(\theta_k|\theta_{k-1}) \end{aligned} \tag{7}$$

We can now write (6) as

$$p(X_k,\Theta_k|Y_k) = \frac{p(y_k|x_k,\theta_k)p(x_k|\theta_{k-1},x_{k-1})p(\theta_k|\theta_{k-1})p(X_{k-1},\Theta_{k-1}|Y_{k-1})}{p(y_k|Y_{k-1})} \tag{8}$$

and thus

$$\begin{aligned} &I_k(x_k,\theta_k) = \\ &= \max_{X_{k-1},\Theta_{k-1}} \frac{p(y_k|x_k,\theta_k)p(x_k|\theta_{k-1},x_{k-1})p(\theta_k|\theta_{k-1})p(X_{k-1},\Theta_{k-1}|Y_{k-1})}{p(y_k|Y_{k-1})} = \\ &= \max_{x_{k-1},\theta_{k-1}} \frac{p(y_k|x_k,\theta_k)p(x_k|\theta_{k-1},x_{k-1})p(\theta_k|\theta_{k-1})}{p(y_k|Y_{k-1})} I_{k-1}(x_{k-1},\theta_{k-1}) \end{aligned} \tag{9}$$

The denominator of (9) is a normalization constant independent of the state, thus the recursive equation

$$\overline{I}_k(x_k,\theta_k) = \max_{x_{k-1},\theta_{k-1}} p(y_k|x_k,\theta_k)p(x_k|\theta_{k-1},x_{k-1})p(\theta_k|\theta_{k-1})\overline{I}_{k-1}(x_{k-1},\theta_{k-1}) \tag{10}$$

can be used instead. The iteration is initiated with

$$\overline{I}_0(x_0,\theta_0) = p(y_0|x_0,\theta_0)p(x_0)p(\theta_0) \tag{11}$$

4.1 Value Iteration

The procedure outlined in Section 4 applies to general Markov models. Next we will make use of the special structure of the problem studied here. If either $A(\theta_k)$ or $R_w(\theta_k)$ are invertible the recursive JMAP problem can be transformed into a value iteration problem. Here we will pursue the latter.

First using the Gaussian noise assumption we find that

$$\begin{aligned} p(y_k|x_k,\theta_k) &= \frac{1}{(2\pi)^{p/2}\sqrt{\det R_v(\theta_k)}} \exp\left\{-\frac{1}{2}\left\|v_k\right\|^2_{R_v^{-1}(\theta_k)}\right\} \\ v_k &= y_k - C(\theta_k)x_k - D(\theta_k)u_k \end{aligned} \tag{12}$$

and

$$\begin{aligned} p(x_{k+1}|x_k,\theta_k) &= \frac{1}{(2\pi)^{n/2}\sqrt{\det R_w(\theta_k)}} \exp\left\{-\frac{1}{2}\,\|w_k\|^2_{R_w^{-1}(\theta_k)}\right\} \\ w_k &= x_{k+1} - A(\theta_k)x_k - B(\theta_k)u_k \end{aligned} \qquad (13)$$

To simplify the maximization required for computing $\overline{I}(x_{k+1},\theta_{k+1})$ define the value function

$$V^{\text{opt}}_{k+1}(x_{k+1},\theta_{k+1}) = -\log \overline{I}_{k+1}(x_{x+1},\theta_{k+1}) \qquad (14)$$

The maximization problem now becomes a minimization problem on the form

$$V^{\text{opt}}_{k+1}(x_{k+1},\theta_{k+1}) = \min_{x_k,\theta_k}\left\{V^{\text{opt}}_k(x_k,\theta_k) + L_k(x_{k+1},x_k,\theta_{k+1},\theta_k)\right\} \qquad (15)$$

with initial conditions

$$\begin{aligned} V^{\text{opt}}_0(x_0,\theta_0) &= \frac{1}{2}\|x_0 - \check{x}_0\|^2_{P_0^{-1}} + \frac{1}{2}\|v_0\|^2_{R_v^{-1}(\theta_0)} + K(\theta_0) \\ K(\theta_0) &= \frac{1}{2}\log\det R_v(\theta_0) - \log p(\theta_0) \end{aligned} \qquad (16)$$

The step cost L_k becomes

$$\begin{aligned} L_k(x_{k+1},x_k,\theta_{k+1},\theta_k) &= \frac{1}{2}\|v_{k+1}\|^2_{R_v^{-1}(\theta_{k+1})} + \frac{1}{2}\|w_k\|^2_{R_w^{-1}(\theta_k)} + K(\theta_{k+1},\theta_k) \\ K(\theta_{k+1},\theta_k) &= \frac{1}{2}\log\det R_v(\theta_{k+1}) + \frac{1}{2}\log\det R_w(\theta_k) - \log p(\theta_{k+1},\theta_k) + \tilde{K} \end{aligned} \qquad (17)$$

Here $\tilde{K}$ is a constant chosen in such a way that

$$\min_{\theta_{k+1},\theta_k} K(\theta_{k+1},\theta_k) = 0 \qquad (18)$$

The reason for this choice will be clear when the value iteration procedure is approximated.

4.2 Value Function Parametrization

When choosing a value function parametrization, it is essential that the new value function can be parametrized in the same way. To achieve this, first assume a value function on the form

$$V^{\text{opt}}_k(x_k,\theta_k) = \min_{\pi\in\Pi^{\text{opt}}_k(\theta_k)} \begin{bmatrix} x_k \\ 1 \end{bmatrix}^T \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{12}^T & \pi_{22} \end{bmatrix} \begin{bmatrix} x_k \\ 1 \end{bmatrix} \qquad (19)$$

Here Π_k^{opt} denotes a set of matrices and $\Pi_k^{\text{opt}}(\theta_k)$ are disjoint subsets depending on θ_k. Next note that the step cost L_k can be written as

$$L_k(x_{k+1}, x_k, \theta_{k+1}, \theta_k) = \left\| \Lambda(\theta_k, \theta_{k+1}) \begin{bmatrix} x_{k+1} \\ 1 \\ x_k \end{bmatrix} \right\|^2_{Q(\theta_k, \theta_{k+1})} \tag{20}$$

We can now write the new value function as

$$V_{k+1}^{\text{opt}}(x_{k+1}, \theta_{k+1}) = \min_{\theta_k, \pi \in \Pi_k^{\text{opt}}(\theta_k)} \min_{x_k} \begin{bmatrix} x_{k+1} \\ 1 \\ x_k \end{bmatrix}^T U(\theta_k, \theta_{k+1}, \pi) \begin{bmatrix} x_{k+1} \\ 1 \\ x_k \end{bmatrix} \tag{21}$$

