Multi Sensor Data Fusion

Hugh Durrant-Whyte
Australian Centre for Field Robotics
The University of Sydney NSW 2006
Australia
hugh@acfr.usyd.edu.au

January 22, 2001

Version 1.2

Contents

1 Introduction

Data fusion is the process of combing information from a number of different sources to provide a robust and complete description of an environment or process of interest. Data fusion is of special significance in any application where a large amounts of data must be combined, fused and distilled to obtain information of appropriate quality and integrity on which decisions can be made. Data fusion finds application in many military systems, in civilian surveillance and monitoring tasks, in process control and in information systems. Data fusion methods are particularly important in the drive toward autonomous systems in all these applications. In principle, automated data fusion processes allow essential measurements and information to be combined to provide knowledge of sufficient richness and integrity that decisions may be formulated and executed autonomously.

This course provides a practical introduction to data fusion methods. It aims to introduce key concepts in multi-sensor modeling, estimation, and fusion. The focus of this course is on mathematical methods in probabilistic and estimation-theoretic data fusion. Associated with this course is a series of computer-based laboratories which provide the opportunity to implement and evaluate multi-sensor tracking, estimation and identification algorithms.

Data fusion is often (somewhat arbitrarily) divided into a hierarchy of four processes. Levels 1 and 2 of this process are concerned with the formation of track, identity, or estimate information and the fusion of this information from several sources. Level 1 and 2 fusion is thus generally concerned with numerical information and numerical fusion methods (such as probability theory or Kalman filtering). Level 3 and 4 of the data fusion process is concerned with the extraction of "knowledge" or decisional information. Very often this includes qualitative reporting or secondary sources of information or knowledge from human operators or other sources. Level 3 and 4 fusion is thus concerned with the extraction of high-level knowledge (situation awareness for example) from low level fusion processes, the incorporation of human judgment and the formulation of decisions and actions. This hierarchy is not, by any means, the only way of considering the general data fusion problem. It is perhaps appropriate for many military data fusion scenarios, but is singularly inappropriate for many autonomous systems or information fusion problems. The imposition of a "hierarchical" structure to the problem at the outset can also serve to mislead the study of distributed, decentralised and network-centric data fusion structures. Nevertheless, the separate identification of numerical problems (tracking, identification and estimation) from decisional and qualitative problems (situation awareness, qualitative reporting and threat assessment) is of practical value.

This course focus mainly on level 1-2 type data fusion problems. These are concerned with the fusion of information from sensors and other sources to arrive at an estimate of location and identity of objects in an environment. It encompasses both the direct fusion of sensor information and the indirect fusion of estimates obtained from local fusion centers. The primary methods in level 1-2 fusion methods are probabilistic. These include multi-target tracking, track-to-track fusion, and distributed data fusion methods. Level 3-4 data fusion problems are considered in less detail. These involve the modeling of qualitative information sources, the use of non-probabilistic methods in describing uncertainty and general decision making processes. Level 3-4 data fusion, obviously, builds on level 1-2 methods.

1.1 Data Fusion Methods

In any data fusion problem, there is an environment, process or quantity whose true value, situation or state is unknown. It would be unreasonable to expect that there is some single source of perfect and complete knowledge about the problem of interest and so information must be obtained *indirectly* from sources which provide imperfect and incomplete knowledge, using these to infer the information needed. There may well be many sources of information that could be used to help obtain the required knowledge: some effect characteristic of a particular state may be observed, prior beliefs about possible states may be available, or knowledge about certain constraints and relations may exist. To use the available information to its maximum effect it is important to describe precisely the way in which this information relates to the underlying state of the world.

In sensor data fusion problems these concepts of 'world', 'state', 'information' and 'observation' can be made precise:

- The quantity of interest will be denoted by $\mathbf{x}$. This quantity may describe an environment, process, statement or single number. Employing the terminology of decision theory, the quantity $\mathbf{x}$ will be called **the state of nature** or simply 'the state'. The state of nature completely describes the world of interest. The state of nature can take on a variety of values all of which are contained in the set $\mathcal{X}$ of possible states of nature; $\mathbf{x} \in \mathcal{X}$. The current state of the world is simply a singleton of this set. An environment model comprises this set of possible states together with any knowledge of how elements of this set are related.

- To obtain information about the state of nature an experiment or observation is conducted yielding a quantity $\mathbf{z}$. The observation can take on a variety of values all of which are contained in the **sample space** $\mathcal{Z}$. The information obtained by observation is a single realization $\mathbf{z} \in \mathcal{Z}$ from this set. If the information we obtain through observation is to be of any value we must know how this information depends on the true state of nature. That is, for each specific state of nature $\mathbf{x} \in \mathcal{X}$ an **observation model** is required that describes what observations we will make $\mathbf{z} = \mathbf{z}(x) \in \mathcal{Z}$. The observation model describes precisely the sensing process.

- Given the information obtained through observation $\mathbf{z}$, the goal of the data fusion process is to infer an underlying state of nature $\mathbf{x}$. To do this we describe a function or **decision rule** δ which maps observations to states of nature; $\delta(z) \rightarrow \mathbf{x} \in \mathcal{X}$. The decision model must incorporate information about the nature of the observation process, prior beliefs about the world, and a measure of the value placed on accuracy and error in the state of the world. The decision rule is essential in data fusion problems; it is the function that takes in all information and produces a single decision and resulting action.

These three elements; the state, the observation model and the decision rule, are the essential components of the data fusion problem.

The most important problem in data fusion is the development of appropriate models of uncertainty associated with both the state and observation process. This course begins in Section 2 by describing key methods for representing and reasoning about uncertainty. The focus is on the use of probabilistic and information-theoretic methods for sensor modeling and for data fusion. Probability theory provides an extensive and unified set of methods for describing and manipulating uncertainty. It is demonstrated how basic probability models can be used to fuse information, to describe different data fusion architectures and to manage sensors and sensor information. Many of the data fusion methods described in the remainder of this course are based on these probabilistic methods. However, probabilistic modeling methods do have limitations. Alternatives methods for describing uncertainty including fuzzy logic, evidential reasoning, and qualitative reasoning methods, are briefly considered. In certain circumstances one of these alternative techniques may provide a better method of describing uncertainty than probabilistic models.

Estimation is the single most important problem in sensor data fusion. Fundamentally, an estimator is a decision rule which takes as an argument a sequence of observations and whose action is to compute a value for the parameter or state of interest. Almost all data fusion problems involve this estimation process: we obtain a number of observations from a group of sensors and using this information we wish to find some *estimate* of the true state of the environment we are observing. In particular, estimation encompasses problems such as multi-target tracking. Section 3 of this course focuses on the multi-sensor estimation problem. The Kalman filter serves as the basis for most multi-sensor estimation problems. The multi-sensor Kalman filter is introduced. Various formulations of the multi-sensor Kalman filter are discussed. The problem of asynchronous and delayed data is described and solved. Alternatives to the Kalman filter such as likelihood filters, are discussed. The problem of data association; relating observations to tracks, is introduced. The three key methods for data association; the nearest neighbour filter, probabilistic data association and the multiple hypothesis and track splitting filters, are described. Finally, the track-to-track fusion algorithm is described as a method for combining estimates generated by local estimators.

An increasingly important facet of data fusion is the design of system architecture. As the complexity of data fusion systems increases, as more sensors and more information sources are incorporated, so the need to provide distributed data fusion architectures increases. While many block-diagrams of distributed data fusion systems exist, there are few mathematical methods for manipulating and fusing information in these architectures. Section 4 describes the use of information-theoretic estimation methods in distributed and decentralised data fusion systems. It is demonstrated how multiple-target tracking and identification problems can be effectively decentralised amongst a network of sensors nodes. A detailed study of the required communication algorithms is performed. This focuses on issues of delayed, intermittent, and limited band-width communications. Advanced areas of interest including sensor management, model distribution and organisation control are also briefly described.

The final component in data fusion is the need to make decisions on the basis of information obtained from fusion. These decisions encompass the allocation and management of sensing resources, the identification of specific situations, and the control or deployment of sensing platforms. Section 5 briefly introduces key elements of decision theory methods. These include ideas of utility, risk, and robustness. Some applications in sensor management are described.

1.2 Bibliography

Data fusion is a generally poorly defined area of study. This is because data fusion methods are based on techniques established in many other diverse fields. As a consequence many data fusion books tend to be either quite broad summaries of methods, or are more qualitative discussions of architectures and principles for specific applications. Some of the key recommended books are described here:

Blackman: The recent book by Blackman and Popoli [9] is probably the most comprehensive book on data fusion methods. It covers both level 1-2 multi-target tracking and identification problems as well as level 3-4 methods in situation assessment and sensor management. Notably, it covers a number of current military systems in some detail and gives develops a number of specific examples of multi-sensor systems.
Waltz and Linas: The book by Waltz and Linas [43] has become something of a classic in the field. It was one of the first books that attempted to define and establish the field of data fusion. The book consists of mainly qualitative descriptions of various data fusion systems (with an emphasis on military systems). There are some quantitative methods introduced but they are described only at a superficial level. However the general overview provided is still of significant value.
BarShalom: Yakkov Barshalom has written and edited a number of books on data fusion and tracking. Of note are his classic book [7] on tracking and data association, and his two edited volumes on data fusion [5, 6]. Together these are reasonably definitive in terms of current tracking and data association methods.
Artech House series: Artech house (www.artechhouse.com) are a specialist publishing house in the radar, data fusion and military systems area. They publish by far the most important and advanced technical books in the data fusion area.

Additional references are provided at the end of these notes.

2 Probabilistic Data Fusion

Uncertainty lies at the heart of all descriptions of the sensing and data fusion process. An explicit measure of this uncertainty must be provided to enable sensory information to be fused in an efficient and predictable manner. Although there are many methods of representing uncertainty, almost all of the theoretical developments in this course are based on the use of *probabilistic* models. There is a huge wealth of experience and methods associated with the development of probabilistic models of environments and information. Probabilistic models provide a powerful and consistent means of describing uncertainty in a broad range of situations and leads naturally into ideas of information fusion and decision making.

It could be, and indeed often is, argued that probability is the only rational way to model uncertainty. However, the practical realities of a problem often suggest alternative uncertainty modeling methods. Probabilistic models are good at many things but it is often the case that such a model cannot capture all the information that we need to define and describe the operation of a sensing and data fusion system. For example it would be difficult to imagine using probabilistic models alone to capture the uncertainty and error introduced by the dependence of, say, an active electro-magnetic sensor information on beam-width and radiation frequency effects. Alternative methods of modeling uncertainty are briefly described in Section 2.4.

Even with limitations, probabilistic modeling techniques play an essential role in developing data fusion methods. Almost all conventional data fusion algorithms have probabilistic models as a central component in their development. It would be impossible to talk about data fusion in any coherent manner without first obtaining a thorough understanding of probabilistic modeling methods.

This section begins by briefly introducing the essential elements of probabilistic modeling: probability densities, conditional densities, and Bayes theorem. With these basic elements, the basic data fusion problem is described and it is shown how information can be combined in a probabilistic manner. From this, various data fusion architectures are described and, mathematically, how they are implemented. The idea of information and entropy is also introduced as a natural way of describing the flow of 'information' in different data fusion architectures. These tools, together with the Kalman filter are the underlying basis for all the data fusion methods developed in this course.

2.1 Probabilistic Models

In the following, familiarity with essential probability theory is assumed, and some simple notation and rules to be used throughout this course are introduced. A probability density function (*pdf*) $P_y(\cdot)$ is defined on a random variable $\mathbf{y}$, generally written as $P_y(\mathbf{y})$ or simply $P(\mathbf{y})$ when the dependent variable is obvious. The random variable may be a scalar or vector quantity, and may be either discrete or continuous in measure.

The *pdf* is considered as a (probabilistic) *model* of the quantity $\mathbf{y}$; observation or state. The *pdf* $P(\mathbf{y})$ is considered valid if;

1. It is positive; $P(\mathbf{y}) > 0$ for all $\mathbf{y}$, and
2. It sums (integrates) to a total probability of 1;

$$\int_y P(\mathbf{y})\mathrm{d}y = 1.$$

The joint distribution $P_{xy}(\mathbf{x}, \mathbf{y})$ is defined in a similar manner.

Integrating the *pdf* $P_{xy}(\mathbf{x}, \mathbf{y})$ over the variable $\mathbf{x}$ gives the marginal *pdf* $P_y(\mathbf{y})$ as

$$P_y(\mathbf{y}) = \int_x P_{xy}(\mathbf{x}, \mathbf{y})\mathrm{d}x, \tag{1}$$

and similarly integrating over $\mathbf{y}$ gives the marginal *pdf* $P_x(\mathbf{x})$. The joint *pdf* over n variables, $P(\mathbf{x}_1, \cdots, \mathbf{x}_n)$, may also be defined with analogous properties to the joint *pdf* of two variables.

The conditional *pdf* $P(\mathbf{x} \mid \mathbf{y})$ is defined by

$$P(\mathbf{x} \mid \mathbf{y}) \triangleq \frac{P(\mathbf{x}, \mathbf{y})}{P(\mathbf{y})}, \tag{2}$$

and has the usual properties of a *pdf* with $\mathbf{x}$ the dependent variable given that $\mathbf{y}$ takes on specific fixed values. The conditional *pdf* $P(\mathbf{y} \mid \mathbf{x})$ is similarly defined.

The *chain-rule* of conditional distributions can be used to expand a joint *pdf* in terms of conditional and marginal distributions. From Equation 2,

$$P(\mathbf{x}, \mathbf{y}) = P(\mathbf{x} \mid \mathbf{y})P(\mathbf{y}). \tag{3}$$

The chain-rule can be extended to any number of variables in the following form

$$P(\mathbf{x}_1, \cdots, \mathbf{x}_n) = P(\mathbf{x}_1 \mid \mathbf{x}_2 \cdots, \mathbf{x}_n) \cdots P(\mathbf{x}_{n-1} \mid \mathbf{x}_n)P(\mathbf{x}_n), \tag{4}$$

where the expansion may be taken in any convenient order. Substitution of Equation 3 into Equation 1 gives an expression for the marginal distribution of one variable in terms of the marginal distribution of a second variable as

$$P_y(\mathbf{y}) = \int_x P_{x|y}(\mathbf{y} \mid \mathbf{x})P_x(\mathbf{x})\mathrm{d}x. \tag{5}$$

This important equation is known as the total probability theorem. It states that the total probability in a state $\mathbf{y}$ can be obtained by considering the ways in which $\mathbf{y}$ can occur given that the state $\mathbf{x}$ takes a specific value (this is encoded in $P_{x|y}(\mathbf{y} \mid \mathbf{x})$), weighted by the probability that each of these values of $\mathbf{x}$ is true (encoded in $P_x(\mathbf{x})$).

If it happens that knowledge of the value of $\mathbf{y}$ does not give us any more information about the value of $\mathbf{x}$ then $\mathbf{x}$ and $\mathbf{y}$ are said to be independent as

$$P(\mathbf{x} \mid \mathbf{y}) = P(\mathbf{x}). \tag{6}$$

With Equation 6 substituted into Equation 3

$$P(\mathbf{x}, \mathbf{y}) = P(\mathbf{x})P(\mathbf{y}). \tag{7}$$

A weaker form of independence can be defined through the important idea of conditional independence. Given three random variables $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, the conditional distribution of $\mathbf{x}$ given both $\mathbf{y}$ and $\mathbf{z}$ is defined as $P(\mathbf{x} \mid \mathbf{yz})$. If knowledge of the value of $\mathbf{z}$ makes the value of $\mathbf{x}$ independent of the value of $\mathbf{y}$ then

$$P(\mathbf{x} \mid \mathbf{y}, \mathbf{z}) = P(\mathbf{x} \mid \mathbf{z}). \tag{8}$$

This may be the case for example if $\mathbf{z}$ indirectly contains all the information contributed by $\mathbf{y}$ to the value of $\mathbf{x}$. Conditional independence can be exploited in a number of different ways. In particular, applying the chain-rule to the joint probability density function on three random variables $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$

$$\begin{aligned} P(\mathbf{x}, \mathbf{y}, \mathbf{z}) &= P(\mathbf{x}, \mathbf{y} \mid \mathbf{z})P(\mathbf{z}) \\ &= P(\mathbf{x} \mid \mathbf{y}, \mathbf{z})P(\mathbf{y} \mid \mathbf{z})P(\mathbf{z}), \end{aligned} \tag{9}$$

together with the conditional independence result of Equation 8 the intuitive result

$$P(\mathbf{x}, \mathbf{y} \mid \mathbf{z}) = P(\mathbf{x} \mid \mathbf{z})P(\mathbf{y} \mid \mathbf{z}), \tag{10}$$

is obtained. That is, if $\mathbf{x}$ is independent of $\mathbf{y}$ given knowledge of $\mathbf{z}$ then the joint probability density function of $\mathbf{x}$ and $\mathbf{y}$ conditioned on $\mathbf{z}$ is simply the product of the marginal distributions of $\mathbf{x}$ and $\mathbf{y}$ each conditioned on $\mathbf{z}$, analogously to Equation 7.

The idea of conditional independence underlies many of the data fusion algorithms developed in this course. Consider the state of a system $\mathbf{x}$ and two observations of this state $\mathbf{z}_1$ and $\mathbf{z}_2$. It should be clear that the two observations are not independent,

$$P(\mathbf{z}_1, \mathbf{z}_2) \neq P(\mathbf{z}_1)P(\mathbf{z}_2),$$

as they must both depend on the common state $\mathbf{x}$. Indeed, if the two observations were independent (were unrelated to each other), there would be little point fusing the information they contain ! Conversely, it is quite reasonable to assume that the *only* thing the two observations have in common is the underlying state, and so the observations *are* independent once the state is known; that is, the observations are conditionally independent given the state as

$$P(\mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{x}) = P(\mathbf{z}_1 \mid \mathbf{x})P(\mathbf{z}_2 \mid \mathbf{x}).$$

Indeed, for the purposes of data fusion, this would not be a bad definition of the state; simply what the two information sources have in common.

2.2 Probabilistic Methods

2.2.1 Bayes Theorem

Bayes theorem is arguably the most important result in the study of probabilistic models. Consider two random variables $\mathbf{x}$ and $\mathbf{z}$ on which is defined a joint probability density function $P(\mathbf{x}, \mathbf{z})$. The chain-rule of conditional probabilities can be used to expand this density function in two ways

$$\begin{aligned} P(\mathbf{x}, \mathbf{z}) &= P(\mathbf{x} \mid \mathbf{z})P(\mathbf{z}) \\ &= P(\mathbf{z} \mid \mathbf{x})P(\mathbf{x}). \end{aligned} \tag{11}$$

Rearranging in terms of one of the conditional densities, Bayes theorem is obtained

$$P(\mathbf{x} \mid \mathbf{z}) = \frac{P(\mathbf{z} \mid \mathbf{x})P(\mathbf{x})}{P(\mathbf{z})}. \tag{12}$$

The value of this result lies in the interpretation of the probability density functions $P(\mathbf{x} \mid \mathbf{z})$, $P(\mathbf{z} \mid \mathbf{x})$, and $P(\mathbf{x})$. Suppose it is necessary to determine the various likelihoods of different values of an unknown state of nature $\mathbf{x} \in \mathcal{X}$. There may be prior beliefs about what values of $\mathbf{x}$ might be expect, encoded in the form of relative likelihoods in the **prior probability density function** $P(\mathbf{x})$. To obtain more information about the state $\mathbf{x}$ an observation $\mathbf{z} \in \mathcal{Z}$ is made. The observations made are modeled as a conditional probability density function $P(\mathbf{z} \mid \mathbf{x})$ which describes, for each fixed state of nature $\mathbf{x} \in \mathcal{X}$, the likelihood that the observation $\mathbf{z} \in \mathcal{Z}$ will be made; the probability of $\mathbf{z}$ given $\mathbf{x}$. The new likelihoods associated with the state of nature $\mathbf{x}$ must now be computed from the original prior information and the information gained by observation. This is encoded in the **posterior distribution** $P(\mathbf{x} \mid \mathbf{z})$ which describes the likelihoods associated with $\mathbf{x}$ *given* the observation $\mathbf{z}$. The marginal distribution $P(\mathbf{z})$ simply serves to normalize the posterior. The value of Bayes theorem is now clear, it provides a direct means of combining observed information with prior beliefs about the state of the world. Unsurprisingly, Bayes theorem lies at the heart of many data fusion algorithms.

The conditional distribution $P(\mathbf{z} \mid \mathbf{x})$ serves the role of a *sensor model.* This distribution can be thought of in two ways. First, in building a sensor model, the distribution is constructed by fixing the value[1] of $\mathbf{x} = x$ and then asking what *pdf* in the variable $\mathbf{z}$ results. Thus, in this case, $P(\mathbf{z} \mid x)$ is considered as a distribution on $\mathbf{z}$. For example, if we know the true range to a target (x), then $P(\mathbf{z} \mid x)$ is the distribution on the actual observation of this range. Conversely, once the sensor model exists, observations (numbers, not distributions) are made and $\mathbf{z} = z$ is fixed. From this however, we want to infer the state $\mathbf{x}$. Thus the distribution $P(z \mid \mathbf{x})$ is now considered as a distribution in $\mathbf{x}$. In this latter, case, the distribution is known as the **Likelihood Function** and the dependence on $\mathbf{x}$ is made clear by writing $\Lambda(\mathbf{x}) = P(z \mid \mathbf{x})$.

[1]Random variables are denoted by a bold face font, specific values taken by the random variable are denoted by normal fonts.

In a practical implementation of Equation 12, $P(\mathbf{z} \mid \mathbf{x})$ is constructed as a function of both variables (or a matrix in discrete form). For each fixed value of $\mathbf{x}$, a distribution in $\mathbf{z}$ is defined. Therefore as $\mathbf{x}$ varies, a family of distributions in $\mathbf{z}$ is created. The following two examples, one in continuous variables and one in discrete variables, makes these ideas clear.

Example 1

Consider a continuous valued state $\mathbf{x}$*, the range to target for example, and an observation* $\mathbf{z}$ *of this state. A commonly used model for such an observation is where the observation made of true state is Normally (Gaussian) distributed with mean* $\mathbf{x}$ *and a variance* σ_z^2 *as*

$$P(\mathbf{z} \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\left(-\frac{1}{2}\frac{(\mathbf{z}-\mathbf{x})^2}{\sigma_z^2}\right). \tag{13}$$

It should be clear that this is a simple function of both $\mathbf{z}$ *and* $\mathbf{x}$*. If we know the true value of the state,* $\mathbf{x} = x$*, then the distribution is a function of* $\mathbf{z}$ *only; describing the probability of observing a particular value of range (Normally distributed around the true range* x *with variance* σ_z^2*). Conversely, if we make a specific observation,* $\mathbf{z} = z$*, then the distribution is a function of* $\mathbf{x}$ *only; describing the probability of the true range value (Normally distributed around the range observation* z *with variance* σ_z^2*). In this case, the distribution is the Likelihood Function.*

Now assume that we have some prior belief about the true state $\mathbf{x}$ *encoded in a Gaussian prior as*

$$P(\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}-x_p)^2}{\sigma_x^2}\right).$$

Note, this is a function of a single variable $\mathbf{x}$ *(with* x_p *fixed). Bayes theorem can be directly applied to combine this prior information with information from a sensor, modeled by 13. First, an observation (technically, an experimental realisation),* z*, is made and instantiated in Equation 13. Then the prior and sensor model are multiplied together to produce a posterior distribution (which is a function of* $\mathbf{x}$ *only, and is sometimes referred to as the posterior likelihood) as*

$$\begin{aligned} P(\mathbf{x} \mid \mathbf{z}) &= C\frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}-z)^2}{\sigma^2}\right).\frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}-x_p)^2}{\sigma_x^2}\right) && (14) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}-\overline{x})^2}{\sigma^2}\right) && (15) \end{aligned}$$

where C *is a constant independent of* $\mathbf{x}$ *chosen to ensure that the posterior is appropriately normalized, and* $\overline{x}$ *and* σ^2 *are given by*

$$\overline{x} = \frac{\sigma_x^2}{\sigma_x^2+\sigma_z^2}z + \frac{\sigma_z^2}{\sigma_x^2+\sigma_z^2}x_p,$$

and

$$\sigma^2 = \frac{\sigma_z^2 \sigma_x^2}{\sigma_z^2 + \sigma_x^2} = \left(\frac{1}{\sigma_z^2} + \frac{1}{\sigma_x^2} \right)^{-1}$$

Thus the posterior is also Gaussian with a mean that is the weighted average of the means for the original prior and likelihood, and with a variance equal to the parallel combination of the original variances.

Example 2

Consider a simple example of the application of Bayes theorem to estimating a discrete parameter on the basis of one observation and some prior information. The environment of interest is modeled by a single state $\mathbf{x}$ *which can take on one of three values:*

x_1*:* $\mathbf{x}$ *is a type 1 target.*
x_2*:* $\mathbf{x}$ *is a type 2 target.*
x_3*: No visible target.*

A single sensor observes $\mathbf{x}$ *and returns three possible values:*

z_1*: Observation of a type 1 target.*
z_2*: Observation of a type 2 target.*
z_3*: No target observed.*

The sensor model is described by the likelihood matrix $P_1(\mathbf{z} \mid \mathbf{x})$*:*

	z_1	z_2	z_3
x_1	0.45	0.45	0.1
x_2	0.45	0.45	0.1
x_3	0.1	0.1	0.8

Note, that this likelihood matrix is a function of both $\mathbf{x}$ *and* $\mathbf{z}$*. For a fixed value of the true state, it describes the probability of a particular observation being made (the rows of the matrix). When a specific observation is made, it describes a probability distribution over the values of true state (the columns) and is then the Likelihood Function* $\Lambda(\mathbf{x})$*.*

The posterior distribution of the true state $\mathbf{x}$ *after making an observation* $\mathbf{z} = z_i$ *is given by*

$$P(\mathbf{x} \mid \ z_i) = \alpha P_1(z_i \mid \mathbf{x}) P(\mathbf{x})$$

where α *is simply a normalizing constant set by requiring the sum, over* $\mathbf{x}$*, of posterior probabilities to be equal to 1.*

In the first instance we will assume that we do not have any prior information about the possible likelihood of target types 1 and 2, and so we set the prior probability vector to $P(\mathbf{x}) = (0.333, 0.333, 0.333)$*. If we now observe* $\mathbf{z} = z_1$*, then clearly the posterior distribution will be given by* $P(\mathbf{x} \mid z_1) = (0.45, 0.45, 0.1)$ *(i.e. the first column of the likelihood matrix above; the likelihood function given* z_1 *has been observed).*

If now we subsequently use this posterior distribution as the prior for a second observation $P(\mathbf{x}) = (0.45, 0.45, 0.1)$*, and we again make the observation* $\mathbf{z} = z_1$*, then the new posterior distribution will be given by*

$$\begin{aligned} P(\mathbf{x} \mid z_1) &= \alpha P_1(z_1 \mid \mathbf{x}) P(\mathbf{x}) \\ &= \alpha \times (0.45, 0.45, 0.1) \otimes (0.45, 0.45, 0.1) \\ &= (0.488, 0.488, 0.024). \end{aligned}$$

(where the notation $\otimes$ *denotes an element-wise product).*

Notice that the result of this integration process is to increase the probability in both type 1 and type 2 targets at the expense of the no-target hypothesis. Clearly, although this sensor is good at detecting targets, it is not good at distinguishing between targets of different types.

2.2.2 Data Fusion using Bayes Theorem

It is possible to apply Bayes theorem directly to the integration of observations from several different sources. Consider the set of observations

$$\mathbf{Z}^n \triangleq \{\mathbf{z}_1 \in \mathcal{Z}_1, \cdots, \mathbf{z}_n \in \mathcal{Z}_n\} .$$

It is desired to use this information to construct a posterior distribution $P(\mathbf{x} \mid \mathbf{Z}^n)$ describing the relative likelihoods of the various values of the state of interest $\mathbf{x} \in \mathcal{X}$ given the information obtained. In principle, Bayes theorem can be directly employed to compute this distribution function from

$$\begin{aligned} P(\mathbf{x} \mid \mathbf{Z}^n) &= \frac{P(\mathbf{Z}^n \mid \mathbf{x}) P(\mathbf{x})}{P(\mathbf{Z}^n)} \\ &= \frac{P(\mathbf{z}_1, \cdots, \mathbf{z}_n \mid \mathbf{x}) P(\mathbf{x})}{P(\mathbf{z}_1, \cdots, \mathbf{z}_n)}. \end{aligned} \tag{16}$$

In practice it would be difficult to do this because it requires that the joint distribution $P(\mathbf{z}_1, \cdots, \mathbf{z}_n \mid \mathbf{x})$ is known completely; that is, the joint distribution of all possible combinations of observations conditioned on the underlying state[2]. However, it is usually quite reasonable to assume that *given* the true state $\mathbf{x} \in \mathcal{X}$, the information obtained from the i^{th} information source is independent of the information obtained from other sources. The validity of this assumption is discussed below. With this assumption, Equation 8 implies that

$$P(\mathbf{z}_i \mid \mathbf{x}, \mathbf{z}_1, \cdots, \mathbf{z}_{i-1}, \mathbf{z}_{i+1}, \cdots, \mathbf{z}_n) = P(\mathbf{z}_i \mid \mathbf{x}), \tag{17}$$

and from Equation 10 this gives

$$P(\mathbf{z}_1, \cdots, \mathbf{z}_n \mid \mathbf{x}) = P(\mathbf{z}_1 \mid \mathbf{x}) \cdots P(\mathbf{z}_n \mid \mathbf{x}) = \prod_{i=1}^{n} P(\mathbf{z}_i \mid \mathbf{x}). \tag{18}$$

[2]In Example 2 above, this would require the construction of likelihood matrix of size m^n where m is the number of possible outcomes for each observation and where n is the number of observations made.

Substituting this back into Equation 16 gives

$$P(\mathbf{x} \mid \mathbf{Z}^n) = [P(\mathbf{Z}^n)]^{-1} P(\mathbf{x}) \prod_{i=1}^{n} P(\mathbf{z}_i \mid \mathbf{x}). \tag{19}$$

Thus the updated likelihoods in the state, the posterior distribution on $\mathbf{x}$, is simply proportional to the product of prior likelihood and individual likelihoods from each information source. The marginal distribution $P(\mathbf{Z}^n)$ simply acts as a normalising constant. Equation 19 provides a simple and direct mechanism for computing the relative likelihood in different values of a state from any number of observations or other pieces of information.

Equation 19 is known as the *independent likelihood pool* [8]. In practice, the conditional probabilities $P(\mathbf{z}_i \mid \mathbf{x})$ are stored *a priori* as functions of both $\mathbf{z}_i$ and $\mathbf{x}$. When an observation sequence $\mathbf{Z}^n = \{z_1, z_2, \cdots, z_n\}$ is made, the observed values are instantiated in this probability distribution and likelihood functions $\Lambda_i(\mathbf{x})$ are constructed, which are functions only of the unknown state $\mathbf{x}$. The product of these likelihood functions with the prior information $P(\mathbf{x})$, appropriately normalised, provides a posterior distribution $P(\mathbf{x} \mid \mathbf{Z}^n)$, which is a function of $\mathbf{x}$ only for a specific observation sequence $\{z_1, z_2, \cdots, z_n\}$. Figure 1 shows the structure of the independent likelihood pool in a centralised architecture.

The effectiveness of Equation 19 relies crucially on the assumption that the information obtained from different information sources is independent when conditioned on the true underlying state of the world; this is defined in Equation 17. It would be right to question if this assumption is reasonable. It is clearly unreasonable to state that the information obtained is unconditionally independent;

$$P(\mathbf{z}_1, \cdots, \mathbf{z}_n) \neq P(\mathbf{z}_1) \cdots P(\mathbf{z}_n), \tag{20}$$

because each piece of information depends on a *common* underlying state $\mathbf{x} \in \mathcal{X}$. If the information obtained were independent of this state, and therefore unconditionally independent of other information sources, there would be little value in using it to improve knowledge of the state. It is precisely because the information obtained is unconditionally dependent on the underlying state that it has value as an information source. Conversely, it is generally quite reasonable to assume that the underlying state is the *only* thing in common between information sources and so once the state has been specified it is correspondingly reasonable to assume that the information gathered is conditionally independent given this state. There are sometimes exceptions to this general rule, particularly when the action of sensing has a non-trivial effect on the environment. More complex dependencies are considered in Section 2.2.4

Example 3

Consider again Example 2 of the discrete observation of target type. A second sensor is obtained which makes the same three observations as the first sensor, but whose likelihood

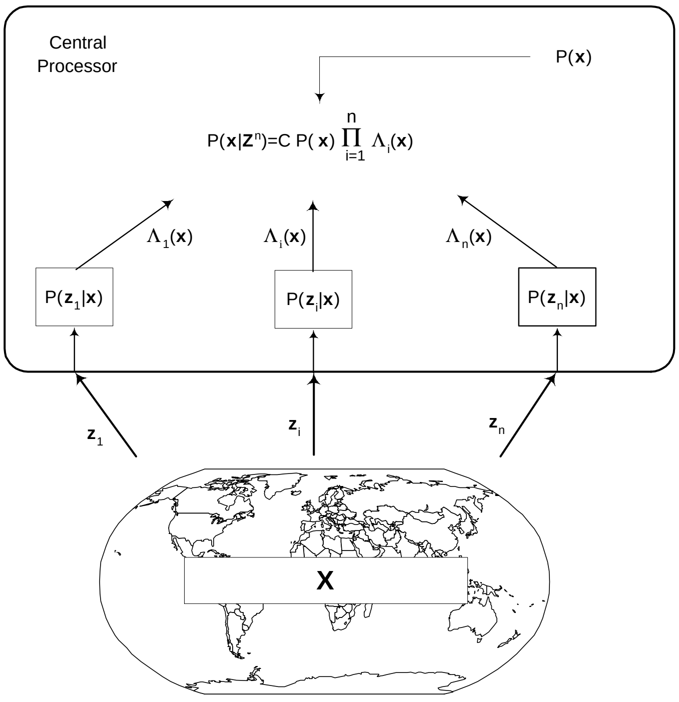


Figure 1: The centralised implementation of the independent likelihood pool as a method for combining information from a number of sources. The central processor maintains a model of each sensor i in terms of a conditional probability distribution $P_i(\mathbf{z}_i \mid \mathbf{x})$, together with any prior probabilistic knowledge $P(\mathbf{x})$. On arrival of a measurement set $\mathbf{Z}^n$, each sensor model is instantiated with the associated observation to form a likelihood $\Lambda_i(\mathbf{x})$. The normalised product of these yields the posterior distribution $P(\mathbf{x} \mid \mathbf{Z}^n)$.

matrix $P_2(\mathbf{z}_2 \mid \mathbf{x})$ is described by

	z_1	z_2	z_3
x_1	0.45	0.1	0.45
x_2	0.1	0.45	0.45
x_3	0.45	0.45	0.1

Whereas the first sensor was good at detecting targets but not at distinguishing between different target types, this second sensor has poor overall detection probabilities but good target discrimination capabilities. So for example, with a uniform prior, if we observe $\mathbf{z} = z_1$ with this second sensor, the posterior distribution on possible true states will be given by $P(\mathbf{x} \mid z_1) = (0.45, 0.1, 0.45)$ (i.e the first column of the likelihood matrix) .

It clearly makes sense to combine the information from both sensors to provide a system with both good detection and good discrimination capabilities. From Equation 19, the product of the two likelihood functions gives us an overall likelihood function for the combined system as $P_{12}(\mathbf{z}_1, \mathbf{z}_2 \mid \mathbf{x}) = P_1(\mathbf{z}_1 \mid \mathbf{x})P_2(\mathbf{z}_2 \mid \mathbf{x})$. Thus if we observe $\mathbf{z}_1 = z_1$

using the first sensor, and $\mathbf{z}_2 = z_1$ *with the second sensor (assuming a uniform prior), then the posterior likelihood in* $\mathbf{x}$ *is given by*

$$\begin{aligned} P(\mathbf{x} \mid z_1, z_1) &= \alpha P_{12}(z_1, z_1 \mid \mathbf{x}) \\ &= \alpha P_1(z_1 \mid \mathbf{x}) P_2(z_1 \mid \mathbf{x}) \\ &= \alpha \times (0.45, 0.45, 0.1) \otimes (0.45, 0.1, 0.45) \\ &= (0.6924, 0.1538, 0.1538) \end{aligned}$$

Comparing this to taking two observations of z_1 *with sensor 1 (in which the resulting posterior was* $(0.488, 0.488, 0.024)$*) it can be seen that sensor 2 adds substantial target discrimination power at the cost of a slight loss of detection performance for the same number of observations.*

Repeating this calculation for each $\mathbf{z}_1$*,* $\mathbf{z}_2$ *observation pair, results in the combined likelihood matrix*

$\mathbf{z}_1 = z_1$

$\mathbf{z}_2 =$	z_1	z_2	z_3
x_1	0.6924	0.1538	0.4880
x_2	0.1538	0.6924	0.4880
x_3	0.1538	0.1538	0.0240

$\mathbf{z}_1 = z_2$

$\mathbf{z}_2 =$	z_1	z_2	z_3
x_1	0.6924	0.1538	0.4880
x_2	0.1538	0.6924	0.4880
x_3	0.1538	0.1538	0.0240

$\mathbf{z}_1 = z_3$

$\mathbf{z}_2 =$	z_1	z_2	z_3
x_1	0.1084	0.0241	0.2647
x_2	0.0241	0.1084	0.2647
x_3	0.8675	0.8675	0.4706

The combined sensor provides substantial improvements in overall system performance[3]*. If for example we observe target 1 with the first sensor (the array block* $\mathbf{z}_1 = z_1$*) and again*

[3]Note that summing over any column still come to 1. In practical implementations, it is often sufficient to encode relative likelihoods of different events and to normalize only when computing the posterior distribution.

observe target 1 with the second sensor (the first column of this block), then the posterior distribution in the three hypotheses is

$$P(\mathbf{x} \mid \mathbf{z}_1, \mathbf{z}_2) = (0.692, 0.154, 0.154),$$

and so target 1 is clearly the most probable target. If however, we observe a type 2 target with the second sensor after having observed a type 1 target with the first sensor, a similar calculation gives the posterior as $(0.154, 0.692, 0.154)$, *that is target type 2 has high probability. This is because although sensor 1 observed a type 1 target, the likelihood function for sensor 1 tells us that it is poor at distinguishing between target types and so sensor 2 information is used for this purpose. If now we observe no target with sensor 2, having detected target type 1 with the first sensor, the posterior given both observations is given by* $(0.488, 0.488, 0.024)$. *That is we still believe that there is a target (because we know sensor 1 is much better at target detection than sensor 2), but we still have no idea which of target 1 or 2 it is as sensor 2 has been unable to make a valid detection. The analysis for sensor 1 detecting target 2 is identical to that for detection of target 1. Finally, if sensor 1 gets no detection, but sensor 2 detects target type 1, then the posterior likelihood is given by* $(0.108, 0.024, 0.868)$. *That is we still believe there is no target because we know sensor 1 is better at providing this information (and perversely, sensor 2 confirms this even though it has detected target type 1).*

Practically, the joint likelihood matrix is never constructed (it is easy to see why here, with $n = 3$ *sensors, and* $m = 3$ *possible observations and* $k = 3$ *possible outcomes, the dimension of the joint likelihood matrix has* $k \times m^n = 27$ *entries.) Rather, the likelihood matrix is constructed for each sensor and these are only combined when instantiated with an observation. Storage then reduces to* n *arrays of dimension* $k \times m$, *at the cost of a* k *dimensional vector multiply of the instantiated likelihood functions. This is clearly a major saving in storage and complexity and underlines the importance of the conditional independence assumption to reduction in computational complexity.*

2.2.3 Recursive Bayes Updating

The integration of information using Equation 19 would, in principle, require that all past information is remembered and, on arrival of new information in the form $P(\mathbf{z}_k \mid \mathbf{x})$, that the total likelihood be recomputed based on all information gathered up to this time. However, Bayes theorem, in the form of Equation 19, also lends itself to the incremental or recursive addition of new information in determining a revised posterior distribution on the state. With $\mathbf{Z}^k \triangleq \{\mathbf{z}_k, \mathbf{Z}^{k-1}\}$

$$\begin{aligned} P(\mathbf{x}, \mathbf{Z}^k) &= P(\mathbf{x} \mid \mathbf{Z}^k) P(\mathbf{Z}^k) && (21) \\ &= P(\mathbf{z}_k, \mathbf{Z}^{k-1} \mid \mathbf{x}) P(\mathbf{x}) \\ &= P(\mathbf{z}_k \mid \mathbf{x}) P(\mathbf{Z}^{k-1} \mid \mathbf{x}) P(\mathbf{x}), && (22) \end{aligned}$$

where it is assumed conditional independence of the observation sequence. Equating both sides of this expansion gives

$$\begin{aligned} P(\mathbf{x} \mid \mathbf{Z}^k)P(\mathbf{Z}^k) &= P(\mathbf{z}_k \mid \mathbf{x})P(\mathbf{Z}^{k-1} \mid \mathbf{x})P(\mathbf{x}) && (23) \\ &= P(\mathbf{z}_k \mid \mathbf{x})P(\mathbf{x} \mid \mathbf{Z}^{k-1})P(\mathbf{Z}^{k-1}). && (24) \end{aligned}$$

Noting that $P(\mathbf{Z}^k)/P(\mathbf{Z}^{k-1}) = P(\mathbf{z}_k \mid \mathbf{Z}^{k-1})$ and rearranging gives

$$P(\mathbf{x} \mid \mathbf{Z}^k) = \frac{P(\mathbf{z}_k \mid \mathbf{x})P(\mathbf{x} \mid \mathbf{Z}^{k-1})}{P(\mathbf{z}_k \mid \mathbf{Z}^{k-1})}. \tag{25}$$

The advantage of Equation 25 is that we need compute and store only the posterior likelihood $P(\mathbf{x} \mid \mathbf{Z}^{k-1})$ which contains a complete summary of all past information. When the next piece of information $P(\mathbf{z}_k \mid \mathbf{x})$ arrives, the previous posterior takes on the role of the current prior and the product of the two becomes, when normalised, the new posterior. Equation 25 thus provides a significant improvement in computational and memory requirements over Equation 19.