Minimizing over x_k and θ_k gives a new value function on the same form

$$V_{k+1}^{\text{opt}}(x_{k+1}, \theta_{k+1}) = \min_{\pi \in \Pi_{k+1}^{\text{opt}}(\theta_{k+1})} \begin{bmatrix} x_{k+1} \\ 1 \end{bmatrix}^T \pi \begin{bmatrix} x_{k+1} \\ 1 \end{bmatrix} \tag{22}$$

Expressions for $\Lambda(\theta_k, \theta_{k+1})$, $Q(\theta_k, \theta_{k+1})$, $U(\theta_k, \theta_{k+1}, \pi)$ and $\pi \in \Pi_{k+1}^{\text{opt}}(\theta_{k+1})$ are given in Appendix A.1. Because the new value function is on the same form, it is theoretically possible to continue the value iteration. However, the size of the set Π_{k+1}^{opt} will in general be M times larger than Π_k^{opt}. Thus the size grows exponentially with k. The fact that $K(\theta_{k+1}, \theta_k) \geq 0$ by construction ensures that all $\pi \in \Pi_{k+1}^{\text{opt}}$ are positive semi-definite. This property will be useful when the exact value iteration is approximated next.

4.3 Relaxed Value Iteration

As proposed in [Lincoln and Rantzer, 2006] the value iteration (15) can be relaxed if the Bellman equality is replaced by two inequalities instead. First define upper and lower bounds on V_{k+1} as

$$\begin{aligned} \overline{V}_{k+1}(x_{k+1}, \theta_{k+1}) &= \min_{x_k, \theta_k} \{V_k(x_k, \theta_k) + \overline{\alpha} L_k(x_{k+1}, x_k, \theta_{k+1}, \theta_k)\} \\ \underline{V}_{k+1}(x_{k+1}, \theta_{k+1}) &= \min_{x_k, \theta_k} \{V_k(x_k, \theta_k) + \underline{\alpha} L_k(x_{k+1}, x_k, \theta_{k+1}, \theta_k)\} \end{aligned} \tag{23}$$

Now it is possible to replace (15) with the two inequalities

$$\underline{V}_{k+1}(x_{k+1}, \theta_{k+1}) \leq V_{k+1}(x_{k+1}, \theta_{k+1}) \leq \overline{V}_{k+1}(x_{k+1}, \theta_{k+1}) \tag{24}$$

Here the scalars $\overline{\alpha} > 1$ and $\underline{\alpha} < 1$ are slack parameters that can be chosen to trade off optimality against complexity. By the introduction of inequalities instead of equalities it is in principle possible to fit a simpler value function function between the upper and lower bounds.

If, in each step, (24) holds with the upper ($\overline{V}_{k+1}$) and lower ($\underline{V}_{k+1}$) bounds computed as in (23), the obtained solution will satisfy

$$\underline{\alpha} V_k^{\text{opt}}(x_k, \theta_k) \leq V_k(x_k, \theta_k) \leq \overline{\alpha} V_k^{\text{opt}}(x_k, \theta_k) \tag{25}$$

which gives guarantees on how far from optimal the approximate solution is.

The approximate value function can be parametrized in many ways, as long as there exist methods for computing (23) and finding a V_{k+1} satisfying (24). How to choose a good parametrization of the relaxed value function for a state feedback control setup has recently been studied in [Wernrud, 2008].

If V_k is parametrized in the same way as V_k^{opt}, the approximate value function V_{k+1} can be constructed by selecting matrices from the lower bound $\underline{V}_{k+1}$ until (24) is satisfied, see Figure 2 for an illustration. Note that adding a matrix to the set Π_{k+1} decreases the overall function value because of the parametrization used. For a detailed discussion on how to construct V_{k+1} see for example Procedure 1 in [Lincoln and Rantzer, 2006]. The fact that all $\pi \in \Pi_k$ are positive semi-definite simplifies the procedure for testing (24).

4.4 Computing Estimates

Filtered estimates, that is $\hat{x}_{k|k}$ and $\hat{\theta}_{k|k}$, given measurements up to and including y_k can be computed by minimizing the value function $V_k(x_k, \theta_k)$.

$$\{\hat{x}_{k|k}, \hat{\theta}_{k|k}\} = \arg\min_{x_k, \theta_k} V_k(x_k, \theta_k) \tag{26}$$

To compute an estimate of the full trajectory $\{X_k, \Theta_k\}$ we can backtrack using previously computed value functions

$$\{\hat{x}_{i-1|k}, \hat{\theta}_{i-1|k}\} = \arg\min_{x_{i-1}, \theta_{i-1}} \left\{V_{i-1}(x_{i-1}, \theta_{i-1}) + \underline{\alpha} L_{i-1}(\hat{x}_{i|k}, x_{i-1}, \hat{\theta}_{i|k}, \theta_{i-1})\right\} \tag{27}$$

Because $\hat{x}_{i|k}$ is fixed, terms independent of x_{i-1} and θ_{i-1} can be removed from L_{i-1}. Thus the problem reduces to

$$\{\hat{x}_{i-1|k}, \hat{\theta}_{i-1|k}\} = \arg\min_{x_{i-1}, \theta_{i-1}} \left\{V_{i-1}(x_{i-1}, \theta_{i-1}) + \underline{\alpha} \tilde{L}_{i-1}(\hat{x}_{i|k}, x_{i-1}, \hat{\theta}_{i|k}, \theta_{i-1})\right\} \tag{28}$$

where

$$\tilde{L}_{i-1}(\hat{x}_{i|k}, x_{i-1}, \hat{\theta}_{i|k}, \theta_{i-1}) = \frac{1}{2}\|\hat{x}_{i|k} - A(\theta_{i-1})x_{i-1} - B(\theta_{i-1})u_{i-1}\|^2_{R_w^{-1}(\theta_{i-1})} + K(\hat{\theta}_{i|k}, \theta_{i-1}) \tag{29}$$

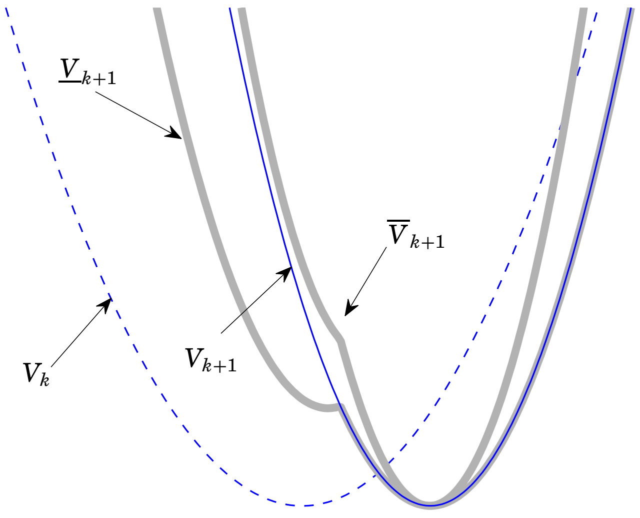


Figure 2. 1-D illustration of how the new approximate value function V_{k+1} is constructed by adding functions from the lower bound until it fits between the bounds. The functions are plotted for a fixed θ.

Note that $\tilde{L}_{i-1}$ does not depend on y_i, thus there is no need to save old measurements. The reason for including $\underline{\alpha}$ in (27) and (26) will be clear from the discussion on stability for switching systems in Section 6.

5. Simulation Results

In this section, we will demonstrate the proposed recursive state estimator on the problem of estimating the state of a Markov chain in colored noise. This problem was also studied in [Logothetis and Krishnamurthy, 1999], from which the parameters are taken. The observed output is

$$y_k = e_k + r_k + v_k \tag{30}$$

where r_k is the Markov chain output $\pm q$, v_k is white Gaussian noise and e_k is colored Gaussian noise. The problem can be formulated as a MJLS on the form (2) with $A(\theta_k) = A$ and $C(\theta_k) = C$ describing the colored noise $e_k = Cx_k$, $B(\theta_k) = 0$, $u_k \equiv 1$ and

$$D(\theta_k) = \begin{cases} q & \text{if } \theta_k = 1 \\ -q & \text{if } \theta_k = 2 \end{cases} \tag{31}$$

As in [Logothetis and Krishnamurthy, 1999] we will use a fourth order model for the colored noise e_k with

$$\begin{aligned}
A &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\
C &= \begin{bmatrix} 1 & -3.83 & 2.62 & -1.15 \end{bmatrix} \\
R_w &= \operatorname{diag}\left(\begin{bmatrix} 10 & 1 & 1 & 1 \end{bmatrix}\right) \cdot 10^{-3} \\
R_v &= 10^{-6}
\end{aligned} \tag{32}$$

Compared to [Logothetis and Krishnamurthy, 1999] some additional noise has been added to make R_w positive definite. In all simulations, the algorithm was run on 300 data points and the results were averaged over a number of independent runs to achieve good confidence levels. In the case of noticeable uncertainty, 95% confidence intervals are indicated in the figures by shaded lines.

In the following three sections, performance in terms of error probability is investigated. The first two scenarios are also studied in [Logothetis and Krishnamurthy, 1999] whereas in the third, relaxation level effects specific to the proposed approximation method are investigated. Two different estimation schemes were considered: fixed interval (of length 1) smoothing $\hat{x}_{k-1|k}$ and as in [Logothetis and Krishnamurthy, 1999] full smoothing $\hat{x}_{k|300}$.

The overall performance of the full smoother $\hat{x}_{k|300}$ was slightly worse than what was reported by the best batch algorithm in Figure 5 in [Logothetis and Krishnamurthy, 1999]. This is expected, however, since a batch algorithm should perform at least as good as a recursive one. Also, the signal to noise ratio was slightly lower in our case due to the extra noise added to make R_w positive definite.

5.1 Signal-to-Noise Ratio Effects

The Markov chain level q was varied from 0.1 to 1 with the dwell probability $p(\theta_{k+1} = i | \theta_k = i)$ fixed at 0.9. The relaxation levels were set to $\underline{\alpha} = 1$ and $\overline{\alpha} = 1.1$. From Figure 3 we can make the following observations:

1. The error probability decreases with increased Markov chain level q as expected. This is a simple implication of the increased signal-to-noise ratio.

2. The variation in performance improvement due to full smoothing over one step smoothing is still unexplained.

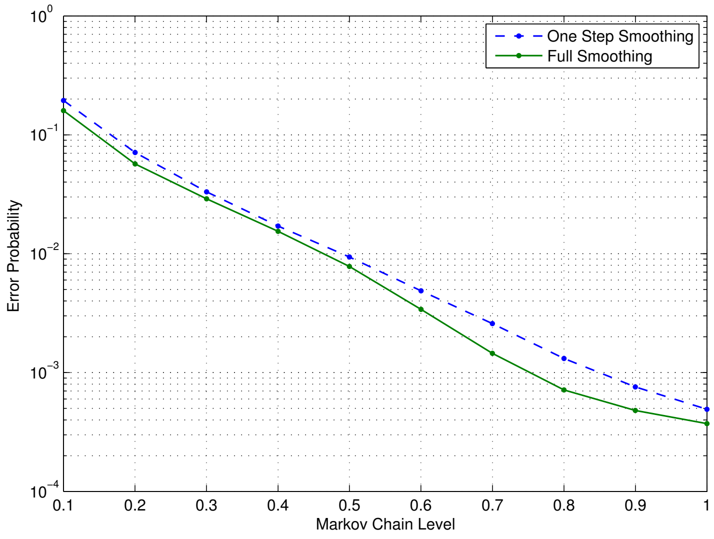


Figure 3. Error probability as a function of the Markov chain level q for the one step smoother $\hat{x}_{k-1|k}$ and the full smoother $\hat{x}_{k|300}$.

5.2 Transition Probability Effects

The Markov chain level was fixed at $q = 0.1$ and the dwell probability $p(\theta_{k+1} = i|\theta_k = i)$ was varied from 0.5 to 0.975. As in the previous setup the relaxation levels were set to $\underline{\alpha} = 1$ and $\overline{\alpha} = 1.1$. From Figure 4 we make the following observations:

1. As expected, the error probability decreases with increased dwell probability. This allows the estimator to average over a longer period, thus reducing the noise level.
2. The performance improvement due to full smoothing over one step smoothing increases with increased dwell probability.