Example 4

An important example of the recursive application of Bayes theorem is in the calculation of the posterior distribution of a scalar $\mathbf{x}$ *under the assumption that the likelihood function for the observations given the true state is Gaussian with known variance* σ^2*;*

$$P(\mathbf{z}_k \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(\mathbf{z}_k - \mathbf{x})^2}{\sigma^2}\right).$$

If we assume that the posterior distribution in $\mathbf{x}$ *after taking the first* $k-1$ *observations is also Gaussian with mean* $\mathbf{x}_{k-1}$ *and variance* σ^2_{k-1}*,*

$$P(\mathbf{x} \mid \mathbf{Z}^{k-1}) = \frac{1}{\sqrt{2\pi\sigma^2_{k-1}}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}_{k-1} - \mathbf{x})^2}{\sigma^2_{k-1}}\right).$$

then the posterior distribution in $\mathbf{x}$ *after the first* k *observations is given by*

$$\begin{aligned} P(\mathbf{x} \mid \mathbf{Z}^k) &= K\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(\mathbf{z}_k - \mathbf{x})^2}{\sigma^2}\right) . \frac{1}{\sqrt{2\pi\sigma^2_{k-1}}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}_{k-1} - \mathbf{x})^2}{\sigma^2_{k-1}}\right) && (26) \\ &= \frac{1}{\sqrt{2\pi\sigma^2_k}} \exp\left(-\frac{1}{2}\frac{(\mathbf{x}_k - \mathbf{x})^2}{\sigma^2_k}\right) && (27) \end{aligned}$$

where K *is a constant independent of* $\mathbf{x}$ *chosen to ensure that the posterior is appropriately normalized, and* $\mathbf{x}_k$ *and* σ^2_k *are given by*

$$\mathbf{x}_k = \frac{\sigma^2_{k-1}}{\sigma^2_{k-1} + \sigma^2}\mathbf{z}_k + \frac{\sigma^2}{\sigma^2_{k-1} + \sigma^2}\mathbf{x}_{k-1}, \tag{28}$$

and

$$\sigma_k^2 = \frac{\sigma^2 \sigma_{k-1}^2}{\sigma^2 + \sigma_{k-1}^2} \tag{29}$$

The most important point to note about the posterior is that it too is a Gaussian; the product of two Gaussian distributions is itself Gaussian. Distributions that have this symmetry property are known as conjugate distributions. Given this property, it is clearly not necessary to go through the process of multiplying distributions together as it is sufficient to simply compute the new mean and variance recursively from Equations 28 and 29 as these completely characterize the associated Gaussian distribution. With these simple recursion equations any number of observations may be fused in a simple manner.

Example 5

Quite general prior and likelihood distributions can be handled by direct application of Bayes Theorem. Consider the problem in which we are required to determine the location of a target in a defined area. Figure 2(a) shows a general prior probability distribution defined on the search area. The distribution is simply defined as a probability value $P(x_i, y_j)$ *on a grid point at location* x_i, y_j*, of area* $\mathrm{d}x_i$*,* $\mathrm{d}y_j$*. The only constraints placed on this distribution are that*

$$P(x_i, y_j) > 0, \qquad \forall x_i, y_j$$

and that

$$\sum_i \sum_j P(x_i, x_j) \mathrm{d}x_i \mathrm{d}y_i = 1$$

(which can easily be satisfied by appropriate normalisation.)

A sensor (sensor 1) now takes observations of the target from a sensor located at $x = 15, y = 0km$*. The likelihood function generated from this sensor following an observation* z_1 *is shown in Figure 2(b). This likelihood* $P_1(z_1 \mid \mathbf{x}_i, \mathbf{y}_j)$ *again consists of a general location probability defined on the* x_i y_j *grid. The likelihood shows that the bearing resolution of the sensor is high, whereas it has almost no range accuracy (the likelihood is long and thin with probability mass concentrated on a line running from sensor to target). The posterior distribution having made this first observation is shown in Figure 2(c) and is computed from the point-wise product of prior and likelihood,*

$$P(\mathbf{x}_i, \mathbf{y}_j \mid z_1) = \alpha \times P_1(z_1 \mid \mathbf{x}_i, \mathbf{y}_j) \otimes P(\mathbf{x}_i, \mathbf{y}_j),$$

where α *is simply a normalising constant. It can be seen that the distribution defining target location is now approximately restrained to a line along the detected bearing. Figure 2(d) shows the posterior* $P(\mathbf{x}_i, \mathbf{y}_j \mid z_1, z_2)$ *following a second observation* z_2 *by the same sensor. This is again computed by point-wise multiplication of the likelihood* $P_1(z_2 \mid \mathbf{x}_i, \mathbf{y}_j)$ *with the new prior (the posterior from the previous observation* $P(\mathbf{x}_i, \mathbf{y}_j \mid z_1)$*). It can be seen that there is little improvement in location density following this second observation; this is to be expected as there is still little range data available.*

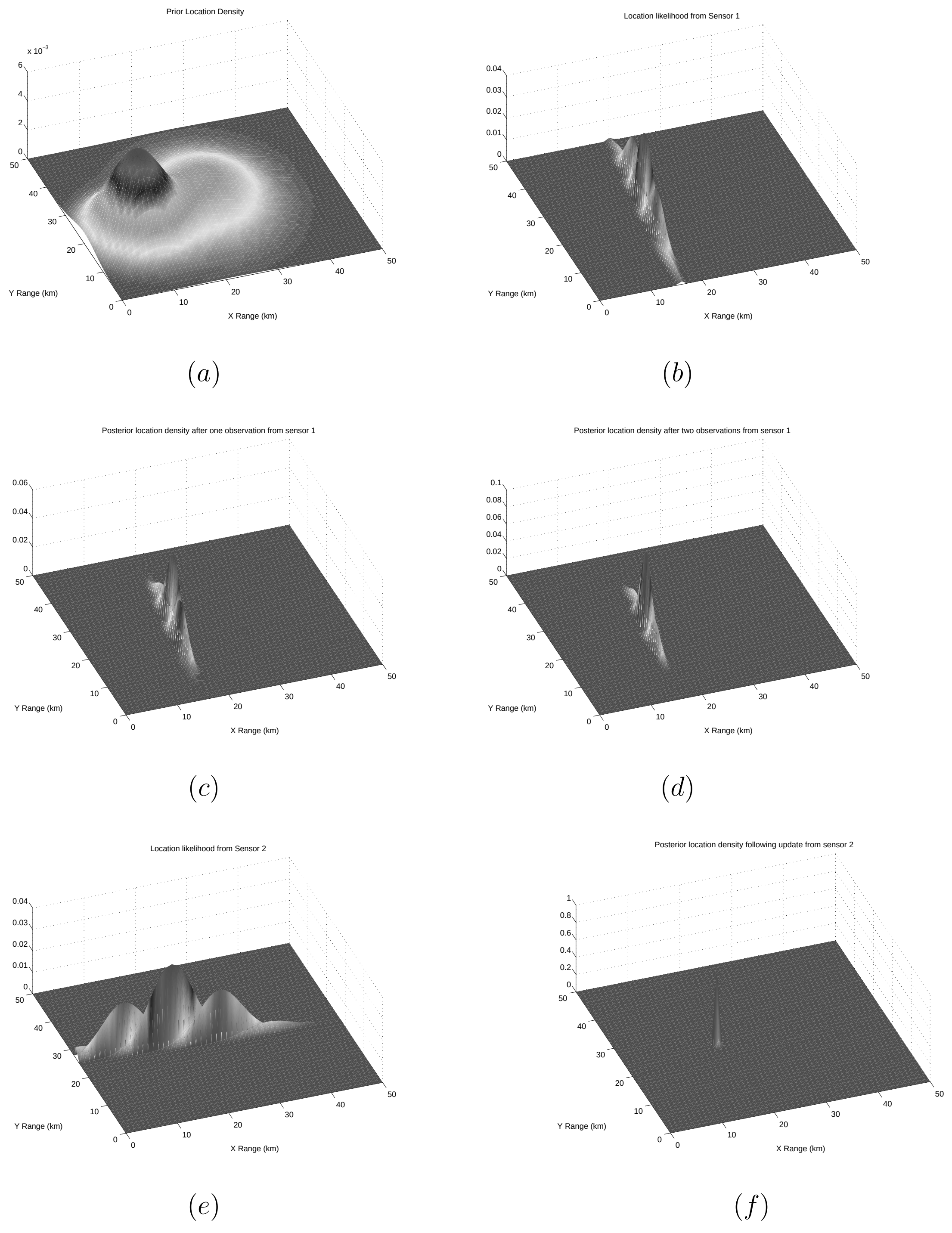


Figure 2: Generalised Bayes Theorem. The figures show plots of two-dimensional distribution functions defined on a grid of x and y points: (a) Prior distribution; (b) likelihood function for first sensor; (c) posterior after one application of Bayes Theorem; (d) posterior after two applications; (e) likelihood function for second sensor; (f) final posterior.

A second sensor (sensor 2) now takes observations of the target from a location $x = 50, y = 20$. Figure 2(e) shows the target likelihood $P_2(z_3 \mid \mathbf{x}_i, \mathbf{y}_j)$ following an observation z_3 by this sensor. It can be seen that this sensor (like sensor 1), has high bearing resolution, but almost no range resolution. However, because the sensor is located at a different site, we would expect that the combination of bearing information from the two sensors would provide accurate location data. Indeed, following point-wise multiplication of the second sensor likelihood with the new prior (the posterior $P(\mathbf{x}_i, \mathbf{y}_j \mid z_1, z_2)$ from the previous two observations of sensor 1), we obtain the posterior $P(\mathbf{x}_i, \mathbf{y}_j \mid z_1, z_2, z_3)$ shown in Figure 2(f) which shows all probability mass highly concentrated around a single target location.

2.2.4 Data Dependency and Bayes Networks

Bayes Theorem provides a general method for reasoning about probabilities and fusing information. As has been seen, in complex problems involving many sources of information, it is usual to assume conditional independence of observation information (that is, the observations are conditionally independent given the underlying state). This is necessary to avoid the problem of specifying very large numbers of likelihoods associated with each permutation of observations (if there are p possible outcomes for each observation then the specification of the joint likelihood $P(\mathbf{z}_1, \cdots, \mathbf{z}_n \mid \mathbf{x})$ requires the specification of p^n probabilities, whereas if conditional independence is assumed, we need only specify n likelihoods $P_i(\mathbf{z}_i \mid \mathbf{x})$, or a total of $p \times n$ probabilities).

In many situations it is not possible to simply assume conditional independence of observations, and indeed there is often a clear dependency between sources of information (when each source depends on some underlying assumption or report from a third source, for example). In such cases, it is easy to be swamped by the complexity of the various dependencies between information reports. However, even in such cases, it is still possible to make good use of the rules of conditioning to exploit any independence that may exist. Recall that in general, without any assumptions of independence, the joint distribution on any set of random variables $\mathbf{x}_i$ may be expanded using the chain-rule of conditional probabilities as

$$P(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n) = P(\mathbf{x}_1 \mid \mathbf{x}_2, \cdots, \mathbf{x}_n) \cdots P(\mathbf{x}_{n-1} \mid \mathbf{x}_n) P(\mathbf{x}_n),$$

where the expansion may be taken in any convenient order. If, all variables depend on all other variables, then there is little to be gained by this expansion. If however a variable $\mathbf{x}_j$, for example, depends directly on only one or a few other variables $\mathbf{x}_h$, then the appropriate expansion can substantially reduce the number of probabilities that need be stored,

$$P(\mathbf{x}_j, \mathbf{x}_h, \cdots, \mathbf{x}_n) = P(\mathbf{x}_j \mid \mathbf{x}_h) P(\mathbf{x}_h \cdots \mathbf{x}_n).$$

Such direct conditional independencies are common if only because the acquisition of information is causal. For example, information about a state $\mathbf{x}_{k+1}$ at time $k + 1$ will depend on all information obtained up to time $k+1$. However, if all past information up to time k is summarised in the variable $\mathbf{x}_k$, then the new variable $\mathbf{x}_{k+1}$ becomes conditionally independent of all past information (up to k), once $\mathbf{x}_k$ is specified (essentially $\mathbf{x}_k$ tells us all we need to know about past observations and so re-stating these observations adds no more knowledge; this is the well-known Markov property).

The issue of dependency between variables is often best expressed graphically, in terms of a network describing relationships. Here, the relations between different states are explicitly represented in a graph structure, with the vertices of the graph representing individual states and their probability distributions, and the edges of the graph representing conditional probabilities of adjacent nodes. In the discrete case, the graph requires the specification of a conditional probability matrix of dimension $n_i \times n_j$ for each edge relating two states $\mathbf{x}_i$ and $\mathbf{x}_j$ which can take on respectively n_i and n_j distinct values. From this a detailed computational scheme can be developed for propagating probabilistic information through the graph. These networks are referred to as Bayes Networks, Belief Networks or Probabilistic Networks. The landmark book by Pearl [35], describes the construction and use of these dependency networks in probabilistic reasoning. A detailed discussion of Bayes networks is beyond the scope of this course. The following example provides a brief introduction.

Example 6

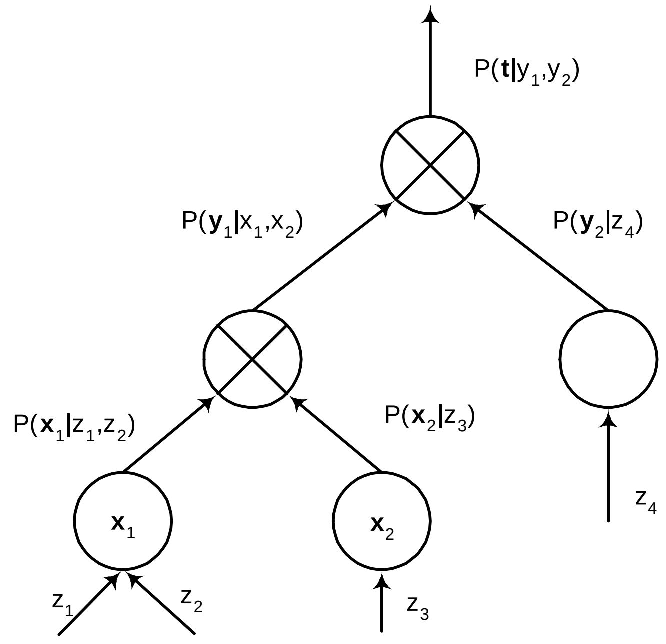


Figure 3: A Simple Bayesian Network (see text for details).

It is required to determine a target identity $\mathbf{t}$. *Target identity probabilities are obtained from two independent sources* $\mathbf{y}_1$ *and* $\mathbf{y}_2$. *The first source bases target reporting on two sensor sources* $\mathbf{x}_1$ *and* $\mathbf{x}_2$, *the first of these two sensors employing a pair of dependent observations* $\mathbf{z}_1$, $\mathbf{z}_2$, *and the second from a single observation* $\mathbf{z}_3$. *The second source* $\mathbf{y}_2$

bases its report directly on an observation $\mathbf{z}_4$*. The dependency structure is shown in the belief network of Figure 3. The conditional probabilities are specified at each of the nodes in the network (tree) structure. It is possible to specify these at the arcs and use the nodes as places where total probabilities (posteriors)are computed.*

The structure shown in the figure simply exposes the dependencies between variables and is a convenient way of describing, computationally, the role of each likelihood. The tree structure shown is common in causal processes. It is sometimes the case that reports are based on a common report or observation, in which case the tree becomes a general network structure. These cases are difficult to deal with other than explicitly computing the resulting dependence between report variables.

2.2.5 Distributed Data Fusion with Bayes Theorem

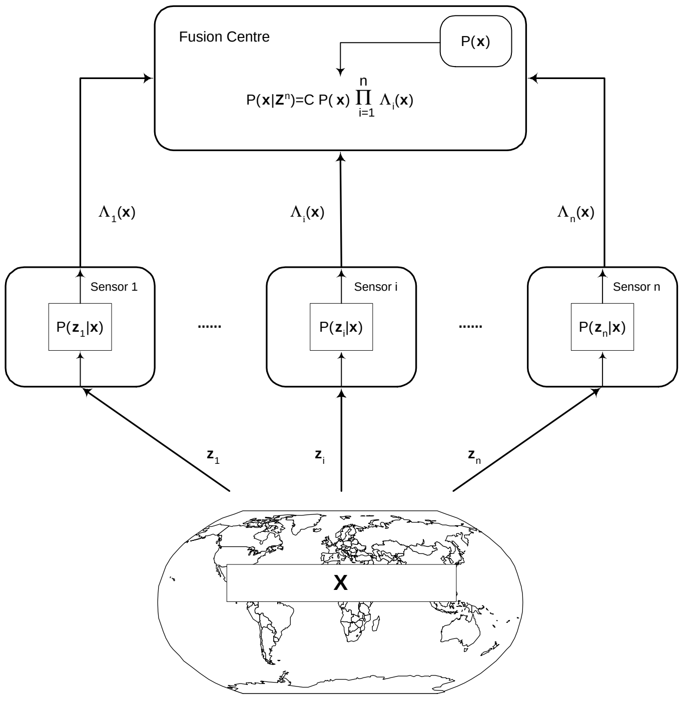


Figure 4: The distributed implementation of the independent likelihood pool. Each sensor maintains it's own model in the form of a conditional probability distribution $P_i(\mathbf{z}_i \mid \mathbf{x})$. On arrival of a measurement, z_i, the sensor model is instantiated with the associated observation to form a likelihood $\Lambda_i(\mathbf{x})$. This is transmitted to a central fusion centre were the normalised product of likelihoods and prior yields the posterior distribution $P(\mathbf{x} \mid \mathbf{Z}^n)$.

Providing the essential rules of conditional probabilities and Bayes Theorem are followed, it is not difficult to develop methods for distributing the data fusion problem. Figure 4 shows one such distributed architecture. In this case, the sensor models, in the form of likelihood functions, are maintained locally at each sensor site. When an observation is made, these likelihoods are instantiated to provide a likelihood function $\Lambda_i(\mathbf{x})$ describing a probability distribution over the true state of the world. Importantly, it is this likelihood that is transmitted to the fusion centre. Thus, the sensor talks to the centre in terms of the underlying state of the world, and not in terms of the raw observation (as is the case in Figure 1). This has the advantage that each sensor is 'modular' and talks in a common 'state' language. However, it has the disadvantage that a complete likelihood, rather than a single observation, must be communicated. The central fusion center simply computes the normalised product of communicated likelihoods and prior to yield a posterior distribution.

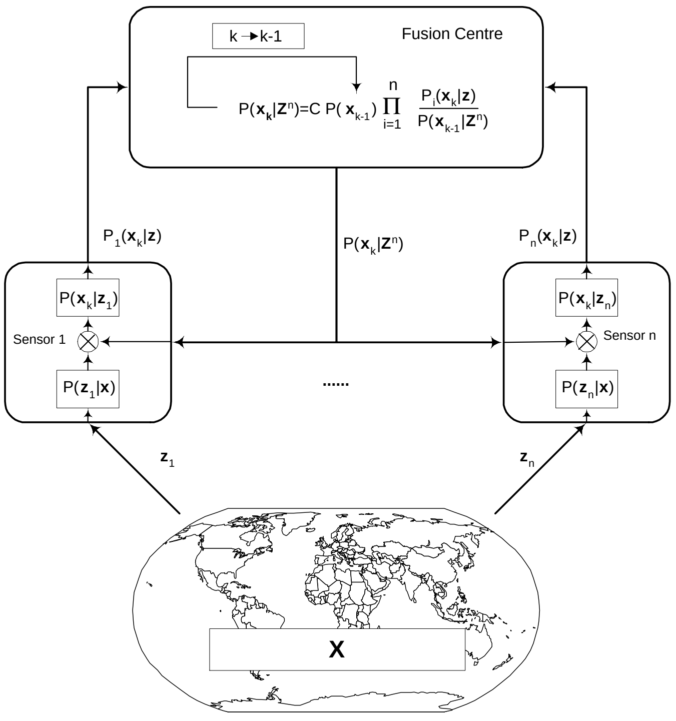

k → k-1
Fusion Centre
$P(\mathbf{x}_k|\mathbf{Z}^n)=C\,P(\mathbf{x}_{k-1})\prod_{i=1}^{n}\frac{P_i(\mathbf{x}_k|\mathbf{z})}{P(\mathbf{x}_{k-1}|\mathbf{Z}^n)}$
$P_1(\mathbf{x}_k|\mathbf{z})$
$P(\mathbf{x}_k|\mathbf{Z}^n)$
$P_n(\mathbf{x}_k|\mathbf{z})$
$P(\mathbf{x}_k|\mathbf{z}_1)$
Sensor 1
$P(\mathbf{z}_1|\mathbf{x})$
......
$P(\mathbf{x}_k|\mathbf{z}_n)$
Sensor n
$P(\mathbf{z}_n|\mathbf{x})$
$\mathbf{z}_1$
$\mathbf{z}_n$
X

Figure 5: A distributed implementation of the independent opinion pool in which each sensor maintains both it's own model and also computes a local posterior. The complete posterior is made available to all sensors and so they become, in some sense, autonomous. The figure shows Bayes Theorem in a recursive form.

A second approach to distributed data fusion using Bayes Theorem is shown in Figure 5. In this case, each sensor computes a likelihood but then combines this, locally, with the prior from the previous time-step, so producing a local posterior distribution on the state. This is then communicated to the central fusion centre. The fusion centre then recovers the new observation information by dividing each posterior by the communicated (global) prior and then taking a normalised product to produce a new global posterior. This posterior is then communicated back to the sensors and the cycle repeats in recursive form. The advantage of this structure is that global information, in state form, is made available at each local sensor site. This makes each sensor, in some sense, autonomous. The disadvantage of this architecture is the need to communicate state distributions both to and from the central fusion center. Distributed data fusion methods will be discussed in detail later in this course.

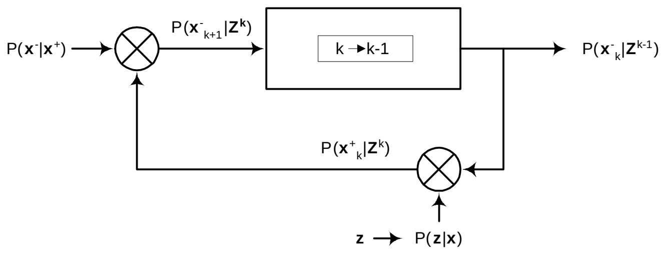


Figure 6: A Bayesian formulation of the classical feedback aided navigation system, fusing rate data with external observation information (see text for details).

Almost any data fusion structure is possible as long as the essential rules of conditional probability and Bayes theorem are followed. Figure 6 shows a Bayes filter in feedback form. The filter outputs state predictions in the form of a *pdf* $P(\mathbf{x}_k^- \mid \mathbf{Z}^{k-1})$ of the state $\mathbf{x}_k^-$ at time k given all observations $\mathbf{Z}^{k-1}$ up to time $k-1$. This is essentially a prior distribution and provides a distribution on the state at time k prior to an observation being made at this time (and thus is marked with a '$-$' superscript to denote 'before update'). The prior is fed-back and multiplied by the likelihood $\Lambda(\mathbf{x})$ associated with an incoming observation. This multiplication essentially implements Bayes Theorem to produce the posterior $P(\mathbf{x}_k^+ \mid \mathbf{Z}^k)$, where the superscript '+' denotes 'after observation'. The likelihood itself is generated from the sensor model $P(\mathbf{z} \mid \mathbf{x})$ in the normal manner. The posterior is fed-back and multiplied by another distribution $P(\mathbf{x}^- \mid \mathbf{x}^+)$ which essentially predicts the future state on the basis of current state. Multiplying this distribution by the fed-back posterior generates a new prior. The cycle, after delay, then repeats. This structure is a Bayesian implementation of the classical feed-back aided navigation system. The state prediction distribution $P(\mathbf{x}^- \mid \mathbf{x}^+)$ has the same role as an inertial system, using integrated rate information to generate predictions of state. The feedback loop is corrected by reference to some external observation (the aiding sensor) modeled by the likelihood function $P(\mathbf{z} \mid \mathbf{x})$. Thus rate and external information are fused in a predictor-corrector arrangement.

2.2.6 Data Fusion with Log-Likelihoods

In data fusion problems and in Bayes networks, it is often easier to work with the log of a probability, rather than the probability itself. These 'log-likelihoods' are more convenient computationally than probabilities as additions and subtractions rather than multiplications and divisions are employed in fusing probability information. Further, log-likelihoods are also more closely related to formal definitions of information.

The log-likelihood or conditional log-likelihood are defined as;

$$\mathbf{l}(\mathbf{x}) \stackrel{\triangle}{=} \log P(\mathbf{x}), \qquad \mathbf{l}(\mathbf{x} \mid \mathbf{y}) \stackrel{\triangle}{=} \log P(\mathbf{x} \mid \mathbf{y}). \tag{30}$$

It is clear that the log-likelihood is always less than zero and is only equal to zero when all probability mass is assigned to a single value of $\mathbf{x}$; $\mathbf{l}(\mathbf{x}) \leq 0$. The log-likelihood itself can sometimes be a useful and efficient means of implementing probability calculations. For example, taking logs of both sides of Equation 12 we can rewrite Bayes theorem in terms of log-likelihood as

$$\mathbf{l}(\mathbf{x} \mid \mathbf{z}) = \mathbf{l}(\mathbf{z} \mid \mathbf{x}) + \mathbf{l}(\mathbf{x}) - \mathbf{l}(\mathbf{z}). \tag{31}$$

Example 7

Consider again the two-sensor discrete target identification example (Example 3). The log-likelihood matrix for the first sensor (using natural logs) is

$$\begin{array}{cccc} & z_1 & z_2 & z_3 \\ x_1 & -0.799 & -2.303 & -0.799 \\ x_2 & -2.303 & -0.799 & -0.799 \\ x_3 & -0.799 & -0.799 & -2.303 \end{array}$$

and for the second

$$\begin{array}{cccc} & z_1 & z_2 & z_3 \\ x_1 & -0.799 & -2.303 & -0.799 \\ x_2 & -2.303 & -0.799 & -0.799 \\ x_3 & -0.799 & -0.799 & -2.303 \end{array}$$

The posterior likelihood (given a uniform prior) following observation of target 1 by sensor 1 and target 1 by sensor 2 is the sum *of the first columns of each of the likelihood matrices*

$$\begin{aligned} l(\mathbf{x} \mid z_1, z_1) &= l_1(z_1 \mid \mathbf{x}) + l_2(z_1 \mid \mathbf{x}) + C \\ &= (-0.7985, -0.7985, -2.3026) + (-0.7985, -2.3026, -0.7985) + C \\ &= (-1.5970, -3.1011, -3.1011) + C \\ &= (-0.3680, -1.8721, -1.8721) \end{aligned}$$

where the constant $C = 1.229$ is found through normalisation (which in this case requires that the anti-logs sum to one). The first thing to note is that the computation is obviously simpler than in the case of obtaining products of probability distributions, particularly as the dimension of the state vectors increase. The second thing is that the normalising constant is additive. Thus, normalisation need only occur if probabilities are required, otherwise, the relative magnitude of the log-likelihoods are sufficient to indicate relative likelihoods (the smaller the log-likelihood, the nearer the probability is to one).

Example 8

A second interesting example of log-likelihoods is in the case where prior and sensor likelihoods are Gaussian as in Example 4. Taking logs of Equation 27, gives

$$\begin{aligned} l(\mathbf{x} \mid \mathbf{Z}^k) &= -\frac{1}{2}\frac{(\mathbf{x}_k - \mathbf{x})^2}{\sigma_k^2} \\ &= -\frac{1}{2}\frac{(\mathbf{z}_k - \mathbf{x})^2}{\sigma^2} - \frac{1}{2}\frac{(\mathbf{x}_{k-1} - \mathbf{x})^2}{\sigma_{k-1}^2} + C. \end{aligned} \quad (32)$$

as before, completing squares gives

$$\mathbf{x}_k = \frac{\sigma_{k-1}^2}{\sigma_{k-1}^2 + \sigma^2}\mathbf{z}_k + \frac{\sigma^2}{\sigma_{k-1}^2 + \sigma^2}\mathbf{x}_{k-1},$$

and

$$\sigma_k^2 = \frac{\sigma^2 \sigma_{k-1}^2}{\sigma^2 + \sigma_{k-1}^2},$$

thus the log-likelihood is quadratic in $\mathbf{x}$*; for each value of* $\mathbf{x}$*, a log-likelihood is specified as* $-\frac{1}{2}\frac{(\mathbf{x}_k - \mathbf{x})^2}{\sigma_k^2}$*, modulo addition of a constant* C*.*

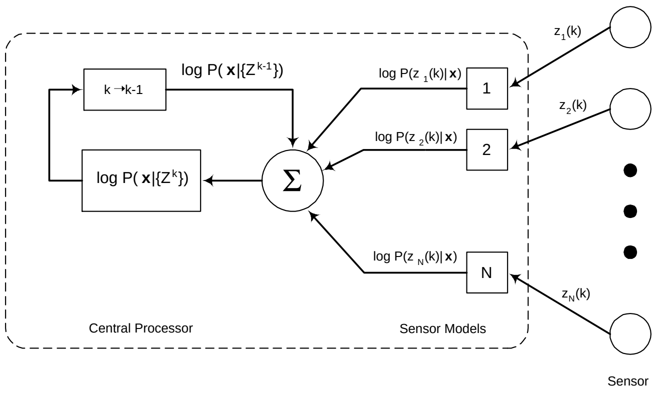


Figure 7: A log-likelihood implementation of a fully centralised data fusion architecture.

Log-likelihoods are a convenient way of implementing distributed data fusion architectures. Figure 7 shows a log-likelihood implementation of a fully centralised data fusion architecture (equivalent to Figure 1) in which observations are transmitted directly to a central fusion centre which maintains the sensor likelihood functions. Fusion of information is simply a matter of summing log-likelihoods. Figure 8 shows a log-likelihood implementation of the independent likelihood architecture (equivalent to Figure 4). In this case, each sensor maintains its own model and communicates log-likelihoods to a central processor. The fusion of log-likelihoods is again simply a summation. Figure 9 shows a log-likelihood implementation of the independent opinion pool (equivalent to Figure 5)

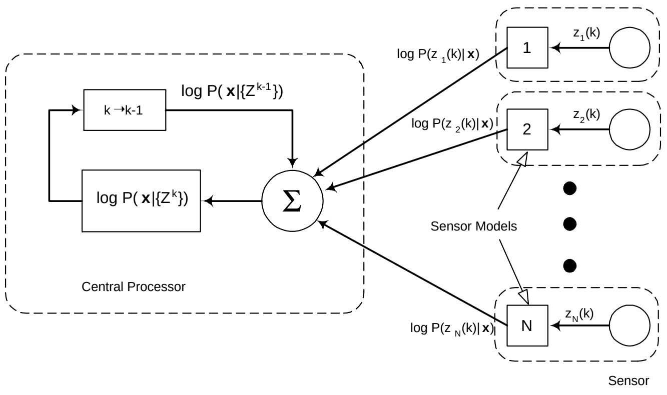


Figure 8: A log-likelihood implementation of the independent likelihood pool architecture.

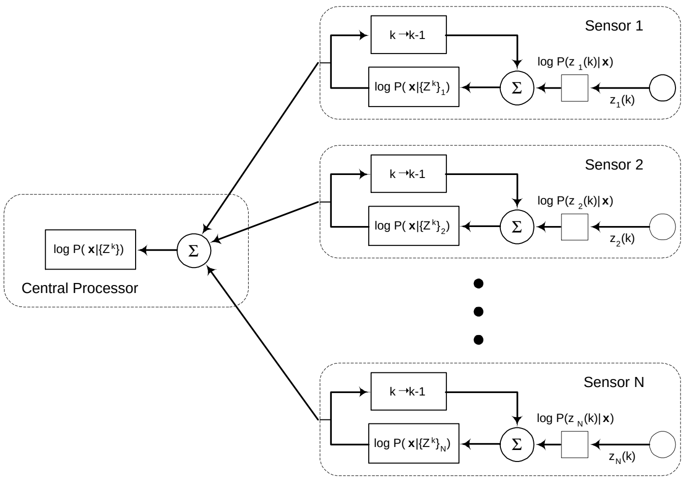


Figure 9: A log-likelihood implementation of the independent opinion pool architecture.

in which each local sensor maintains a posterior likelihood which is communicate to the central fusion center. In all cases, the log-likelihood implementation involves only simple addition and subtraction in the form of classical feed-back loops. These can be easily manipulated to yield a wide variety of different architectures.

2.3 Information Measures

Probabilities and log-likelihoods are defined on states or observations. It is often valuable to also measure the amount of *information* contained in a given probability distribution. Formally, information is a measure of the compactness of a distribution; logically if a probability distribution is spread evenly across many states, then it's information content is low, and conversely, if a probability distribution is highly peaked on a few states, then it's information content is high. Information is thus a function of the distribution, rather than the underlying state. Information measures play an important role in designing and managing data fusion systems. Two probabilistic measures of information are of particular value in data fusion problems; the Shannon information (or entropy) and the Fisher information.

2.3.1 Entropic Information

The entropy or Shannon information[4] $H_P(\mathbf{x})$ associated with a probability distribution $P(\mathbf{x})$, defined on a random variable $\mathbf{x}$, is defined as the expected value of minus the log-likelihood. For continuous-valued random variables this is given by (see [34] Chapter 15)

$$H_P(\mathbf{x}) \overset{\triangle}{=} -\mathrm{E}\{\log P(\mathbf{x})\} = -\int_{-\infty}^{\infty} P(\mathbf{x}) \log P(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{33}$$

and for discrete random variables

$$H_P(\mathbf{x}) \overset{\triangle}{=} -\mathrm{E}\{\log P(\mathbf{x})\} = -\sum_{\mathbf{x}\in\mathcal{X}} P(\mathbf{x}) \log P(\mathbf{x}). \tag{34}$$

Note that following convention, we have used $\mathbf{x}$ as an argument for $H_P(\cdot)$ even though the integral or sum is taken over values of $\mathbf{x}$ so $H_P(\cdot)$ is not strictly a function of $\mathbf{x}$ but is rather a function of the distribution $P(\cdot)$.

The entropy $H_P(\cdot)$ measures the compactness of a density $P(\cdot)$. It achieves a minimum of zero when all probability mass is assigned to a single value of $\mathbf{x}$; this agrees with an intuitive notion of a 'most informative' distribution. Conversely, when probability mass is uniformly distributed over states, the entropy in the distribution is a maximum; this too agrees with the idea of least informative (maximum entropy) distributions. Maximum entropy distributions are often used as prior distributions when no useful prior information is available. For example, if the random variable $\mathbf{x}$ can take on at most n discrete values

[4]Properly, information should be defined as the negative of entropy; when entropy is a minimum, information is a maximum. As is usual, we shall ignore this distinction and usually talk about entropy minimization when we really mean information maximisation.

in the set $\mathcal{X}$, then the least informative (maximum entropy) distribution on $\mathbf{x}$ is one which assigns a uniform probability $1/n$ to each value. This distribution will clearly have an entropy of $\log n$. When $\mathbf{x}$ is continuous-valued, the least informative distribution is also uniform. However, if the range of $\mathbf{x}$ is continuous then the distribution is technically not well defined as probability mass must be assigned equally over an infinite range. If the information is to be used as a prior in Bayes rule this technical issue is often not a problem as $P(\mathbf{x})$ can be set to any convenient constant value over the whole range of $\mathbf{x}$;[5] $P(\mathbf{x}) = 1, \forall \mathbf{x}$, for example, without affecting the values computed for the posterior $P(\mathbf{x} \mid \mathbf{z})$.

It can be shown that, up to a constant factor and under quite general conditions of preference ordering and preference boundedness, this definition of entropy is the *only* reasonable definition of 'informativenss'. An excellent proof of this quite remarkable result (first shown by Shannon) can be found in [14]. The implications of this in data fusion problems are many-fold. In particular it argues that entropy is a uniquely appropriate measure for evaluating and modeling information sources described by probabilistic models. Such ideas will be examined in later sections on system performance measures, organization and management.

2.3.2 Conditional Entropy

The basic idea of entropy can logically be extended to include conditional entropy; for continuous-valued random variables

$$H_P(\mathbf{x} \mid z_j) \stackrel{\triangle}{=} -\mathrm{E}\{\log P(\mathbf{x} \mid z_j)\} = -\int_{-\infty}^{\infty} P(\mathbf{x} \mid z_j) \log P(\mathbf{x} \mid z_j) \mathrm{d}\mathbf{x} \tag{35}$$

and for discrete random variables

$$H_P(\mathbf{x} \mid z_j) \stackrel{\triangle}{=} -\mathrm{E}\{\log P(\mathbf{x} \mid z_j)\} = -\sum_{\mathbf{x}} P(\mathbf{x} \mid z_j) \log P(\mathbf{x} \mid z_j). \tag{36}$$

This should be interpreted as the information (entropy) about the state $\mathbf{x}$ contained in the distribution $P(\cdot \mid \mathbf{z})$ following an observation z_j. Note that $H_P(\mathbf{x} \mid \mathbf{z})$ is still a function of $\mathbf{z}$, and thus depends on the observation made.

The mean conditional entropy, $\overline{H}(\mathbf{x} \mid \mathbf{z})$, taken over all possible values of $\mathbf{z}$, is given by

$$\begin{aligned} \overline{H}(\mathbf{x} \mid \mathbf{z}) &\stackrel{\triangle}{=} \mathrm{E}\{H(\mathbf{x} \mid \mathbf{z})\} \\ &= \int_{-\infty}^{+\infty} P(\mathbf{z}) H(\mathbf{x} \mid \mathbf{z}) \mathrm{d}\mathbf{z} \\ &= -\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} P(\mathbf{z}) P(\mathbf{x} \mid \mathbf{z}) \log P(\mathbf{x} \mid \mathbf{z}) \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z} \\ &= -\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} P(\mathbf{x}, \mathbf{z}) \log P(\mathbf{x} \mid \mathbf{z}) \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z}, \end{aligned} \tag{37}$$

[5] A distribution such as this which clearly violates the constraint that $\int_{-\infty}^{+\infty} P(\mathbf{x})\mathrm{d}\mathbf{x} = 1$ is termed an *improper* distribution, or in this case an improper prior.

for continuous random variables and

$$\begin{aligned}\overline{H}(\mathbf{x} \mid \mathbf{z}) &\stackrel{\triangle}{=} \sum_{\mathbf{z}} H(\mathbf{x} \mid \mathbf{z}) P(\mathbf{z}) \\ &= -\sum_{\mathbf{z}}\sum_{\mathbf{x}} P(\mathbf{x},\mathbf{z}) \log P(\mathbf{x} \mid \mathbf{z}). \end{aligned} \tag{38}$$

for discrete random variables. Note that $\overline{H}(\mathbf{x} \mid \mathbf{z})$ is not a function of either $\mathbf{x}$ or $\mathbf{z}$. It is a essentially a measure of the information that will be obtained (on the average) by making an observation before the value of the observation is known.

Example 9

Recall example 3 of two sensors observing a discrete state; type 1 target, type 2 target, and no target. Consider first sensor 1. Assuming a uniform prior distribution, the information (entropy) obtained about the state given that an observation $\mathbf{z} = z_1$ *has been made (first column of the likelihood matrix for sensor 1), is simply given by (using natural logs)*

$$H_{P_1}(\mathbf{x} \mid z_1) = -\sum_{i=1}^{3} P_1(x_i \mid z_1) \log P_1(x_i \mid z_1) = 0.9489. \tag{39}$$

For the first sensor, the conditional entropy for each possible observation is

$$H_{P_1}(\mathbf{x} \mid \mathbf{z}) = -\sum_{i=1}^{3} P_1(x_i \mid \mathbf{z}) \log P_1(x_i \mid \mathbf{z}) = (0.9489, 0.9489, 0.6390). \tag{40}$$

Thus, observing either z_1 *or* z_2 *is equally as informative (indeed it provides exactly the same information as sensor 1 can not distinguish between targets), but observing* z_3 *(no target) is* most *informative because probability mass is relatively concentrated on the no target state. Successive observation of the* z_1 *or* z_2 *with sensor 1, yields a posterior density on* $\mathbf{x}$ *of* $(0.5, 0.5, 0.0)$ *which has an information value (entropy) of* $\log(2) = 0.6931$. *Successive observation of* z_3 *yields a posterior of* $(0.0, 0.0, 1.0)$ *which has an entropy of zero (log*(1)*, the minimum entropy).*

Similar calculations for the likelihood matrix of sensor 2 give the conditional entropy for each observation as $H_{P_2}(\mathbf{x} \mid \mathbf{z}) = (0.948, 0.948, 0.948)$*; that is any observation is equally as informative. This is because, in the likelihood function for sensor 2, the relative distribution of probability mass is the same for any observation, even though the mass itself is placed on different states. Successive observation of any of* z_1*,* z_2*, or* z_3 *results in probability being assigned to only two of three states and thus for the posterior entropy to achieve a minimum of* $\log(2) = 0.6331$. *Note that, as with sensor 1 observing* z_1 *or* z_2*, entropy achieves a lower bound not equal to zero.*

To find the mean conditional entropy, the joint distribution $P(x, y)$ *is required. This* *is computed from*

$$\begin{aligned}
P(\mathbf{x},\mathbf{z}) &= P(\mathbf{z}\mid\mathbf{x})P(\mathbf{x}) \\
&= \begin{bmatrix} 0.45 & 0.45 & 0.1 \\ 0.45 & 0.45 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \otimes \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \\
&= \begin{bmatrix} 0.1500 & 0.1500 & 0.0333 \\ 0.1500 & 0.1500 & 0.0333 \\ 0.0333 & 0.0333 & 0.2667 \end{bmatrix},
\end{aligned}$$

(Note the sum of elements is equal to 1). Substituting these values into Equation 38 and summing over both $\mathbf{x}$ *and* $\mathbf{z}$ *gives the mean conditional entropy as* 0.8456.