5.3 Relaxation Level Effects

The Markov chain level was fixed at $q = 0.5$ and the dwell probability was fixed at $p(\theta_{k+1} = i|\theta_k = i) = 0.75$ while the relaxation level $\overline{\alpha}$ was varied from 1.1 to 100. The lower limit $\underline{\alpha}$ was held at 1. In addition to the error probability for the full smoother, the complexity measured as the number of matrices in Π_k, was also studied. From Figure 5 we make the following observations:

1. As expected, the complexity decreases with increased relaxation level almost everywhere. The reason for the non-monotonic decrease is the

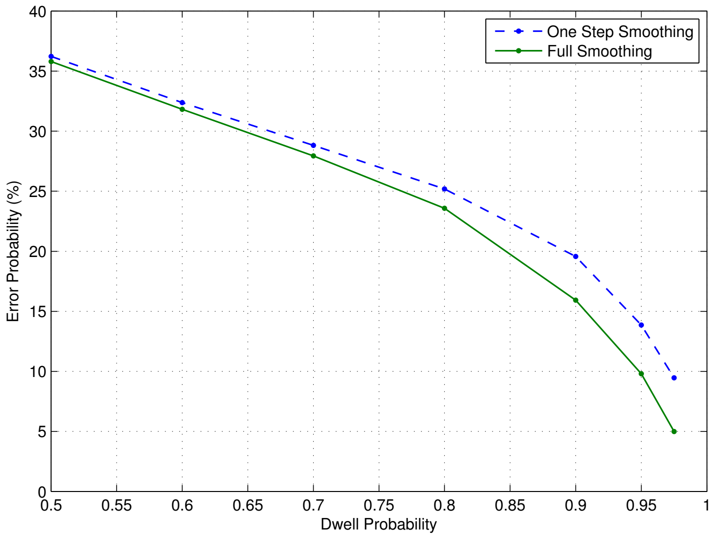


Figure 4. Error probability as a function of the dwell probability for the one step smoother $\hat{x}_{k-1|k}$ and the full smoother $\hat{x}_{k|300}$.

fact that a lower relaxation level might force the estimator too keep a function that will later prove useful to reduce the complexity.

2. Also for the error probability the general trend is that it increases with an increased relaxation level, but here the non-monotonic behaviour is even more evident.

3. Notice how a complexity reduction from more than 40 to 20 functions barely effects the error probability.

6. Switching Systems

Switching systems is a sub-class of the general class presented in Section 2. In a Switching System, the mode θ_k is assumed to change in an arbitrary fashion. Here we also assume that the noise covariance matrices don't depend on the mode θ_k. This implies that the constant $K(\theta_k, \theta_{k+1})$ will not depend on the mode and thus it will be equal to zero by construction. We will also restrict the discussion to the case when $u_k = 0$. Under

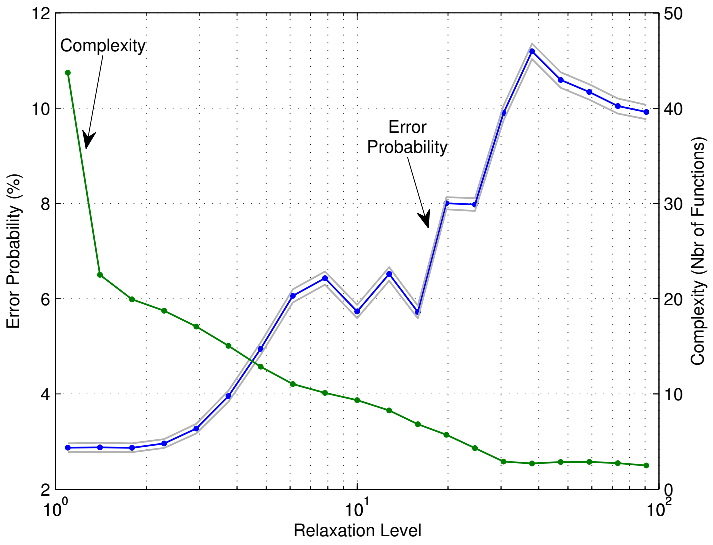


Figure 5. Error probability for the full smoother $\hat{x}_{k|300}$ and complexity as a function of relaxation level $\overline{\alpha}$. The shaded regions indicate 95% confidence intervals.

these assumptions the step cost becomes:

$$\begin{aligned} L_k(x_{k+1}, x_k, \theta_{k+1}, \theta_k) &= \|v_{k+1}\|^2_{R_v^{-1}} + \|w_k\|^2_{R_w^{-1}} \\ v_{k+1} &= y_{k+1} - C(\theta_{k+1})x_{k+1} \\ w_k &= x_{k+1} - A(\theta_k)x_k \end{aligned} \tag{33}$$

6.1 Stability

Switched systems have two very different types of states, the continuous state x_k and the discrete mode θ_k, thus what is meant by stability needs to be specified. In this section, the properties of the continuous state estimate will be studied. Particularly the following type of stability will be addressed:

DEFINITION 1
An estimator is an asymptotically stable observer for the noise-free system

$$\begin{aligned} x_{k+1} &= A(\theta_k)x_k \\ y_k &= C(\theta_k)x_k \end{aligned} \tag{34}$$

if there exists an integer $T_s \geq 0$ such that for every initial condition x_0 and mode sequence Θ_{T+T_s} the estimate $\hat{x}_{T|T+T_s} \to x_T$ as $T \to \infty$. □

Next, a few concepts concerning observability need to be established. The field of observability for switched systems is not as well developed as for linear systems and thus there are quite a few notions of observability. Here we will use a definition of observability similar to the one of State Observability in [Babaali and Egerstedt, 2004].

DEFINITION 2
The system (34) is said to be (N_x, N)-observable if there exists a non-negative integer N and a finite $\kappa(i)$ such that

$$\|\tilde{A}(\Theta_{i-1})x_0 - \tilde{A}(\hat{\Theta}_{i-1})\hat{x}_0\|^2 \leq \kappa(i)\|O(\Theta_{N-1})x_0 - O(\hat{\Theta}_{N-1})\hat{x}_0\|^2 \tag{35}$$

holds for all $x_0, \hat{x}_0, \Theta_{N-1}, \hat{\Theta}_{N-1}$ and $0 \leq i \leq N_x$. □

Checking $(0,N)$-observability is equivalent to checking State Observability as defined in [Babaali and Egerstedt, 2004], which is shown to be decidable. To the best of our knowledge, the only way of checking for general (N_x, N)-observability is by directly checking (35) for all possible mode sequences of length N. The numbers N and N_x can be interpreted in the following way: If the output is observed from 0 to $N-1$, the state sequence $x_0, \ldots, x_{N_x}$ can be uniquely determined for all mode sequences, regardless if they were estimated correctly or not.

EXAMPLE 1
The second order system with two modes

$$\begin{aligned} A(1) &= \begin{bmatrix} 0.8 & 0.9 \\ -0.9 & 0.5 \end{bmatrix} & A(2) &= \begin{bmatrix} 0.8 & 0.9 \\ -0.9 & 0 \end{bmatrix} \\ C(1) &= \begin{bmatrix} 1 & 0 \end{bmatrix} & C(2) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \end{aligned} \tag{36}$$

is $(0,2)$-observable with $\kappa(0) \approx 2.54$. In this particular example $O(\Theta)$ does not depend on Θ and thus $\kappa(0) = \lambda_{\min}^{-1}(O^T O)$. For $i = 1$ for example $x_0 = \hat{x}_0 = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$ makes the right hand side of (35) equal to zero whereas the left hand side becomes 0.25, thus there does not exist a finite $\kappa(1)$. □

EXAMPLE 2
The second order system taken from [Babaali and Egerstedt, 2004]

$$\begin{aligned} A(1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} & A(2) &= \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \\ C(1) &= \begin{bmatrix} 1 & 0 \end{bmatrix} & C(2) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \end{aligned} \tag{37}$$

is $(2,3)$-observable with $\kappa(2) \approx 54.2$. □

Next, we need to establish a connection between the value function $V_k(x_k, \theta_k)$ and the size of the estimation error $\|\hat{x} - x\|$. A desirable property is that if the value function is small the estimation error is also small. This fact is made explicit by the following lemma, which is a modified version of Lemma 4.4.2 in [Rao, 2000].

Lemma 2
If the system (34) is (N_x, N)-observable then there exists a finite $\mu(i)$ such that

$$\begin{aligned}\|x_{T-N+1+i} - \hat{x}_{T-N+1+i}\|^2 \\ \leq 4\mu(i)\left(\sum_{k=T-N+1}^{T} \|\hat{v}_k\|^2_{R_v^{-1}} + \sum_{k=T-N+1}^{T-1} \|\hat{w}_k\|^2_{R_w^{-1}}\right) \qquad (38)\end{aligned}$$

for all $0 \leq i \leq N_x$.

Proof. The proof is given in Appendix A.2. □

Note that Lemma 2 holds for any estimate, regardless of how it was generated, but only for x_k generated by (34). We are now ready to formulate a theorem regarding stability in the sense of Definition 1 for the class of systems referred to as switching systems.

Theorem 1
If the system (34) is (N_x, N)-observable then the proposed recursive state estimator, using the step cost defined in (33), will be asymptotically stable in the sense of Definition 1, for $N - 1 - N_x \leq T_s < N$.

Proof. First note that

$$\begin{aligned} V_T^* &= \min_{x_T, \theta_T} V_T(x_T, \theta_T) = V_T(\hat{x}_{T|T}, \hat{\theta}_{T|T}) \\ &\geq \min_{x_{T-1}, \theta_{T-1}} \{V_{T-1}(x_{T-1}, \theta_{T-1}) + \underline{\alpha} L_{T-1}(\hat{x}_{T|T}, x_{T-1}, \hat{\theta}_{T|T}, \theta_{T-1})\} \\ &= V_{T-1}(\hat{x}_{T-1|T}, \hat{\theta}_{T-1|T}) + \underline{\alpha} L_{T-1}(\hat{x}_{T|T}, \hat{x}_{T-1|T}, \hat{\theta}_{T|T}, \hat{\theta}_{T-1|T}) \qquad (39)\end{aligned}$$

where the inequality is due to relaxed value iteration. Note that the estimates $\hat{x}_{T|T}$ and $\hat{x}_{T-1|T}$ are computed in the same way as prescribed by (26) and (27) in Section 4.4, thus explaining why $\underline{\alpha}$ was included there.

Now repeating the same procedure as in (39) we get

$$\begin{aligned} V_T^* \geq V_{T-N}(\hat{x}_{T-N|T}, \hat{\theta}_{T-N|T}) \\ + \sum_{k=T-N}^{T-1} \underline{\alpha} L_k(\hat{x}_{k+1|T}, \hat{x}_{k|T}, \hat{\theta}_{k+1|T}, \hat{\theta}_{k|T}) \end{aligned} \tag{40}$$

Substituting (33) for L_k and using optimality of V_{T-N}^* gives

$$V_T^* \geq V_{T-N}^* + \underline{\alpha} \sum_{k=T-N}^{T-1} \|\hat{v}_{k+1|T}\|^2_{R_v^{-1}} + \|\hat{w}_{k|T}\|^2_{R_w^{-1}} \tag{41}$$

Rearranging the sum and using positivity of the terms we get

$$V_T^* \geq V_{T-N}^* + \underline{\alpha} \left(\sum_{k=T-N+1}^{T} \|\hat{v}_{k|T}\|^2_{R_v^{-1}} + \sum_{k=T-N+1}^{T-1} \|\hat{w}_{k|T}\|^2_{R_w^{-1}} \right) \tag{42}$$

Now putting $N = 1$ gives that $V_T^* - V_{T-1}^* \geq 0$, thus the sequence of optimal costs is non-decreasing. Using the fact that there are no disturbances acting on the system the optimal cost is bounded from above as

$$\begin{aligned} V_T^* &= \min_{x_T, \theta_T} V_T(x_T, \theta_T) \\ &\leq \min_{x_T, \theta_T} \overline{\alpha} V^{\text{opt}}(x_T, \theta_T) \leq \overline{\alpha} \|x_0 - \check{x}_0\|^2_{P_0^{-1}} \end{aligned} \tag{43}$$

where $\check{x}_0$ is the initial guess and x_0 is the true initial value. Because the sequence $\{V_T^*\}$ is non-decreasing and bounded from above it will converge to $V_\infty^* < \infty$ as $T \to \infty$. Thus the partial sum

$$\left(\sum_{k=T-N+1}^{T} \|\hat{v}_{k|T}\|^2_{R_v^{-1}} + \sum_{k=T-N+1}^{T-1} \|\hat{w}_{k|T}\|^2_{R_w^{-1}} \right) \to 0 \tag{44}$$

as $T \to \infty$. Now using Lemma 2 we can conclude that

$$\left\| x_{T|T+Ts} - \hat{x}_{T|T+Ts} \right\|^2 \to 0 \tag{45}$$

as $T \to \infty$ for $N - 1 - N_x \leq Ts \leq N - 1$. □

If a system is only (N_x, N)-observable, with $N_x < N - 1$, we can not conclude that the estimation error approaches zero unless fixed lag smoothing is used. We will now give one example, where we indeed need fixed lag smoothing.