Example 10

It is of considerable future interest to provide a measure for the entropy of a Gaussian (normal) distribution. The pdf for an n-dimensional Gaussian is given by:

$$P(\mathbf{x}) = \mathbf{N}(\overline{\mathbf{x}},\mathbf{P}) = \mid 2\pi\mathbf{P}\mid^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\overline{\mathbf{x}})^T\mathbf{P}^{-1}(\mathbf{x}-\overline{\mathbf{x}})\right], \tag{41}$$

where $\overline{\mathbf{x}}$ *is the mean of the distribution and* $\mathbf{P}$ *the covariance, and where* $\mid\cdot\mid$ *denotes matrix determinant. The entropy for this distribution is obtained as follows*

$$\begin{aligned}
H_P(\mathbf{x}) &= \mathrm{E}\{\log P(\mathbf{x})\} \\
&= -\frac{1}{2}\mathrm{E}\{(\mathbf{x}-\overline{\mathbf{x}})^T\mathbf{P}^{-1}(\mathbf{x}-\overline{\mathbf{x}}) + \log[(2\pi)^n\mid\mathbf{P}\mid]\} \\
&= -\frac{1}{2}\mathrm{E}\{\textstyle\sum_{ij}(\mathbf{x}_i-\overline{\mathbf{x}}_i)\mathbf{P}^{-1}_{ij}(\mathbf{x}_j-\overline{\mathbf{x}}_j)\} - \frac{1}{2}\log[(2\pi)^n\mid\mathbf{P}\mid] \\
&= -\frac{1}{2}\sum_{ij}\mathrm{E}\{(\mathbf{x}_j-\overline{\mathbf{x}}_j)(\mathbf{x}_i-\overline{\mathbf{x}}_i)\}\ \mathbf{P}^{-1}_{ij} - \frac{1}{2}\log[(2\pi)^n\mid\mathbf{P}\mid] \\
&= -\frac{1}{2}\sum_j\sum_i\mathbf{P}_{ji}\mathbf{P}^{-1}_{ij} - \frac{1}{2}\log[(2\pi)^n\mid\mathbf{P}\mid] \\
&= -\frac{1}{2}\sum_j(\mathbf{P}\mathbf{P}^{-1})_{jj} - \frac{1}{2}\log[(2\pi)^n\mid\mathbf{P}\mid] \\
&= -\frac{1}{2}\sum_j\mathbf{1}_{jj} - \frac{1}{2}\log[(2\pi)^n\mid\mathbf{P}\mid] \\
&= -\frac{n}{2} - \frac{1}{2}\log[(2\pi)^n\mid\mathbf{P}\mid] \\
&= -\frac{1}{2}\log[(2\pi e)^n\mid\mathbf{P}\mid].
\end{aligned} \tag{42}$$

Thus the entropy of a Gaussian distribution is defined only by the state vector length n and the covariance P. The entropy is proportional to the log of the determinant of the *covariance. The determinant of a matrix is a volume measure (recall that the determinant is the product of the eigenvalues of a matrix and the eigenvalues define axis lengths in n space). Consequently, the entropy is a measure of the volume enclosed by the covariance matrix and consequently the compactness of the probability distribution.*

If the Gaussian is scalar with variance σ^2, then the entropy is simply given by

$$H(\mathbf{x}) = \log \sigma\sqrt{2\pi e}$$

For a two random variables $\mathbf{x}$ and $\mathbf{z}$ which are jointly Gaussian, with correlation $\rho_{zx} = \frac{\sigma_{xz}}{\sqrt{\sigma_{xx}^2\sigma_{zz}^2}}$, the conditional entropy is given by

$$H(\mathbf{x} \mid \mathbf{z}) = \log \sigma_{xx}\sqrt{2\pi e(1-\rho_{xz}^2)}.$$

Thus when the variables are uncorrelated, $\rho_{xz} = 0$, the conditional entropy is just the entropy in $P(\mathbf{x})$, $H(\mathbf{x} \mid \mathbf{z}) = H(\mathbf{x})$, as $\mathbf{z}$ provides no additional information about $\mathbf{x}$. Conversely, when the variables are highly correlated $\rho_{xz} \to 1$, the conditional entropy goes to zero as complete knowledge of $\mathbf{z}$ implies complete knowledge of $\mathbf{x}$.

2.3.3 Mutual Information

With these definitions of entropy and conditional entropy, it is possible to write an 'information form' of Bayes theorem. Taking expectations of Equation 31 with respect to both the state $\mathbf{x}$ and the observation $\mathbf{z}$ gives (we will now drop the suffix P when the context of the distribution is obvious)

$$\overline{H}(\mathbf{x} \mid \mathbf{z}) = \overline{H}(\mathbf{z} \mid \mathbf{x}) + H(\mathbf{x}) - H(\mathbf{z}). \tag{43}$$

Simply, this describes the change in entropy or information following an observation from a sensor modeled by the likelihood $P(\mathbf{z} \mid \mathbf{x})$.

Being able to describe changes in entropy leads naturally to asking an important question; what is the most informative observation I can make? This question may be answered through the idea of mutual information.

The mutual information $I(\mathbf{x}, \mathbf{z})$ obtained about a random variable $\mathbf{x}$ with respect to a second random variable $\mathbf{z}$ is now defined as

$$\begin{aligned} I(\mathbf{x}, \mathbf{z}) &= -\mathrm{E}\{\log \frac{P(\mathbf{x}, \mathbf{z})}{P(\mathbf{x})P(\mathbf{z})}\} \\ &= -\mathrm{E}\{\log \frac{P(\mathbf{x} \mid \mathbf{z})}{P(\mathbf{x})}\} \\ &= -\mathrm{E}\{\log \frac{P(\mathbf{z} \mid \mathbf{x})}{P(\mathbf{z})}\} \qquad (44) \end{aligned}$$

Mutual information is an *a priori* measure of the information to be gained through observation. It is a function of the ratio of the density $P(\mathbf{x} \mid \mathbf{z})$ following an observation to the prior density $P(x)$. The expectation is taken over $\mathbf{z}$ and $\mathbf{x}$, so the mutual information gives an average measure of the gain to be expected *before* making the observation. If the underlying probability distributions are continuous then

$$I(\mathbf{x},\mathbf{z}) = -\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} P(\mathbf{x},\mathbf{z}) \log \frac{P(\mathbf{x},\mathbf{z})}{P(\mathbf{x})P(\mathbf{z})} \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z}, \tag{45}$$

and for discrete distributions

$$I(\mathbf{x},\mathbf{z}) = -\sum_{\mathbf{z}\in\mathcal{Z}}\sum_{\mathbf{x}\in\mathcal{X}} P(\mathbf{x},\mathbf{z}) \log \frac{P(\mathbf{x},\mathbf{z})}{P(\mathbf{x})P(\mathbf{z})}. \tag{46}$$

Noting that

$$\frac{P(\mathbf{x},\mathbf{z})}{P(\mathbf{x})P(\mathbf{z})} = \frac{P(\mathbf{x}\mid\mathbf{z})}{P(\mathbf{x})}, \tag{47}$$

then if $\mathbf{x}$ and $\mathbf{z}$ are independent, the expressions in Equation 47 become equal to one and (taking logs) the mutual information becomes equal to zero. This is logical; if knowledge of the state is independent of the observation, the information to be gained by taking an observation (the mutual information) is zero. Conversely, as $\mathbf{x}$ becomes more dependent on $\mathbf{z}$, then $P(\mathbf{x}\mid\mathbf{z})$ becomes more peaked or compact relative to the prior distribution $P(\mathbf{x})$ and so mutual information increases. Note that mutual information is always positive (it is not possible to lose information by taking observations).

Equation 44 can be written in terms of the component entropies as

$$\begin{aligned} I(\mathbf{x},\mathbf{z}) &= H(\mathbf{x}) + H(\mathbf{z}) - H(\mathbf{x},\mathbf{z}) \\ &= H(\mathbf{x}) - \overline{H}(\mathbf{x}\mid\mathbf{z}) \\ &= H(\mathbf{z}) - \overline{H}(\mathbf{z}\mid\mathbf{x}) \end{aligned} \tag{48}$$

Equation 48 measures the 'compression' of the probability mass caused by an observation. Mutual information provides an average measure of how much more information we would have about the random variable $\mathbf{x}$ if the value of $\mathbf{z}$ where known. Most importantly mutual information provides a *pre-experimental* measure of the usefulness of obtaining information (through observation) about the value of $\mathbf{z}$.

Example 11

Continuing the two sensors target identification example (Example 9), it is interesting to see what value mutual information has in deciding which sensor should be used for correctly identifying a target (this is a sensor management problem).

First assume that the prior information on $\mathbf{x}$ *is uniform so that*

$$P(\mathbf{x}) = [1/3, 1/3, 1/3]^T.$$

Consider taking an observation with sensor 1. The total probability of observation $P(\mathbf{z})$ *can be found by summing the joint probability* $P(\mathbf{x},\mathbf{z})$ *(computed in Example 9) to obtain*

$P(\mathbf{z}) = [1/3, 1/3, 1/3]$. *The mutual information,* $I(\mathbf{x}, \mathbf{z})$*, on using sensor 1 to take an observation is then*

$$\begin{array}{rcl} I(\mathbf{x}, \mathbf{z}) & = & -\sum_j P(z_j) \log P(z_j) - \left(-\sum_j \sum_i P(x_i, z_j) \log P(z_j \mid x_i)\right) \\ & = & 1.0986 - 0.8456 = 0.2530. \end{array}$$

Suppose we now go ahead and take an observation with sensor 1 and the result is z_1*, an observation of target 1. The prior is now updated as (Example 2)*

$$P(\mathbf{x}) = P_1(\mathbf{x} \mid z_1) = [0.45, 0.45, 0.1].$$

The mutual information gain from using sensor 1 given this new prior is $I(\mathbf{x}, \mathbf{z}) = 0.1133$*. Note, that this is smaller than the mutual information obtained with a uniform prior. This is because we now have observation information and so the value of obtaining more information is less. Indeed, mutual information shows that the more we observe, the less value there is in taking a new observation. In the limit, if we repeatedly observe* z_1 *with sensor 1, then the prior becomes equal to* $[0.5, 0.5, 0.0]$ *(Example 2). In this case the predicted mutual information gain is zero. That is, there is no value in using sensor 1, yet again, to obtain information.*

Similarly, again assuming a uniform prior, the mutual information gain predicted for using sensor 2 is 0.1497*.This is lower than sensor 1, when the no-target density is more highly peaked. If we observe target 1 with sensor 2 then the updated prior becomes* $P(x) = [0.45, 0.1, 0.45]$*. Using this prior, the mutual information gain for using sensor 2 is only* 0.1352*; smaller, but not as small as the corresponding mutual information gain from sensor 1 after the first measurement. Continual observation of target 1 with sensor 2 results in the posterior becoming equal to* $[0.5, 0.0, 0.5]$ *and if, this is substituted into the equation for mutual information gain, we obtain a predicted mutual information gain of* 0.1205*; even after taking an infinite number of measurements. The reason for this is subtle; as sensor 2 can not distinguish targets, any observation other than that of target 1 will still provide additional information and as mutual information is an average measure, it is still predicts, on the average, an increase in information (even though further observation of target 1 will not provide any more information). Continual observation of target 1 with sensor 2 would never happen in practice. As the likelihood suggests, if the true state were target 1, then sensor 2 would be as likely to not to detect a target at all as report a target 1. Indeed, if we mix detections as the likelihood suggests, the posterior tends to* $[1, 0, 0]$ *or* $[0, 1, 0]$ *which then suggests zero mutual information gain from sensor 2.*

Now, suppose sensor 1 is used to obtain the first measurement and target 1 is detected so that the new prior is $[0.45, 0.45, 0.1]$*. Which of sensor 1 or 2 should be used to make the second observation ? With this prior, the predicted mutual information gain from using sensor 1 is* 0.1133*, and from using sensor 2 is* 0.1352*. This (logically) tells us to use sensor 2. Now sensor 2 is equally as likely to detect the correct target or return a no-detect. Sensor 2 is now employed and a no-detect is returned yielding a posterior* $[0.4880, 0.4880, 0.0241]$*. With this as a prior, mutual information will tell us to continue* *with sensor 2 until we get a return for either target 1 or 2. If this (target 2, say) happens next time round, then the posterior will be* $[0.1748, 0.7864, 0.0388]$*; we now have, for the first time a target preference (2) and sensor 1 and 2 can both be used to refine this estimate. With this prior, the mutual information for sensor 1 is* 0.0491 *and for sensor 2 is* 0.0851*; thus sensor 2 will be used on the fourth sensing action.*

This process of predicting information gain, making a decision on sensing action, then sensing, is an efficient and effective way of managing sensing resources and of determining optimal sensing policies. However, it should be noted that this is an average *policy. As we saw with the behaviour of sensor 2, it is sometimes not the most logical policy if we value things in a manner other than on the average. This issue will be dealt with in more detail in the section on decision making.*

Example 12

A second interesting application of mutual information gain is Example 5. In this example, general probability distributions, defining target location, are defined on a location grid. Entropy measures can be obtained for distributions on this grid from

$$\begin{aligned} H(\mathbf{x},\mathbf{y}) &= -\mathrm{E}\{\log P(\mathbf{x},\mathbf{y})\} \\ &= \int\int P(\mathbf{x},\mathbf{y})\log P(\mathbf{x},\mathbf{y})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \\ &\approx \sum\sum P(x_i,y_j)\log P(x_i,y_j)\delta x_i \delta y_j \end{aligned} \tag{49}$$

The total entropy associated with the prior shown in Figure 2(a), computed with Equation 49 is $H(\mathbf{x}) = 12.80$*. The total entropy associated with the sensor 1 likelihood shown in Figure 2(b) is* $\overline{H}(\mathbf{z} \mid \mathbf{x}) = 4.2578$*, and the entropy associated with the resulting posterior of Figure 2(c) is* $\overline{H}(\mathbf{x} \mid \mathbf{x}) = 3.8961$*. The predicted mutual information gain from this observation is therefore* $I(\mathbf{x},\mathbf{z}) = 8.9039$ *(a big gain). However, the predicted mutual information gain from a second observation using sensor 1 is only* 0.5225*. This is reflected in the slight change in posterior from Figure 2(c) to Figure 2(d). Conversely, the predicted mutual information gain from sensor 2, with likelihood shown in Figure 2(e), is* 2.8598 *(a relatively large gain). This is reflected in the posterior shown in Figure 2(f) which has an entropy of only* 0.5138*.*

This example shows that it is straight-forward to apply principles of entropy to any probability type of probabilistic information; to measure compactness of information and the prediction of information gain using mutual information.

2.3.4 Fisher Information

A second measure of information commonly used in probabilistic modeling and estimation is the Fisher information. Unlike Shannon information, Fisher information may only be defined on continuous distributions. The Fisher information $J(\mathbf{x})$ is defined as the second derivative of the log-likelihood[6]

$$J(\mathbf{x}) = \frac{\mathrm{d}^2}{\mathrm{d}\mathbf{x}^2} \log P(\mathbf{x}). \tag{50}$$

In general, if $\mathbf{x}$ is a vector, then $J(\mathbf{x})$ will be a matrix, usually called the Fisher Information Matrix. The Fisher information describes the information content about the values of $\mathbf{x}$ contained in the distribution $P(\mathbf{x})$. The Fisher information measures the surface of a bounding region containing probability mass. Thus, like entropy, it measures compactness of a density function. However, entropy measures a volume and is thus a single number, whereas Fisher information is a series of numbers (and generally a matrix) measuring the axes of the bounding surface.

The Fisher information is particularly useful in the estimation of continuous valued quantities. If $P(\mathbf{x})$ describes all the (probabilistic) information we have about the quantity $\mathbf{x}$, then the smallest variance that we can have on an estimate of the true value of $\mathbf{x}$ is known as the Cramer-Rao lower bound, and is equal to the Fisher information $J(\mathbf{x})$. An estimator that achieves this lower bound is termed 'efficient'.

Example 13

The simplest example of Fisher information is the information associated with a vector $\mathbf{x}$ *known to be Gaussian distributed with mean* $\overline{\mathbf{x}}$ *and covariance* $\mathbf{P}$*;*

$$P(\mathbf{x}) = N(\mathbf{x}; \overline{\mathbf{x}}; \mathbf{P}) = \frac{1}{(2\pi)^{n/2}|P|} \exp\left(-\frac{1}{2}(\mathbf{x}-\overline{\mathbf{x}})\mathbf{P}^{-1}(\mathbf{x}-\overline{\mathbf{x}})^T\right) \tag{51}$$

Taking logs of this distribution, and differentiating twice with respect to $\mathbf{x}$ *gives* $\mathbf{J}(\mathbf{x}) = \mathbf{P}^{-1}$*. That is, the Fisher information is simply the inverse covariance. This agrees with intuition; Gaussian distributions are often drawn as a series of ellipsoids containing probability mass.* $\mathbf{P}^{-1}$ *describes the surface of this ellipsoid, the square root of it's eigenvalues being the dimensions of each axis of the ellipsoid.*

The Fisher information plays an important role in multi-sensor estimation problems. In conventional estimation problems when only a single source of information is used to obtain information about an unknown state, it is common to talk of an estimate of this state together with some associated uncertainty (normally a variance). However, in multi-sensor estimation problems it is difficult to describe any (statistical) relations there may be between the different estimates produced by different combinations of sensors. This problem can only be overcome by dealing directly with the likelihood functions associated with the observations themselves and by explicitly accounting for any dependencies between different estimates. The Fisher information provides a direct means of accounting for these dependencies as it makes explicit the information available in the likelihood function. We will return to this again in the Chapter on multisensor estimation.

[6]The first derivative of the log-likelihood is called the score function

2.3.5 The Relation between Shannon and Fisher Measures

The question arises that if entropy is considered to be the only reasonable measure of information content in a distribution, why consider Fisher information at all as it must, by definition, be an 'unreasonable' measure of information. Fortunately, there is a sensible explanation for this problem, in that for continuous variables, Fisher information and Shannon information are indeed related by the log-likelihood function. Broadly, entropy is related to the volume of a set (formally a 'typical set') containing a specified probability mass. Fisher information is related in the surface area of this typical set. Thus maximization of Fisher information is equivalent to minimization of entropy. A detailed development of this relation using the Asymptotic Equipartion Property (AEP) is given in Cover [16].

Example 14

If the pdf associated with a random variable $\mathbf{x}$ *is known to be Gaussian distributed with mean* $\overline{\mathbf{x}}$ *and covariance* $\mathbf{P}$ *(as Equation 51), then an explicit relation for the relation between Fisher and Shannon information can be obtained, and indeed is given in Equation 42 (Example 10) as;*

$$H(\mathbf{x}) = -\frac{1}{2}\log[(2\pi e)^n \mid \mathbf{P} \mid]$$

This clearly shows the relation between the Fisher surface measure of information $\mathbf{P}^{-1}$ *and the entropic volume measure through the determinant of* $\mathbf{P}$.

2.4 Alternatives to Probability

The representation of uncertainty is so important to the problem of information fusion that a number of alternative modeling techniques have been proposed to deal with perceived limitations in probabilistic methods[7].

There are four main perceived limitations of probabilistic modeling techniques:

1. Complexity: the need to specify a large number of probabilities to be able to apply probabilistic reasoning methods correctly.

2. Inconsistency: the difficulties involved in specifying a consistent set of beliefs in terms of probability and using these to obtain consistent deductions about states of interest.

3. Precision of Models: the need to be precise in the specification of probabilities for quantities about which little is known.

4. Uncertainty about uncertainty: the difficulty in assigning probability in the face of uncertainty, or ignorance about the source of information.

[7]In the literature on the subject, there appears to be no middle ground; you are either a religious Bayesian or a rabid non-conformist.

There are four main techniques put forward to address these issues; interval calculus, fuzzy logic, and the theory of evidence (Dempster-Shafer methods). We briefly discuss each of these in turn.

2.4.1 Interval Calculus

The representation of uncertainty using an interval to bound true parameter values has a number of potential advantages over probabilistic techniques. In particular, intervals provide a good measure of uncertainty in situations where there is a lack of probabilistic information, but in which sensor and parameter error is known to be bounded. In interval techniques, the uncertainty in a parameter x is simply described by a statement that the true value of the state x is known to be bounded from below by a, and from above by b; $x \in [a, b]$. It is important that no other additional probabilistic structure is implied, in particular the statement $x \in [a, b]$ does not necessarily imply that x is equally probable (uniformly distributed) over the interval $[a, b]$.

There are a number of simple and basic rules for the manipulation of interval errors. These are described in detail in the book by Moore [30] (whose analysis was originally aimed at understanding limited precision computer arithmetic). Briefly, with $a, b, c, d \in \Re$, addition, subtraction, multiplication and division are defined by the following algebraic relations;

$$[a, b] + [c, d] = [a + c, b + d] \tag{52}$$

$$[a, b] - [c, d] = [a - d], b - c] \tag{53}$$

$$[a, b] \times [c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)] \tag{54}$$

$$[a, b]/[c, d] = [a, b] \times [1/d, 1/c], \qquad 0 \notin [c, d]. \tag{55}$$

Interval addition and multiplication are both associative and commutative; with intervals $A, B, C \subset \Re$,

$$A + (B + C) = (A + B) + C, \quad A \times (B \times C) = (A \times B) \times C, \tag{56}$$

$$A + B = B + A, \qquad A \times B = B \times A. \tag{57}$$

The distributive law does not always hold, however a weaker law of *subdistributivity* can be applied on the basis that an interval is simply a set of real numbers;

$$A \times (B + C) \subseteq A \times B + A \times C, \tag{58}$$

with equality when $A = [a, a] = a$ (a single number), or when $B \times C > 0$ (the intervals B and C contain numbers of the same sign.

Interval arithmetic admits an obvious metric distance measure;

$$d([a, b], [c, d]) = \max(|a - c|, |b - d|) \tag{59}$$

It follows that

$$d(A, B) = d(B, A), \qquad d(A, B) = 0 \text{ iff } A = B, \qquad d(A, B) + d(B, C) \geq d(A, C) \tag{60}$$

Matrix arithmetic using intervals is also possible, but substantially more complex, particularly when matrix inversion is required.

Interval calculus methods are sometimes used for detection. However, they are not generally used in data fusion problems as: i) it is difficult to get results that converge to anything of value (it is too pessimistic) and; ii) it is hard to encode dependencies between variables which are at the core of many data fusion problems.

2.4.2 Fuzzy Logic

Fuzzy logic has found wide-spread popularity as a method for representing uncertainty particularly in applications such as supervisory control and high-level data fusion tasks. It is often claimed that fuzzy logic provides an ideal tool for inexact reasoning, particularly in rule-based systems. Certainly, fuzzy logic has had some notable success in practical application. However, the jury is still out on weather fuzzy logic offers more or less than conventional probabilistic methods.

A great deal has been written about fuzzy sets and fuzzy logic (see for example [17] and the discussion in [9] Chapter 11). Here we briefly describe the main definitions and operations without any attempt to consider the more advanced features of fuzzy logic methods.

Consider a Universal set consisting of the elements x; $\mathcal{X} = \{x\}$. Consider a proper subset $\mathcal{A} \subseteq \mathcal{X}$ such that

$$\mathcal{A} = \{x \mid x \quad \text{has some specific property}\} .$$

In conventional logic systems, we can define a membership function $\mu_A(x)$ which reports if a specific element $x \in \mathcal{X}$ is a member of this set:

$$\mathcal{A} \rightleftharpoons \mu_A(x) = \{ \begin{matrix} 1 & \text{if} & x \in A \\ 0 & \text{if} & x \notin A \end{matrix}$$

For example $\mathcal{X}$ may be the set of all aircraft. The set $\mathcal{A}$ may be the set of all supersonic aircraft. In the fuzzy logic literature, this is known as a "crisp" set.

In contrast, a fuzzy set is one in which there is a *degree of membership*, ranging between 0 and 1. A fuzzy membership function $\mu_A(x)$ then defines the degree of membership of an element $x \in \mathcal{X}$ in the set $\mathcal{A}$. For example, if $\mathcal{X}$ is again the set of all aircraft, $\mathcal{A}$ may be the set of all "fast" aircraft. Then the fuzzy membership function $\mu_A(x)$ assigns a value between 0 and 1 indicating the degree of membership of every aircraft x to this set. Formally

$$\mathcal{A} \rightleftharpoons \mu_A \mapsto [0, 1].$$

Figure 10 shows an instance of this fuzzy membership function. Clearly, despite the "fuzzy" nature of the set, it must still be quantified in some form.

Composition rules for fuzzy sets follow the composition processes for normal crisp sets, specifically

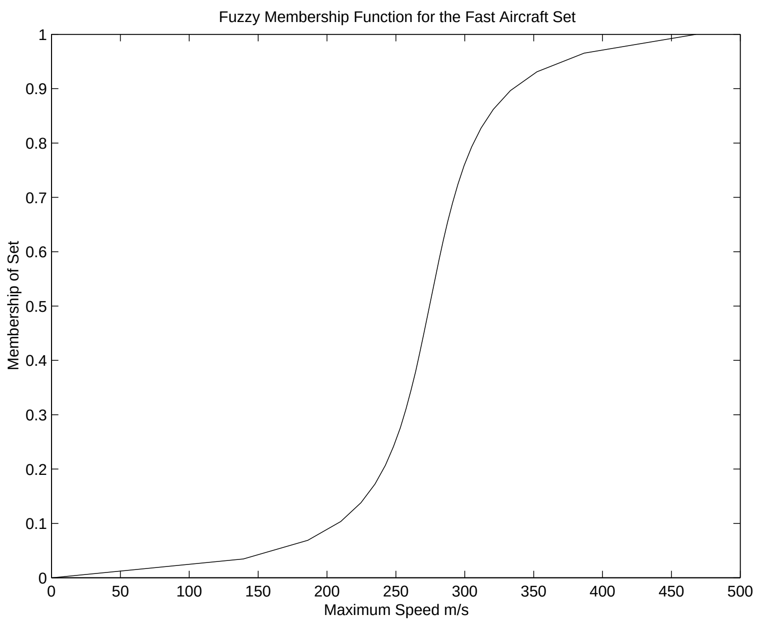


Figure 10: Example fuzzy membership function

AND is implemented as a minimum:

$$\mathcal{A} \cap \mathcal{B} \rightleftharpoons \mu_{A \cap B}(x) = \mathbf{min}\left[\mu_A(x), \mu_B(x)\right]$$

OR is implemented as a maximum:

$$\mathcal{A} \cup \mathcal{B} \rightleftharpoons \mu_{A \cup B}(x) = \mathbf{max}\left[\mu_A(x), \mu_B(x)\right]$$

NOT is implemented as a compliment:

$$\overline{\mathcal{B}} \rightleftharpoons \mu_{\overline{B}}(x) = 1 - \mu_B(x)$$

The normal properties associated with binary logic now hold; commutativity, associativity, idempotence, distributivity De Morgan's law and absorption. The only exception is that the law of the excluded middle is no longer true

$$\mathcal{A} \cup \overline{\mathcal{A}} \neq \mathcal{X}, \qquad \mathcal{A} \cap \overline{\mathcal{A}} \neq \emptyset$$

Together these definitions and laws provide a systematic means of reasoning about inexact values.

The relationship between fuzzy set theory and probability is still hotly debated. It should however be noted that the minimum and maximum operations above are highly suggestive of the results provided by probability theory when the two sets are either wholly dependent or wholly independent.

2.4.3 Evidential Reasoning

Evidential reasoning (often called the Dempster-Shafer theory of evidence after the originators of these ideas) has seen intermittent success particularly in automated reasoning applications.

Evidential reasoning is qualitatively different from either probabilistic methods or fuzzy set theory in the following sense: Consider a universal set $\mathcal{X}$. In probability theory or fuzzy set theory, a belief mass may be placed on any element $x_i \in \mathcal{X}$ and indeed on any subset $\mathcal{A} \subseteq \mathcal{X}$. In evidential reasoning, belief mass can not only be placed on elements and sets, but also sets of sets. Specifically, while the domain of probabilistic methods is all possible subsets, the domain of evidential reasoning is the domain of all sets of all subsets.

Example 15

Consider the set $\mathcal{X} = \{rain,\ no\ rain\}$. The set is complete and mutually exclusive (at least about the event rain). In probability theory, by way of a forecast, we might assign a probability to each possible event. For example, $P(rain) = 0.3$, and thus $P(no\ rain) = 0.7$.

In evidential reasoning, we construct the set of all sets, otherwise called the power set as

$$2^{\mathcal{X}} = \{\{rain, no\ rain\}\ , \{rain\}\ , \{no\ rain\}\ , \{\quad\}\ \}\ ,$$

and belief mass is assigned to all elements of this set as

$$\begin{aligned} m(\{rain, no\ rain\}\ ) &= 0.5 \\ m(\{rain\}\ ) &= 0.3 \\ m(\{no\ rain\}\ ) &= 0.2 \\ m(\{\quad\}\ ) &= 0.0 \end{aligned}$$

(traditionally, the empty set is assigned a belief mass of zero for normalisation purposes). The interpretation of this is that there is a 30% chance of rain, a 20% chance of no rain and a 50% chance of either rain or no rain. In effect, the measure placed on the set containing both rain and no rain, is a measure of ignorance or inability to distinguish between the two alternatives.

Evidential reasoning thus provides a method of capturing ignorance or an inability to distinguish between alternatives. In probability theory, this would be dealt with in a very different manner by assigning an equal or uniform probability to each alternative. Yet, stating that there is a 50% chance of rain is clearly *not* the same as saying that it is unknown if it will rain or not.

The use of the power set as the "frame of discernment" allows a far richer representation of beliefs. However, this comes at the cost of a substantial increase in complexity. If there are n elements in the original set $\mathcal{X}$, then there will be 2^n (hence the definition of the power set) possible sets of subsets on which a belief mass will be assigned. For large

n, this is clearly intractable. Further, when the set is continuous, the set of all subsets is not even measurable.

Given a frame of discernment, two mass assignments, $m(\mathcal{A})$ and $m(\mathcal{B})$, can be combined to yield a third mass assignment, $m(\mathcal{C}) = m(\mathcal{C} \mid \mathcal{A}, \mathcal{B})$ using Dempster's rule of combination

$$m(\mathcal{C} \mid \mathcal{A}, \mathcal{B}) = \frac{1}{1-\kappa} \sum_{i,j \mid \mathcal{A}_i \cap \mathcal{B}_j = \mathcal{C}} m(\mathcal{A}) m(\mathcal{B}) \tag{61}$$

where κ is a measure of inconsistency between the two mass assignments $m(\mathcal{A})$ and $m(\mathcal{B})$ defined as the mass product from the sets $\mathcal{A}$ and $\mathcal{B}$ which have zero intersection

$$\kappa = \sum_{i,j \mid \mathcal{A}_i \cap \mathcal{B}_j = \emptyset} m(\mathcal{A}) m(\mathcal{B}) \tag{62}$$

Note that the magnitude of κ is a measure of the degree to which the two mass distributions are in conflict. Since mass distribution can 'move freely' between singletons of the set, two sets are not conflicting unless they have no common elements.

In addition to basic mass assignments, evidential reasoning introduces the concepts of support $\mathrm{Spt}(\mathcal{C})$ and plausibility $\mathrm{Pls}(\mathcal{C})$ of any proposition $\mathcal{C} \subseteq \mathcal{X}$ as

$$\begin{aligned}
\mathrm{Spt}(\mathcal{C} \mid \mathcal{A}, \mathcal{B}) &= \frac{1}{1-\kappa} \sum_{i,j \mid \mathcal{A}_i \cap \mathcal{B}_j \subseteq \mathcal{C}} m(\mathcal{A}_i) m(\mathcal{B}_j) \\
&= \sum_{\mathcal{D} \mid \mathcal{D} \subseteq \mathcal{C}} m(\mathcal{D} \mid \mathcal{A}, \mathcal{B})
\end{aligned} \tag{63}$$

and

$$\begin{aligned}
\mathrm{Pls}(\mathcal{C} \mid \mathcal{A}, \mathcal{B}) &= \frac{1}{1-\kappa} \sum_{i,j \mid (\mathcal{A}_i \cap \mathcal{B}_j) \cap \mathcal{C} \neq \emptyset} m(\mathcal{A}_i) m(\mathcal{B}_j) \\
&= \sum_{\mathcal{D} \mid \mathcal{D} \cap \mathcal{C} \neq \emptyset} m(\mathcal{D} \mid \mathcal{A}, \mathcal{B}) \\
&= 1 - \mathrm{Spt}(\overline{\mathcal{C}} \mid \mathcal{A}, \mathcal{B})
\end{aligned} \tag{64}$$

The support for a proposition $\mathcal{C}$ is the evidence directly assigned to the proposition or any subset of the proposition $\mathcal{C}$ (that is, any proposition which implies $\mathcal{C}$). The plausibility of a proposition $\mathcal{C}$ is the sum of all the mass assigned to propositions which have a non-null intersection with $\mathcal{C}$ (that is, all propositions which do not contradict $\mathcal{C}$).

Example 16

(After [9], p511). Spt(EFA) is the sum of masses assigned directly to the proposition that an aircraft is any variant of the EFA. In contrast, the sum for Pls(EFA) would include mass assigned to any proposition that does not contradict the EFA. The plausibility sum would include the mass propositions such as aircraft, fixed wing aircraft, fighter or friend. The plausibility sum would not include mass of propositions such as foe bomber or helicopter.

There is no doubt that evidential reasoning will find a role in advanced data fusion systems, particularly in areas such as attribute fusion, and situation assessment. However, it should be noted that the role played by probability theory in this is still generally unresolved (see also [11, 13, 43]).

3 Multi-Sensor Estimation

Estimation is the single most important problem in sensor data fusion. Fundamentally, an estimator is a decision rule which takes as an argument a sequence of observations and whose action is to compute a value for the parameter or state of interest. Almost all data fusion problems involve this estimation process: we obtain a number of observations from a group of sensors and using this information we wish to find some *estimate* of the true state of the environment we are observing. Estimation encompasses all important aspects of the data fusion problem. Sensor models are required to understand what information is provided, environment models are required to relate observations made to the parameters and states to be estimated, and some concept of information value is needed to judge the performance of the estimator. Defining and solving an estimation problem is almost always the key to a successful data fusion system.

This section begins with a brief summary of the Kalman filter algorithm. The intention is to introduce notation and key data fusion concepts; prior familiarity with the basic Kalman Filter algorithm is assumed (see either [18] or the numerous excellent books on Kalman filtering [12, 7, 28]). The multi-sensor Kalman filter is then discussed. Three main algorithms are considered; the group-sensor method, the sequential sensor method and the inverse covariance form. The track-to-track fusion algorithm is also described. The problem of multiple-target tracking and data association are described. The three most important algorithms for data association are introduced. Finally, alternative estimation methods are discussed. In particular, maximum likelihood filters and various probability-distribution oriented methods. Subsequent sections consider the distributed Kalman filter and different data fusion architectures.

3.1 The Kalman Filter

The Kalman Filter is a recursive linear estimator which successively calculates an estimate for a continuous valued state, that evolves over time, on the basis of periodic observations that of this state. The Kalman Filter employs an explicit statistical model of how the parameter of interest $\mathbf{x}(t)$ evolves over time and an explicit statistical model of how the observations $\mathbf{z}(t)$ that are made are related to this parameter. The gains employed in a Kalman Filter are chosen to ensure that, with certain assumptions about the observation and process models used, the resulting estimate $\hat{\mathbf{x}}(t)$ minimises mean-squared error

$$L(t) = \int_{-\infty}^{\infty} \left(\mathbf{x}(t) - \hat{\mathbf{x}}(t)\right)^T \left(\mathbf{x}(t) - \hat{\mathbf{x}}(t)\right) P(\mathbf{x}(t) \mid \mathbf{Z}^t)\mathrm{d}\mathbf{x}. \tag{65}$$

Differentiation of Equation 65 with respect to $\mathbf{x}(t)$ and setting equal to zero gives

$$\hat{\mathbf{x}}(t) = \int_{-\infty}^{\infty} \mathbf{x}(t) P(\mathbf{x}(t) \mid \mathbf{Z}^t)\mathrm{d}\mathbf{x}, \tag{66}$$

which is simply the conditional mean $\hat{\mathbf{x}}(t) = \mathrm{E}\{\mathbf{x}(t) \mid \mathbf{Z}^t\}$. The Kalman filter, and indeed any mean-squared-error estimator, computes an estimate which is the conditional mean; an average, rather than a most likely value.

The Kalman filter has a number of features which make it ideally suited to dealing with complex multi-sensor estimation and data fusion problems. In particular, the explicit description of process and observations allows a wide variety of different sensor models to be incorporated within the basic algorithm. In addition, the consistent use statistical measures of uncertainty makes it possible to quantitatively evaluate the role each sensor places in overall system performance. Further, the linear recursive nature of the algorithm ensure that its application is simple and efficient. For these reasons, the Kalman filter has found wide-spread application in many different data fusion problems [38, 5, 7, 28].

3.1.1 State and Sensor Models

The starting point for the Kalman filter algorithm is to define a model for the states to be estimated in the standard state-space form;

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{G}(t)\mathbf{v}(t), \tag{67}$$

where

$\mathbf{x}(t) \in \Re^n$ is the state vector of interest,

$\mathbf{u}(t) \in \Re^s$ is a known control input,

$\mathbf{v}(t) \in \Re^q$ is a random variable describing uncertainty in the evolution of the state,

$\mathbf{F}(t)$ is the $n \times n$ state (model) matrix,

$\mathbf{B}(t)$ is the $n \times s$ input matrix, and

$\mathbf{G}(t)$ is the $n \times q$ noise matrix.

An observation (output) model is also defined in standard state-space form;

$$\mathbf{z}(t) = \mathbf{H}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{w}(t), \tag{68}$$

where

$\mathbf{z}(t) \in \Re^m$ is the observation vector,

$\mathbf{w}(t) \in \Re^r$ is a random variable describing uncertainty in the observation,

$\mathbf{H}(t)$ is the $m \times n$ observation (model) matrix,

$\mathbf{D}(t)$ is the $m \times r$ observation noise matrix.

These equations define the evolution of a continuous-time system with continuous observations being made of the state. However, the Kalman filter is almost always implemented in discrete-time. It is straight-forward to obtain a discrete-time version of Equations 67 and 68.

First, a discrete-time set $\mathbf{t} = \{t_0, t_1, \cdots t_k, \cdots\}$ is defined. Equation 68 can be written in discrete time as

$$\mathbf{z}(t_k) = \mathbf{H}(t_k)\mathbf{x}(t_k) + \mathbf{D}(t_k)\mathbf{w}(t_k), \qquad \forall t_k \in \mathbf{t} \tag{69}$$

where $\mathbf{z}(t_k)$, $\mathbf{x}(t_k)$ and $\mathbf{w}(t_k)$ are the discrete-time observation, state and noise vectors respectively, and $\mathbf{H}(t_k)$ and $\mathbf{D}(t_k)$ the observation and noise models evaluated at the discrete time instant t_k. The discrete-time form of the state equation requires integration of Equation 67 over the interval (t_k, t_{k-1}) as

$$\mathbf{x}(t_k) = \mathbf{\Phi}(t_k, t_{k-1})\mathbf{x}(t_{k-1}) + \int_{t_{k-1}}^{t_k} \mathbf{\Phi}(t_k, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)\mathrm{d}\tau + \int_{t_{k-1}}^{t_k} \mathbf{\Phi}(t_k, \tau)\mathbf{G}(\tau)\mathbf{v}(\tau)\mathrm{d}\tau. \tag{70}$$

where $\mathbf{\Phi}(\cdot, \cdot)$ is the state transition matrix satisfying the matrix differential equation

$$\dot{\mathbf{\Phi}}(t_k, t_{k-1}) = \mathbf{F}(t_k)\mathbf{\Phi}(t_k, t_{k-1}), \qquad \mathbf{\Phi}(t_{k-1}, t_{k-1}) = \mathbf{1}. \tag{71}$$

The state transition matrix has three important properties that should be noted:

1. It is uniquely defined for all t, t_0 in $[0, \infty]$.
2. (The semi-group property) $\mathbf{\Phi}(t_3, t_1) = \mathbf{\Phi}(t_3, t_2)\mathbf{\Phi}(t_2, t_1)$.
3. $\mathbf{\Phi}(t_k, t_{k-1})$ is non singular and $\mathbf{\Phi}^{-1}(t_k, t_{k-1})$=$\mathbf{\Phi}(t_{k-1}, t_k)$.

When $\mathbf{F}(t) = \mathbf{F}$ is a constant matrix, the state transition matrix is given by

$$\mathbf{\Phi}(t_k, t_{k-1}) = \mathbf{\Phi}(t_k - t_{k-1}) = \exp \mathbf{F}(t_k - t_{k-1}). \tag{72}$$

which clearly satisfies these three properties.

If $\mathbf{u}(t) = \mathbf{u}(t_k)$ and $\mathbf{v}(t) = \mathbf{v}(t_k)$ remain approximately constant over the interval (t_{k-1}, t_k) then the following discrete-time models can be defined;

$$\begin{aligned} \mathbf{F}(t_k) &\stackrel{\triangle}{=} \mathbf{\Phi}(t_k, t_{k-1}) \\ \mathbf{B}(t_k) &\stackrel{\triangle}{=} \textstyle\int_{t_{k-1}}^{t_k} \mathbf{\Phi}(t_k, \tau)\mathbf{B}(\tau)\mathrm{d}\tau \\ \mathbf{G}(t_k) &\stackrel{\triangle}{=} \textstyle\int_{t_{k-1}}^{t_k} \mathbf{\Phi}(t_k, \tau)\mathbf{G}(\tau)\mathrm{d}\tau. \end{aligned} \tag{73}$$

Equation 70 can then be written in a discrete-time from equivalent to Equation 67 as

$$\mathbf{x}(t_k) = \mathbf{F}(t_k)\mathbf{x}(t_{k-1}) + \mathbf{B}(t_k)\mathbf{u}(t_k) + \mathbf{G}(t_k)\mathbf{v}(t_k). \tag{74}$$

The accuracy of this model could be improved by taking mean values for both $\mathbf{u}(t)$ and $\mathbf{v}(t)$ over the sampling interval.

In almost all cases the time interval $\Delta t(k) \stackrel{\triangle}{=} t_k - t_{k-1}$ between successive samples of the state remains constant. In this case it is common particle to drop the time argument and simply index variables by the sample number. In this case Equation 74 is written as

$$\mathbf{x}(k) = \mathbf{F}(k)\mathbf{x}(k-1) + \mathbf{B}(k)\mathbf{u}(k) + \mathbf{G}(k)\mathbf{v}(k), \tag{75}$$

and Equation 69 as

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{D}(k)\mathbf{w}(k) \tag{76}$$

Equations 75 and 76 are the model forms that will be used throughout this section unless discussion of asynchronous data is relevant.

A basic assumption in the derivation of the Kalman filter is that the random sequences $\mathbf{v}(k)$ and $\mathbf{w}(k)$ describing process and observation noise are all Gaussian, temporally uncorrelated and zero-mean

$$\mathrm{E}\{\mathbf{v}(k)\} = \mathrm{E}\{\mathbf{w}(k)\} = \mathbf{0}, \qquad \forall k, \tag{77}$$

with known covariance

$$\mathrm{E}\{\mathbf{v}(i)\mathbf{v}^T(j)\} = \delta_{ij}\mathbf{Q}(i), \quad \mathrm{E}\{\mathbf{w}(i)\mathbf{w}^T(j)\} = \delta_{ij}\mathbf{R}(i). \tag{78}$$

It is also generally assumed that the process and observation noises are also uncorrelated

$$\mathrm{E}\{\mathbf{v}(i)\mathbf{w}^T(j)\} = \mathbf{0}, \qquad \forall i, j. \tag{79}$$

These are effectively equivalent to a Markov property requiring observations and successive states to be conditionally independent. If the sequences $\mathbf{v}(k)$ and $\mathbf{w}(k)$ are temporally correlated, a shaping filter can be used to 'whiten' the observations; again making the assumptions required for the Kalman filter valid [28]. If the process and observation noise sequences are correlated, then this correlation can also be accounted for in the Kalman filter algorithm [2]. If the sequence is not Gaussian, but is symmetric with finite moments, then the Kalman filter will still produce good estimates. If however, the sequence has a distribution which is skewed or otherwise 'pathological', results produced by the Kalman filter will be misleading and there will be a good case for using a more sophisticated Bayesian filter [40]. Problems in which the process and observation models are non-linear are dealt in Section 3.1.7

We will use the following standard example of constant-velocity particle motion as the basis for many of the subsequent examples on tracking and data fusion:

Example 17

Consider the linear continuous-time model for the motion of particle moving with approximately constant velocity:

$$\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ v(t) \end{bmatrix}.$$

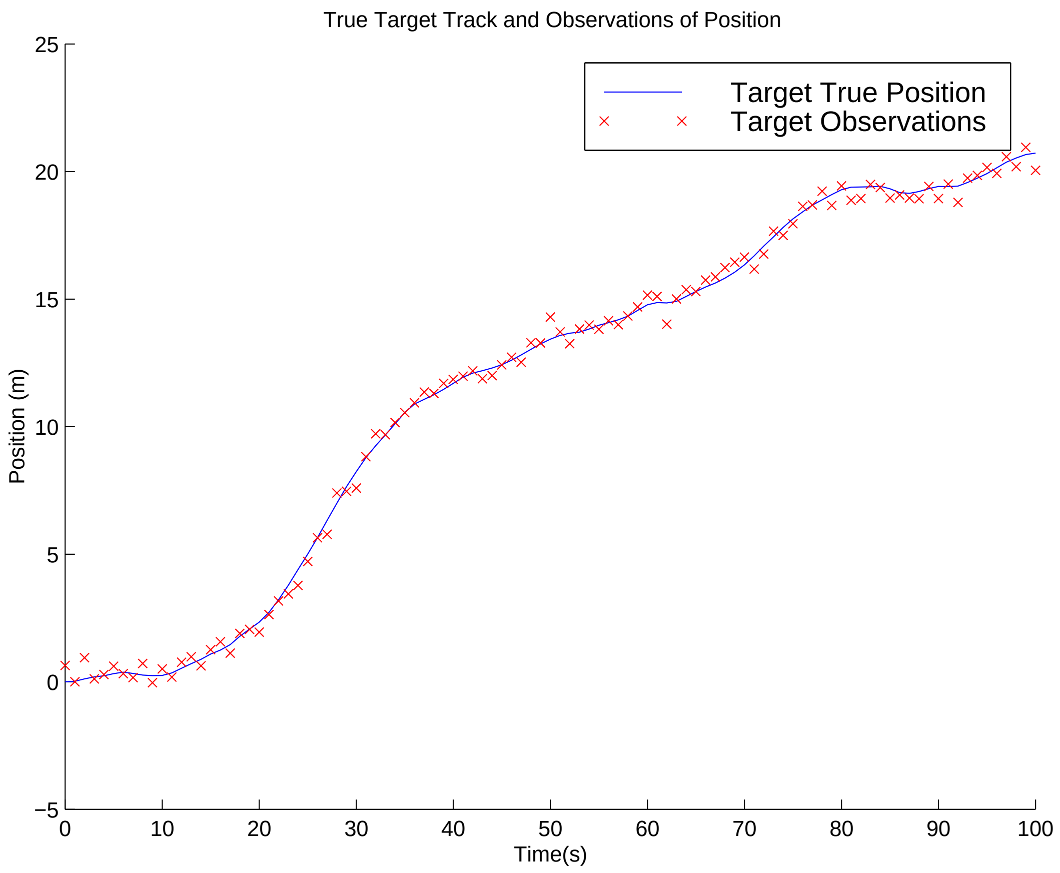


Figure 11: Plot of true target position and observations made of target position for the constant velocity target model described in Example 17, with $\sigma_q = 0.01m/s$ and $\sigma_r = 0.1m$

In this case the state-transition matrix from Equation 72 over a time interval ΔT *is given by*

$$\mathbf{\Phi}(\Delta T) = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}.$$

On the assumption that the process noise is white and uncorrelated with $\mathrm{E}[v(t)] = 0$ *and* $\mathrm{E}[v(t)v(\tau)] = \sigma_q^2(t)\delta(t-\tau)$*, then the equivalent discrete-time noise process is given by*

$$\mathbf{v}(k) = \int_0^{\Delta T} \begin{bmatrix} \tau \\ 1 \end{bmatrix} v(k\Delta T + \tau)\mathrm{d}\tau = \begin{bmatrix} \Delta T^2/2 \\ \Delta T \end{bmatrix} v(k).$$

With the definitions given in Equation 73, the equivalent discrete-time model is given by

$$\begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(k-1) \\ \dot{x}(k-1) \end{bmatrix} + \begin{bmatrix} \Delta T^2/2 \\ \Delta T \end{bmatrix} v(k).$$

If observations are made at each time step k *of the location of the particle the observation model will be in the form*

$$z_x = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + w(k), \qquad \mathrm{E}\{w^2(k)\} \; = \sigma_r^2$$

Figure 11 shows a typical constant-velocity target motion generated according to these models. The target position executes a random walk. Observations are randomly dispersed around true target position.

These equations can trivially be extended to two (and three) dimensions giving a two dimensional model in the form:

$$\begin{bmatrix} x(k) \\ \dot{x}(k) \\ y(k) \\ \dot{y}(k) \end{bmatrix} = \begin{bmatrix} 1 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(k-1) \\ \dot{x}(k-1) \\ y(k-1) \\ \dot{y}(k-1) \end{bmatrix} + \begin{bmatrix} \Delta T^2/2 & 0 \\ \Delta T & 0 \\ 0 & \Delta T^2/2 \\ 0 & \Delta T \end{bmatrix} \begin{bmatrix} v_x(k) \\ v_y(k) \end{bmatrix}, \tag{80}$$

and

$$\begin{bmatrix} z_x \\ z_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \dot{x}(k) \\ y(k) \\ \dot{y}(k) \end{bmatrix} + \begin{bmatrix} w_x(k) \\ w_y(k) \end{bmatrix} \tag{81}$$

with

$$\mathbf{R}(k) = \mathrm{E}\{\mathbf{w}(k)\mathbf{w}^T(k)\} = \begin{bmatrix} \sigma_{rx}^2 & 0 \\ 0 & \sigma_{ry}^2 \end{bmatrix}.$$

This motion model is widely used in target tracking problems (in practice as well as theory) as it is simple, linear and admits easy solution for multiple-target problems. Figure 12 shows a typical $x-y$ target plot generated using this model. Target velocity and heading can be deduced from the magnitude and orientation of the estimated velocity vector $[\dot{x}(k), \dot{y}(k)]^T$ (Figure 12 (c) and (d)). The plots show that this simple model is capable of accommodating many reasonable motion and maneuver models likely to be encountered in typical target tracking problems. It should however be noted that this track is equivalent to running independent models for both x and y as shown in Figure 11. In reality, motions in x and y directions will be physically coupled, and therefore correlated, by the heading of the target. This (potentially valuable information) is lost in this target model.

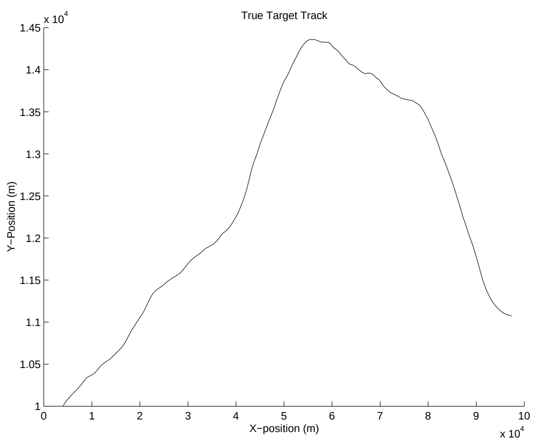
As this example shows, it is always best to start with a continuous-time model for the state and then construct a discrete model, rather than stating the discrete model at the outset. This allows for correct identification of noise transfers and noise correlations.

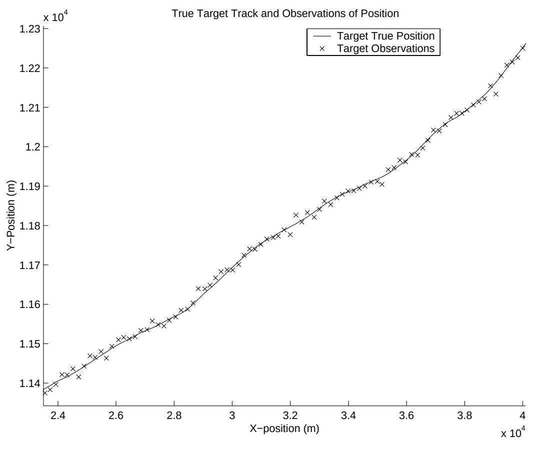
3.1.2 The Kalman Filter Algorithm

The Kalman filter algorithm produces estimates that minimise mean-squared estimation error conditioned on a given observation sequence

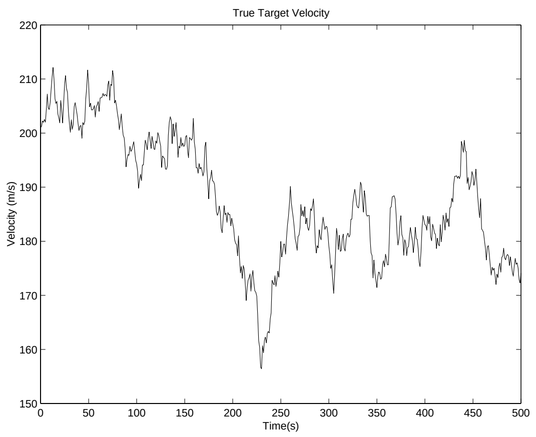
$$\hat{\mathbf{x}}(i \mid j) = \arg \min_{\hat{\mathbf{x}}(i \mid j) \in \Re^n} \mathrm{E}\{(\mathbf{x}(i) - \hat{\mathbf{x}})(\mathbf{x}(i) - \hat{\mathbf{x}})^T \mid \mathbf{z}(1), \cdots, \mathbf{z}(j)\}. \tag{82}$$

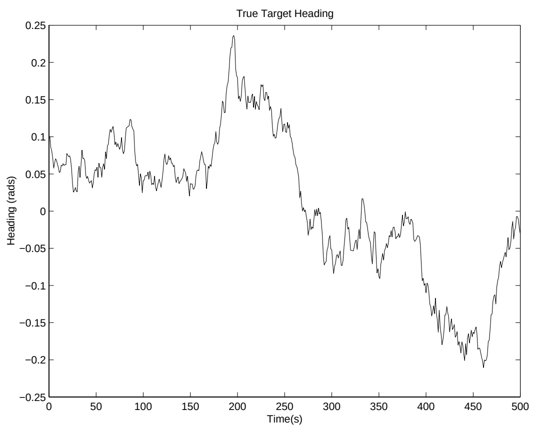
As has been previously demonstrated (Equation 66) the estimate obtained is simply the expected value of the state at time i conditioned on the observations up to time j. The





(a) (b)





(c) (d)

Figure 12: True target track for linear the uncoupled linear target model of Example 17 (a) $x - y$ track; (b) Detail of track and corresponding observations; (c) Deduced target velocity; (d) Deduced target heading.

estimate is thus defined as the conditional mean

$$\hat{\mathbf{x}}(i \mid j) \stackrel{\triangle}{=} \mathrm{E}\{\mathbf{x}(i) \mid \mathbf{z}(1), \cdots, \mathbf{z}(j)\} \stackrel{\triangle}{=} \mathrm{E}\{\mathbf{x}(i) \mid \mathbf{Z}^j\} \, . \tag{83}$$

The estimate variance is defined as the mean squared error in this estimate

$$\mathbf{P}(i \mid j) \stackrel{\triangle}{=} \mathrm{E}\{(\mathbf{x}(i) - \hat{\mathbf{x}}(i \mid j))(\mathbf{x}(i) - \hat{\mathbf{x}}(i \mid j))^T \mid \mathbf{Z}^j\} \, . \tag{84}$$

The estimate of the state at a time k given all information up to time k will be written as $\hat{\mathbf{x}}(k \mid k)$. The estimate of the state at a time k given only information up to time $k-1$ is called a one-step-ahead prediction (or just a prediction) and is written as $\hat{\mathbf{x}}(k \mid k-1)$.

The Kalman filter algorithm is now stated without proof. Detailed derivations can be found in many books on the subject, [28, 7] for example (see also [18]). The state is assumed to evolve in time according to Equation 75. Observations of this state are made at regular time intervals according to Equation 76. The assumptions about the noise processes entering the system, as described by Equations 77, 78 and 79, are assumed true. It is also assumed that an estimate $\hat{\mathbf{x}}(k-1 \mid k-1)$ of the state $\mathbf{x}(k-1)$ at time $k-1$ based on all observations made up to and including time $k-1$ is available, and that this estimate is equal to the conditional mean of the true state $\mathbf{x}(k-1)$ conditioned on these observations. The conditional variance $\mathbf{P}(k-1 \mid k-1)$ in this estimate is also assumed known. The Kalman filter then proceeds recursively in two stages:

Prediction: A prediction $\hat{\mathbf{x}}(k \mid k-1)$ of the state at time k and its covariance $\mathbf{P}(k \mid k-1)$ is computed according to

$$\hat{\mathbf{x}}(k \mid k-1) = \mathbf{F}(k)\hat{\mathbf{x}}(k-1 \mid k-1) + \mathbf{B}(k)\mathbf{u}(k) \tag{85}$$

$$\mathbf{P}(k \mid k-1) = \mathbf{F}(k)\mathbf{P}(k-1 \mid k-1)\mathbf{F}^T(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^T(k). \tag{86}$$

Update: At time k an observation $\mathbf{z}(k)$ is made and the updated estimate $\hat{\mathbf{x}}(k \mid k)$ of the state $\mathbf{x}(k)$, together with the updated estimate covariance $\mathbf{P}(k \mid k)$ is computed from the state prediction and observation according to

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\left(\mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1)\right) \tag{87}$$

$$\mathbf{P}(k \mid k) = (\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k))\mathbf{P}(k \mid k-1)(\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k))^T + \mathbf{W}(k)\mathbf{R}(k)\mathbf{W}^T(k) \tag{88}$$

where the gain matrix $\mathbf{W}(k)$ is given by

$$\mathbf{W}(k) = \mathbf{P}(k \mid k-1)\mathbf{H}(k)\Big[\mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}^T(k) + \mathbf{R}(k)\Big]^{-1} \tag{89}$$

The Kalman filter is recursive or cyclic (see Figure 13). We start with an estimate, generate a prediction, make an observation, then update the prediction to an estimate. The filter makes explicit use of the process model in generating a prediction and the

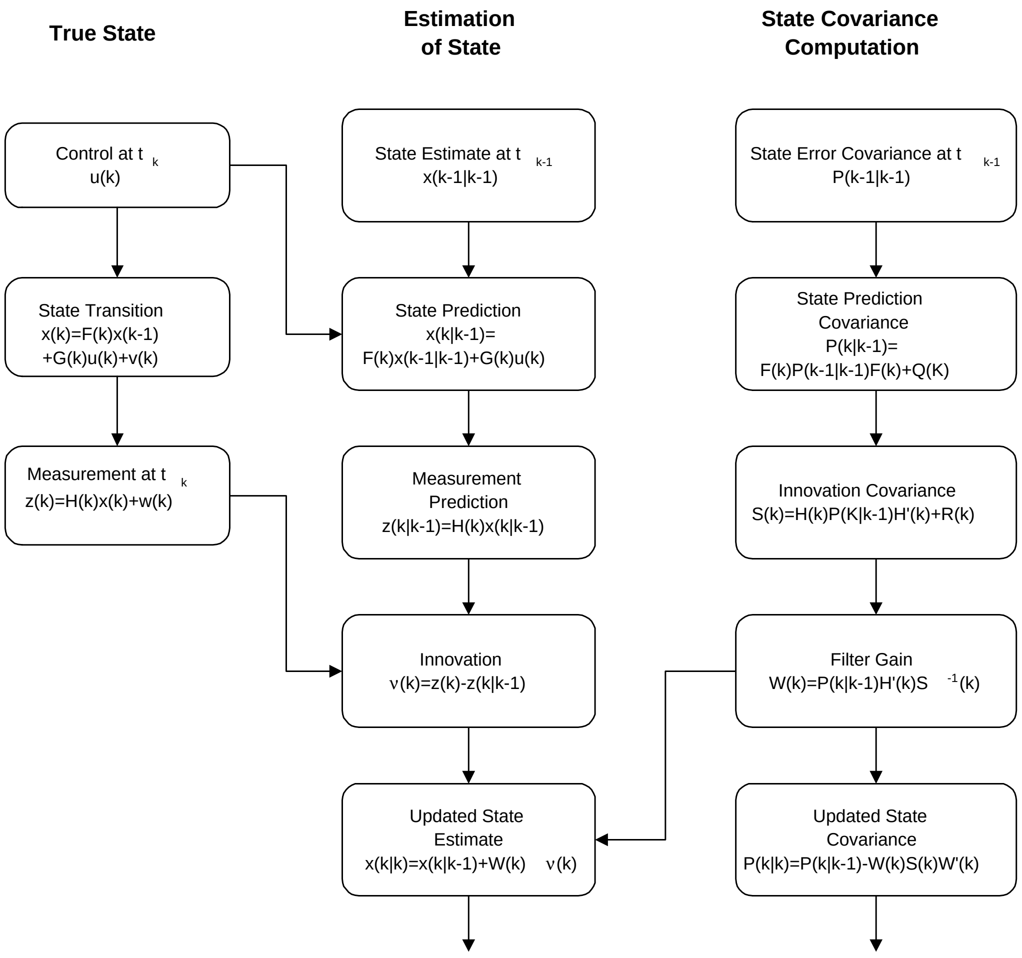


Figure 13: Block diagram of the Kalman filter cycle (after Barshalom and Fortmann 1988 [7])

observation model in generating an update. The update stage of the filter is clearly linear, with a weight $\mathbf{W}(k)$ being associated with the observation $\mathbf{z}(k)$ and a weight $\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k)$ being associated with the prediction $\hat{\mathbf{x}}(k \mid k-1)$. The Kalman filter also provides a propagation equation for the covariance in the prediction and estimate.

Example 18

It is straight-forward to build a state-estimator for the target and observation model of example Example 17. The Kalman filter Equations 87–89 are implemented with $\mathbf{F}(k)$, $\mathbf{Q}(k)$ *as defined in Equation 80 and with* $\mathbf{H}(k)$ *and* $\mathbf{R}(k)$ *as defined in Equation 81. Initial conditions for* $\hat{\mathbf{x}}(0 \mid 0)$ *are determined from the first few observations and with* $\mathbf{P}(0 \mid 0) = 10\mathbf{Q}(k)$ *(see [18] for details). The results of the filter implementation are shown in Figure 14. The Figure shows results for the x component of the track. It should be noted that the estimate always lies between the observation and prediction (it is a weighted sum of these terms). Note also the error (between true and estimated) velocity* *and heading is zero mean and white indicating a good match between model and estimator.*

3.1.3 The Innovation

A prediction can be made as to what observation will be made at a time k based on the observations that have been made up to time $k-1$ by simply taking expectations of the observation Equation 76 conditioned on previous observations:

$$\begin{aligned}\hat{\mathbf{z}}(k \mid k-1) &\stackrel{\triangle}{=} \mathrm{E}\{\mathbf{z}(k) \mid \mathbf{Z}^{k-1}\} \\ &= \mathrm{E}\{\mathbf{H}(k)\mathbf{x}(k) + \mathbf{W}(k) \mid \mathbf{Z}^{k-1}\} \\ &= \mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1) \qquad (90)\end{aligned}$$

The difference between the observation $\mathbf{z}(k)$ and the predicted observation $\mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1)$ is termed the *innovation* or residual $\nu(k)$:

$$\nu(k) = \mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1) \qquad (91)$$

The innovation is an important measure of the deviation between the filter estimates and the observation sequence. Indeed, because the 'true' states are not usually available for comparison with the estimated states, the innovation is often the only measure of how well the estimator is performing. The innovation is particularly important in data association.

The most important property of innovations is that they form an orthogonal, uncorrelated, white sequence,

$$\mathrm{E}\{\nu(k) \mid \mathbf{Z}^{k-1}\} = \mathbf{0}, \qquad \mathrm{E}\{\nu(i)\nu^T(j)\} = \mathbf{S}(i)\delta_{ij}, \qquad (92)$$

where

$$\mathbf{S}(k) = \mathbf{R}(k) + \mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}(k) \qquad (93)$$

is the innovation covariance. This can be exploited in monitoring of filter performance.

The innovation and the innovation variance can be used to express an alternative, simpler form of the update equations:

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\nu(k) \qquad (94)$$

$$\mathbf{P}(k \mid k) = \mathbf{P}(k \mid k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^T(k) \qquad (95)$$

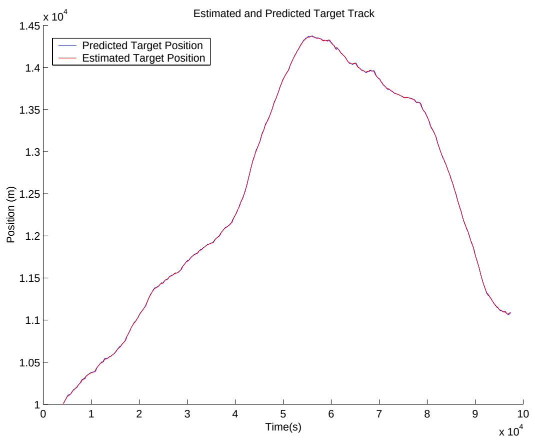
and from Equation 89

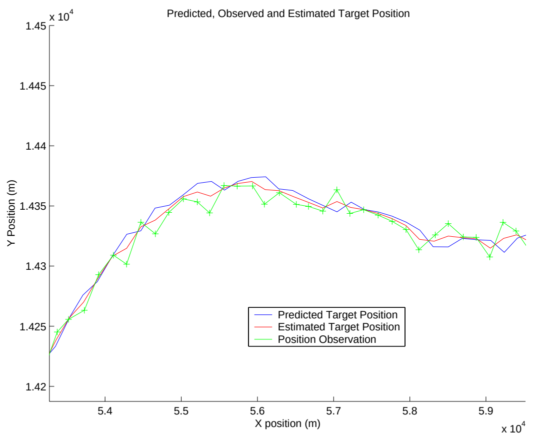
$$\mathbf{W}(k) = \mathbf{P}(k \mid k-1)\mathbf{H}(k)\mathbf{S}^{-1}(k) \qquad (96)$$

This is the preferred form of the update Equations in the remainder of this course.

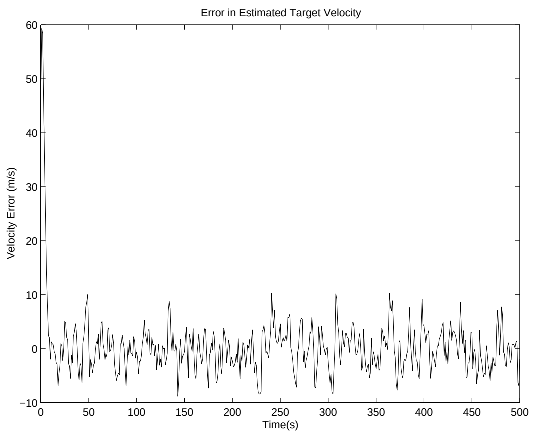
Example 19

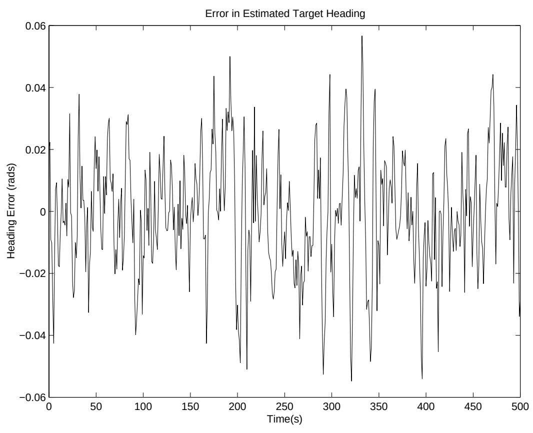
Continuing Example 18, the innovation and innovation covariance can be calculated from Equations 91 and 93. These are shown in Figure 15(a). The most important points to note are that the innovation sequence is zero mean and white, and that approximately





(*a*) (*b*)





(*c*) (*d*)

Figure 14: Estimated target track for the linear uncoupled target model of Examples 17 and 18 (a) Estimate and prediction of $x - y$ track; (b) Detail of estimated track and corresponding observations. Note the estimate always lies between the observation and prediction; (c) Error in estimated target velocity; (d) Error in estimated target heading. Note both heading and velocity errors are zero mean and white.

65% of all innovations lie within the 'one-sigma' standard deviation bounds. Figure 15(b) shows the standard deviation (estimated error) in state prediction and state estimate (for the position state). It is clear that the standard deviation, and therefore covariance, rapidly converge to a constant value. The innovation covariance and gain matrix will also therefore converge to a constant value.

3.1.4 Understanding the Kalman Filter

There are three different, but related ways of thinking about the Kalman filter algorithm: as a simple weighted average; as a conditional mean computed from knowledge of the correlation between observation and state; and as an orthogonal decomposition and projection operation. Each of these interpretations is exploited at different points in this course.

The interpretation of the Kalman filter as weighted average follows directly from Equation 87 as

$$\hat{\mathbf{x}}(k \mid k) = [\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1)] + \mathbf{W}(k)\mathbf{z}(k) \tag{97}$$

It is interesting to explicitly determine the weights, $\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k)$ and $\mathbf{W}(k)$, associated with the prediction and observation respectively. Pre-multiplying Equation 87 by $\mathbf{H}(k)$, substituting for the gain matrix and simplifying, gives

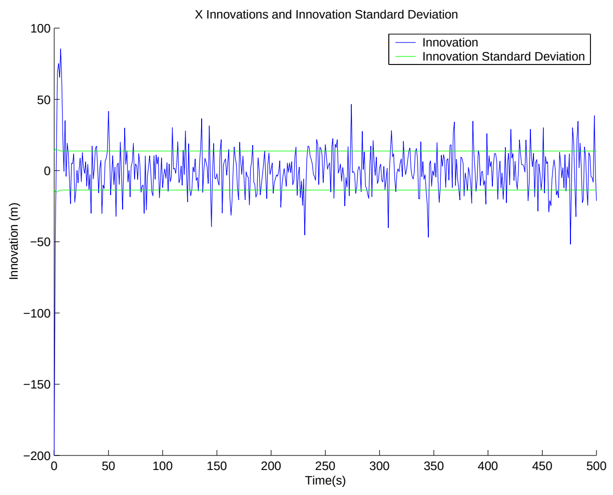
$$\begin{aligned} \mathbf{H}(k)\hat{\mathbf{x}}(k \mid k) = &\left[(\mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}(k))^{-1} + \mathbf{R}^{-1}(k)\right]^{-1} \\ \times &\left[(\mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}(k))^{-1}\mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1) + \mathbf{R}^{-1}(k)\mathbf{z}(k)\right]. \end{aligned}$$

This is just a weighted sum of the predicted observation $\mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1)$ and the actual observation $\mathbf{z}(k)$ in which $(\mathbf{H}(k)\mathbf{P}(k+1 \mid k)\mathbf{H}(k))^{-1}$ is the inverse of the prediction covariance projected in to observation space and is the 'confidence' in the predicted observation, and $\mathbf{R}^{-1}(k)$ is the inverse observation noise covariance and is the 'confidence' in the observation itself. The sum is normalised by the total confidence $[(\mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}(k))^{-1} + \mathbf{R}^{-1}(k)]^{-1}$ and the result is a new estimate for the state projected into the observation space as $\mathbf{H}(k)\hat{\mathbf{x}}(k \mid k)$. The covariance in this projected estimate is given simply by the normalisation factor of the weighted sum

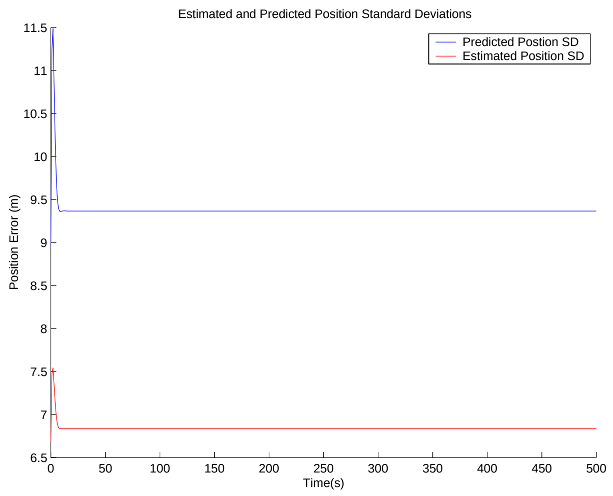
$$\mathbf{H}(k)\mathbf{P}(k \mid k)\mathbf{H}(k) = \left[(\mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}(k))^{-1} + \mathbf{R}^{-1}(k)\right]^{-1} \tag{98}$$

This averaging process works by first projecting the state vector into observation space as $\mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1)$ and corresponding covariance as $\mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}(k)$, where the observation $\mathbf{z}(k)$ and its corresponding covariance $\mathbf{R}(k)$ are directly available, and then computes an updated estimate $\mathbf{H}(k)\hat{\mathbf{x}}(k \mid k)$ as a weighted sum of these, again in observation space. Clearly, this interpretation of the Kalman filter holds only for those states which can be written as a linear combination of the observation vector.

The Kalman filter is also able to provide estimates of states which are not a linear combination of the observation vector. It does this by employing the cross correlation



(*a*)



(*b*)

Figure 15: Track Errors for the linear uncoupled target model of Examples 17 and 18: (a) Innovation and innovation variance in x estimate; (b) Estimated error (standard deviation) in x estimate and x prediction.

between observed and unobserved states to find that part of the observation information which is correlated with the unobserved states. In this sense, the Kalman filter is sometimes referred to as a **linear observer**. This interpretation of the Kalman filter algorithm is most easily appreciated by considering the estimator as derived from the conditional mean, computed directly from Bayes theorem [14, 28]. In this case the estimator of Equation 87 may be written as

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{P}_{xz}\mathbf{P}_{zz}^{-1}\left[\mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1)\right] \tag{99}$$

with Equation 88 being written as

$$\mathbf{P}(k \mid k) = \mathbf{P}_{xx} - \mathbf{P}_{xz}\mathbf{P}_{zz}^{-1}\mathbf{P}_{xz}^{T} \tag{100}$$

where

$$\begin{aligned}
\mathbf{P}_{xz} &= \mathrm{E}\{(\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k-1))(\mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1))^{T} \mid \mathbf{Z}^{k-1}\} \\
\mathbf{P}_{zz} &= \mathrm{E}\{(\mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1))(\mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1))^{T} \mid \mathbf{Z}^{k-1}\} \\
\mathbf{P}_{xx} &= \mathbf{P}(k \mid k-1)
\end{aligned}$$

are respectively the cross correlation between prediction and observation, the observation covariance and the prediction covariance. Equation 99 shows how new observations $\mathbf{z}(k)$ contribute to the updated estimate $\hat{\mathbf{x}}(k \mid k)$. First, the difference between observation and predicted observation is computed. Then the term $\mathbf{P}_{xz}$, by definition, tells us how the states are correlated with the observations made, and finally $\mathbf{P}_{zz}^{-1}$ is used to 'normalise' this correlation by the confidence we have in the new observation. This shows that provided we have knowledge of how the states are correlated to the observations made, we may use the observation information to estimate the state even in cases where the state is not a linear combination of the observation vector.

A third method of interpreting the Kalman filter is as geometric operation that projects the true state $\mathbf{x}(k)$ on to the subspace spanned by the observation sequence $\mathbf{Z}^k$. Figure 16 shows this graphically using multi-dimensional space representation. The space spanned by the set $\mathbf{Z}^k$ is shown as a plane embedded in the overall state space, $\mathbf{Z}^{k-1}$ as a line embedded in the $\mathbf{Z}^k$ plane, the new observation $\mathbf{z}(k)$ as a line so arranged that $\mathbf{Z}^{k-1}$ and $\mathbf{z}(k)$ together generate $\mathbf{Z}^k$. The innovation $\nu(k)$ is described as a line orthogonal to $\mathbf{Z}^{k-1}$ also embedded in the $\mathbf{Z}^k$. The prediction $\hat{\mathbf{x}}(k \mid k-1)$ lies in the space $\mathbf{Z}^{k-1}$, $\mathbf{W}(k)\nu(k)$ lies in the space orthogonal to $\mathbf{Z}^{k-1}$, and the estimate $\hat{\mathbf{x}}(k \mid k)$ lies in the space $\mathbf{Z}^k$ and is again the projection of $\mathbf{x}(k)$ in to this space. Figure 16 makes clear the role of $\mathbf{W}(k)$ and $[\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k)]$ as complimentary projections. The estimate can be thought of either as the orthogonal sum of prediction $\hat{\mathbf{x}}(k \mid k-1)$ and weighted innovation $\mathbf{W}(k)\nu(k)$, or as the sum of the parallel vectors $[\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k)]\hat{\mathbf{x}}(k \mid k-1)$ and $\mathbf{W}(k)\mathbf{z}(k)$.

3.1.5 Steady-State Filters

When filtering is synchronous, and state and observation matrices time-invariant, the state and observation models may be written as

$$\mathbf{x}(k) = \mathbf{F}\mathbf{x}(k-1) + \mathbf{G}\mathbf{u}(k) + \mathbf{v}, \tag{101}$$

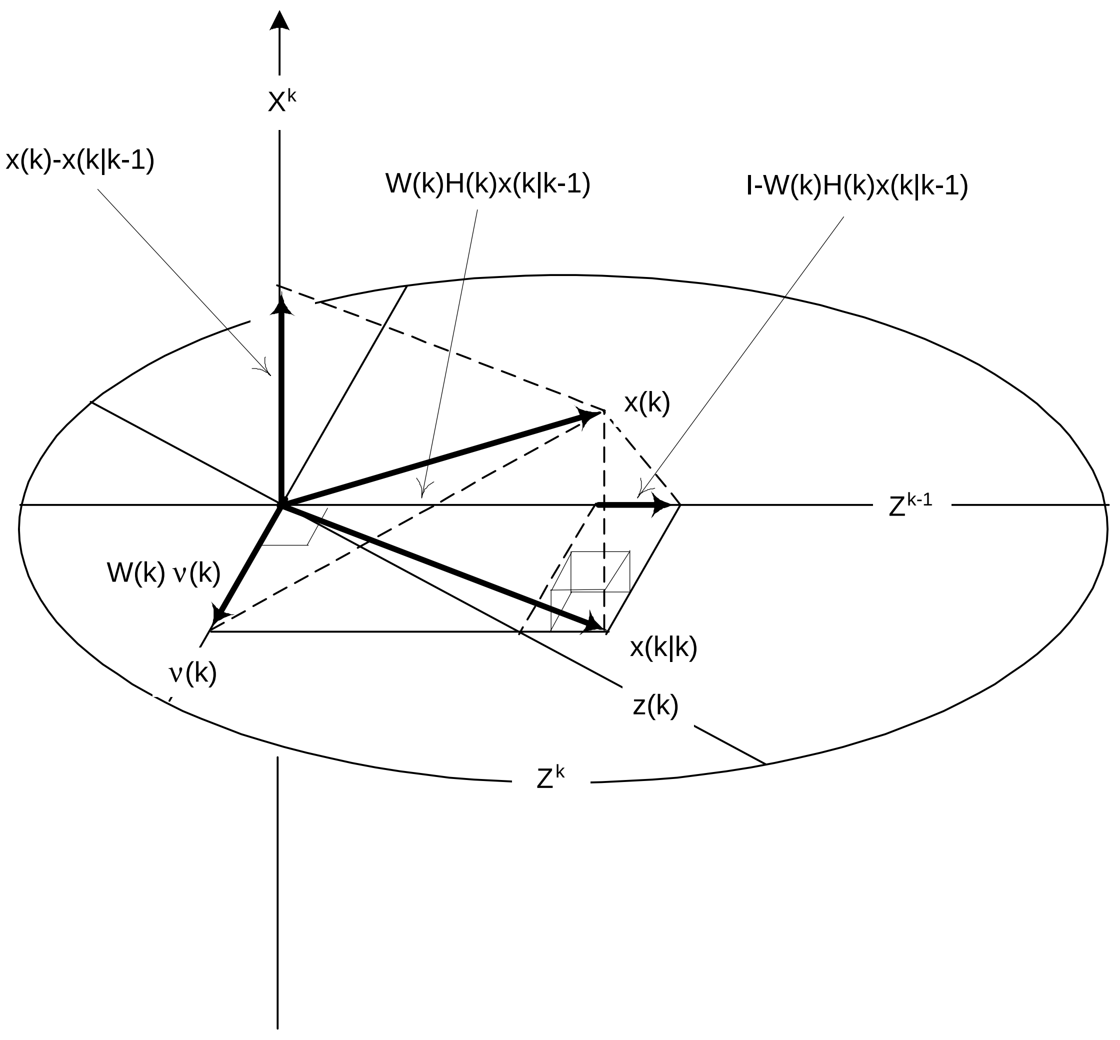


Figure 16: A projection diagram for the Kalman filter. The complete space is the true state space $\mathbf{X}^k$. The sub-space spanned by the first k observations is shown as the ellipse $\mathbf{Z}^k$. The space $\mathbf{Z}^{k-1}$ spanned by the first $k-1$ observations is shown as a line sub-space of the space $\mathbf{Z}^k$. The true state $\mathbf{x}(k)$ lies in the space $\mathbf{X}^k$, the estimate $\hat{\mathbf{x}}(k \mid k)$ lies in the space $\mathbf{Z}^k$ (because $\hat{\mathbf{x}}(k \mid k)$ is a linear combination of observations up to time k), prediction $\hat{\mathbf{x}}(k \mid k-1)$ lies in the space $\mathbf{Z}^{k-1}$. The estimate is a perpendicular projection of the true state on to the space $\mathbf{Z}^k$. By construction, the innovation space $\nu(k)$ is perpendicular to the space $\mathbf{Z}^{k-1}$. The new estimate $\hat{\mathbf{x}}(k \mid k)$ is the sum of: 1) the projection of $\mathbf{x}(k)$ onto $\mathbf{Z}^{k-1}$ along $\mathbf{z}(k)$, given by $[\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k)]\hat{\mathbf{x}}(k \mid k-1)$; and 2) the projection of $\mathbf{x}(k)$ onto $\mathbf{z}(k)$ along $\mathbf{Z}^{k-1}$, given by $\mathbf{W}(k)\mathbf{z}(k)$. For comparison, the estimate that would result based only on the observation $\mathbf{z}(k)$ is $\hat{\mathbf{x}}(k \mid \mathbf{z}(k))$, the projection of $\mathbf{x}(k)$ onto $\mathbf{z}(k)$. Diagramatic method from [41] (see also [14])

$$\mathbf{z}(k) = \mathbf{H}\mathbf{x}(k) + \mathbf{w}, \tag{102}$$

with

$$\mathrm{E}\{\mathbf{v}\mathbf{v}^T\} = \mathbf{Q}, \qquad \mathrm{E}\{\mathbf{w}\mathbf{w}^T\} = \mathbf{R}. \tag{103}$$

The calculations for the state estimate covariances and filter gains do not depend on the value of the observations made. Consequently, if **F**, **H**, **Q**, and **R** are time invariant, then the innovation covariance **S** will be constant and the covariance and gain matrices will also tend to a constant steady state value

$$\mathbf{P}(k \mid k) \to \mathbf{P}^+_\infty, \qquad \mathbf{P}(k \mid k-1) \to \mathbf{P}^-_\infty, \qquad \mathbf{W}(k) \to \mathbf{W}_\infty \tag{104}$$

This convergence is often rapid as is graphically demonstrated in Figure 15.