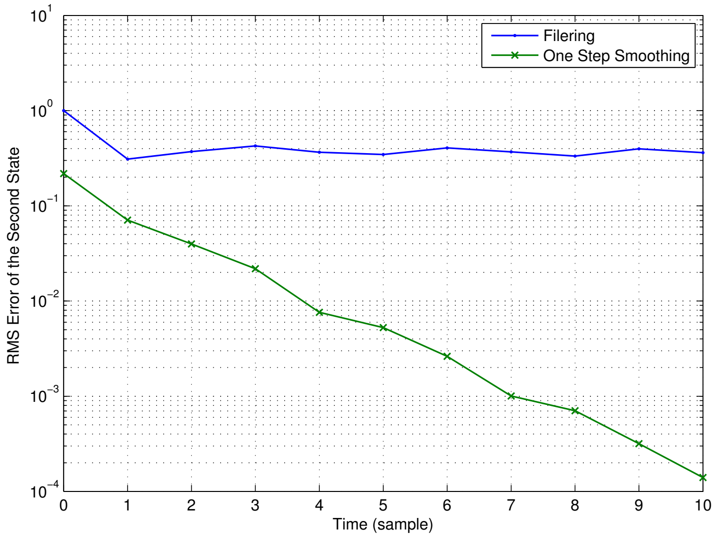


Figure 6. Root mean square error for filtering and one step smoothing estimates of the unobserved second state in Example 3. Note that smoothing is required for the estimation error to approach zero.

Example 3

Applying the proposed recursive estimator with $\underline{\alpha} = 1$, $\overline{\alpha} = 1.5$, $R_v = 0.1$ and $R_w = \operatorname{diag}([\,0.1 \quad 0.1\,])$ on the system from Example 1 gives the root mean square errors shown in Figure 6. The mode sequence was generated as the output of a Markov chain with transition probability matrix

$$P = \begin{bmatrix} 0.3 & 0.7 \\ 0.4 & 0.6 \end{bmatrix} \tag{46}$$

which yields a mean square stable system. In this case, fixed lag smoothing is required for the estimation error to approach zero. Because the system was shown to be $(0, 2)$-observable, Theorem 1 states that if $T_s = 1$, that is one step smoothing is used, the estimation error will approach zero. □

In this section we have assumed a rather restrictive, that is hard to fulfill, criterion for observability. One common relaxation is to assume a minimum dwell time between switches, this was done for example in [Alessandri *et al.*, 2005b].

7. Conclusions

In this paper, we have presented a recursive joint maximum a posteriori (JMAP) state estimation algorithm for Markov jump linear systems. The JMAP estimation problem was first transformed into a recursive problem using dynamic programming. To conquer the complexity associated with dynamic programming, we used a technique often referred to as relaxed dynamic programming. In relaxed dynamic programming, the Bellman equality is replaced by two inequalities which permits the value function to be simplified in each step, keeping the complexity at manageable levels.

The proposed state estimator was applied to the problem of estimating the state of a Markov chain in colored noise. The achieved performance was similar to that of a non-recursive estimator presented in [Logothetis and Krishnamurthy, 1999].

In the special case of a switching system, that is when the mode at time k is independent of the mode at time $k-1$, we proved that the proposed recursive state estimator is a stable observer for a noise free system.

Acknowledgement

This work was supported by the Swedish Research Council.

A. Appendix

A.1 Parametrizations

Step Cost.

$$\Lambda(\theta_k, \theta_{k+1}) = \begin{bmatrix} I & -B(\theta_k)u_k & -A(\theta_k) \\ -C(\theta_{k+1}) & y_{k+1} - D(\theta_{k+1})u_{k+1} & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{47}$$

$$Q(\theta_k, \theta_{k+1}) = \begin{bmatrix} \frac{1}{2}R_w^{-1}(\theta_k) & 0 & 0 \\ 0 & \frac{1}{2}R_v^{-1}(\theta_{k+1}) & 0 \\ 0 & 0 & K(\theta_k, \theta_{k+1}) \end{bmatrix} \tag{48}$$

$$U(\theta_k,\theta_{k+1},\pi) = \Lambda^T(\theta_k,\theta_{k+1})Q(\theta_k,\theta_{k+1})\Lambda(\theta_k,\theta_{k+1}) + \begin{bmatrix} 0 & 0 & 0 \\ 0 & \pi_{22} & \pi_{12}^T \\ 0 & \pi_{12} & \pi_{11} \end{bmatrix} \tag{49}$$

When computing the upper and lower bounds (23), $Q(\theta_k,\theta_{k+1})$ is replaced by $\overline{\alpha}Q(\theta_k,\theta_{k+1})$ and $\underline{\alpha}Q(\theta_k,\theta_{k+1})$ respectively.

Value Function. For $\pi \in \Pi_0$

$$\pi = \Lambda^T(\theta_0)Q(\theta_0)\Lambda(\theta_0) \tag{50}$$

where

$$\Lambda(\theta_0) = \begin{bmatrix} I & -\check{x}_0 \\ -C(\theta_0) & y_0 - D(\theta_0)u_0 \\ 0 & 1 \end{bmatrix}^T \tag{51}$$

$$Q(\theta_0) = \begin{bmatrix} \frac{1}{2}P_0^{-1} & 0 & 0 \\ 0 & \frac{1}{2}R_v^{-1}(\theta_0) & 0 \\ 0 & 0 & K(\theta_0) \end{bmatrix} \tag{52}$$

For $\pi \in \Pi_{k+1}$, $k \geq 0$

$$\pi = \begin{bmatrix} U_{11} & U_{12} \\ U_{12}^T & U_{22} \end{bmatrix} - \begin{bmatrix} U_{13} \\ U_{23} \end{bmatrix} U_{33}^{-1} \begin{bmatrix} U_{13} \\ U_{23} \end{bmatrix}^T$$

with $U(\theta_k,\theta_{k+1},\pi)$ partitioned as

$$\begin{bmatrix} x_{k+1} \\ 1 \\ x_k \end{bmatrix}^T \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ U_{12}^T & U_{22} & U_{23} \\ U_{13}^T & U_{23}^T & U_{33} \end{bmatrix} \begin{bmatrix} x_{k+1} \\ 1 \\ x_k \end{bmatrix} \tag{53}$$

A.2 Proof of Lemma 2

Before proving Lemma 2 we first establish the following two lemmas concerning quadratic forms.