If the covariances and gain matrix all tend to a constant steady-state value after only a few time-steps, it seems sensible that the computationally intensive process of computing these values on-line be avoided by simply inserting these constant values in to the filter from the start. In effect, this means that a constant gain matrix is used in the computation of the estimates

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}\left[\mathbf{z}(k) - \mathbf{H}\hat{\mathbf{x}}(k \mid k-1)\right]. \tag{105}$$

The steady-state value of the gain matrix **W** is most easily computed by simply simulating the covariance calculation off-line (which does not require that any observations are made).

In the case of a 'constant velocity' model, the gain matrix can be described by two dimensionless coefficients α and β,

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \begin{bmatrix} \alpha \\ \beta/T \end{bmatrix}\left[\mathbf{z}(k) - \mathbf{H}\hat{\mathbf{x}}(k \mid k-1)\right]. \tag{106}$$

This is known as the α–β filter. In the case of a 'constant acceleration' model, the gain matrix can be described by three dimensionless coefficients α, β and γ,

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \begin{bmatrix} \alpha \\ \beta/T \\ \gamma/T^2 \end{bmatrix}\left[\mathbf{z}(k) - \mathbf{H}\hat{\mathbf{x}}(k \mid k-1)\right]. \tag{107}$$

This is known as the α–β–γ filter. These constant parameters are usually obtained through simulation and a degree of judgement.

The steady-state filter provides a substantial reduction in the amount of on-line computation that must be performed by the filter. In many situations this may be invaluable; particularly in multi-target tracking applications where many thousands of filters must be employed simultaneously. Providing that the assumption of constant system and noise models applies, the error incurred by using a steady-state filter is insignificant. In practice, steady-state filters are used even in situations where the basic assumptions of linear time-invariance and constant noise injection are violated.

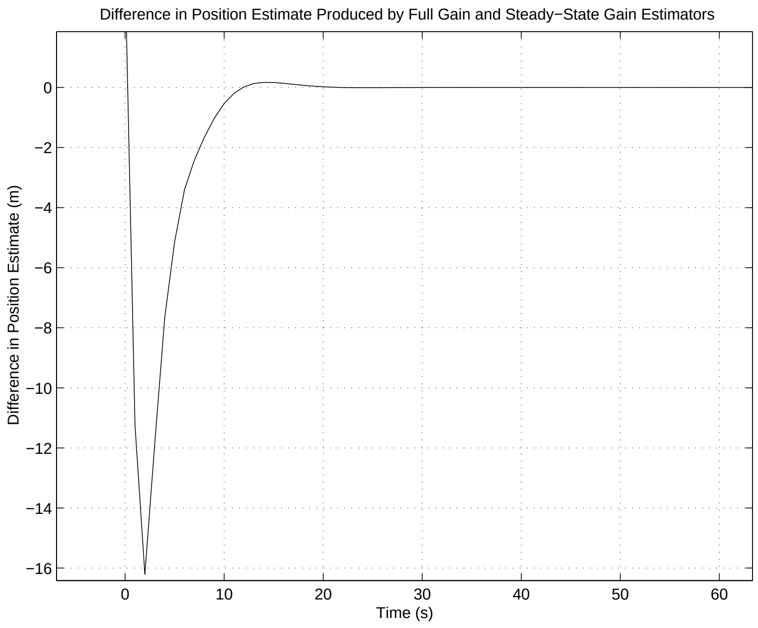


Figure 17: Plot of the difference between x position estimates computed using a full gain history and a steady state-gain for the constant velocity target model described in Example 17.

Example 20

Consider again the Example 18 of constant-velocity particle motion. With a process and observation noise standard deviations given by $\sigma_q = 0.01$, *and* $\sigma_r = 0.1$ *respectively and with a constant sample period of one second, the steady-state gain matrix is found to be* $\mathbf{W} = [0.4673, 0.1460]^T$. *The equivalent steady-state filter (with* $T = 1$*) will therefore have parameters* $\alpha = 0.4673$ *and* $\beta = 0.1460$. *Figure 17 shows the error between estimates produced by the constant-velocity filter (with full covariance and gain calculation), and the estimates produced by the equivalent* α*–*β *filter. It can be seen that after an initially large error, but after only a few time-steps, the error between these two filters is negligible. The large error at the start is caused by initialisation with the steady-state values of gain.*

3.1.6 Asynchronous, Delayed and Asequent Observations

In data fusion systems the observation process is almost never synchronous. This is because information must come from many different sensors, which will each have different sample rates and latencies associated with the acquisition and communication of observations.

Generally, three timing issues of increasing complexity must be addressed;

- Asynchronous data: Information which arrives at random intervals but in a timely manner.

- Delayed data: Information that arrives both at random intervals and late; after an estimate has already been constructed.
- Asequent data: Information that arrives randomly, late and out of time sequence (temporal order).

The asynchronous data problem is relatively easy to accommodate in the existing Kalman filter framework. For asynchronous observations, the general discrete-time state and observation models of Equations 74 and 69 apply. With these definitions, the Kalman filter for asynchronous observations remains essentially the same as for the synchronous case. It is assumed the existence of an estimate for the state $\hat{\mathbf{x}}(t_{k-1} \mid t_{k-1})$ at time t_{k-1} that includes all observations made up to this time. At a later time $t_k > t_{k-1}$ an observation is made according to Equation 69. The first step is simply to generate a prediction of the state at the time the observation was made according to

$$\hat{\mathbf{x}}(t_k \mid t_{k-1}) = \mathbf{F}(t_k)\hat{\mathbf{x}}(t_{k-1} \mid t_{k-1}) + \mathbf{B}(t_k)\mathbf{u}(t_k), \tag{108}$$

together with an associated prediction covariance

$$\mathbf{P}(t_k \mid t_{k-1}) = \mathbf{F}(t_k)\mathbf{P}(t_{k-1} \mid t_{k-1})\mathbf{F}^T(t_k) + \mathbf{G}(t_k)\mathbf{Q}(t_k)\mathbf{G}^T(t_k). \tag{109}$$

An innovation and innovation covariance are then computed according to

$$\nu(t_k) = \mathbf{z}(t_k) - \mathbf{H}(t_k)\hat{\mathbf{x}}(t_k \mid t_{k-1}), \tag{110}$$

$$\mathbf{S}(t_k) = \mathbf{H}(t_k)\mathbf{P}(t_k \mid t_{k-1})\mathbf{H}^T(t_k) + \mathbf{R}(t_k), \tag{111}$$

from which an updated state estimate and associated covariance can be found from

$$\hat{\mathbf{x}}(t_k \mid t_k) = \hat{\mathbf{x}}(t_k \mid t_{k-1}) + \mathbf{W}(t_k)\nu(t_k), \tag{112}$$

$$\mathbf{P}(t_k \mid t_k) = \mathbf{P}(t_k \mid t_{k-1}) - \mathbf{W}(t_k)\mathbf{S}(t_k)\mathbf{W}^T(t_k), \tag{113}$$

where

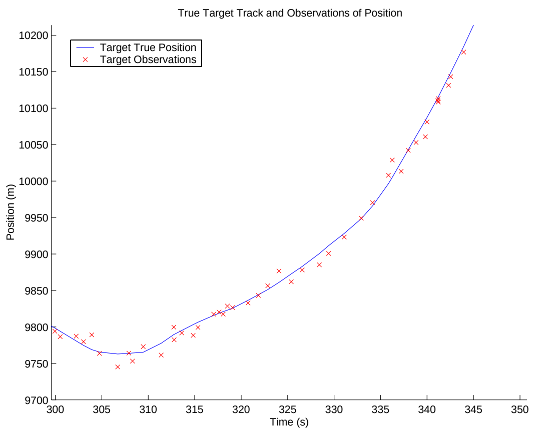
$$\mathbf{W}(t_k) = \mathbf{P}(t_k \mid t_{k-1})\mathbf{H}^T(t_k)\mathbf{S}(t_k) \tag{114}$$

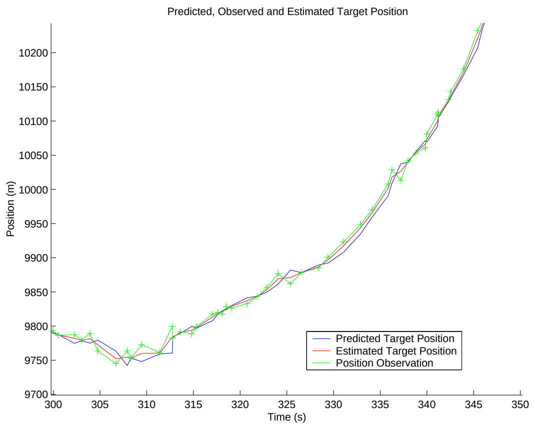
is the associated gain matrix. The main point to note here is that the injected process noise variance $\mathbf{Q}(t_k)$ and all the model matrices ($\mathbf{F}(t_k)$, $\mathbf{B}(t_k)$, $\mathbf{G}(t_k)$, and $\mathbf{H}(t_k)$) are all functions of the time interval $t_k - t_{k-1}$. Consequently, the computed variances $\mathbf{P}(t_k \mid t_k)$, $\mathbf{P}(t_k \mid t_{k-1})$, $\mathbf{S}(t_k)$, and the gain matrix $\mathbf{W}(t_k)$ will not remain constant and must be computed at every time slot.

Example 21

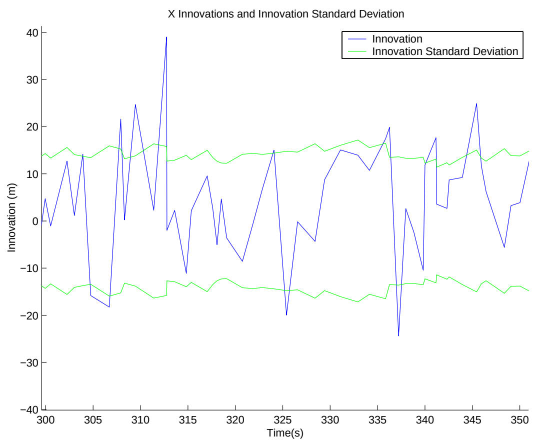
Consider again the example of the constant-velocity particle motion described by the continuous-time process model

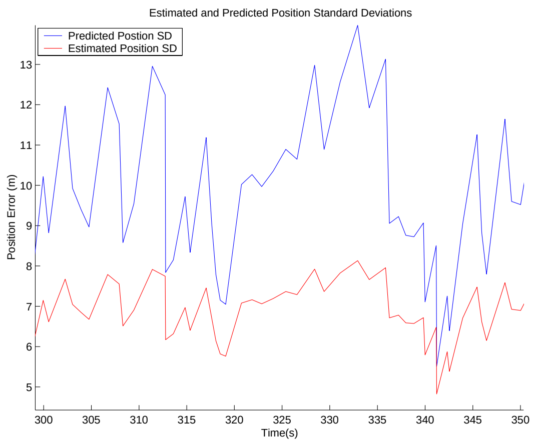
$$\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ v(t) \end{bmatrix}.$$





(a) (b)





(c) (d)

Figure 18: Estimated target track for the linear constant velocity particle model with asynchronous observations and updates. A time-section shown of: (a) True state and asynchronous observations; (b) State predictions and estimates, together with observations; (c) Innovations and innovation standard deviations; (d) State prediction and state estimate standard deviations (estimated errors).

We assume observations of the position of the particle are made asynchronously at times t_k according to

$$z_x(t_k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x(t_k) \\ \dot{x}(t_k) \end{bmatrix} + w(t_k), \qquad \mathrm{E}\{w^2(t_k)\} \; = \sigma_r^2.$$

We define $\delta t_k = t_k - t_{k-1}$ as the time difference between any two successive observations made at time t_{k-1} and t_k. With the definitions given by Equation 73, the discrete-time process model becomes

$$\begin{bmatrix} x(t_k) \\ \dot{x}(t_k) \end{bmatrix} = \begin{bmatrix} 1 & \delta t_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(t_{k-1}) \\ \dot{x}(t_{k-1}) \end{bmatrix} + \begin{bmatrix} \delta t^2 \\ \delta t \end{bmatrix} v(t_k),$$

with

$$\mathbf{Q}(t_k) = \mathrm{E}[\mathbf{v}(t_k)\mathbf{v}^T(t_k)] = \int_0^{\delta t} \begin{bmatrix} \delta t \\ 1 \end{bmatrix} \begin{bmatrix} \delta t & 1 \end{bmatrix} q \mathrm{d}\tau = \begin{bmatrix} \delta t^3/3 & \delta t^2/2 \\ \delta t^2/2 & \delta t \end{bmatrix} q.$$

The implementation of an asynchronous filter looks similar to the implementation of a synchronous filter except that the matrices defined in Equation 73 must be recomputed at every time-step. However, the variances computed by the asynchronous filter look quite different from those computed in the synchronous case even if they have the same average value. Figure 18 shows the results of an asynchronous constant-velocity filter. The process and observation noise standard-deviations have been set, as before, to $\sigma_q = 0.01$ and $\sigma_r = 0.1$. Observations are generated with random time intervals uniformly distributed between 0 and 2 seconds (an average of 1 second). Figure 18(a) shows the computed position estimate and prediction standard deviations . Clearly, these covariances are not constant, unsurprisingly, the variance in position estimate grows in periods where there is a long time interval between observations and reduces when there is a small interval between observations. Figure 18(b) shows the filter innovations and associated computed standard deviations. Again, these are not constant, however they do still satisfy criteria for whiteness with 95% of innovations falling within their corresponding $\pm 2\sigma$ bounds.

To deal with delayed observations is to maintain two estimates, one associated with the true time at which the last observation was obtained, and the second a prediction, based on this last estimate, which describes the state at the current time. When a new, delayed, observation is made, the current prediction is discarded and a new prediction up to the time of this delayed observation is computed, based on the estimate at the time of the previous delayed observation. This is then combined with the new observation to produce a new estimate which is itself predicted forward to the current time. Let t_c be the current time at which a new delayed observation is made available, let t_p be the previous time a delayed observation was made available and let $t_c > t_k > t_{k-1}$, and $t_c > t_p > t_{k-1}$. We assume that we already have an estimate $\hat{\mathbf{x}}(t_{k-1} \mid t_{k-1})$ and an estimate (strictly a prediction) $\hat{\mathbf{x}}(t_p \mid t_{k-1})$ and that we acquire an observation $\mathbf{z}(t_k)$ taken at time t_k. We start by simply discarding the estimate $\hat{\mathbf{x}}(t_p \mid t_{k-1})$, and generating a new prediction $\hat{\mathbf{x}}(t_k \mid t_{k-1})$ and prediction covariance $\mathbf{P}(t_k \mid t_{k-1})$ from Equations 108 and 109. These together with the delayed observation are used to compute a new estimate $\hat{\mathbf{x}}(t_k \mid t_k)$, at the time the delayed observation was made, from Equations 112, 113, and 114. This estimate is then predicted forward to the current time to produce an estimate $\hat{\mathbf{x}}(t_c \mid t_k)$ and its associated covariance $\mathbf{P}(t_c \mid t_k)$ according to

$$\hat{\mathbf{x}}(t_p \mid t_k) = \mathbf{F}(t_p)\hat{\mathbf{x}}(t_k \mid t_k) + \mathbf{B}(t_p)\mathbf{u}(t_p), \tag{115}$$

$$\mathbf{P}(t_p \mid t_k) = \mathbf{F}(t_p)\mathbf{P}(t_k \mid t_k)\mathbf{F}^T(t_p) + \mathbf{G}(t_p)\mathbf{Q}(t_p)\mathbf{G}^T(t_p). \tag{116}$$

Both estimates $\hat{\mathbf{x}}(t_k \mid t_k)$ and $\hat{\mathbf{x}}(t_p \mid t_k)$ together with their associated covariances should be maintained for the next observations.

It should be clear that if the observations are delayed, then the estimate provided at the current time will not be as good as the estimates obtained when the observations are obtained immediately. This is to be expected because the additional prediction required injects additional process noise into the state estimate.

It is also useful to note that the same techniques can be used to produce estimates for any time in the future as well as simply the current time. This is sometimes useful in providing advance predictions that are to be used to synchronize with incoming observations.

Asequent data occurs when the observations made are delayed in such a way that they arrive at the filter for processing out of time-order. Although this does not often happen in single-sensor systems, it is a common problem in multi-sensor systems where the pre-processing and communication delays may differ substantially between different sensors. The essential problem here is that the gain matrix with and without the delayed observation will be different and the previous estimate, corresponding to the time at which the delayed observation was taken, cannot easily be "unwrapped" from the current estimate. With the tools we have so far developed, there is no simple way of dealing with this problem other than by remembering past estimates and recomputing the current estimate every time an out-of-order observation is obtained. However, a solution to this problem is possible using the inverse-covariance filter which we will introduce latter in this chapter.

3.1.7 The Extended Kalman Filter

The extended Kalman filter (EKF) is a form of the Kalman filter that can be employed when the state model and/or the observation model are non-linear. The EKF is briefly described in this section.

The state models considered by the EKF are described in state-space notation by a first order non-linear vector differential equation or state model of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{v}(t), t], \tag{117}$$

where

$\mathbf{x}(t) \in \Re^n$ is the state of interest,

$\mathbf{u}(t) \in \Re^r$ is a known control input,

$\mathbf{f}[\cdot,\cdot,\cdot]$ a mapping of state and control input to state 'velocity', and

$\mathbf{v}(t)$ a random vector describing both dynamic driving noise and uncertainties in the state model itself ($\mathbf{v}(t)$ is often assumed additive).

The observation models considered by the EKF are described in state-space notation by a non-linear vector function in the form

$$\mathbf{z}(t) = \mathbf{h}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t), t] \tag{118}$$

where

$\mathbf{z}(t) \in \Re^m$ is the observation made at time t,

$\mathbf{h}[\cdot,\cdot,\cdot]$ is a mapping of state and control input to observations, and

$\mathbf{w}(t)$ a random vector describing both measurement corruption noise and uncertainties in the measurement model itself ($\mathbf{w}(t)$ is often assumed additive).

The EKF, like the Kalman filter, is almost always implemented in discrete-time. To do this, a discrete form of Equations 117 and 118 are required. First, a discrete-time set $\mathbf{t} = \{t_0, t_1, \cdots t_k, \cdots\}$ is defined. Equation 118 can then be written in discrete time as

$$\mathbf{z}(t_k) = \mathbf{h}[\mathbf{x}(t_k), \mathbf{u}(t_k), \mathbf{w}(t_k), t_k], \qquad \forall t_k \in \mathbf{t} \tag{119}$$

where $\mathbf{z}(t_k)$, $\mathbf{x}(t_k)$ and $\mathbf{w}(t_k)$ are the discrete-time observation, state and noise vectors evaluated at the discrete time instant t_k. The discrete-time form of the state equation requires integration of Equation 117 over the interval (t_k, t_{k-1}) as

$$\mathbf{x}(t_k) = \mathbf{x}(t_{k-1}) + \int_{t_{k-1}}^{t_k} \mathbf{f}[\mathbf{x}(\tau), \mathbf{u}(\tau), \mathbf{v}(\tau), \tau]\mathrm{d}\tau. \tag{120}$$

In practice, this integration is normally solved using a simple Euler (backward difference) approximation as

$$\mathbf{x}(t_k) = \mathbf{x}(t_{k-1}) + \Delta T_k \mathbf{f}[\mathbf{x}(t_{k-1}), \mathbf{u}(t_{k-1}), \mathbf{v}(t_{k-1}), t_{k-1}], \tag{121}$$

where $\Delta T_k = t_k - t_{k-1}$. As with the Kalman filter, when the sample interval is constant, time is indexed by k and Equations 121 and 119 are written as

$$\mathbf{x}(k) = \mathbf{x}(k-1) + \Delta T \mathbf{f}[\mathbf{x}(k-1), \mathbf{u}(k-1), \mathbf{v}(k-1), k], \tag{122}$$

and

$$\mathbf{z}(k) = \mathbf{h}[\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}(k), k]. \tag{123}$$

To apply the Kalman filter algorithm to estimation problems characterised by non-linear state and observation models, perturbation methods are used to linearise true non-linear system models around some nominal state trajectory to yield a model which is itself linear in the error. Given a non-linear system model in the form of Equation 117, a nominal state trajectory is described using the same process model (assuming $\mathbf{v}(t)$ is zero mean),

$$\dot{\mathbf{x}}_n(t) = \mathbf{f}[\mathbf{x}_n(t), \mathbf{u}_n(t), t]. \tag{124}$$

Then, Equation 117 is expanded about this nominal trajectory as a Taylor series;

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{f}[\mathbf{x}_n(t), \mathbf{u}_n(t), t] \\ &\quad +\nabla\mathbf{f}_{\mathbf{x}}(t)\delta\mathbf{x}(t) + O\left[\delta\mathbf{x}(t)^2\right] \\ &\quad +\nabla\mathbf{f}_{\mathbf{u}}(t)\delta\mathbf{u}(t) + O\left[\delta\mathbf{u}(t)^2\right] \\ &\quad +\nabla\mathbf{f}_{\mathbf{v}}(t)\mathbf{v}(t),\end{aligned} \tag{125}$$

where

$$\nabla\mathbf{f}_{\mathbf{x}}(t) \stackrel{\triangle}{=} \left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\right|_{\substack{\mathbf{x}(t)=\mathbf{x}_n(t)\\ \mathbf{u}(t)=\mathbf{u}_n(t)}}, \quad \nabla\mathbf{f}_{\mathbf{u}}(t) \stackrel{\triangle}{=} \left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right|_{\substack{\mathbf{x}(t)=\mathbf{x}_n(t)\\ \mathbf{u}(t)=\mathbf{u}_n(t)}}, \quad \nabla\mathbf{f}_{\mathbf{v}}(t) \stackrel{\triangle}{=} \left.\frac{\partial\mathbf{f}}{\partial\mathbf{v}}\right|_{\substack{\mathbf{x}(t)=\mathbf{x}_n(t)\\ \mathbf{u}(t)=\mathbf{u}_n(t)}} \tag{126}$$

and

$$\delta\mathbf{x}(t) \stackrel{\triangle}{=} (\mathbf{x}(t) - \mathbf{x}_n(t)), \qquad \delta\mathbf{u}(t) \stackrel{\triangle}{=} (\mathbf{u}(t) - \mathbf{u}_n(t)). \tag{127}$$

Subtracting Equation 124 from Equation 125 provides a linear error model in the form

$$\delta\dot{\mathbf{x}}(t) = \nabla\mathbf{f}_{\mathbf{x}}(t)\delta\mathbf{x}(t) + \nabla\mathbf{f}_{\mathbf{u}}(t)\delta\mathbf{u}(t) + \nabla\mathbf{f}_{\mathbf{v}}(t)\mathbf{v}(t). \tag{128}$$

Identifying $\mathbf{F}(t) = \nabla\mathbf{f}_{\mathbf{x}}(t)$, $\mathbf{B}(t) = \nabla\mathbf{f}_{\mathbf{u}}(t)$, and $\mathbf{G}(t) = \nabla\mathbf{f}_{\mathbf{v}}(t)$, Equation 128, is now in the same form as Equation 67 and may be solved for the perturbed state vector $\delta\mathbf{x}(t)$ in closed form through Equation 70, yielding a linear discrete-time equation for error propagation. Clearly this approximation is only valid when terms of second order and higher are small enough to be neglected; when the true and nominal trajectories are close and $\mathbf{f}(\cdot)$ is suitably smooth. With judicious design of the estimation algorithm this can be achieved surprisingly often. It is also possible to retain or approximate higher-order terms from Equation 125 and so improve the validity of the approximation.

Similar arguments can be applied to linearise Equation 68 to provide an observation error equation in the form

$$\delta\dot{\mathbf{z}}(t) = \nabla\mathbf{h}_{\mathbf{x}}(t)\delta\mathbf{x}(t) + \nabla\mathbf{f}_{\mathbf{h}}(t)\delta\mathbf{w}(t). \tag{129}$$

Identifying $\mathbf{H}(t) = \nabla\mathbf{f}_{\mathbf{x}}(t)$, and $\mathbf{D}(t) = \nabla\mathbf{f}_{\mathbf{w}}(t)$, Equation 129, is now in the same form as Equation 68.

The discrete-time extended Kalman filter algorithm can now be stated. With appropriate identification of discrete time states and observations, the state model is written as

$$\mathbf{x}(k) = \mathbf{f}\left(\mathbf{x}(k-1), \mathbf{u}(k), \mathbf{v}(k), k\right), \tag{130}$$

and the observation model as

$$\mathbf{z}(k) = \mathbf{h}\left(\mathbf{x}(k), \mathbf{w}(k)\right). \tag{131}$$

Like the Kalman filter, it is assumed that the noises $\mathbf{v}(k)$ and $\mathbf{w}(k)$ are all Gaussian, temporally uncorrelated and zero-mean with known variance as defined in Equations 77–79. The EKF aims to minimise mean-squared error and therefore compute an approximation to the conditional mean. It is assumed therefore that an estimate of the state at time $k-1$ is available which is approximately equal to the conditional mean,

$$\hat{\mathbf{x}}(k-1 \mid k-1) \approx \mathrm{E}\{\mathbf{x}(k-1) \mid Z^{k-1}\}\ . \tag{132}$$

The extended Kalman filter algorithm will now be stated without proof. Detailed derivations may be found in any number of books on the subject. The principle stages in the derivation of the EKF follow directly from those of the linear Kalman filter with additional step that the process and observation models are linearised as a Taylor's series about the estimate and prediction respectively. The algorithm has two stages:

Prediction: A prediction $\hat{\mathbf{x}}(k \mid k-1)$ of the state at time k and its covariance $\mathbf{P}(k \mid k-1)$ is computed according to

$$\hat{\mathbf{x}}(k \mid k-1) = \mathbf{f}\left(\hat{\mathbf{x}}(k-1 \mid k-1), \mathbf{u}(k)\right) \tag{133}$$

$$\mathbf{P}(k \mid k-1) = \nabla\mathbf{f}_{\mathbf{x}}(k)\mathbf{P}(k-1 \mid k-1)\nabla^T\mathbf{f}_{\mathbf{x}}(k) + \nabla\mathbf{f}_{\mathbf{v}}(k)\mathbf{Q}(k)\nabla^T\mathbf{f}_{\mathbf{v}}(k) \tag{134}$$

Update: At time k an observation $\mathbf{z}(k)$ is made and the updated estimate $\hat{\mathbf{x}}(k \mid k)$ of the state $\mathbf{x}(k)$, together with the updated estimate covariance $\mathbf{P}(k \mid k)$ is computed from the state prediction and observation according to

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\left[\mathbf{z}(k) - \mathbf{h}(\hat{\mathbf{x}}(k \mid k-1))\right] \tag{135}$$

$$\mathbf{P}(k \mid k) = \mathbf{P}(k \mid k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^T(k) \tag{136}$$

where

$$\mathbf{W}(k) = \mathbf{P}(k \mid k-1)\nabla^T\mathbf{h}_{\mathbf{x}}(k)\mathbf{S}^{-1}(k) \tag{137}$$

and

$$\mathbf{S}(k) = \nabla\mathbf{h}_{\mathbf{x}}(k)\mathbf{P}(k \mid k-1)\nabla^T\mathbf{h}_{\mathbf{x}}(k) + \nabla\mathbf{h}_{\mathbf{w}}(k)\mathbf{R}(k)\nabla^T\mathbf{h}_{\mathbf{w}}(k). \tag{138}$$

and where the Jacobian $\nabla\mathbf{f}_{\cdot}(k)$ is evaluated at $\mathbf{x}(k-1) = \hat{\mathbf{x}}(k-1 \mid k-1)$ and $\nabla\mathbf{h}_{\cdot}(k)$ is evaluated at and $\mathbf{x}(k) = \hat{\mathbf{x}}(k \mid k-1)$.

A comparison of Equations 85–96 with Equations 133–138 makes it clear that the extended Kalman filter algorithm is very similar to the linear Kalman filter algorithm, with the substitutions $\mathbf{F}(k) \to \nabla\mathbf{f}_{\mathbf{x}}(k)$ and $\mathbf{H}(k) \to \nabla\mathbf{h}_{\mathbf{x}}(k)$ being made in the equations for the variance and gain propagation. Thus, the extended Kalman filter is, in effect, a linear estimator for a state error which is described by a *linear* equation and which is being observed according to a *linear* equation of the form of Equation 76.

The extended Kalman filter works in much the same way as the linear Kalman filter with some notable caveats:

- The Jacobians $\nabla\mathbf{f}_\mathbf{x}(k)$ and $\nabla\mathbf{h}_\mathbf{x}(k)$ are typically not constant, being functions of both state and timestep. This means that unlike the linear filter, the covariances and gain matrix must be computed on-line as estimates and predictions are made available, and will not in general tend to constant values. This significantly increase the amount of computation which must be performed on-line by the algorithm.
- As the linearised model is derived by perturbing the true state and observation models around a predicted or nominal trajectory, great care must be taken to ensure that these predictions are always 'close enough' to the true state that second order terms in the linearisation are indeed insignificant. If the nominal trajectory is too far away from the true trajectory then the true covariance will be much larger than the estimated covariance and the filter will become poorly matched. In extreme cases the filter may also become unstable.
- The extended Kalman filter employs a linearised model which must be computed from an approximate knowledge of the state. Unlike the linear algorithm, this means that the filter must be accurately initialized at the start of operation to ensure that the linearised models obtained are valid. If this is not done, the estimates computed by the filter will simply be meaningless.

3.2 The Multi-Sensor Kalman Filter

Many of the techniques developed for single sensor Kalman filters can be applied directly to multi-sensor estimation and tracking problems. In principle, a group of sensors can be considered as a single sensor with a large and possibly complex observation model. In this case the Kalman filter algorithm is directly applicable to the multi-sensor estimation problem. However, as will be seen, this approach is practically limited to relatively small numbers of sensors.

A second approach is to consider each observation made by each sensor as a separate and independent realization, made according to a specific observation model, which can be incorporate into the estimate in a sequential manner. Again, single-sensor estimation techniques, applied sequentially, can be applied to this formulation of the multi-sensor estimation problem. However, as will be seen, this approach requires that a new prediction and gain matrix be calculated for each observation from each sensor at every time-step, and so is computationally very expensive.

A third approach is to explicitly derive equations for integrating multiple observations made at the same time into a common state estimate. Starting from the formulation of the multi-sensor Kalman filter algorithm, employing a single model for a group of sensors, a set of recursive equations for integrating individual sensor observations can be derived. As will be seen, these equations are more naturally expressed in terms of 'information' rather than state and covariance.

The systems considered to this point are all 'centralized'; the observations made by sensors are reported back to a central processing unit in a raw form where they are processed by a single algorithm in much the same way as single sensor systems. It is also possible to formulate the multi-sensor estimation problem in terms of a number of local sensor filters, each generating state estimates, which are subsequently communicated in processed form back to a central fusion centre. This distributed processing structure has a number of advantages in terms of modularity of the resulting architecture. However, the algorithms required to fuse estimate or track information at the central site can be quite complex.

In this section, the multi-sensor estimation problem is first defined in terms of a set of observation models and a single common process model. Each of the three centralised processing algorithms described above will be developed and compared. techniques described above; deriving appropriate equations and developing simple examples. The following section will then show how these multi-sensor estimation algorithms are applied in simple tracking problems.

Example 22

This example introduces a fundamental tracking problem which is further developed in the remainder of this section. The problem consists of the tracking of a number of targets from a number of tracking stations. The simulated target models and observations are non-linear, while the tracking algorithms used at the sensor sites are linear. This is fairly typical of actual tracking situations.

The targets to be tracked are modeled as 2-dimensional platforms with controllable heading and velocity. The continuous model is given by:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\phi}(t) \end{bmatrix} = \begin{bmatrix} V(t)\cos(\phi+\gamma) \\ V(t)\sin(\phi+\gamma) \\ \frac{V(t)}{\kappa}\sin(\gamma) \end{bmatrix}$$

where $(x(t), y(t))$ is the target position, $\phi(t)$ is the target orientation, $V(t)$ and $\gamma(t)$ are the platform velocity and heading, and κ is a constant minimum instantaneous turn radius for the target. Then $\mathbf{x}(t) = [x(t), y(t), \phi(t)]^T$ is defined as the state of the target and $\mathbf{u}(t) = [V(t), \gamma(t)]^T$ as the target input.

This model can be converted into a discrete time model using a constant (Euler) integration rule over the (asynchronous) sample interval $\Delta T_k = t_k - t_{k-1}$ as

$$\begin{bmatrix} x(k) \\ y(k) \\ \phi(k) \end{bmatrix} = \begin{bmatrix} x(k-1) + \Delta T V(k)\cos(\phi(k-1)+\gamma(k)) \\ y(k-1) + \Delta T V(k)\sin(\phi(k-1)+\gamma(k)) \\ \Delta T\phi(k-1) + \frac{V(k)}{\kappa}\sin(\gamma(k)) \end{bmatrix}$$

which is in the standard form $\mathbf{x}(k) = \mathbf{f}(\mathbf{x}(k-1), \mathbf{u}(k))$. For the purposes of simulation, the model is assumed driven by Brownian models for velocity and heading,

$$\dot{V}(t) = V(t) + v_V(t), \qquad \dot{\gamma}(t) = \gamma(t) + v_\gamma(t),$$

where $\mathbf{v}(t) = [v_V(t), v_\gamma(t)]^T$ is a zero mean white driving sequence with strength

$$\mathrm{E}\{\mathbf{v}(t)\mathbf{v}^T(t)\} = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\gamma^2 \end{bmatrix}$$

The discrete-time model for this is simply

$$V(k) = V(k-1) + \Delta T_k v_V(k), \qquad \gamma(k) = \gamma(k-1) + \Delta T_k v_\gamma(k).$$

A group of typical trajectories generated by this model is shown in Figure 19(a).

The targets are observed using sensors (tracking stations) $i = 1, \cdots, N$ whose location and pointing angles $\mathbf{T}_i(t) = [X_i(t), Y_i(t), \psi_i(t)]^T$ are known. This does not preclude the sensors from actually being in motion. Each sensor site i is assumed to make range and bearing observations to the j^{th} targets as

$$\begin{bmatrix} z^r_{ij}(k) \\ z^\theta_{ij}(k) \end{bmatrix} = \begin{bmatrix} \sqrt{(x_j(k) - X_i(k))^2 + (y_j(k) - Y_i(k))^2} \\ \arctan\left(\frac{y_j(k)-Y_i(k)}{x_j(k)-X_i(k)}\right) - \psi_i(k) \end{bmatrix} + \begin{bmatrix} w^r_{ij}(k) \\ w^\theta_{ij}(k) \end{bmatrix},$$

where the random vector $\mathbf{w}_{ij}(k) = [r^r_{ij}(k), r^\theta_{ij}(k)]^T$ describes the noise in the observation process due to both modeling errors and uncertainty in observation. Observation noise errors are taken to be zero mean and white with constant variance

$$\mathrm{E}\{\mathbf{w}_{ij}(k)\mathbf{w}^T_{ij}(k)\} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}.$$

A typical set of observations generated by this model for a track is shown in Figure 19(b).

3.2.1 Observation Models

Figure 20 shows the centralised data fusion architecture developed in the following three sections. A common model of the true state is provided in the usual linear discrete-time form;

$$\mathbf{x}(k) = \mathbf{F}(k)\mathbf{x}(k-1) + \mathbf{G}(k)\mathbf{u}(k) + \mathbf{v}(k), \tag{139}$$

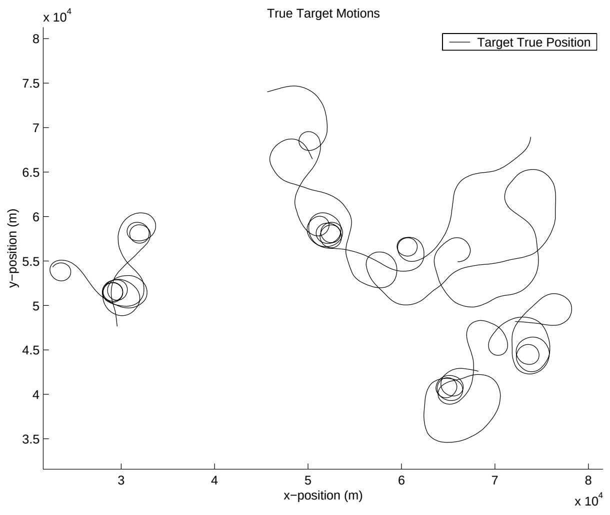
where $\mathbf{x}(\cdot)$ is the state of interest, $\mathbf{F}(k)$ is the state transition matrix, $\mathbf{G}(k)$the control input model, $\mathbf{u}(k)$ the control input vector, and $\mathbf{v}(k)$ a random vector describing model and process uncertainty, assumed zero mean and temporally uncorrelated;

$$\begin{aligned} \mathrm{E}\{\mathbf{v}(k)\} &= \mathbf{0}, \qquad \forall k, \\ \mathrm{E}\{\mathbf{v}(i)\mathbf{v}^T(j)\} &= \delta_{ij}\mathbf{Q}(i). \end{aligned} \tag{140}$$

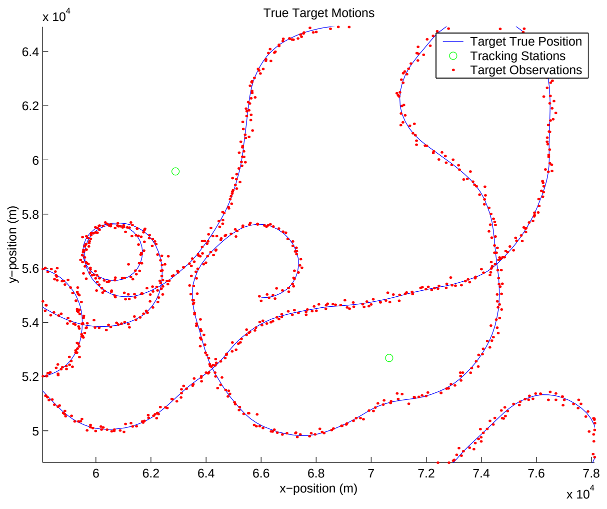
It is important to emphasise that, because all sensors are observing the same state (there would be little point in the data fusion problem otherwise) this process model must be *common* to all sensors.

Observations of the state of this system are made synchronously by a number of different sensors according to a set of linear observation models in the form

$$\mathbf{z}_s(k) = \mathbf{H}_s(k)\mathbf{x}(k) + \mathbf{w}_s(k), \qquad s = 1, \cdots, S. \tag{141}$$



(a)



(b)

Figure 19: Typical tracks and observations generated by the 'standard' multi-target tracking example of Example 22: (a) true x–y tracks; (b) detail of track observations.