LEMMA 3

$$\left\| G \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right\|_Q^2 \leq \lambda_{\max}\left(G^TQG\right)\left(\|x_1\|^2 + \|x_2\|^2\right) \tag{54}$$

Proof.

$$\begin{aligned}\left\|G\begin{bmatrix}x_1\\x_2\end{bmatrix}\right\|_Q^2 &= \begin{bmatrix}x_1\\x_2\end{bmatrix}^T G^TQG\begin{bmatrix}x_1\\x_2\end{bmatrix} = \operatorname{tr}\left(G^TQG\begin{bmatrix}x_1\\x_2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\right)\\ &\le \lambda_{\max}\left(G^TQG\right)\operatorname{tr}\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\right) = \lambda_{\max}\left(G^TQG\right)\left(\|x_1\|^2+\|x_2\|^2\right)\end{aligned} \tag{55}$$

□

LEMMA 4
For R_1 and R_2 invertible

$$\left\|\begin{bmatrix}A & B\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}\right\|_Q^2 \le \gamma\left(\|x_1\|_{R_1}^2+\|x_2\|_{R_2}^2\right) \tag{56}$$

with

$$\gamma = \lambda_{\max}\left(\begin{bmatrix}AR_1^{-1/2} & BR_2^{-1/2}\end{bmatrix}^T Q\begin{bmatrix}AR_1^{-1/2} & BR_2^{-1/2}\end{bmatrix}\right) \tag{57}$$

Proof. Let $y_1 = R_1^{1/2}x_1$ and $y_2 = R_2^{1/2}x_2$

$$\begin{aligned}\left\|\begin{bmatrix}A & B\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}\right\|_Q^2 &= \left\|\begin{bmatrix}AR_1^{-1/2} & BR_2^{-1/2}\end{bmatrix}\begin{bmatrix}y_1\\y_2\end{bmatrix}\right\|_Q^2\\ &\le \gamma\left(\|y_1\|^2+\|y_2\|^2\right) = \left(\|x_1\|_{R_1}^2+\|x_2\|_{R_2}^2\right)\end{aligned} \tag{58}$$

□

Proof. [Lemma 2] Without loss of generality let us assume that $T = N-1$. Next introduce the following notation, for which subscript sizes of some quantities such as $O(\Theta_{N-1})$ has been dropped where they can be inferred from context. First let

$$\hat{W} = \begin{bmatrix}\hat{w}_0\\ \vdots\\ \hat{w}_{N-2}\end{bmatrix} \quad \hat{V} = \begin{bmatrix}\hat{v}_0\\ \vdots\\ \hat{v}_{N-1}\end{bmatrix} \quad \hat{X} = \begin{bmatrix}\hat{x}_0\\ \vdots\\ \hat{x}_{N-1}\end{bmatrix} \quad X = \begin{bmatrix}x_0\\ \vdots\\ x_{N-1}\end{bmatrix} \tag{59}$$

Now we can express the vector of state estimates $\hat{X}$ and the true state X as

$$\begin{aligned}\hat{X} &= \mathcal{A}(\hat{\Theta})\hat{x}_0 + M\hat{W}\\ X &= \mathcal{A}(\Theta)x_0\end{aligned} \tag{60}$$

where

$$M = \begin{bmatrix} M_0 \\ \vdots \\ M_{N-1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ I & & & \\ \tilde{A}(\hat{\Theta}_{1:1}) & I & & \\ & & \ddots & \\ \tilde{A}(\hat{\Theta}_{1:N-2}) & \tilde{A}(\hat{\Theta}_{2:N-2}) & \dots & I \end{bmatrix} \tag{61}$$

Next express the sum as a quadratic form

$$\begin{aligned} &\sum_{k=0}^{N-1} \|y_k - C(\theta_k)\hat{x}_k\|^2_{R_v^{-1}} + \sum_{k=0}^{N-2} \|\hat{x}_{k+1} - A(\hat{\theta}_k)\hat{x}_k\|^2_{R_w^{-1}} \\ &\qquad = \|\hat{V}\|^2_{\overline{R}_v^{-1}} + \|\hat{W}\|^2_{\overline{R}_w^{-1}} = \|\overline{Y} - G(\hat{\Theta})\hat{X}\|^2_{\overline{R}^{-1}} \end{aligned} \tag{62}$$

We can now bound $\|x_i - \hat{x}_i\|^2$ for $0 \le i \le N_x$ from above as

$$\begin{aligned} &\|x_i - \hat{x}_i\|^2 \\ &= \left\|\tilde{A}(\Theta_{i-1})x_0 - \tilde{A}(\hat{\Theta}_{i-1})\hat{x}_0 - M_i\hat{W}\right\|^2 \\ &\overset{(56)}{\le} 2\left(\left\|\tilde{A}(\Theta_{i-1})x_0 - \tilde{A}(\hat{\Theta}_{i-1})\hat{x}_0\right\|^2 + \gamma(i)\left\|\hat{W}\right\|^2\right) \\ &\overset{(35)}{\le} 2\left(\kappa(i)\left\|\mathcal{O}(\Theta_{N-2})x_0 - \mathcal{O}(\hat{\Theta}_{N-2})\hat{x}_0\right\|^2 + \gamma(i)\left\|\hat{W}\right\|^2\right) \\ &= 2\left\|\begin{bmatrix} \mathcal{O}(\Theta)x_0 - \mathcal{O}(\hat{\Theta})\hat{x}_0 \\ \hat{W} \end{bmatrix}\right\|^2_{\begin{bmatrix} \kappa(i)I & 0 \\ 0 & \gamma(i)I \end{bmatrix}} \\ &= 2\left\|\begin{bmatrix} I & \mathcal{C}(\hat{\Theta})M \\ 0 & I \end{bmatrix}\begin{bmatrix} \mathcal{O}(\Theta)x_0 - \mathcal{O}(\hat{\Theta})\hat{x}_0 - \mathcal{C}(\hat{\Theta})M\hat{W} \\ \hat{W} \end{bmatrix}\right\|^2_{\begin{bmatrix} \kappa(i)I & 0 \\ 0 & \gamma(i)I \end{bmatrix}} \\ &\overset{(57)}{\le} 2\mu(i)\left(\left\|\mathcal{O}(\Theta)x_0 - \mathcal{O}(\hat{\Theta})\hat{x}_0 - \mathcal{C}(\hat{\Theta})M\hat{W}\right\|^2_{\overline{R}_v^{-1}} + \left\|\hat{W}\right\|^2_{\overline{R}_w^{-1}}\right) \\ &= 2\mu(i)\left(\left\|\mathcal{C}(\Theta)X - \mathcal{C}(\hat{\Theta})\hat{X}\right\|^2_{\overline{R}_v^{-1}} + \left\|\hat{W}\right\|^2_{\overline{R}_w^{-1}}\right) \\ &= 2\mu(i)\left(\left\|G(\Theta)X - G(\hat{\Theta})\hat{X}\right\|^2_{\overline{R}^{-1}}\right) \\ &\overset{(56)}{\le} 4\mu(i)\left(\|\overline{Y} - G(\Theta)X\|^2_{\overline{R}^{-1}} + \left\|\overline{Y} - G(\hat{\Theta})\hat{X}\right\|^2_{\overline{R}^{-1}}\right) \\ &= 4\mu(i)\left\|\overline{Y} - G(\hat{\Theta})\hat{X}\right\|^2_{\overline{R}^{-1}} \end{aligned}$$