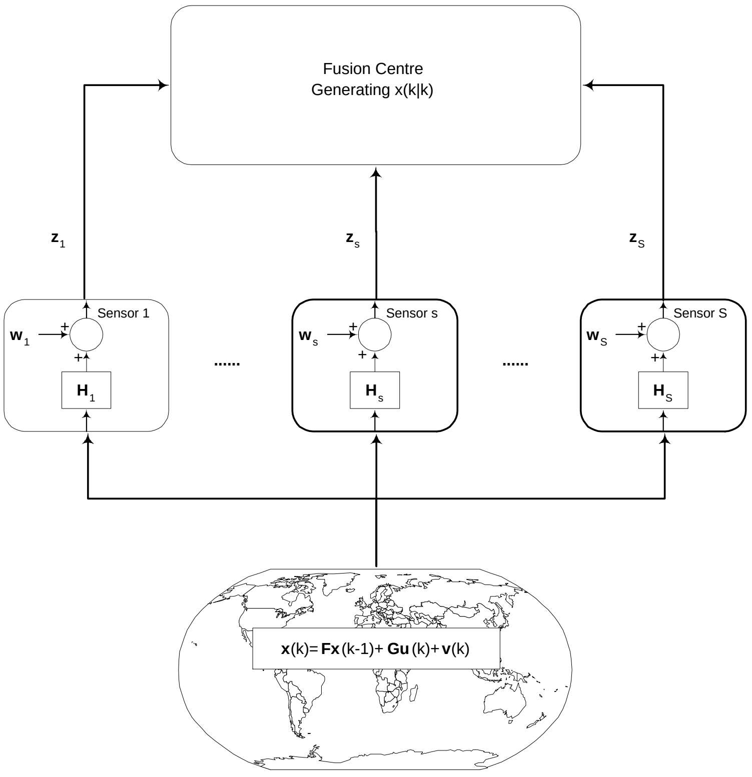


Figure 20: The structure of a centralised data fusion architecture. Each sensor has a different observation model but observes a common process. Observations are sent in unprocessed form to a central fusion center that combines observations to provide an estimate of the common state.

where $\mathbf{z}_s(k)$ is the observation made at time k by sensor s of a common state $\mathbf{x}(k)$ according to the observation model $\mathbf{H}_s(k)$ in additive noise $\mathbf{w}_s(k)$. The case in which the observations made by the sensors are asynchronous, in which different sensors make observations at different rates, can be dealt with by using the observation model $\mathbf{H}_s(t_k)$.

It is assumed that the observation noise models $\mathbf{w}_s(k)$ are all zero mean, uncorrelated between sensors and also temporally uncorrelated;

$$\begin{aligned} \mathrm{E}\{\mathbf{w}_s(k)\} &= \mathbf{0}, \qquad s = 1, \cdots, S, \quad \forall k \\ \mathrm{E}\{\mathbf{w}_p(i)\mathbf{w}_q(j)\} &= \delta_{ij}\delta_{pq}\mathbf{R}_p(i). \end{aligned} \tag{142}$$

It is also assumed that the process and observation noises are uncorrelated

$$\mathrm{E}\{\mathbf{v}(i)\mathbf{w}_s^T(j)\} = \mathbf{0}, \qquad \forall i, j, s. \tag{143}$$

This assumption is not absolutely necessary. It is relatively simple, but algebraically complex, to include a term accounting for correlated process and observation errors (see [12] Chapter 7, or [21].)

3.2.2 The Group-Sensor Method

The simplest way of dealing with a multi-sensor estimation problem is to combine all observations and observation models in to a single composite 'group sensor' and then to deal with the estimation problem using an identical algorithm to that employed in single-sensor systems. Defining a composite observation vector by

$$\mathbf{z}(k) \triangleq \left[\mathbf{z}_1^T(k), \cdots, \mathbf{z}_S^T(k)\right]^T, \tag{144}$$

and a composite observation model by

$$\mathbf{H}(k) \triangleq \left[\mathbf{H}_1^T(k), \cdots, \mathbf{H}_s^T(k)]\right]^T, \tag{145}$$

with

$$\mathbf{w}(k) \triangleq \left[\mathbf{w}_1^T(k), \cdots, \mathbf{w}_s^T(k)\right]^T, \tag{146}$$

where from Equation 142

$$\begin{aligned}\mathbf{R}(k) &= \mathrm{E}\{\mathbf{w}(k)\mathbf{w}^T(k)\} \\ &= \mathrm{E}\{\left[\mathbf{w}_1^T(k), \cdots, \mathbf{w}_s^T(k)\right]^T \left[\mathbf{w}_1^T(k), \cdots, \mathbf{w}_s^T(k)\right]\} \\ &= \mathrm{blockdiag}\{\mathbf{R}_1(k), \cdots, \mathbf{R}_s(k)\},\end{aligned} \tag{147}$$

the observation noise covariance is block-diagonal with blocks equal to the individual sensor observation noise covariance matrices. The set of observation equations defined by Equation 141 may now be rewritten as a single 'group' observation model in standard form

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{w}(k). \tag{148}$$

With a process model described by Equation 139 and a group observation model defined by Equation 148, estimates of state can in principle be computed using the standard Kalman filter algorithm given by Equations 94, 95, and 96.

Example 23

Consider again the tracking of a particle moving with constant velocity with process model as defined in Example 17. Suppose we have two sensors, the first observing the position and the second observing the velocity of the particle. The two observation models will be

$$z_1 = [\,1 \quad 0\,]\begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + w_1(k), \qquad \mathrm{E}\{w_1^2(k)\} = \sigma_{r_1}^2,$$

and

$$z_2 = [\,0 \quad 1\,]\begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + w_2(k), \qquad \mathrm{E}\{w_2^2(k)\} = \sigma_{r_2}^2.$$

The composite group sensor model is simply given by

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \qquad \mathrm{E}\{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}[\,w_1 \quad w_2\,]\} = \begin{bmatrix} \sigma_{r_1}^2 & 0 \\ 0 & \sigma_{r_2}^2 \end{bmatrix}.$$

We could add a third sensor making additional measurements of position according to the observation model

$$z_3 = [\,1 \quad 0\,] \begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + w_3(k), \qquad \mathrm{E}\{w_3^2(k)\} \; = \sigma_{r3}^2,$$

in which case the new composite group sensor model will be given by

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix},$$

$$\mathrm{E}\{ \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} [\,w_1 \quad w_2 \quad w_3\,]\} \; = \begin{bmatrix} \sigma_{r_1}^2 & 0 & 0 \\ 0 & \sigma_{r_2}^2 & 0 \\ 0 & 0 & \sigma_{r_3}^2 \end{bmatrix}.$$

It should be clear that there is no objection in principle to incorporating as many sensors as desired in this formulation of the multi-sensor estimation problem.

The prediction phase of the multi-sensor Kalman filter algorithm makes no reference to the observations that are made and so is identical in every respect to the prediction phase of the single-sensor filter. However, the update phase of the cycle, which incorporates measurement information from sensors, will clearly be affected by an increase in the number of sensors. Specifically, if we have a state vector $\mathbf{x}(k)$ of dimension n and S sensors each with an observation vector $\mathbf{z}_s(k), s = 1, \cdots, S$ of dimension m_s together with an observation model $\mathbf{H}_s(k)$ of dimension $m_s \times n$ then the group observation vector $\mathbf{z}(k)$ will have dimension $\mathbf{m} = \sum_{s=1}^{S} m_s$ and the group observation model $\mathbf{H}(k)$ will have dimension $\mathbf{m} \times n$. The consequence of this lies in Equations 94, 95 and 96. Clearly the group-sensor innovation $\nu(k) \triangleq \left[\nu_1^T(k), \cdots, \nu_1^T(k)\right]^T$ will now have dimension $\mathbf{m}$, and the group-sensor innovation covariance matrix $\mathbf{S}(k)$ will have dimension $\mathbf{m} \times \mathbf{m}$. As the number of sensors incorporated in the group sensor model increases, so does the dimension of the innovation vector and innovation covariance matrix. This is a problem because the *inverse* of the innovation covariance is required to compute the group-sensor gain matrix $\mathbf{W}(k)$ in Equation 96. It is well known that the calculation of a matrix inverse increase in proportion to the square of its dimension.

For a small number of sensors and so for a relatively small innovation dimension, the group-sensor approach to multi-sensor estimation may be the most practical to implement. However, as the number of sensors increases, the value of this monolithic approach to the data fusion problem becomes limited.

Example 24

It is common practice to use linear models to track targets that are clearly not linear, particularly in multiple-target tracking problems. Consider again the tracks and observations generated by Example 22. The range and bearing observation vector $\mathbf{z}_{ij}(k) =$

$[z^r_{ij}(k), z^\theta_{ij}(k)]^T$ *can be converted into an equivalent observation vector in absolute cartesian coordinates as*

$$\begin{bmatrix} z^x_{ij}(k) \\ z^y_{ij}(k) \end{bmatrix} = \begin{bmatrix} X_i(k) + z^r_{ij}(k)\cos z^\theta_{ij}(k) \\ Y_i(k) + z^r_{ij}(k)\sin z^\theta_{ij}(k) \end{bmatrix}.$$

The observation covariance matrix can also be converted into absolute cartesian coordinates using the relationship

$$\mathbf{R}_{xy}(k) = \begin{bmatrix} \sigma^2_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma^2_{yy} \end{bmatrix} = Rot(z^\theta_{ij}(k)) \begin{bmatrix} \sigma^2_r & 0 \\ 0 & (z^r_{ij}(k))^2\sigma^2_\theta \end{bmatrix} Rot^T(z^\theta_{ij}(k))$$

where $Rot(\theta)$ is the rotation matrix

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

Note now that observation variance is strongly range dependent and is lined up with the sensor bore-sight.

Once the observation vector has been converted into a global cartesian coordinate frame, a linear filter can be used to estimate a linear target track. The two-dimensional particle model of Example 17 with process model defined in Equation 80 and with an observation model (now not constant) defined in Equation 81, can be used in the filter defined by Equations 87–89 in Example 18.

Figure 21 shows the results of tracking four targets from two (stationary) sensor sites. First note that the state variances and innovation variances are not constant because the observation variances are strongly dependent on the relative position of observer and target. The asynchronous nature of the observations also contributes to this non-constancy. It can be observed in Figure 21(c) and (d) the rising and falling as variance values as the target shown comes closer to and then further away from the tracking sites.

The algorithm implemented here is equivalent to the group sensor method, but the results will also be the same for the sequential and inverse covariance algorithms.

3.2.3 The Sequential-Sensor Method

A second approach to the multi-sensor estimation problem is to consider each sensor observation as an independent, sequential update to the state estimate and for each observation to compute an appropriate prediction and gain matrix. In contrast to the group-sensor approach in which a single sensor model was constructed by combining all sensor models, the sequential update approach considers each sensor model individually, one by one. This means that the dimension of the innovation and innovation covariance matrix at each update stage remains the same size as their single-sensor equivalents at the cost of computing a new gain matrix for each observation from each sensor.

The description of the state to be estimated is assumed in the form of Equation 139. Observations are made of this common state by S sensors according to Equation 141. It

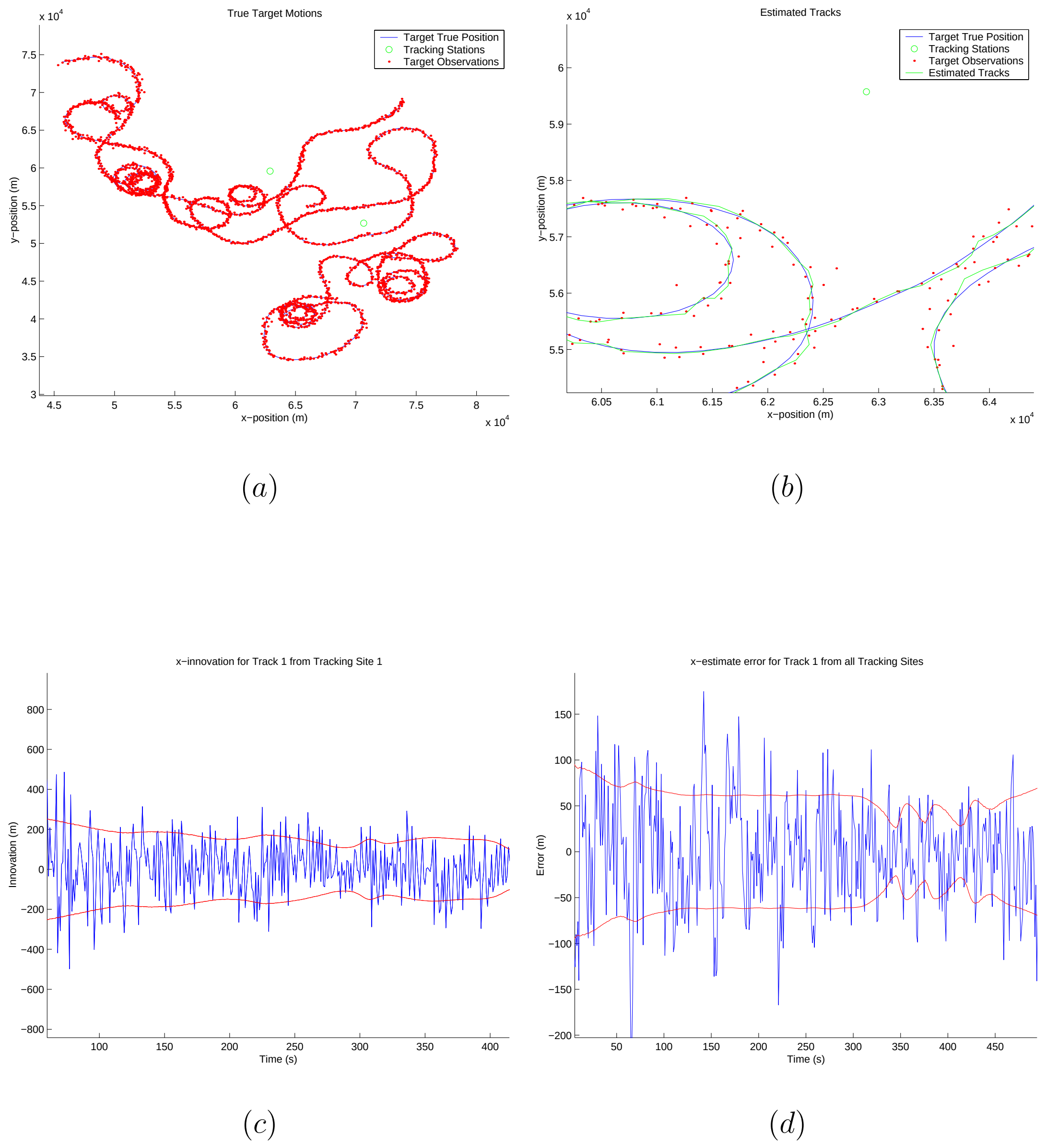


Figure 21: A multiple sensor multiple target tracking example with four targets and two tracking stations: (a) True state and asynchronous observations; (b) Detail of true states, track estimates, and observations; (c) Innovations and innovation standard deviations for a particular track and tracking station; (d) Track position estimates from all sites together with standard deviations (estimated errors).

is assumed that every sensor takes an observation synchronously at every time step. The case of asynchronous observations is straight-forward in this sequential-sensor algorithm.

It is assumed that it is possible to consider the observations made by the sensors at any one time-step in a specific but otherwise arbitrary order so that the observation $\mathbf{z}_p(k)$, $1 \leq p \leq S$, from the p^{th} sensor will be before the observation $\mathbf{z}_{p+1}(k)$ from the $(p+1)^{th}$. The set of observations made by the first p sensors at time k is denoted by (caligraphic notation is used to denote sets across sensors compared to bold-face notation for sets associated with a single sensor)

$$\mathcal{Z}_p(k) \stackrel{\triangle}{=} \{\mathbf{z}_1(k), \cdots, \mathbf{z}_p(k)\} , \tag{149}$$

so that at every time-step the observation set $\mathcal{Z}_S(k)$ is available to construct a new state estimate. The set of all observations made by the p^{th} sensor up to time k will be denoted by

$$\mathbf{Z}_p^k \stackrel{\triangle}{=} \{\mathbf{z}_p(1), \cdots, \mathbf{z}_p(k)\} , \tag{150}$$

and the set of all observations made by the first p sensors up to time k by

$$\mathcal{Z}_p^k \stackrel{\triangle}{=} \{\mathbf{Z}_1^k, \cdots, \mathbf{Z}_p^k\} , \qquad p \leq S. \tag{151}$$

Of particular interest is the set of observations consisting of all observations made by the first p sensors up to a time i and all observations made by the first q sensors up to a time j

$$\mathcal{Z}_{p,q}^{i,j} = \{\mathcal{Z}_p^i \cup \mathcal{Z}_q^j\} \ = \{\mathbf{Z}_1^i, \cdots, \mathbf{Z}_p^i, \mathbf{Z}_{p+1}^j, \cdots, \mathbf{Z}_q^j\} , \qquad p < q, i \geq j. \tag{152}$$

and in particular the set consisting of all observations made by all sensors up to a time $k-1$ together with the observations made by the first p sensors up to time k

$$\begin{aligned} \mathcal{Z}_{p,S}^{k,k-1} &= \{\mathcal{Z}_p^k \cup \mathcal{Z}_S^{k-1}\} \\ &= \{\mathbf{Z}_1^k, \cdots, \mathbf{Z}_p^k, \mathbf{Z}_{p+1}^{k-1}, \cdots, \mathbf{Z}_S^{k-1}\} \\ &= \{\mathcal{Z}_p(k) \cup \mathcal{Z}_S^{k-1}\} , \qquad p < S. \end{aligned} \tag{153}$$

The estimate constructed at each time-step on the basis of all observations from all sensors up to time $k-1$ and on the observations made by the first p sensors up to time k is defined by

$$\hat{\mathbf{x}}(k \mid k, p) = \mathrm{E}\{\mathbf{x}(k) \mid \mathcal{Z}_{p,S}^{k,k-1}\} , \tag{154}$$

where in particular

$$\hat{\mathbf{x}}(k \mid k, 0) = \hat{\mathbf{x}}(k \mid k-1) = \mathrm{E}\{\mathbf{x}(k) \mid \mathcal{Z}_S^{k-1}\} , \tag{155}$$

is the prediction of the state at time k before any observations are made and

$$\hat{\mathbf{x}}(k \mid k, S) = \hat{\mathbf{x}}(k \mid k) = \mathrm{E}\{\mathbf{x}(k) \mid \mathcal{Z}_S^k\} , \tag{156}$$

is the estimate at time k based on the observations made by all sensors.

It is assumed that Equations 140, 142, and 143 hold. For the first observation considered at any one time-step, the prediction stage for the sequential-sensor estimator is very similar to the prediction stage for the single-sensor estimator. The state prediction is simply

$$\begin{aligned} \hat{\mathbf{x}}(k \mid k, 0) &= \hat{\mathbf{x}}(k \mid k-1) \\ &= \mathbf{F}(k)\hat{\mathbf{x}}(k-1 \mid k-1) + \mathbf{G}(k)\mathbf{u}(k) \qquad (157) \\ &= \mathbf{F}(k)\hat{\mathbf{x}}(k-1 \mid k-1, S) + \mathbf{G}(k)\mathbf{u}(k), \end{aligned}$$

with corresponding covariance

$$\begin{aligned} \mathbf{P}(k \mid k, 0) &= \mathbf{P}(k \mid k-1) \\ &= \mathbf{F}(k)\mathbf{P}(k-1 \mid k-1)\mathbf{F}^T(k) + \mathbf{Q}(k) \qquad (158) \\ &= \mathbf{F}(k)\mathbf{P}(k-1 \mid k-1, S)\mathbf{F}^T(k) + \mathbf{Q}(k). \end{aligned}$$

The new estimate found by integrating the observation $\mathbf{z}_1(k)$ made by the first sensor at time k is then simply given by

$$\hat{\mathbf{x}}(k \mid k, 1) = \hat{\mathbf{x}}(k \mid k, 0) + \mathbf{W}_1(k)\left[\mathbf{z}_1(k) - \mathbf{H}(i)k\hat{\mathbf{x}}(k \mid k, 0)\right] \qquad (159)$$

with covariance

$$\mathbf{P}(k \mid k, 1) = \mathbf{P}(k \mid k, 0) - \mathbf{W}_1(k)\mathbf{S}_1(k)\mathbf{W}_1^T(k), \qquad (160)$$

where

$$\mathbf{W}_1(k) = \mathbf{P}(k \mid k, 0)\mathbf{H}_1^T(k)\mathbf{S}_1^{-1}(k) \qquad (161)$$

and

$$\mathbf{S}_1(k) = \mathbf{H}_1(k)\mathbf{P}(k \mid k, 0)\mathbf{H}_1^T(k) + \mathbf{R}_1(k) \qquad (162)$$

To now integrate the observation made by the second sensor we need to employ the estimate $\hat{\mathbf{x}}(k \mid k, 1)$ to generate a prediction. However, if the observations made by the sensors are assumed synchronous, the state does not evolve between successive sensor readings from the same time-step and so $\mathbf{F}(k) = \mathbf{1}$ and $\mathbf{Q}(k) = \mathbf{0}$ (this is not true in the asynchronous case). This means that the estimate $\hat{\mathbf{x}}(k \mid k, 1)$ is itself the prediction for the next update. In general, the estimate of the state $\hat{\mathbf{x}}(k \mid k, p)$ at time k based on observations made by all sensors up to time $k-1$ and by the first p sensors up to time k can be computed from the estimate $\hat{\mathbf{x}}(k \mid k, p-1)$ as

$$\hat{\mathbf{x}}(k \mid k, p) = \hat{\mathbf{x}}(k \mid k, p-1) + \mathbf{W}_p(k)\left[\mathbf{z}_p(k) - \mathbf{H}_p(k)\hat{\mathbf{x}}(k \mid k, p-1)\right] \qquad (163)$$

with covariance

$$\mathbf{P}(k \mid k, p) = \mathbf{P}(k \mid k, p) - \mathbf{W}_p(k)\mathbf{S}_p(k)\mathbf{W}_p^T(k), \qquad (164)$$

where

$$\mathbf{W}_p(k) = \mathbf{P}(k \mid k, p-1)\mathbf{H}_p^T(k)\mathbf{S}_p^{-1}(k) \tag{165}$$

and

$$\mathbf{S}_p(k) = \mathbf{H}_p(k)\mathbf{P}(k \mid k, p-1)\mathbf{H}_p^T(k) + \mathbf{R}_p(k) \tag{166}$$

The state update at each time-step could be computed in a batch-mode by explicitly expanding Equation 163 to become

$$\begin{aligned} \hat{\mathbf{x}}(k \mid k) \;&=\; \left[\prod_{s=1}^{S} \left(\mathbf{1} - \mathbf{W}_i(k)\mathbf{H}_i(k)\right)\right] \hat{\mathbf{x}}(k \mid k-1) \\ &\quad + \sum_{i=1}^{S} \left[\prod_{j=i+1}^{S} \left(\mathbf{1} - \mathbf{W}_j(k)\mathbf{H}_j(k)\right)\right] \mathbf{W}_i(k)\mathbf{z}_i(k) \end{aligned} \tag{167}$$

The case in which the observations are not synchronous may be dealt with in a similar way to the single-sensor asynchronous case providing that due care is taken to generate a proper prediction at each time each sensor records an observation.

As with the group-sensor approach to multi-sensor estimation, this method works well for small numbers of sensors[8]. However the amount of computation that must be performed increases with an increase in the number of sensors as a new gain matrix must be computed for each observation from each sensor at each time step. Although this increase is linear (the dimension of the innovation covariance matrix which must be inverted does not increase), it may still become a problem when large numbers of sensors are employed.

3.2.4 The Inverse-Covariance Form

Neither the group-sensor nor the sequential-sensor algorithms are of much help when the number of sensors becomes large. In such cases it would be useful to find an explicit set of equations and provide an algorithm which would allow the direct integration of multiple observations into a single composite estimate.

The multi-sensor estimation problem would be considerably simplified if it were possible to write the estimate as a simple linear combination of the innovations and prediction in the standard Kalman filter form. Unfortunately

$$\hat{\mathbf{x}}(k \mid k) \neq \hat{\mathbf{x}}(k \mid k-1) + \sum_{i=1}^{S} \mathbf{W}_i(k)\left[\mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}(k \mid k-1)\right], \tag{168}$$

with

$$\mathbf{W}_i(k) = \mathbf{P}(k \mid k-1)\mathbf{H}_i^T(k)\mathbf{S}^{-1}(k), \qquad i = 1, \cdots, S,$$

and

$$\mathbf{S}_i(k) = \mathbf{H}_i(k)\mathbf{P}(k \mid k-1)\mathbf{H}_i^T(k) + \mathbf{R}_i(k), \qquad i = 1, \cdots, S.$$

[8]Indeed the sequential-sensor method is to be preferred when the observations are not synchronous as in the asynchronous case a new prediction and gain matrix must be computed for each new observation regardless.

If the equality in Equation 168 were to hold, the group-sensor gain matrix would need to be in block-diagonal form. For the gain-matrix to be block-diagonal, Equation 96 shows that the group-sensor innovation covariance matrix must also be in block-diagonal form. Unfortunately, this is never the case. For example with two sensors, the group-sensor observation vector is

$$\mathbf{z}(k) = \left[\mathbf{z}_1^T(k), \mathbf{z}_2^T(k)\right]^T,$$

the group-sensor observation matrix is

$$\mathbf{H}(k) = \left[\mathbf{H}_1^T(k), \mathbf{H}_2^T(k)\right]^T,$$

and the group-sensor observation noise covariance matrix

$$\mathbf{R}(k) = blockdiag\{\mathbf{R}_1(k), \mathbf{R}_2(k)\}.$$

The group sensor innovation covariance is thus

$$\begin{aligned}
\mathbf{S}(k) &= \mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}^T(k) + \mathbf{R}(k) \\
&= \begin{bmatrix} \mathbf{H}_1(k) \\ \mathbf{H}_2(k) \end{bmatrix} \mathbf{P}(k \mid k-1) \left[\, \mathbf{H}_1^T(k) \quad \mathbf{H}_2^T(k) \,\right] + \begin{bmatrix} \mathbf{R}_1(k) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_2(k) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{H}_1(k)\mathbf{P}(k \mid k-1)\mathbf{H}_1^T(k) + \mathbf{R}_1(k) \\ \mathbf{H}_2(k)\mathbf{P}(k \mid k-1)\mathbf{H}_1^T(k)\mathbf{H}_1(k)\mathbf{P}(k \mid k-1)\mathbf{H}_2^T(k) \\ \mathbf{H}_2(k)\mathbf{P}(k \mid k-1)\mathbf{H}_2^T(k) + \mathbf{R}_1(k) \end{bmatrix},
\end{aligned} \tag{169}$$

which is clearly not block-diagonal; so the gain matrix

$$\mathbf{W}(k) = \mathbf{P}(k \mid k-1)\mathbf{H}^T(k)\mathbf{S}^{-1}(k)$$

will also not be block-diagonal.

Fundamentally, the reason why Equation 168 may be used to integrate single-sensor observations over time but not for integrating multiple observations at a single time-step is that the innovations are correlated[9]. This correlation is due to the fact that the innovations at a single time-step have a common prediction. This correlation is reflected in the off-diagonal terms of Equation 169 of the form $\mathbf{H}_i(k)\mathbf{P}(k \mid k-1)\mathbf{H}_j^T(k)$.

The fact that the innovations generated by each sensor are correlated results in significant problems in the implementation of multi-sensor estimation algorithms. Ignoring correlations between innovations (assuming equality in Equation 168 and constructing an estimate from a simple weighted sum of innovations) will result in disaster. This is because the information due to the prediction will be 'double counted' in each update thus implying considerably more information (and confidence) than is actually the case. The use of 'forgetting factors' or 'fading memory filters' are routinely used to address this issue, although these are really no more than domain-specific hacks. It should also be pointed out that the correlatedness of innovations in multi-sensor estimation exposes the popular fallacy that it is always 'cheaper' to communicate innovations in such algorithms.

[9]the innovations from one time-step to the next are uncorrelated but the innovations generated by many sensors at a single time are correlated.

Example 25

For a two-sensor system we can explicitly evaluate the combined innovation covariance matrix using the matrix inversion lemma and compute the appropriate gain matrices for the innovations of both sensors. Dropping all time subscripts, the inverse innovation covariance may be written as [24]

$$\mathbf{S}^{-1} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{12}^T & \mathbf{S}_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{\Delta}^{-1} & -\mathbf{\Delta}^{-1}\mathbf{S}_{12}\mathbf{S}_{22}^{-1} \\ -\mathbf{S}_{22}^{-1}\mathbf{S}_{12}^T\mathbf{\Delta}^{-1} & \mathbf{S}_{22}^{-1} + \mathbf{S}_{22}^{-1}\mathbf{S}_{12}^T\mathbf{\Delta}^{-1}\mathbf{S}_{12}\mathbf{S}_{22}^{-1} \end{bmatrix}, \tag{170}$$

where

$$\mathbf{\Delta} = \mathbf{S}_{11} - \mathbf{S}_{12}\mathbf{S}_{22}^{-1}\mathbf{S}_{12}^T.$$

Making appropriate substitutions from Equation 169, and employing the matrix inversion lemma twice, we have

$$\begin{aligned}
\mathbf{\Delta}^{-1} &= \left[\mathbf{S}_{11} - \mathbf{S}_{12}\mathbf{S}_{22}^{-1}\mathbf{S}_{12}^T\right]^{-1} \\
&= \left[\mathbf{H}_1\mathbf{P}\mathbf{H}_1^T + \mathbf{R}_1 - \mathbf{H}_1\mathbf{P}\mathbf{H}_2^T\left[\mathbf{H}_2\mathbf{P}\mathbf{H}_2^T + \mathbf{R}_2\right]^{-1}\mathbf{H}_2\mathbf{P}\mathbf{H}_1^T\right]^{-1} \\
&= \left[\mathbf{R}_1 + \mathbf{H}_1\left(\mathbf{P} - \mathbf{P}\mathbf{H}_2^T\left[\mathbf{H}_2\mathbf{P}\mathbf{H}_2^T + \mathbf{R}_2\right]^{-1}\mathbf{H}_2\mathbf{P}\right)\mathbf{H}_1^T\right]^{-1} \\
&= \left[\mathbf{R}_1 + \mathbf{H}_1\left[\mathbf{P}^{-1} + \mathbf{H}_2^T\mathbf{R}_2^{-1}\mathbf{H}_2\right]^{-1}\mathbf{H}_1^T\right]^{-1} \\
&= \mathbf{R}_1^{-1} - \mathbf{R}_1^{-1}\mathbf{H}_1\left[\mathbf{P}^{-1} + \mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1 + \mathbf{H}_2^T\mathbf{R}_2^{-1}\mathbf{H}_2\right]^{-1}\mathbf{H}_1^T\mathbf{R}_1^{-1}.
\end{aligned}$$

The two gain matrices may now be computed from

$$[\,\mathbf{W}_1 \quad \mathbf{W}_2\,] = \mathbf{P}\begin{bmatrix} \mathbf{H}_1^T \\ \mathbf{H}_2^T \end{bmatrix}\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{12}^T & \mathbf{S}_{22} \end{bmatrix}^{-1}.$$

Substituting in from Equations 169 and 170 we have for the first gain

$$\begin{aligned}
\mathbf{W}_1 &= \mathbf{P}\mathbf{H}_1^T\mathbf{\Delta}^{-1} - \mathbf{P}\mathbf{H}_2^T\left[\mathbf{H}_2\mathbf{P}\mathbf{H}_2^T + \mathbf{R}_2\right]^{-1}\mathbf{H}_2\mathbf{P}\mathbf{H}_1^T\mathbf{\Delta}^{-1} \\
&= \left(\mathbf{P} - \mathbf{P}\mathbf{H}_2^T\left[\mathbf{H}_2\mathbf{P}\mathbf{H}_2^T + \mathbf{R}_2\right]^{-1}\mathbf{H}_2\mathbf{P}\right)\mathbf{H}_1^T\mathbf{\Delta}^{-1} \\
&= \left[\mathbf{P}^{-1} + \mathbf{H}_2^T\mathbf{R}_2^{-1}\mathbf{H}_2\right]^{-1}\mathbf{H}_1^T\mathbf{\Delta}^{-1} \\
&= \left[\mathbf{P}^{-1} + \mathbf{H}_2^T\mathbf{R}_2^{-1}\mathbf{H}_2\right]^{-1} \\
&\quad \times\left(\mathbf{1} - \mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1\left[\mathbf{P}^{-1} + \mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1 + \mathbf{H}_2^T\mathbf{R}_2^{-1}\mathbf{H}_2\right]^{-1}\right)\mathbf{H}_1^T\mathbf{R}_1^{-1} \\
&= \left[\mathbf{P}^{-1} + \mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1 + \mathbf{H}_2^T\mathbf{R}_2^{-1}\mathbf{H}_2\right]^{-1}\mathbf{H}_1^T\mathbf{R}_1^{-1},
\end{aligned}$$

and similarly for the second gain

$$\mathbf{W}_2 = \left[\mathbf{P}^{-1} + \mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1 + \mathbf{H}_2^T\mathbf{R}_2^{-1}\mathbf{H}_2\right]^{-1}\mathbf{H}_2^T\mathbf{R}_2^{-1}$$

To derive a set of explicit equations for multi-sensor estimation problems, we could begin by employing the matrix inversion lemma to invert the innovation matrix for the two-sensor case given in Equation 169, and then proceed to simplify the equation for the group-sensor gain matrix. However, it is easier in the first instance to write the gain and update equations for the group-sensor system in *inverse covariance form*.
Rewriting the weights associated with the prediction and observation.

$$\begin{aligned}\mathbf{I} - \mathbf{W}(k)\mathbf{H}(k) &= \left[\mathbf{P}(k \mid k-1) - \mathbf{W}(k)\mathbf{H}(k)\mathbf{P}(k \mid k-1)\right]\mathbf{P}^{-1}(k \mid k-1) \\ &= \left[\mathbf{P}(k \mid k-1) - \mathbf{W}(k)\mathbf{S}(k)\left(\mathbf{S}^{-1}(k)\mathbf{H}(k)\mathbf{P}(k \mid k-1)\right)\right] \\ &\quad \times\mathbf{P}^{-1}(k \mid k-1) \\ &= \left[\mathbf{P}(k \mid k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^T(k)\right]\mathbf{P}^{-1}(k \mid k-1) \\ &= \mathbf{P}(k \mid k)\mathbf{P}^{-1}(k \mid k-1).\end{aligned} \tag{171}$$

Similarly,

$$\mathbf{W}(k) = \mathbf{P}(k \mid k-1)\mathbf{H}^T(k)\left[\mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}^T(k) + \mathbf{R}^{-1}(k)\right]^{-1}$$

$$\mathbf{W}(k)\left[\mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}^T(k) + \mathbf{R}^{-1}(k)\right] = \mathbf{P}(k \mid k-1)\mathbf{H}^T(k)$$

$$\mathbf{W}(k)\mathbf{R}(k) = \left[\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k)\right]\mathbf{P}(k \mid k-1)\mathbf{H}^T(k)$$

so

$$\mathbf{W}(k) = \mathbf{P}(k \mid k)\mathbf{H}^T(k)\mathbf{R}^{-1}(k). \tag{172}$$

Substituting Equations 171 and 172 into Equation 94 gives the state update equation as

$$\hat{\mathbf{x}}(k \mid k) = \mathbf{P}(k \mid k)\left[\mathbf{P}^{-1}(k \mid k-1)\hat{\mathbf{x}}(k \mid k-1) + \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k)\right]. \tag{173}$$

From Equations 95, 96 and 171 we have

$$\mathbf{P}(k \mid k) = \left[\mathbf{I} - \mathbf{W}(k)\mathbf{H}(k)\right]\mathbf{P}(k \mid k-1)\left[\mathbf{I} - \mathbf{W}(k)\mathbf{H}(k)\right]^T + \mathbf{W}(k)\mathbf{R}(k)\mathbf{W}^T(k). \tag{174}$$

Substituting in Equations 171 and 172 gives

$$\begin{aligned}\mathbf{P}(k \mid k) &= \left[\mathbf{P}(k \mid k)\mathbf{P}^{-1}(k \mid k-1)\right]\mathbf{P}(k \mid k-1)\left[\mathbf{P}(k \mid k)\mathbf{P}^{-1}(k \mid k-1)\right]^T \\ &\quad + \left[\mathbf{P}(k \mid k)\mathbf{H}^T(k)\mathbf{R}^{-1}(k)\right]\mathbf{R}(k)\left[\mathbf{P}(k \mid k)\mathbf{H}^T(k)\mathbf{R}^{-1}(k)\right]^T.\end{aligned} \tag{175}$$

Pre and post multiplying by $\mathbf{P}^{-1}(k \mid k)$ then simplifying gives the covariance update equation as

$$\mathbf{P}(k \mid k) = \left[\mathbf{P}^{-1}(k \mid k-1) + \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k)\right]^{-1}. \tag{176}$$

Thus, in the inverse-covariance filter[10], the state covariance is first obtained from Equation 176 and then the state itself is found from Equation 173.

Consider now S sensors, each observing a common state according to

$$\mathbf{z}_s(k) = \mathbf{H}_s(k)\mathbf{x}(k) + \mathbf{v}_s(k), \qquad s = 1, \cdots, S, \tag{177}$$

where the noise $\mathbf{v}_s(k)$ is assumed to be white and uncorrelated in both time and between sensors;

$$\begin{aligned} \mathrm{E}\{\mathbf{v}_s(k)\} &= \mathbf{0}, \\ \mathrm{E}\{\mathbf{v}_s(i)\mathbf{v}_p(j)\} &= \delta_{ij}\delta_{sp}\mathbf{R}_i(k) \qquad s, p = 1, \cdots, S; \quad i, j = 1, \cdots. \end{aligned} \tag{178}$$

A composite observation vector $\mathbf{z}(k)$ comprising a stacked vector of observations from the S sensors may be constructed in the form

$$\mathbf{z}(k) = \begin{bmatrix} \mathbf{z}_1(k) \\ \cdots \\ \mathbf{z}_S^T(k) \end{bmatrix}, \tag{179}$$

a composite observation matrix $\mathbf{H}(k)$ comprising the stacked matrix of individual sensor observation matrices

$$\mathbf{H}(k) = \begin{bmatrix} \mathbf{H}_1(k) \\ \cdots \\ \mathbf{H}_S(k) \end{bmatrix}, \tag{180}$$

and a composite noise vector $\mathbf{v}(k)$ comprising a stacked vector of noise vectors from each of the sensors

$$\mathbf{v}(k) = \begin{bmatrix} \mathbf{v}_1(k) \\ \cdots \\ \mathbf{v}_S^T(k) \end{bmatrix}. \tag{181}$$

From Equation 178, the covariance in this composite noise vector is a block diagonal matrix

$$\mathbf{R}(k) = \mathrm{E}\{\mathbf{v}(k)\mathbf{v}^T(k)\} = \mathit{blockdiag}\left(\mathbf{R}_1(k), \cdots, \mathbf{R}_S(k)\right). \tag{182}$$

With these definitions, we have

$$\begin{aligned} \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k) &= \left[\mathbf{H}_1^T(k) \quad \mathbf{H}_2^T(k) \quad \cdots \quad \mathbf{H}_S^T(k)\right] \\ &\quad \times \begin{bmatrix} \mathbf{R}_1^{-1}(k) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_2^{-1}(k) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}_S^{-1}(k) \end{bmatrix} \begin{bmatrix} \mathbf{z}_1(k) \\ \mathbf{z}_2(k) \\ \vdots \\ \mathbf{z}_S(k) \end{bmatrix} \\ &= \sum_{i=1}^{S} \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{z}_i(k), \end{aligned} \tag{183}$$

[10]Strictly, the inverse covariance filter is defined using the inverse of Equation 176 [28].

and

$$\begin{aligned}\mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k) &= [\,\mathbf{H}_1^T(k) \quad \mathbf{H}_2^T(k) \quad \cdots \quad \mathbf{H}_S^T(k)\,] \\ &\quad \times \begin{bmatrix} \mathbf{R}_1^{-1}(k) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_1^{-1}(k) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}_S^{-1}(k) \end{bmatrix} \begin{bmatrix} \mathbf{H}_1(k) \\ \mathbf{H}_2(k) \\ \vdots \\ \mathbf{H}_S(k) \end{bmatrix} \\ &= \textstyle\sum_{i=1}^S \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{H}_i(k). \end{aligned} \tag{184}$$

Substituting Equation 183 into Equation 173, the state update equation becomes

$$\hat{\mathbf{x}}(k \mid k) = \mathbf{P}(k \mid k)\left[\mathbf{P}^{-1}(k \mid k-1)\hat{\mathbf{x}}(k \mid k-1) + \sum_{i=1}^S \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{z}_i(k)\right]. \tag{185}$$

Substituting Equation 184 into Equation 176, the state-covariance update equation becomes

$$\mathbf{P}(k \mid k) = \left[\mathbf{P}^{-1}(k \mid k-1) + \sum_{i=1}^S \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{H}_i(k)\right]^{-1}. \tag{186}$$

The multi-sensor inverse-covariance filter thus consists of a conventional state and state-covariance prediction stage given by Equations 85 and 86, followed by a state and state-covariance update from Equations 185 and 186.

Example 26

Consider again the multi-sensor tracking problem of Example 23 with three sensors:

$$z_1 = [\,1 \quad 0\,]\begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + w_1(k), \qquad \mathrm{E}\{w_1^2(k)\} = \sigma_{r_1}^2,$$

$$z_2 = [\,0 \quad 1\,]\begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + w_2(k), \qquad \mathrm{E}\{w_2^2(k)\} = \sigma_{r_2}^2,$$

and

$$z_3 = [\,1 \quad 0\,]\begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + w_3(k), \qquad \mathrm{E}\{w_3^2(k)\} = \sigma_{r_3}^2.$$

Thus the group sensor model is given by

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \mathbf{H}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix},$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 & 0 & 0 \\ 0 & \mathbf{R}_2 & 0 \\ 0 & 0 & \mathbf{R}_3 \end{bmatrix} = \begin{bmatrix} \sigma_{r_1}^2 & 0 & 0 \\ 0 & \sigma_{r_2}^2 & 0 \\ 0 & 0 & \sigma_{r_3}^2 \end{bmatrix}.$$

It follows that

$$\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_{r_1}^2} & 0 & 0 \\ 0 & \frac{1}{\sigma_{r_2}^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_{r_3}^2} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} \frac{z_1}{\sigma_{r_1}^2} + \frac{z_3}{\sigma_{r_3}^2} \\ \frac{z_2}{\sigma_{r_2}^2} \end{bmatrix} = \sum_{i=1}^{3} \mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{z}_i,$$

and

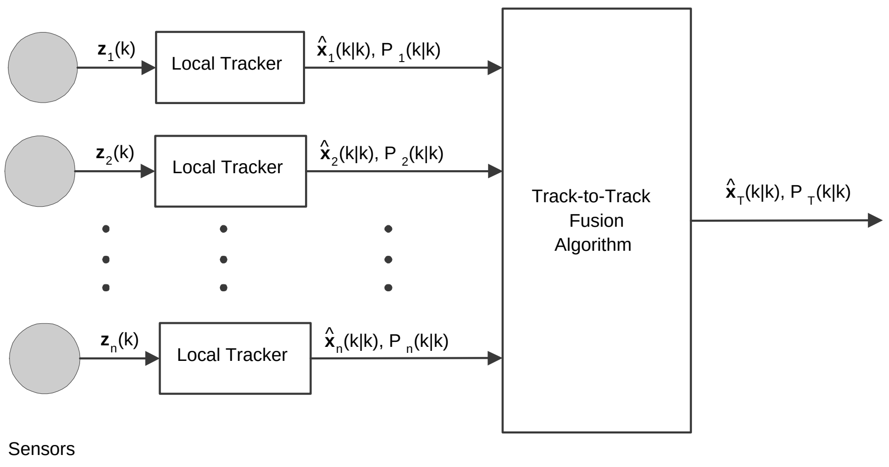
$$\begin{aligned} \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_{r_1}^2} & 0 & 0 \\ 0 & \frac{1}{\sigma_{r_2}^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_{r_3}^2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{\sigma_{r_1}^2} + \frac{1}{\sigma_{r_3}^2} & 0 \\ 0 & \frac{1}{\sigma_{r_2}^2} \end{bmatrix} \\ &= \sum_{i=1}^{3} \mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i, \end{aligned}$$

The advantages of the information-filter form over the group-sensor and asynchronous approaches to the multi-sensor estimation problem derive directly from the formulation of the update equations. Regardless of the number of sensors employed, the largest matrix inversion required is of dimension the state vector. The addition of new sensors simply requires that the new terms $\mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{z}_i(k)$ and $\mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{H}_i(k)$ be constructed and added together. Thus the complexity of the update algorithm grows only linearly with the number of sensors employed. In addition, the update stage can take place in one single step.

The advantages of the inverse-covariance estimator only become obvious when significant numbers of sensors are employed. It is clear from Equation 186 that both the prediction covariance and the updated inverse covariance matrix must be inverted in each cycle of the filter, and so the inverse-covariance filter obtains an advantage only when the dimension of the composite observation vector is approximately two times the dimension of the common state vector. As we shall see later, it is possible to write the prediction stage directly in terms of the inverse covariance, although this does not significantly reduce the amount of computation required.

3.2.5 Track-to-Track Fusion

Track-to-track fusion encompasses algorithms which combine *estimates* from sensor sites. This is distinct from algorithms that combine *observations* from different sensors; the former is often called track fusion, the latter "scan" fusion. In track-to-track fusion algorithms, local sensor sites generate local track estimates using a local Kalman filter. These tracks are then communicated to a central fusion site where they are combined to



$z_1(k)$
Local Tracker
$\hat{x}_1(k|k)$, $P_1(k|k)$
$z_2(k)$
Local Tracker
$\hat{x}_2(k|k)$, $P_2(k|k)$
$z_n(k)$
Local Tracker
$\hat{x}_n(k|k)$, $P_n(k|k)$
Track-to-Track Fusion Algorithm
$\hat{x}_T(k|k)$, $P_T(k|k)$
Sensors

Figure 22: A typical track-to-track fusion architecture in which local tracks are generated at local sensor sites and then communicated to a central fusion centre where a global track file is assimilated.

generate a global track estimate. A typical track-to-track fusion architecture is shown in Figure 22 In some configurations, the global estimate is then communicated back to the local sensor sites (called a feed-back configuration). Track-to-track fusion algorithms have a number of potential advantages over scan fusion methods:

1. Local track information is made available for use locally at each sensor sites.
2. Track information may be communicated at a lower, more compact, rate to the central site for fusion.

However track-to-track fusion algorithms also add additional complexity to a system. Detailed expositions of the track-to-track fusion method can be found in [4, 9].

The track-to-track fusion method begins by assuming a *common* underlying model of the state being observed in the standard form

$$\mathbf{x}(k) = \mathbf{F}(k)\mathbf{x}(k-1) + \mathbf{B}(k)\mathbf{u}(k) + \mathbf{G}(k)\mathbf{v}(k). \tag{187}$$

The state is assumed to be observed by a number of sensors each with different observation models in the form

$$\mathbf{z}_i(k) = \mathbf{H}_i(k)\mathbf{x}(k) + \mathbf{w}_i(k), \qquad i = 1, 2, \tag{188}$$

but where observation noises are assumed independent.

$$\mathrm{E}\{\mathbf{w}_i(k)\mathbf{w}_j(k)\} \;= \delta_{ij}\mathbf{R}_i(k). \tag{189}$$

A local track is formed by each sensor node, on the basis of only local observations, using the normal Kalman filter algorithm as

$$\hat{\mathbf{x}}_i(k \mid k) = \hat{\mathbf{x}}_i(k \mid k-1) + \mathbf{W}_i(k)\left[\mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}_i(k \mid k-1)\right], \tag{190}$$

and

$$\mathbf{P}_i(k \mid k) = \mathbf{P}_i(k \mid k-1) - \mathbf{W}_i(k)\mathbf{S}_i(k)\mathbf{W}_i^T(k), \tag{191}$$

where

$$\mathbf{W}_i(k) = \mathbf{P}_i(k \mid k-1)\mathbf{H}_i^T(k)\mathbf{S}_i^{-1}(k), \tag{192}$$

and

$$\mathbf{S}_i(k) = \left[\mathbf{H}_i(k)\mathbf{P}_i(k \mid k-1)\mathbf{H}_i^T(k) + \mathbf{R}_i(k)\right]. \tag{193}$$

Local state predictions are generated from a common state model

$$\hat{\mathbf{x}}_i(k \mid k-1) = \mathbf{F}(k)\hat{\mathbf{x}}_i(k-1 \mid k-1) + \mathbf{B}(k)\mathbf{u}(k) \tag{194}$$

and

$$\mathbf{P}_i(k \mid k-1) = \mathbf{F}(k)\mathbf{P}_i(k-1 \mid k-1)\mathbf{F}^T(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^T(k) \tag{195}$$

A straight-forward track fusion algorithm is simply to take the variance weighted average of tracks as follows:

$$\hat{\mathbf{x}}_T(k \mid k) = \mathbf{P}_T(k \mid k)\sum_{i=1}^{N}\mathbf{P}_i^{-1}(k \mid k)\hat{\mathbf{x}}_i(k \mid k) \tag{196}$$

$$\mathbf{P}_T(k \mid k) = \left[\sum_{i=1}^{N}\mathbf{P}_i^{-1}(k \mid k)\right]^{-1}. \tag{197}$$

This is a commonly practically used track-to-track fusion method.

However, a central issue in track-to-track fusion is that any two tracks $\hat{\mathbf{x}}_i(k \mid k)$ and $\hat{\mathbf{x}}_j(k \mid k)$ are correlated because they have a common prediction error resulting from a common process model. This correlation is intrinsic to the problem; it is only because the states have this process model in common that there is a reason to fuse the two. Thus it is *not* possible in general to compute a fused track $\hat{\mathbf{x}}_T(k \mid k)$ from a simple linear weighted sum of local tracks.

To address the general track-to-track fusion problem, it is necessary to find an expression for the correlation between two tracks. When this is known, it is possible to fuse the two estimates. To begin, we compute an expression for the track estimate and prediction errors as

$$\begin{aligned}
\tilde{\mathbf{x}}_i(k \mid k) &= \mathbf{x}(k) - \hat{\mathbf{x}}_i(k \mid k) \\
&= \mathbf{x}(k) - \hat{\mathbf{x}}_i(k \mid k-1) - \mathbf{W}_i(k)\left[\mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}_i(k \mid k-1)\right] \\
&= \mathbf{x}(k) - \hat{\mathbf{x}}_i(k \mid k-1) - \mathbf{W}_i(k)\left[\mathbf{H}_i(k)\mathbf{x}(k) + \mathbf{w}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}_i(k \mid k-1)\right] \\
&= \left[\mathbf{1} - \mathbf{W}_i(k)\mathbf{H}_i(k)\right]\tilde{\mathbf{x}}_i(k \mid k-1) - \mathbf{W}_i(k)\mathbf{w}_i(k)
\end{aligned} \tag{198}$$

and

$$\begin{aligned}
\tilde{\mathbf{x}}_i(k \mid k-1) &= \mathbf{x}(k) - \hat{\mathbf{x}}_i(k \mid k-1) \\
&= \mathbf{F}(k)\mathbf{x}(k-1) - \mathbf{F}(k)\hat{\mathbf{x}}_i(k-1 \mid k-1) + \mathbf{G}(k)\mathbf{v}(k) \\
&= \mathbf{F}(k)\tilde{\mathbf{x}}_i(k-1 \mid k-1) + \mathbf{G}(k)\mathbf{v}(k)
\end{aligned} \tag{199}$$

Squaring and taking expectations of Equation 199 yields and expression for the predicted track-to-track cross-correlation

$$\begin{aligned}
\mathbf{P}_{ij}(k \mid k-1) &= \mathrm{E}\{\tilde{\mathbf{x}}_i(k \mid k-1)\tilde{\mathbf{x}}_j^T(k \mid k-1) \mid \mathbf{Z}^{k-1}\} \\
&= \mathrm{E}\{[\mathbf{F}(k)\tilde{\mathbf{x}}_i(k-1 \mid k-1) + \mathbf{G}(k)\mathbf{v}(k)] \\
&\quad \times [\mathbf{F}(k)\tilde{\mathbf{x}}_i(k-1 \mid k-1) + \mathbf{G}(k)\mathbf{v}(k)]^T\} \\
&= \mathbf{F}(k)\mathrm{E}\{\tilde{\mathbf{x}}_i(k-1 \mid k-1)\tilde{\mathbf{x}}_j^T(k-1 \mid k-1) \mid \mathbf{Z}^{k-1}\}\ \mathbf{F}^T(k) \\
&\quad +\mathbf{G}(k)\mathrm{E}\{\mathbf{v}(k)\mathbf{v}^T(k)\}\ \mathbf{G}^T(k) \\
&= \mathbf{F}(k)\mathbf{P}_{ij}(k-1 \mid k-1)\mathbf{F}^T(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^T(k).
\end{aligned} \tag{200}$$

Squaring and taking expectations of Equation 198 yields an expression for the estimate track-to-track cross-correlation

$$\begin{aligned}
\mathbf{P}_{ij}(k \mid k) &= \mathrm{E}\{\tilde{\mathbf{x}}_i(k \mid k)\tilde{\mathbf{x}}_j^T(k \mid k)\} \\
&= \mathrm{E}\{[(\mathbf{1} - \mathbf{W}_i(k)\mathbf{H}_i(k))\,\tilde{\mathbf{x}}_i(k \mid k-1) - \mathbf{W}_i(k)\mathbf{w}_i(k)] \\
&\quad [(\mathbf{1} - \mathbf{W}_j(k)\mathbf{H}_j(k))\,\tilde{\mathbf{x}}_j(k \mid k-1) - \mathbf{W}_j(k)\mathbf{w}_j(k)] \mid \mathbf{Z}^k\} \\
&= [\mathbf{1} - \mathbf{W}_i(k)\mathbf{H}_i(k)]\,\mathrm{E}\{\tilde{\mathbf{x}}_i(k \mid k-1)\tilde{\mathbf{x}}_j^T(k \mid k-1)\}\ [\mathbf{1} - \mathbf{W}_j(k)\mathbf{H}_j(k)]^T \\
&\quad +\mathbf{W}_i(k)\mathrm{E}\{\mathbf{w}_i(k)\mathbf{w}_j^T(k)\}\ \mathbf{W}_j^T(k) \\
&= [\mathbf{1} - \mathbf{W}_i(k)\mathbf{H}_i(k)]\,\mathbf{P}_{ij}(k \mid k-1)\,[\mathbf{1} - \mathbf{W}_j(k)\mathbf{H}_j(k)]^T
\end{aligned} \tag{201}$$

where we have used the fact that $\mathrm{E}\{\mathbf{w}_i(k)\mathbf{w}_j^T(k)\} = \mathbf{0}$. Together, Equations 200 and 201 provide a recursive relationship for computing the cross-correlation $\mathbf{P}_{ij}(k \mid k)$ between the two track estimates $\hat{\mathbf{x}}_i(k \mid k)$ and $\hat{\mathbf{x}}_j(k \mid k)$.

Fusing together two tracks is essentially the same as adding observation information except that the data are correlated. Recall

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{P}_{xz}\mathbf{P}_{zz}^{-1}\,[\mathbf{z}(k) - \hat{\mathbf{z}}(k \mid k-1)] \tag{202}$$

and

$$\mathbf{P}(k \mid k) = \mathbf{P}(k \mid k-1) - \mathbf{P}_{xz}\mathbf{P}_{zz}^{-1}\mathbf{P}_{xz}^T \tag{203}$$

so

$$\hat{\mathbf{x}}_T(k \mid k) = \hat{\mathbf{x}}_i(k \mid k) + \mathbf{P}_{i|j}(k \mid k)\mathbf{P}_{i+j}^{-1}(k \mid k)\,[\hat{\mathbf{x}}_j(k \mid k) - \hat{\mathbf{x}}_i(k \mid k)] \tag{204}$$

and

$$\mathbf{P}_T(k \mid k) = \mathbf{P}_i(k \mid k) - \mathbf{P}_{i|j}(k \mid k)\mathbf{P}_{i+j}^{-1}(k \mid k)\mathbf{P}_{i|j}^T(k \mid k) \tag{205}$$

identify

$$\mathbf{P}_{i|j}(k \mid k) = \mathbf{P}_i(k \mid k) - \mathbf{P}_{ij}(k \mid k) \tag{206}$$

and

$$\mathbf{P}_{i+j}(k \mid k) = \mathbf{P}_i(k \mid k) + \mathbf{P}_j(k \mid k) - \mathbf{P}_{ij}(k \mid k) - \mathbf{P}_{ij}^T(k \mid k) \tag{207}$$

Substituting Equations 206 and 207 into Equation 204 gives

$$\begin{aligned}\hat{\mathbf{x}}_T(k \mid k) &= \hat{\mathbf{x}}_i(k \mid k) + [\mathbf{P}_i(k \mid k) - \mathbf{P}_{ij}(k \mid k)] \\ &\quad \times \left[\mathbf{P}_i(k \mid k) + \mathbf{P}_j(k \mid k) - \mathbf{P}_{ij}(k \mid k) - \mathbf{P}_{ij}^T(k \mid k)\right]^{-1} \\ &\quad \times [\hat{\mathbf{x}}_j(k \mid k) - \hat{\mathbf{x}}_i(k \mid k)] \qquad (208)\end{aligned}$$

as the combined track estimate, and

$$\begin{aligned}\mathbf{P}_T(k \mid k) &= \mathbf{P}_i(k \mid k) - [\mathbf{P}_i(k \mid k) - \mathbf{P}_{ij}(k \mid k)] \\ &\quad \times \left[\mathbf{P}_i(k \mid k) + \mathbf{P}_j(k \mid k) - \mathbf{P}_{ij}(k \mid k) - \mathbf{P}_{ij}^T(k \mid k)\right]^{-1} \\ &\quad \times [\mathbf{P}_i(k \mid k) - \mathbf{P}_{ij}(k \mid k)]^T \qquad (209)\end{aligned}$$

Equations 208 and 209 are in the form of predictor-corrector equations. As written, $\hat{\mathbf{x}}_i(k \mid k)$ is the predicted track, and $\hat{\mathbf{x}}_j(k \mid k) - \hat{\mathbf{x}}_i(k \mid k)$ is the correction term, weighted by a gain proportional to the corresponding covariances. The equations are symmetric so that the role of i and j are interchangeable.

There are a number of basic extensions to the basic track-to-track fusion algorithm. Notable is the use of "equivalent" measurements described in [9]. However, these methods are more appropriately described in context of the information filter.

3.3 Non-linear Data Fusion Methods

In a wide number of practical problems, the errors in process model predictions and in observation model updates are not easily described by symmetric and well-behaved Gaussian distributions. Further, the process and observation models themselves may not be linear or even smooth in the state. In such cases the Kalman filter, as a method for fusing information, may not be the appropriate data fusion algorithm for the task. Instances where this will be the case include:

- Systems in which the process or observation model are highly sensitive to the state prediction values and thus the point around which an extended Kalman filter may be linearised.
- Systems in which the process model is discontinuous, and indeed might have been formed from fitting of lines or curves to experimental data.
- Systems in which the error models for either process or observation or highly skewed so that the mean and the most likely values are substantially different.
- Systems in which the error models are multi-modal, having more than one "likely" value.

In general, the most appropriate course of action in these situations is to return to Bayes Theorem and seek an estimator which can more readily deal with either non Gaussian errors or with non-linear process or observation models.

3.3.1 Likelihood Estimation Methods

The recursive estimation problem can be derived entirely in probabilistic form using Bayes theorem. The general estimation problem requires that the probability distribution

$$P(\mathbf{x}_k \mid \mathbf{Z}^k, \mathbf{U}^k, \mathbf{x}_0) \tag{210}$$

be computed for all times k. This probability distribution describes the posterior density of vehicle state (at time k) given the recorded observations and control inputs up to and including time k together with the initial state. To develop a recursive estimation algorithm, a prior $P(\mathbf{x}_{k-1} \mid \mathbf{Z}^{k-1}, \mathbf{U}^{k-1})$ at time $k-1$ is assumed. The posterior, following a control $\mathbf{u}_k$ and observation $\mathbf{z}_k$, is then computed using Bayes Theorem. This computation requires that a state transition model and an observation model are defined.

The observation model is generally described in the form

$$P(\mathbf{z}_k \mid \mathbf{x}_k), \tag{211}$$

and is assumed conditionally independent, so that

$$\begin{aligned} P(\mathbf{Z}^k \mid \mathbf{X}^k) &= \prod_{i=1}^{k} P(\mathbf{z}_i \mid \mathbf{X}^k) \\ &= \prod_{i=1}^{k} P(\mathbf{z}_i \mid \mathbf{x}_i). \end{aligned} \tag{212}$$

The transition model for the state can be described in terms of a probability distribution on state transitions in the form

$$P(\mathbf{x}_k \mid \mathbf{u}_k, \mathbf{x}_{k-1}). \tag{213}$$

The state transition may reasonably be assumed to be a Markov process in which the next state $\mathbf{x}_k$ depends only on the immediately proceeding state $\mathbf{x}_{k-1}$ and the applied control $\mathbf{u}_k$, and is independent of the observations made.

With these definitions and models, Bayes Theorem may be employed to define a recursive solution to Equation 210.

To derive a recursive update rule for the posterior, the chain rule of conditional probability is employed to expand the joint distribution on the state and observation in terms of the state

$$\begin{aligned} P(\mathbf{x}_k, \mathbf{z}_k \mid \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) &= P(\mathbf{x}_k \mid \mathbf{z}_k, \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) P(\mathbf{z}_k \mid \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) \\ &= P(\mathbf{x}_k \mid \mathbf{Z}^k \mathbf{U}^k, \mathbf{x}_0) P(\mathbf{z}_k \mid \mathbf{Z}^{k-1} \mathbf{U}^k, \mathbf{x}_0), \end{aligned} \tag{214}$$

and then in terms of the observation

$$\begin{aligned} P(\mathbf{x}_k, \mathbf{z}_k \mid \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) &= P(\mathbf{z}_k \mid \mathbf{x}_k, \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) P(\mathbf{x}_k, \mid \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) \\ &= P(\mathbf{z}_k \mid \mathbf{x}_k) P(\mathbf{x}_k \mid \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) \end{aligned} \tag{215}$$

where the last equality employs the assumptions established for the sensor model in Equation 211.

Equating Equations 214 and 215 and rearranging gives

$$P(\mathbf{x}_k \mid \mathbf{Z}^k, \mathbf{U}^k, \mathbf{x}_0) = \frac{P(\mathbf{z}_k \mid \mathbf{x}_k) P(\mathbf{x}_k \mid \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0)}{P(\mathbf{z}_k \mid \mathbf{Z}^{k-1}, \mathbf{U}^k)}. \tag{216}$$

The denominator in Equation 216 is independent of the state and can therefore be set to some normalising constant K. The total probability theorem can be used to rewrite the second term in the numerator in terms of the state process model and the joint posterior from time-step $k-1$ as

$$\begin{aligned} P(\mathbf{x}_k \mid \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) &= \int P(\mathbf{x}_k, \mathbf{x}_{k-1} \mid \mathbf{Z}^{k-1}, \mathbf{U}^k \mathbf{x}_0) \mathrm{d}\mathbf{x}_{k-1} \\ &= \int P(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) P(\mathbf{x}_{k-1} \mid \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) \mathrm{d}\mathbf{x}_{k-1} \\ &= \int P(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_k) P(\mathbf{x}_{k-1} \mid \mathbf{Z}^{k-1}, \mathbf{U}^{k-1}, \mathbf{x}_0) \mathrm{d}\mathbf{x}_{k-1} \qquad (217) \end{aligned}$$

where the last equality follows from the assumed independence of state transition from observations, and from the causality of the control input on state transition.

$$P(\mathbf{x}_k \mid \mathbf{Z}^k, \mathbf{U}^k, \mathbf{x}_0) = K.P(\mathbf{z}_k \mid \mathbf{x}_k) \int P(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_k) P(\mathbf{x}_{k-1} \mid \mathbf{Z}^{k-1}, \mathbf{U}^{k-1}, \mathbf{x}_0) \mathrm{d}\mathbf{x}_{k-1} \tag{218}$$

Equation 218 provides a recursive formulation of the posterior in terms of an observation model and a motion model. It is completely general in allowing any probabilistic and kinematic model for both state transition and observation. It also allows, post posterior, any estimation scheme (maximum likelihood, mini-max, minimum-mean-squared, for example) to be applied.

The following algorithms use this formulation of the estimation problem. They are becoming increasingly important as available computational resources increase and as data fusion problems addressed become more complex. However, time precludes discussing these algorithms in detail.

3.3.2 The Particle Filter

The particle filter is essentially a monte-carlo simulation of Equation 218. See in particular [40] for data fusion application of the particle filter method. This also incorporates other related report information (such as shoreline boundaries) in full probabilistic form).

3.3.3 The Sum-of-Gaussians Method

The sum-of-Gaussians method replaces the distributions in the general formulation with an approximation as a sum of Gaussians. In principle, this can be used to model any distribution. Further, there are efficient update rules for a sum-of-Gaussians approximation. See in particular [22, 1, 29].

3.3.4 The Distribution Approximation Filter (DAF)

The distribution approximation filter (DAF) is a recent method which provides an efficient method of approximating the non-linear transformation of Gaussians. See [23] for details.

3.4 Multi-Sensor Data Association

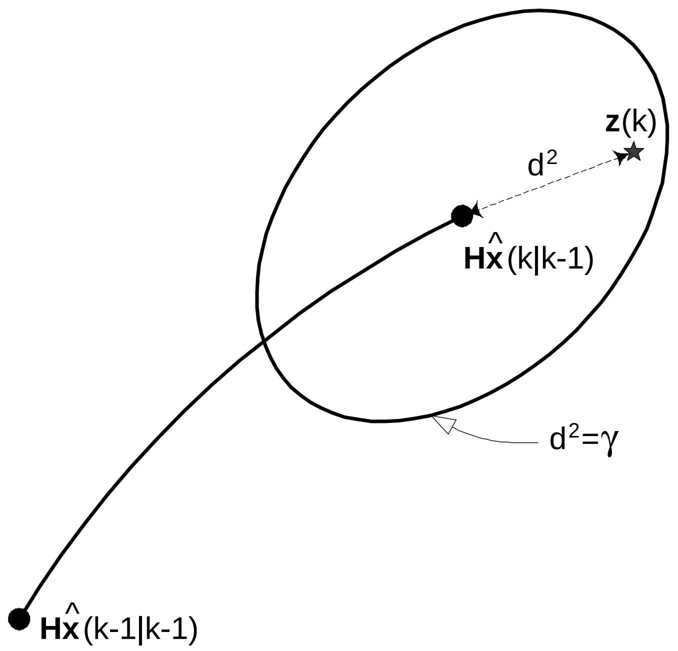


Figure 23: The validation gate. The gate is defined in the observation space and is centered on the observation prediction $\mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1)$. The gate is the normalised innovation. For a fixed value of $d^2 = \gamma$ the gate is ellipsoidal and defined by the eigenvalues of the inverse innovation covariance. The size of the gate is set by requiring the probability of correct association to be above a certain threshold. Observations that fall within this gate are considered 'valid'.

In multi-sensor estimation problems, the many measurements made need to be correctly associated with the underlying states that are being observed. This is the data association problem.

The data association problem includes issues of validating data (ensuring it is not erroneous or arising from clutter[11] for example), associating the correct measurements to

[11]The term *clutter* refers to detections or returns from nearby objects, weather, electromagnetic interference, acoustic anomalies, false alarms, etc., that are generally random in number, location, and intensity. This leads to the occurrence of (possibly) several measurements in the vicinity of the target. The implication of the single target assumption is that the undesirable measurements constitute *ran-*

the correct states (particularly in multi-target tracking problems), and initialising new tracks or states as required. Whereas conventional tracking is really concerned with uncertainty in measurement location, data association is concerned with uncertainty in measurement origin.

The foundations of data association lie in the use of the normalised innovation or 'validation gate'. The innovation $\nu_{ij}(k)$ is the difference between the observation $\mathbf{z}_i(k)$ actually made, and that predicted by the filter $\hat{\mathbf{z}}_j(k \mid k-1) = \mathbf{H}_j(k)\hat{\mathbf{x}}_j(k \mid k-1)$;

$$\nu_{ij}(k) = \mathbf{z}_i(k) - \hat{\mathbf{z}}_j(k \mid k-1) = \mathbf{z}_i(k) - \mathbf{H}_j(k)\hat{\mathbf{x}}_j(k \mid k-1).$$

The innovation variance is simply defined as

$$\begin{aligned}\mathbf{S}_{ij}(k) &= \mathrm{E}\{\nu_{ij}(k)\nu_{ij}^T(k)\} \\ &= \mathbf{H}_j(k)\mathbf{P}_j(k \mid k-1)\mathbf{H}_j^T(k) + \mathbf{R}_i(k).\end{aligned}$$

The normalised innovation between an observation i and a track j is then given by

$$d_{ij}^2(k) = \nu_{ij}^T(k)\mathbf{S}_{ij}^{-1}(k)\nu_{ij}(k). \tag{219}$$

The normalised innovation is a scalar quantity. It can be shown that if the innovations are zero mean and white (the filter is operating correctly) then the normalised innovation is a χ^2 random variable in n_z degrees of freedom (the dimension of the observation vector). It is therefore possible to establish values for d_{ij}^2 which enclose a specific probability of the observation and predicted observation being correctly associated [7, 18]. This is key to establishing data association methods. Referring to Figure 23, for a fixed value of d_{ij}^2 the normalised innovation describes a quadratic centered on the prediction generating an ellipsoidal volume (of dimension n_z) whose axes are proportional to the reciprocal of the eigenvalues of $\mathbf{S}_{ij}^{-1}(k)$. This ellipsoid defines an area or volume in observation space centered on the observation prediction. If an observation falls within this volume then it is considered 'valid'; thus the term validation gate.

The normalised innovation serves as the basis for all data association techniques. The reasons for this are clear. First, the innovation is practically the only measure of divergence of state estimate from observation sequence. Second, it admits a precise probabilistic measure of correct association.

3.4.1 The Nearest-Neighbour Standard Filter

The nearest neighbour standard filter (NNSF) applies the obvious data association policy: It simply chooses the measurement 'nearest' to the predicted measurement as the only validated observation, and the remaining observations are rejected from consideration (Figure 24). The single validated measurement is used for updating the state estimate of the target. The definition of 'nearest' is the observation which yields the minimum

dom interference. A common mathematical model for such interference is a uniform distribution in the measurement space.

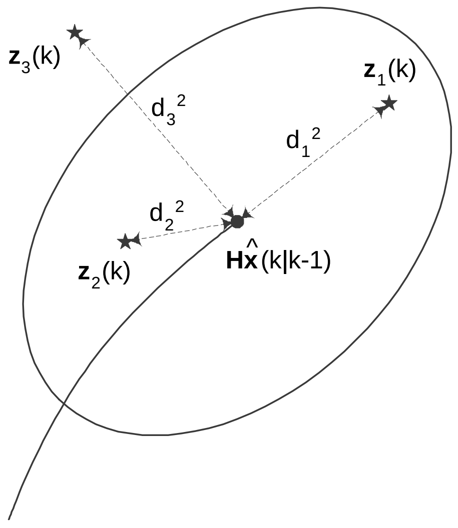


Figure 24: For a single target track, the nearest-neighbour standard filter selects the observation 'nearest' the prediction for association. All other observations are rejected. The proximity measure is the normalised innovation.

normalised innovation as defined in Equation 219. If there are no observations with normalised innovation values less than some defined d_{ij}^2, then no observation is associated. The NNSF algorithm implicitly assumes a high probability of detection for the target and a low rate of clutter in the gate. In such situations, the NNSF algorithm is usually adequate and is practically used in many applications.

There are a number of problems with the basic NNSF algorithm particularly in situations where there is a high degree of clutter and where there are potentially a number of closely spaced target states to be tracked. Figure 25 shows a particular situation in which two track validation gates overlap. In this, and many other cases, the problem of correctly associating an observation with a track is complex. This is first because the obvious pair-wise closest association is not necessarily correct, and second because association is order dependent and so all associations must be considered together to achieve a correct assignment (even when the probability of detect is unity and the false alarm rate is zero). The following two algorithms are the most widely used and common methods of overcoming these two problems (see [7, 9] for detailed expositions of the data association problem).

3.4.2 The Probabilistic Data Association Filter

The Probabilistic Data Association Filter (PDAF) and Joint Probabilistic Data Association Filter (JPDAF) are Bayesian algorithms which compute a probability of correct association between an observation and track. This probability is then used to form a

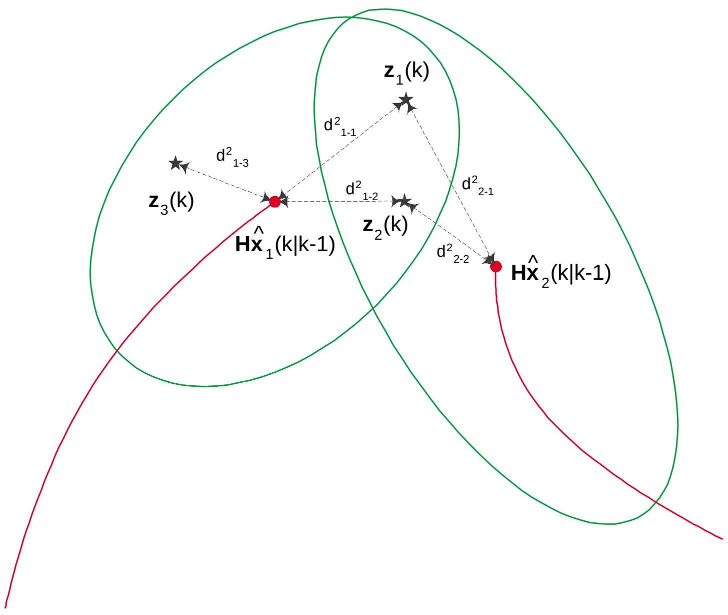


Figure 25: For multiple targets, each target track selects its nearest observation independently of other possible observation-track associations. This can easily lead to erroneous assignments in high target density situations.

weighted average 'correct track'.

The PDAF algorithm starts with a set of validated measurements (those that fall within the initial validation gate). Denoting the set of validated measurements at time k as

$$Z(k) \equiv \{\mathbf{z}_i(k)\}_{i=1}^{m_k}$$

where m_k is the number of measurements in the validation region. The cumulative set of measurements is denoted

$$Z^k \equiv \{Z(j)\}_{j=1}^{k}$$

A basic (sub-optimal) assumption is made that the latest state is assumed normally distributed according to the latest estimate and covariance matrix:

$$p[\mathbf{x}(k) \mid Z^{k-1}] = \mathrm{N}[\mathbf{x}(k); \hat{\mathbf{x}}(k \mid k-1), \mathbf{P}(k \mid k-1)]$$

This assumption means that $Z(k)$ contains measurements from the same elliptical validation gate as used in the NNSF algorithm. This is shown in Figure 26.

In the PDAF algorithm, it is assumed that there is only one target of interest whose track has been initialized (the JPDAF algorithm deals with the more complex problem of coupled multi-target tracks). At each time step a validation region is set up. Among the possibly several validated measurements, one can be target originated, if the target was detected. The remaining measurements are assumed due to false alarms or residual

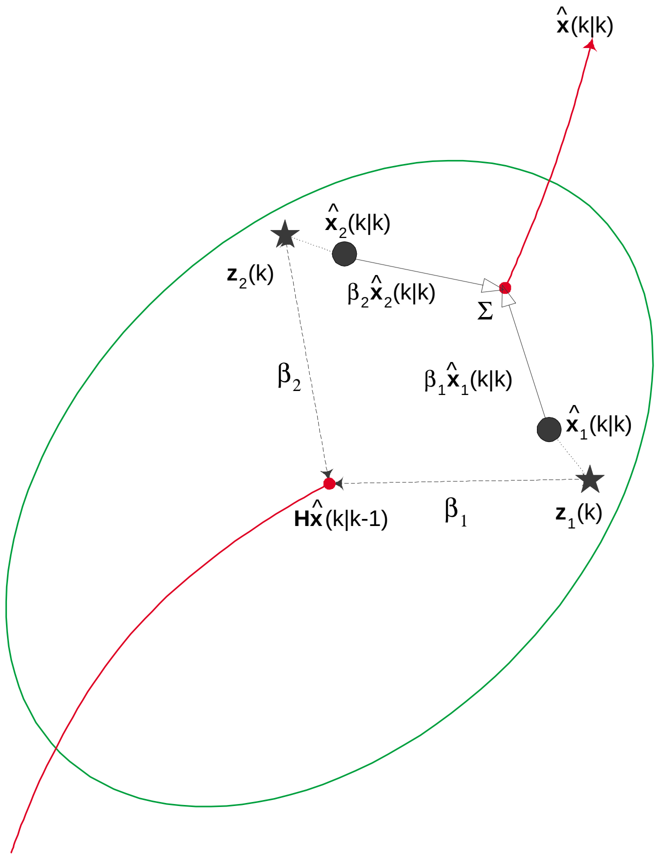

$\hat{\mathbf{x}}(k|k)$
$\hat{\mathbf{x}}_2(k|k)$
$\mathbf{z}_2(k)$
$\beta_2\hat{\mathbf{x}}_2(k|k)$
Σ
β_2
$\beta_1\hat{\mathbf{x}}_1(k|k)$
$\hat{\mathbf{x}}_1(k|k)$
$\mathbf{H}\hat{\mathbf{x}}(k|k-1)$
β_1
$\mathbf{z}_1(k)$

Figure 26: The probabilistic data association algorithm associates all valid observations with a track. For each validated observation, an updated estimate $xjikk$ is computed. A probability of correct association β_i is computed for each such track. Then a combined track is formed from the weighted average of these tracks: $\hat{\mathbf{x}}(k \mid k) = \sum \beta_i \hat{\mathbf{x}}_i(k \mid k)$. For multiple targets, the same process occurs although the probability calculations are more complex.

clutter and are modeled as IID random variables with uniform density. Define $\theta_i(k)$ to be the event that the measurement $\mathbf{z}_i(k)$ originated from the target, and $\theta_0(k)$ as the event that no measurement originated from the target. Let the probabilities of these events be

$$\beta_i(k) \equiv P[\theta_i(k) \mid Z^k], \quad i = 0, 1, \cdots, m_k$$

The events are assumed mutually independent and exhaustive:

$$\sum_{i=0}^{m_k} \beta_i(k) = 1$$

Using the total probability theorem with respect to the above events, the conditional mean of the state at time k can be written as

$$\begin{aligned}
\hat{\mathbf{x}}(k \mid k) &= \mathrm{E}\{\mathbf{x}(k) \mid Z^k\} \\
&= \sum_{i=0}^{m_k} \mathrm{E}\{\mathbf{x}(k) \mid \theta_i(k), Z^k\}\ P(\theta_i(k) \mid Z^k) \\
&= \sum_{i=0}^{m_k} \hat{\mathbf{x}}_i((\mid k)k)\beta_i(k) \qquad (220)
\end{aligned}$$

where $\hat{\mathbf{x}}_i(k \mid k)$ is the updated estimate conditioned on the event $\theta_i(k)$ that the i^{th} validated measurement is correct. This estimate is itself found from

$$\hat{\mathbf{x}}_i(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}_((k)i)\nu_i(k), \quad i = 1, \cdots, m_k \qquad (221)$$

where

$$\nu_i(k) = \mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}(k \mid k-1)$$

and if none of the measurements are correct, the estimate is

$$\hat{\mathbf{x}}_0(k \mid k) = \hat{\mathbf{x}}(k \mid k-1)$$

The PDAF algorithm now proceeds as follows:

1. The set of validated measurements is computed.

2. For each validated measurement an updated track is computed using Equation 221.

3. For each updated track an association probability β_i is computed. The calculation of this probability can be quite complex and dependent on the assumed clutter densities. However, it is normally adequate to set β_i proportional to the normalised innovation for the association.

4. A combined (average) track is computed using Equation 220. A combined average covariance can also be computed although this can become quite complex (see [7] for details.

5. A single prediction of the combined track is then made to the next scan time and the process is repeated.

The PDAF and JPDAF methods are appropriate in situations where there is a high degree of clutter. As pointed out to the author in conversation once: “The great advantage with the PDAF method is that you are never wrong. The problem is you are also never right”.

3.4.3 The Multiple Hypothesis Tracking (MHT) Filter

The Multiple Hypothesis Tracking (MHT) filter (and the related Track Splitting Filter (TSF)), maintain separate tracks for each possible associated observation. At each time step, the predicted observation is used to establish a validation gate (see Figure 27). For each measurement that is found in this validation gate, a new hypothesis track is generated. Thus a single track is split in to n tracks, one associated with each valid measurement plus one track (usually denoted 0) for the no-association hypothesis. Each of these new tracks is then treated independently and used to generate new predictions for the next time step. Since the number of branches into which the track is split can grow exponentially, the likelihood function of each split track is computed and the unlikely ones are discarded.

The MHT algorithm works on complete sequences of observations. That is, the probability that a given branch sequence of observations (from root to leaf) is correct. Denote the l^{th} sequence of measurements up to time k as:

$$\mathbf{Z}^{kl} \equiv \{\mathbf{z}_{i_{1,l}}(1), \cdots, \mathbf{z}_{i_{k,l}}(k)\}$$

where $\mathbf{z}_i(()\mathrm{j})$ is the i^{th} measurement at time j. Let $\Theta^{k,l}$ be the event that the sequence $\mathbf{Z}^{kl}$ is a correct track, then the likelihood function for this event is clearly

$$\Lambda(\Theta^{k,l}) = P(\mathbf{Z}^{k,l} \mid \Theta^{kl}) = P(\mathbf{z}_{i_{1,l}}(1), \cdots, \mathbf{z}_{i_{k,l}}(k) \mid \Theta^{kl}) \tag{222}$$

Denoting $\mathbf{Z}^k$ the cumulative set of all measurements up to time k, Equation 222 reduces to

$$\Lambda(\Theta^{k,l}) = \prod_{j=1}^{k} P(\mathbf{z}_{i_{j,l}} \mid \mathbf{Z}^{j-1}\Theta^{k,l})$$

Under assumptions of linearity and Gaussianity, this reduces to

$$\Lambda(\Theta^{k,l}) = c_k \exp\left[-\frac{1}{2}\sum_{j=1}^{k} \nu^T(j)\mathbf{S}^{-1}(j)\nu(j)\right]$$

where $\nu(j) = \mathbf{z}(j) - \hat{\mathbf{z}}(j \mid j-1)$ is the innovation between track and measurement.

The corresponding modified log-likelihood function is given by

$$\lambda(k) \equiv -2\log\left[\frac{\Lambda(\Theta^{k,l})}{c_k}\right] = \sum_{j=1}^{k} \nu^T(j)\mathbf{S}^{-1}(j)\nu(j) \tag{223}$$

which can be recursively computed as

$$\lambda(k) = \lambda(k-1) + \nu^T(k)\mathbf{S}^{-1}(k)\nu(k). \tag{224}$$

The last term above is a measure of the "goodness of fit" of the measurements. A statistical test for accepting a track is that the log-likelihood function satisfies $\lambda(k) < d$.