$$= 4\mu(i)\left(\sum_{k=0}^{N-1}\|y_k - C(\theta_k)\hat{x}_k\|^2_{R_v^{-1}} + \sum_{k=0}^{N-2}\|\hat{x}_{k+1} - A(\hat{\theta}_k)\hat{x}_k\|^2_{R_w^{-1}}\right) \tag{63}$$

with

$$\gamma(i) = \max_{\hat{\Theta}} \lambda_{\max}\left(M_i^T M_i\right) \tag{64}$$

and

$$\mu(i) = \max_{\hat{\Theta}} \lambda_{\max}\left(\begin{bmatrix} \overline{R}_v^{1/2} & M\overline{R}_w^{1/2} \\ 0 & \overline{R}_w^{1/2} \end{bmatrix}^T \begin{bmatrix} \kappa(i)I & 0 \\ 0 & \gamma(i)I \end{bmatrix} \begin{bmatrix} \overline{R}_v^{1/2} & M\overline{R}_w^{1/2} \\ 0 & \overline{R}_w^{1/2} \end{bmatrix}\right) \tag{65}$$

where the maximization over $\hat{\Theta}$ is due to the fact that M depends on it.

References

Ackerson, G. A. and K. S. Fu (1970): “On state estimation in switching environments.” *IEEE Trans. Automat. Contr.*, **15**, February, pp. 10–17.

Alessandri, A., M. Baglietto, and G. Battistelli (2005a): “Minimum-distance receding-horizon state estimation for switching discrete-time linear systems.” In Findeisen *et al.*, Eds., *Proceedings of International Workshop on Assessment and Future Directions of Nonlinear Model Predictive Control*, pp. 197–204.

Alessandri, A., M. Baglietto, and G. Battistelli (2005b): “Receding-horizon estimation for switching discrete-time linear systems.” *IEEE Transactions on Automatic Control*, **50:11**, pp. 1736–1748.

Alessandri, A. and P. Coletta (2001): “Design of Luenberger observers for a class of hybrid linear systems.” In *HSCC ’01: Proceedings of the 4th International Workshop on Hybrid Systems*, pp. 7–18. Springer-Verlag, London, UK.

Alriksson, P. and A. Rantzer (2006): “Observer synthesis for switched discrete-time linear systems using relaxed dynamic programming.” In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*. Kyoto, Japan.

Babaali, M. and M. Egerstedt (2004): “Observability of switched linear systems.” In *Hybrid Systems: Computation and Control 2004, LNCS 2993*, pp. 48–63. Springer-Verlag Berlin Heidelberg.

Babaali, M. and M. Egerstedt (2005): "Asymptotic observers for discrete-time switched linear systems." In *Proceedings of the 16th IFAC World Congress.*

Babaali, M., M. Egerstedt, and E. W. Kamen (2004): "A direct algebraic approch to observer design under switching measurement equations." *IEEE Transactions on Automatic Control*, **49:11**, pp. 2044–2049.

Balluchi, A., L. Benvenuti, M. D. D. Benedetto, and A. L. Sangiovanni-Vincentelli (2002): "Design of observers for hybrid system." In Tomlin and Greenstreet, Eds., *Hybrid Systems: Computation and Control.* Springer-Verlag, New York, U.S.A.

Balluchi, A., L. Benvenuti, C. Lemma, A. L. Sangiovanni-Vincentelli, and G. Serra (2005): "Actual engaged gear identification: A hybrid observer approch." In *Proceedings of the 16th IFAC World Congress.*

Bemporad, A., D. Mignone, and M. Morari (1999): "Moving horizon estimation for hybrid systems and fault detection." *Proceedings of the 1999 American Control Conference*, **4**, pp. 2471–2475 vol.4.

Blom, H. (1984): "An efficient filter for abruptly changing systems." In *Proceedings of 23rd Conference on Decision and Control.*

Costa, O. L. V. (1994): "Linear minimum mean square error estimation for discrete-time Markovian jump linear systems." *IEEE Transactions on Automatic Control*, **39:8**, pp. 1685–1689.

Cox, H. (1964): "On the estimation of state variables and parameters for noisy dynamic systems." *IEEE Trans. Automat. Contr.*, **9**, January, pp. 5–12.

Doucet, A., N. Gordon, and V. Krishnamurthy (2001): "Particle filters for state estimation of jump Markov linear systems." *IEEE Transactions on Signal Processing*, **49:3**, pp. 613–624.

Goodwin, G. C., M. M. Seron, and J. A. De Doná (2005): *Constrained Control and Estimation : An Optimisation Approach.* Communication and control engineering. Springer.

Ho, Y. C. and R. C. K. Lee (1964): "A Bayesian approach to problems in stochastic estimation and control." *IEEE Trans. Automat. Contr.*, **9**, October, pp. 333–339.

Larson, R. E. and J. Peschon (1966): "A dynamic programming approach to trajectory estimation." *IEEE Trans. Automat. Contr.*, **11**, July, pp. 537–540.

Lincoln, B. and A. Rantzer (2006): "Relaxing dynamic programming." *IEEE Transactions on Automatic Control*, **51:8**, pp. 1249–1260.

Logothetis, A. and V. Krishnamurthy (1999): "Expectation maximization algorithms for MAP estimation of jump Markov linear systems." *IEEE Transactions on Signal Processing*, **47:8**, pp. 2139–2156.

Mazor, E., A. Averbuch, Y. Bar-Shalom, and J. Dayan (Jan 1998): "Interacting multiple model methods in target tracking: a survey." *IEEE Transactions on Aerospace and Electronic Systems*, **34:1**, pp. 103–123.

Oh, S. M., J. Rehg, and F. Dellaert (2006): "Parameterized duration modeling for switching linear dynamic systems." *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, **2**, pp. 1694–1700.

Rao, C. V. (2000): *Moving Horizon Strategies for the Constrained Monitoring and Control of Nonlinear Discrete-Time Systems*. PhD thesis, University of Wisconsin-Madison.

Vidal, R., A. Chiuso, and S. Soatto (2002): "Observability and identifiability of jump linear systems." In *Proceednigs of the 41st IEEE Conference on Decision and Control*, pp. 3614–3619.

Wernrud, A. (2008): *Approximate Dynamic Programming with Applications*. PhD thesis ISRN LUTFD2/TFRT--1082--SE, Department of Automatic Control, Lund University, Sweden.