The MHT algorithm now proceeds as follows

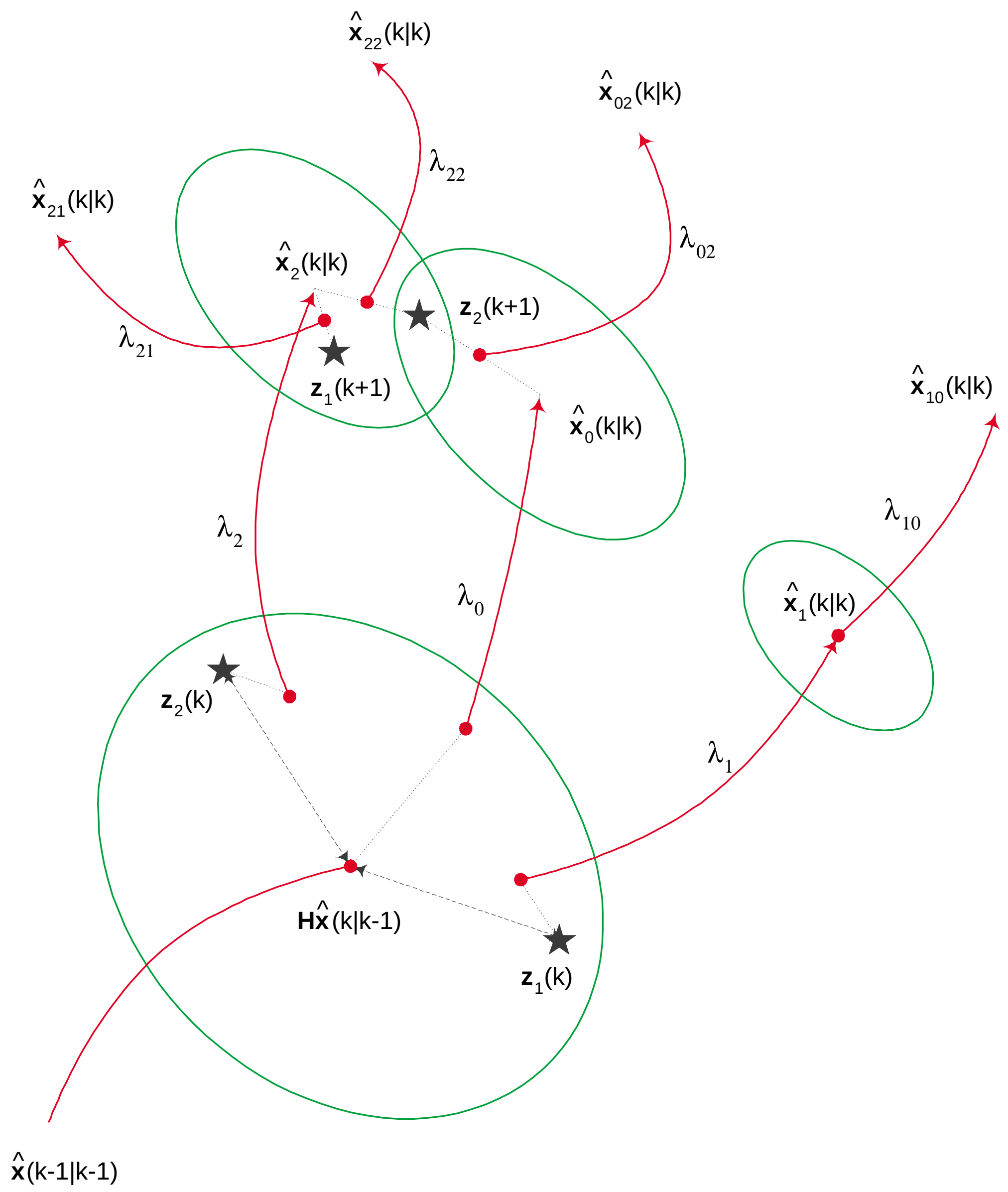

$\hat{\mathbf{x}}_{22}$(k|k)
$\hat{\mathbf{x}}_{02}$(k|k)
$\hat{\mathbf{x}}_{21}$(k|k)
λ_{22}
λ_{02}
$\hat{\mathbf{x}}_{2}$(k|k)
$\mathbf{z}_2$(k+1)
λ_{21}
$\mathbf{z}_1$(k+1)
$\hat{\mathbf{x}}_{0}$(k|k)
$\hat{\mathbf{x}}_{10}$(k|k)
λ_{2}
λ_{10}
λ_{0}
$\hat{\mathbf{x}}_{1}$(k|k)
$\mathbf{z}_2$(k)
λ_{1}
$\mathbf{H}\hat{\mathbf{x}}$(k|k-1)
$\mathbf{z}_1$(k)
$\hat{\mathbf{x}}$(k-1|k-1)

Figure 27: In the track splitting or multiple hypothesis filter, every validated observation $\mathbf{z}_p(k)$ is used to establish a new track $\hat{\mathbf{x}}_p(k \mid k)$. In addition the 'false alarm' and/or 'missed observation' hypothesis also generates a track $\hat{\mathbf{x}}_0(k \mid k)$. These tracks are propagated forward to the next gate and again each track is associated with each valid observation $\mathbf{z}_q(k+1)$ and the tracks are again split into tracks associated with each possible pair-wise association $\hat{\mathbf{x}}_{pq}(k+1 \mid k+1)$. Probabilities or likelihoods λ_{pq} of correct track histories are maintained to prune the resulting hypothesis tree.

1. A predicted observation is produced and a validation region around the prediction established.
2. All observations that fall within the validation gate are considered candidate associations. In addition, the no-association hypothesis '0' is formed.
3. The track is updated with each validated hypothesis according to Equations 221.
4. A likelihood associated with correct association is computed according to Equation 223.
5. The likelihood of association is recursively added to the likelihood of the prediction path having been correctly associated according to Equation 224. This yields a likelihood of the entire sequence of observations to this point being correct.
6. Some pruning of the hypothesis tree may take place at this point.
7. Each track hypothesis is now independently predicted forward to the next time-step, effectively splitting the track equally into each possible competing hypothesis.
8. The process repeats.

There are three points to note about the MHT (and related TSF) algorithm:

- We have implicitly assumed that the detection probability of detection is unity. That is only complete sequences of measurements are considered in the track formation process. ¡issing observations can be taken into account by incorporating a probability of detection at each stage.
- In practice, the likelihood pruning method does not work well with long measurement sequences as it becomes dominated by old measurements. One "hack" around this is to use a fading-memory window.
- The method is clearly dominated by the computational and memory requirements of the splitting algorithm. At each stage each hypothesis can generate many new hypotheses and filters that must be run in parallel.

The MHT algorithm is generally good in situation where there are low clutter rates but high track uncertainty (crossing tracks, maneuvering targets, etc). Practically, the algorithm is dominated by the approach used for pruning unlikely target hypotheses.

3.4.4 Data Association in Track-to-Track Fusion

Data association methods can be extended to association of tracks rather than observations. Such situations occur in distributed architectures and particularly in track-to-track fusion algorithms.

Following the track-to-track fusion algorithm of Equations 208 and 209 a gate equivalent to Equation 219 can be established for testing of two tracks i and j can be associated

$$\begin{aligned} d_{ij}^2(k) &= [\hat{\mathbf{x}}_i(k \mid k) - \hat{\mathbf{x}}_j(k \mid k)] \left[\mathbf{P}_i(k \mid k) + \mathbf{P}_j(k \mid k) - \mathbf{P}_{ij}(k \mid k) - \mathbf{P}_{ij}^T(k \mid k)\right]^{-1} \\ &\quad \times [\hat{\mathbf{x}}_i(k \mid k) - \hat{\mathbf{x}}_j(k \mid k)]^T . \end{aligned} \tag{225}$$

This gate defines a similar ellipsoidal region to the normal validation gate. The test statistic is also χ^2 distributed, but in n_x degrees of freedom (the dimension of the state vector).

Having established this gate, the normal NNSF type algorithm is usually used for associating tracks. However, there is no objection in principle using either PDAF or MHT algorithms for a similar purpose.

3.4.5 Maintaining and Managing Track Files

In multiple target problems in particular, there is a need to maintain a record of how observations and tracks are associated together. This is generally termed a Track File. Track files may be maintained centrally as a 'Joint Assignment Matrix' (JAM)[12]. Track files may also be maintained locally at a sensor site, although these must then be coordinated in some way. A detailed exposition of track file maintenance is beyond the scope of this course (see [9] p603–605 for a detailed discussion).

[12]The Joint Assignment Matrix is used, algorithmically, in a number of advanced data association methods.

4 Distributed and Decentralised Data Fusion Systems

The section addresses the development of data fusion algorithms for distributed and decentralised data fusion architectures.

The nature of data fusion is that there are a number of sensors physically distributed around an environment. In a centralised data fusion system, raw sensor information is then communicated back to a central processor where the information is combined to produce a single fused picture of the environment. In a distributed data fusion system, each sensor has it's own local processor which can generally extract useful information from the raw sensor data prior to communication. This has the advantage that less information is normally communicated, the computational load on the central processor is reduced and the sensors themselves can be constructed in a reasonably modular manner. The degree to which local processing occurs at a sensor site varies substantially from simple validation and data compression up to the full construction of tracks or interpretation of information locally.

While for many systems a centralised approach to data fusion is adequate, the increasing sophistication, functional requirements, complexity and size of data fusion systems, coupled with the ever reducing cost of computing power argues more and more toward some form of distributed processing. The central issue in designing distributed data fusion systems is the development of appropriate algorithms which can operate at a number of distributed sites in a consistent manner. This is the focus of this section.

This section begins with a general discussion of data fusion architectures and the challenges posed in developing distributed data fusion algorithms. The information filter, and more generally the log-likelihood implementations of Bayes theorem are then developed and it is shown how these can readily be mapped to many distributed and decentralised data fusion systems. The issue of communication is a major problem in distributed data fusion systems. This is because of both limited communication bandwidth and also time delays and communication failures that can occur between sensing and fusion processes. The communication problem is dealt with in depth in this section. Finally, some advanced issues in distributed and decentralised data fusion are briefly considered. These include the problem of model distribution, where each local sensor maintains a different model of the environment; sensor management, in which a limited set of sensor resources must be used cooperatively; and system organisation, in which the optimal design of sensor networks is addressed.

4.1 Data Fusion Architectures

Distributed data fusion systems may take many forms. At the simplest level, sensors could communicate information directly to a central processor where it is combined. Little or no local processing of information need take place and the relative advantage of having many sources of information is sacrificed to having complete centralised control over the processing and interpretation of this information. As more processing occurs locally, so computational and communication burden can be removed from the fusion center, but at the cost of reduced direct control of low-level sensor information.

Increasing intelligence of local sensor nodes naturally results in a hierarchical structure for the fusion architecture. This has the advantage of imposing some order on the fusion process, but the disadvantage of placing a specific and often rigid structure on the fusion system.

Other distributed architectures consider sensor nodes with significant local ability to generate tracks and engage in fusion tasks. Such architectures include 'Blackboard' and agent based systems.

Fully decentralised architectures have no central processor and no common communication system. In such systems, nodes can operate in a fully autonomous manner, only coordinating through the anonymous communication information.

The following sections consider these architectures and their associated data fusion algorithms.

4.1.1 Hierarchical Data Fusion Architectures

In a hierarchical structure, the lowest level processing elements transmit information upwards, through successive levels, where the information is combined and refined, until at the top level some global view of the state of the system is made available. Such hierarchical structures are common in many organisations and have many well-known advantages over fully centralised systems; particularly in reducing the load on a centralised processor while maintaining strict control over sub-processor operations.

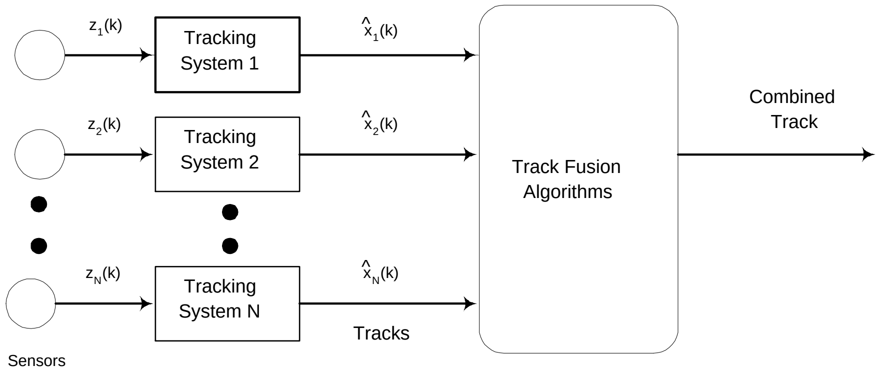


Figure 28: Single Level Hierarchical Multiple Sensor Tracking System

The hierarchical approach to systems design has been employed in a number of data fusion systems and has resulted in a variety of useful algorithms for combining information at different levels of a hierarchical structure. General hierarchical Bayesian algorithms are based on the independent likelihood pool architectures shown in Figures 4 and 5, or on the log-likelihood opinion pools shown in Figures 8 and 9. Here the focus is on hierarchical estimation and tracking algorithms (See Figures 28 and 29).

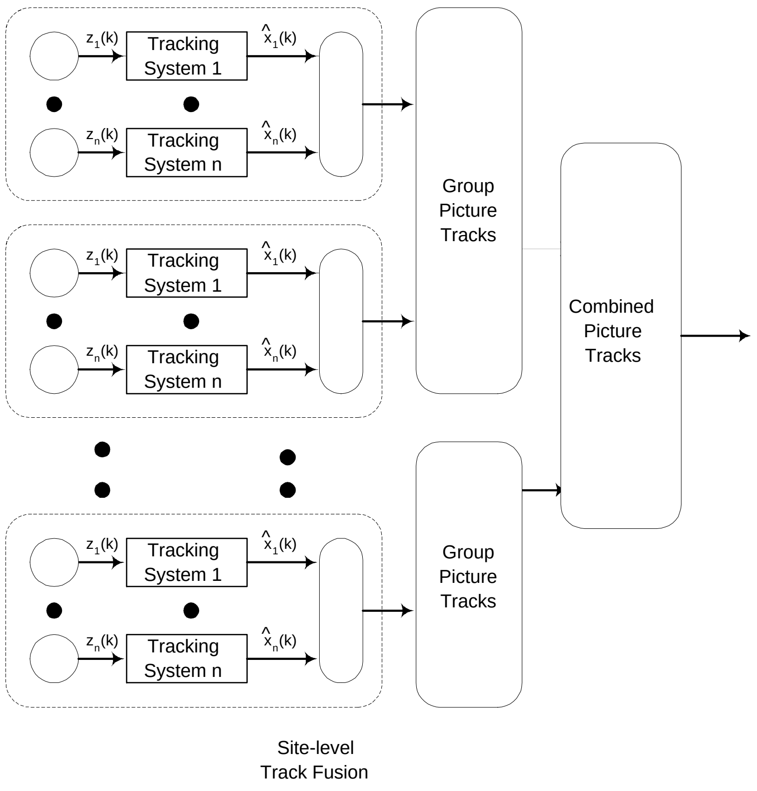


Figure 29: Multiple Level Hierarchical Multiple Sensor Tracking System

First it is assumed that all sensors are observing a common state or track $\mathbf{x}(k)$. Observations are made at local sites of this common state according to a local observation equation in the form

$$\mathbf{z}_i(k) = \mathbf{H}_i(k)\mathbf{x}(k) + \mathbf{w}_i(k), \qquad i = 1, \cdots, S \tag{226}$$

In principle, each site may then operate a conventional Kalman filter or state estimator to provide local state estimates based only on local observations in the form

$$\hat{\mathbf{x}}_i(k \mid k) = \hat{\mathbf{x}}_i(k \mid k-1) + \mathbf{W}_i(k)\left[\mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}_i(k \mid k-1)\right], \qquad i = 1, \cdots, S. \tag{227}$$

These local estimates $\hat{\mathbf{x}}_i(k \mid k)$ may then be passed further up the hierarchy to a central or intermediate processor which combines or fuses tracks to form a global estimate based on all observations in the form

$$\hat{\mathbf{x}}(k \mid k) = \sum_{i=1}^{S} \omega_i(k)\hat{\mathbf{x}}_k(k \mid i), \tag{228}$$

where $\omega_i(k)$ are site weighting matrices.

The essential problem, as described in Section 3, is that each sensor site must be observing a common true state and so the local process models are related through some common global model in the form

$$\mathbf{x}_i(k) = \mathbf{F}_i(k)\mathbf{x}(k) + \mathbf{B}_i(k)\mathbf{u}(k) + \mathbf{G}_i(k)\mathbf{v}(k), \qquad i = 1, \cdots, S.$$

This means that the predictions made at local sites are correlated and so the updated local estimates in Equation 227 must also be correlated despite the fact that the observations made at each site are different. Consequently, the local estimates generated at each site cannot be combined in the independent fashion implied by Equation 228.

The correct method of dealing with this problem is to explicitly account for these correlations in the calculation of the site weighting matrices $\omega_i(k)$. In particular, the track-to-track fusion algorithms described in Section 3.2.5 in the form of Equations 208 and 209 are appropriate to this problem. These algorithms require that the correlations between all sites be explicitly computed in addition to the covariance associated with each local state estimate.

There have been a number of papers on hierarchical estimation systems. The paper by Hashemipour, Roy and Laub [21] is notable in employing, indirectly, the information form of the Kalman filter to derive a hierarchical estimation algorithm. The earlier paper by Speyer [39] has a similar formulation, and although it is concerned with distributed linear quadratic Gaussian (LQG) control problems, also specifically deals with communication or transmission requirements. Other large scale systems in control exhibit similar properties [37]. In addition, the track-to-track fusion techniques described by Bar-Shalom [15, 4] serve as the basis for many derived architectures.

A hierarchical approach to the design of data fusion systems also comes with a number of inherent disadvantages. The ultimate reliance on some central processor or controlling level within the hierarchy means that reliability and flexibility are often compromised. Failure of this central unit leads to failure of the whole system, changes in the system often mean changes in both the central unit and in all related sub-units. Further, the burden placed on the central unit in terms of combining information can often still be prohibitive and lead to an inability of the design methodology to be extended to incorporate an increasing number of sources of information. Finally, the inability of information sources to communicate, other than through some higher level in the hierarchy, eliminates the possibility of any synergy being developed between two or more complimentary sources of information and restricts the system designer to rigid predetermined combinations of information. The limitations imposed by a strict hierarchy have been widely recognised both in human information processing systems as well as in computer-based systems.

4.1.2 Distributed Data Fusion Architectures

The move to more distributed, autonomous, organisations is clear in many information processing systems. This is most often motivated by two main considerations; the desire to make the system more modular and flexible, and a recognition that a centralised or hierarchical structure imposes unacceptable overheads on communication and central computation. The migration to distributed system organisations is most apparent in Artificial Intelligence (AI) application areas, where distributed AI has become a research area in its own right. Many of the most interesting distributed processing organisations have originated in this area.

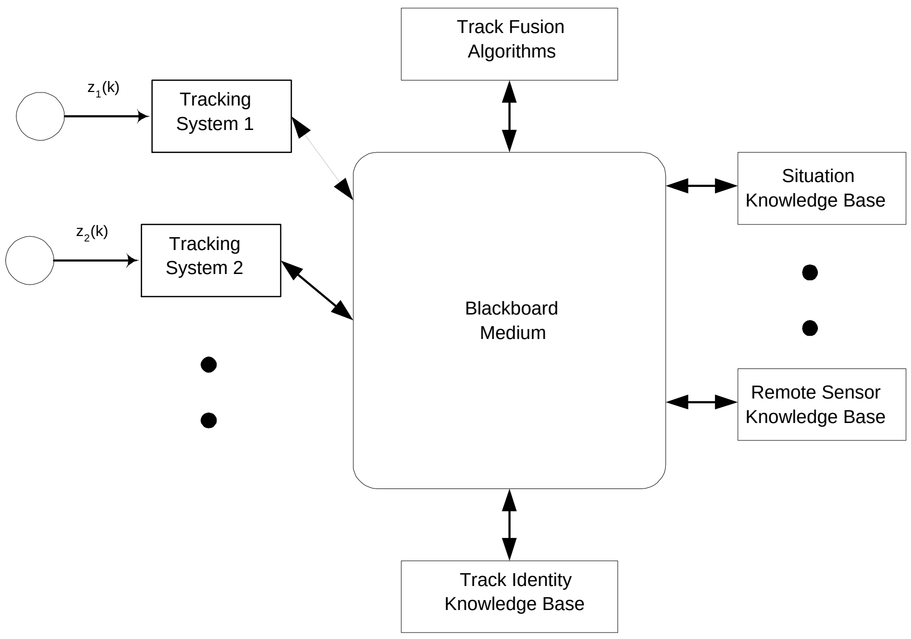


Figure 30: Blackboard Architecture in Data Fusion

Notable is the "Blackboard" architecture (see Figure 30), originally developed in the Hearsay speech understanding programme, but now widely employed in many areas of AI and data fusion research. A Blackboard architecture consists of a number of independent autonomous "agents". Each agent represents a source of expert knowledge or specialised information processing capability. Agents exchange information through a common communication facility or shared memory resource. This resource is called a blackboard. The blackboard is designed to closely replicate its physical analogue. Each agent is able to write information or local knowledge to this resource. Every agent in the system is able to read from this resource, in an unrestricted manner, any information which it considers useful in its current task. In principle, every agent can be made modular and new agents may be added to the system when needed without changing the underlying architecture or operation of the system as a whole. The flexibility of this approach to system organisation has made the Blackboard architecture popular in a range of application domains [33]. In data fusion, the Blackboard approach has been most widely used for knowledge-based data fusion systems in data interpretation and situation assessment (see [19] and [20] for example). However, the structured nature of tracking and identification problems does not lend itself to this anarchic organisational form.

The Blackboard architecture has a number of basic problems. All of these stem from the use of a common communication or memory resource. The core problem is that a central resource naturally entails the need for some type of central control in which a single decision maker is used to sequence and organise the reading and writing of information from the shared resource. Practically, with such a control mechanism, a blackboard architecture becomes no more than a one level hierarchy with consequent lack of flexibility and with the inherent limitations imposed by the use of a central resource.

4.1.3 Decentralised Data Fusion Architectures

A decentralized data fusion system consists of a network of sensor nodes, each with its own processing facility, which together do not require any central fusion or central communication facility. In such a system, fusion occurs locally at each node on the basis of local observations and the information communicated from neighbouring nodes. At no point is there a common place where fusion or global decisions are made.

A decentralised data fusion system is characterised by three constraints:

1. There is no single central fusion center; no one node should be central to the successful operation of the network.
2. There is no common communication facility; nodes cannot broadcast results and communication must be kept on a strictly node-to-node basis.
3. Sensor nodes do not have any global knowledge of sensor network topology; nodes should only know about connections in their own neighbourhood.

Figures 31 and 32 and 33 show three possible realisations of a decentralised data fusion system. The key point is that all these systems have no central fusion center (unlike the 'decentralised' systems often described in the literature which are actually typically distributed or hierarchical).

The constraints imposed provide a number of important characteristics for decentralised data fusion systems:

- Eliminating the central fusion center and any common communication facility ensures that the system is **scalable** as there are no limits imposed by centralized computational bottlenecks or lack of communication bandwidth.
- Ensuring that no node is central and that no global knowledge of the network topology is required for fusion means that the system can be made **survivable** to the on-line loss (or addition) of sensing nodes and to dynamic changes in the network structure.
- As all fusion processes must take place locally at each sensor site and no global knowledge of the network is required *a priori*, nodes can be constructed and programmed in a **modular** fashion.

These characteristics give decentralised systems a major advantage over more traditional sensing architectures, particularly in defense applications.

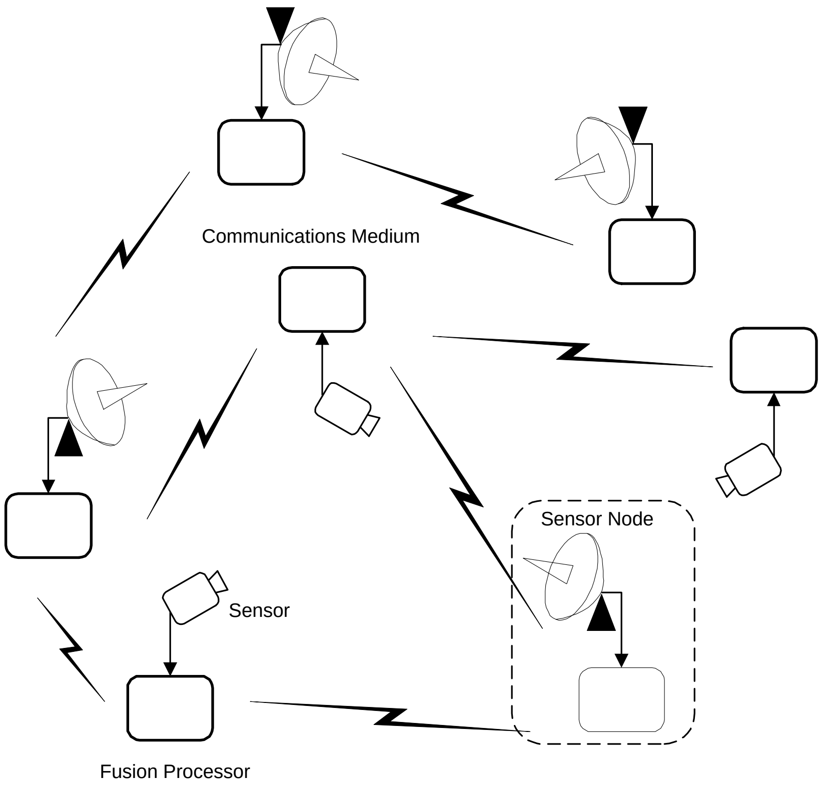


Figure 31: A decentralised data fusion system implemented with a point-to-point communication architecture.

A decentralized organization differs from a distributed processing system in having no central processing or communication facilities. Each sensor node in a decentralized

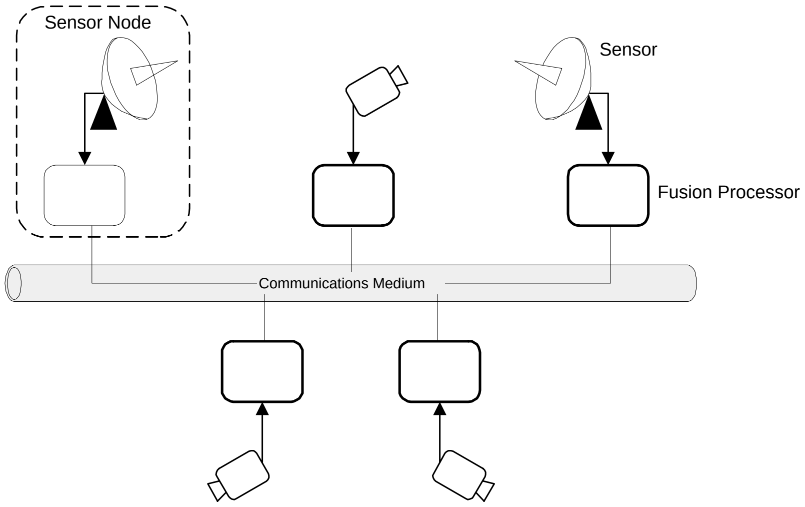


Figure 32: A decentralised data fusion system implemented with a broadcast, fully connected, communication architecture. Technically, a common communication facility violates decentralised data fusion constraints. However a broadcast medium is often a good model of real communication networks.

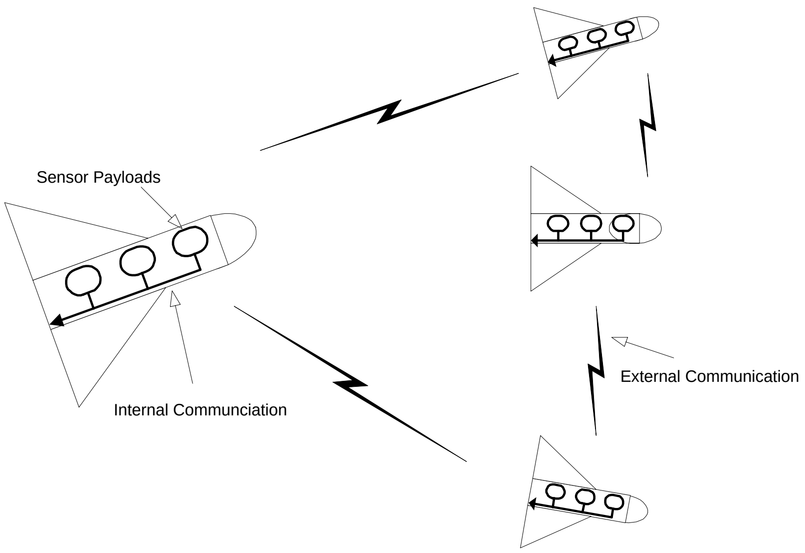


Figure 33: A decentralised data fusion system implemented with a hybrid, broadcast and point-to-point, communication architecture.

organization is entirely self contained and can operate completely independently of any other component in the system. Communication between nodes is strictly one-to-one and requires no remote knowledge of node capability. Throughout this section we distinguish between *decentralized* organizations that have no common resources, and *distributed* organizations where some residual centralized facility is maintained.

4.2 Decentralised Estimation

Decentralised data fusion is based on the idea of using formal *information* measures as the means of quantifying, communicating and assimilating sensory data. In decentralised estimation of continuous valued states, this is implemented in the form of an *information* filter. In this section, the full form of the information filter is derived. It is then demonstrated how the filter may be decentralised amongst a number of sensing nodes. Later sections then describe how to deal with issues of communication and data association in decentralised sensing

4.2.1 The Information Filter

Conventional Kalman filters deal with the estimation of states $\mathbf{x}(i)$, and yield estimates $\hat{\mathbf{x}}(i \mid j)$ together with a corresponding estimate variance $\mathbf{P}(i \mid j)$. The information filter deals instead with the information state vector $\hat{\mathbf{y}}(i \mid j)$ and information matrix $\mathbf{Y}(i \mid j)$ defined as

$$\hat{\mathbf{y}}(i \mid j) = \mathbf{P}^{-1}(i \mid j)\hat{\mathbf{x}}(i \mid j), \qquad \mathbf{Y}(i \mid j) = \mathbf{P}^{-1}(i \mid j). \tag{229}$$

These information quantities have an interpretation related to the underlying probability distributions associated with the estimation problem. The information matrix in particular is closely associated with the Fisher information measures introduced in Section 2.

A set of recursion equations for the information state and information matrix can be derived directly from the equations for the Kalman filter. The resulting information filter is mathematically identical to the conventional Kalman filter.

Recall the update stage for the Kalman filter:

$$\hat{\mathbf{x}}(k \mid k) = (\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k))\hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\mathbf{z}(k) \tag{230}$$

$$\mathbf{P}(k \mid k) = (\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k))\mathbf{P}(k \mid k-1)(\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k))^T + \mathbf{W}(k)\mathbf{R}(k)\mathbf{W}^T(k) \tag{231}$$

Now, from Equations 171 and 172, we have

$$\mathbf{1} - \mathbf{W}(k)\mathbf{H}(k) = \mathbf{P}(k \mid k)\mathbf{P}^{-1}(k \mid k-1), \tag{232}$$

and

$$\mathbf{W}(k) = \mathbf{P}(k \mid k)\mathbf{H}^T(k)\mathbf{R}^{-1}(k). \tag{233}$$

Substituting Equations 232 and 233 into Equation 230 gives

$$\hat{\mathbf{x}}(k \mid k) = \mathbf{P}(k \mid k)\mathbf{P}^{-1}(k \mid k-1)\hat{\mathbf{x}}(k \mid k-1) + \mathbf{P}(k \mid k)\mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k).$$

Pre-multiplying through by $\mathbf{P}^{-1}(k \mid k)$ gives the update equation for the information-state vector as

$$\mathbf{P}^{-1}(k \mid k)\hat{\mathbf{x}}(k \mid k) = \mathbf{P}^{-1}(k \mid k-1)\hat{\mathbf{x}}(k \mid k-1) + \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k). \tag{234}$$

Defining

$$\mathbf{i}(k) \stackrel{\triangle}{=} \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k) \tag{235}$$

as the information-state contribution from an observation $\mathbf{z}(k)$, and with the definitions in Equation 229, Equation 234 becomes

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k). \tag{236}$$

A similar expression can be obtained for the covariance update of Equation 231. Substituting Equations 232 and 233 into Equation 231 and rearranging gives

$$\mathbf{P}^{-1}(k \mid k) = \mathbf{P}^{-1}(k \mid k-1) + \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k). \tag{237}$$

Defining

$$\mathbf{I}(k) \stackrel{\triangle}{=} \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k) \tag{238}$$

as the information matrix associated with the observation, and with the definitions in Equation 229, Equation 237 becomes the information matrix update equation

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \mathbf{I}(k). \tag{239}$$

A comparison of the Kalman filter update stage (Equations 230 and 231) with the information filter update stage (Equations 236 and 239) highlights the simplicity of the information filter update over the Kalman filter update. Indeed, in information form, the update stage is a straight addition of information from a prediction and from an observation. It is this simplicity which gives the information filter it's advantage in multi-sensor estimation problems.

The simplicity of the update stage of the information filter comes at the cost of increased complexity in the prediction stage. Recall the covariance prediction equation

$$\mathbf{P}(k \mid k-1) = \mathbf{F}(k)\mathbf{P}(k-1 \mid k-1)\mathbf{F}^T(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^T(k) \tag{240}$$

To derive the prediction stage for the information filter, the following version of the matrix inversion lemma is noted [28]

$$\left(\mathbf{A} + \mathbf{B}^T\mathbf{C}\right)^T = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}^T\left(\mathbf{1} + \mathbf{C}\mathbf{A}^{-1}\mathbf{B}^T\right)^{-1}\mathbf{C}\mathbf{A}^{-1}.$$

Identifying

$$A = \mathbf{F}(k)\mathbf{P}(k-1 \mid k-1)\mathbf{F}^T(k), \qquad \mathbf{B}^T = \mathbf{G}(k)\mathbf{Q}(k), \qquad \mathbf{C} = \mathbf{G}(k),$$

the inverse of Equation 240 becomes

$$\mathbf{P}^{-1}(k \mid k-1) = \mathbf{M}(k) - \mathbf{M}(k)\mathbf{G}(k)\left[\mathbf{G}^T(k)\mathbf{M}(k)\mathbf{G}(k) + \mathbf{Q}^{-1}(k)\right]^{-1}\mathbf{G}^T(k)\mathbf{M}(k) \quad (241)$$

where

$$\mathbf{M}(k) = \mathbf{F}^{-T}(k)\mathbf{P}^{-1}(k-1 \mid k-1)\mathbf{F}^{-1}(k) \quad (242)$$

when $\mathbf{Q}(k)$ is non singular. Noting the definition of the state transition matrix

$$\mathbf{F}(k) \stackrel{\triangle}{=} \mathbf{\Phi}(t_k, t_{k-1})$$

implies $\mathbf{F}^{-1}(k)$ always exists and indeed

$$\mathbf{F}^{-1}(k) = \mathbf{\Phi}(t_{k-1}, t_k)$$

is simply the state transition matrix defined backwards from a time t_k to t_{k-1}. Now, defining

$$\mathbf{\Sigma}(k) \stackrel{\triangle}{=} \left[\mathbf{G}^T(k)\mathbf{M}(k)\mathbf{G}(k) + \mathbf{Q}^{-1}(k)\right], \quad (243)$$

and

$$\mathbf{\Omega}(k) \stackrel{\triangle}{=} \mathbf{M}(k)\mathbf{G}(k)\left[\mathbf{G}^T(k)\mathbf{M}(k)\mathbf{G}(k) + \mathbf{Q}^{-1}(k)\right]^{-1} = \mathbf{M}(k)\mathbf{G}(k)\mathbf{\Sigma}^{-1}(k), \quad (244)$$

the information matrix prediction equation becomes

$$\mathbf{Y}(k \mid k-1) = \mathbf{P}^{-1}(k \mid k-1) = \mathbf{M}(k) - \mathbf{\Omega}(k)\mathbf{\Sigma}(k)\mathbf{\Omega}^T(k). \quad (245)$$

A number of alternate expressions for the information prediction stage can also be derived:

$$\mathbf{Y}(k \mid k-1) = \left[\mathbf{1} - \mathbf{\Omega}(k)\mathbf{G}^T(k)\right]\mathbf{M}(k) \quad (246)$$

and

$$\mathbf{Y}(k \mid k-1) = \left[\mathbf{1} - \mathbf{\Omega}(k)\mathbf{G}^T\right]\mathbf{M}(k)\left[\mathbf{1} - \mathbf{\Omega}(k)\mathbf{G}^T\right]^T + \mathbf{\Omega}(k)\mathbf{Q}^{-1}(k)\mathbf{\Omega}^T(k). \quad (247)$$

The information-state prediction equations may also be obtained as

$$\begin{aligned}
\hat{\mathbf{y}}(k \mid k-1) &= \left[\mathbf{1} - \mathbf{\Omega}(k)\mathbf{G}^T(k)\right]\mathbf{F}^{-T}(k) \\
&\quad \times \left[\hat{\mathbf{y}}(k-1 \mid k-1) + \mathbf{Y}(k-1 \mid k-1)\mathbf{F}^{-1}(k)\mathbf{B}(k)\mathbf{u}(k)\right] \\
&= \left[\mathbf{1} - \mathbf{\Omega}(k)\mathbf{G}^T(k)\right]\left[\mathbf{F}^{-T}(k)\hat{\mathbf{y}}(k-1 \mid k-1) + \mathbf{M}(k)\mathbf{B}(k)\mathbf{u}(k)\right] \\
&= \left[\mathbf{1} - \mathbf{\Omega}(k)\mathbf{G}^T(k)\right]\mathbf{F}^{-T}(k)\hat{\mathbf{y}}(k-1 \mid k-1) \\
&\quad + \mathbf{Y}(k \mid k-1)\mathbf{B}(k)\mathbf{u}(k). \qquad (248)
\end{aligned}$$

It should be noted that the complexity of the inversion of $\mathbf{\Sigma}(k)$ is only of order the dimension of the driving noise (often scalar). Further $\mathbf{Q}(k)$ is almost never singular, and indeed if it were, singularity can be eliminated by appropriate definition of $\mathbf{G}(k)$. The special case in which $\mathbf{Q}(k) = \mathbf{0}$ yields prediction equations in the form:

$$\mathbf{Y}(k \mid k-1) = \mathbf{M}(k), \qquad \hat{\mathbf{y}}(k \mid k-1) = \mathbf{F}^{-T}(k)\hat{\mathbf{y}}(k-1 \mid k-1) + \mathbf{M}(k)\mathbf{B}(k)\mathbf{u}(k) \quad (249)$$

The information filter is now summarised
Prediction:

$$\hat{\mathbf{y}}(k \mid k-1) = \left[\mathbf{1} - \mathbf{\Omega}(k)\mathbf{G}^T(k)\right]\mathbf{F}^{-T}(k)\hat{\mathbf{y}}(k-1 \mid k-1) + \mathbf{Y}(k \mid k-1)\mathbf{B}(k)\mathbf{u}(k) \quad (250)$$

$$\mathbf{Y}(k \mid k-1) = \mathbf{M}(k) - \mathbf{\Omega}(k)\mathbf{\Sigma}(k)\mathbf{\Omega}^T(k) \quad (251)$$

where

$$\mathbf{M}(k) = \mathbf{F}^{-T}(k)\mathbf{Y}(k-1 \mid k-1)\mathbf{F}^{-1}(k), \quad (252)$$

$$\mathbf{\Omega}(k) = \mathbf{M}(k)\mathbf{G}(k)\mathbf{\Sigma}^{-1}(k), \quad (253)$$

and

$$\mathbf{\Sigma}(k) = \left[\mathbf{G}^T(k)\mathbf{M}(k)\mathbf{G}(k) + \mathbf{Q}^{-1}(k)\right]. \quad (254)$$

Estimate:

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k) \quad (255)$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \mathbf{I}(k). \quad (256)$$

where

$$\mathbf{i}(k) = \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k), \qquad \mathbf{I}(k) = \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k) \quad (257)$$

Given the interpretation of the information matrix $\mathbf{Y}(i \mid j)$ as the Fisher information, the update Equation 256 is simply seen to add the information contributed by the observation. Conversely, the prediction Equation 251 subtracts information caused by process uncertainties.

Example 27

Consider again Example 17 of constant velocity particle motion:

$$\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v(t).$$

where $v(t)$ is a zero mean white noise process with $\mathrm{E}\{v^2(t)\} = q$. The state transition matrix and its inverse over a time interval $\delta t$are given by

$$\mathbf{F}(\delta t) = \begin{bmatrix} 1 & \delta t \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{F}^{-1}(\delta t) = \begin{bmatrix} 1 & -\delta t \\ 0 & 1 \end{bmatrix}$$

and the noise transition matrix by (Equation 73)

$$\mathbf{G}(\delta t) = \int_0^{\delta t} \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathrm{d}\tau = \begin{bmatrix} \delta t^2/2 \\ \delta t \end{bmatrix}$$

Setting

$$\mathbf{Y}(k-1 \mid k-1) = \begin{bmatrix} Y_{xx} & Y_{xv} \\ Y_{xv} & Y_{vv} \end{bmatrix},$$

the propagated information is

$$\mathbf{M}(k) = \mathbf{F}^{-T}(k)\mathbf{Y}(k-1 \mid k-1)\mathbf{F}^{-1}(k) = \begin{bmatrix} Y_{xx} & Y_{xv} - Y_{xx}\delta t \\ Y_{xv} - Y_{xx} & Y_{xx}\delta t^2 - 2Y_{xv}\delta t + Y_{vv} \end{bmatrix}.$$

Then

$$\mathbf{\Sigma}(k) = \left[\mathbf{G}^T(k)\mathbf{M}(k)\mathbf{G}(k) + \mathbf{Q}^{-1}(k)\right] = \delta t^2 \left[Y_{xx}\delta t^2/4 - Y_{xv}\delta t + Y_{vv}\right] + q^{-1}$$

and

$$\begin{aligned} \mathbf{\Omega}(k) &= \mathbf{M}(k)\mathbf{G}(k)\mathbf{\Sigma}^{-1}(k) \\ &= \frac{1}{Y_{xx}\delta t^4/4 - Y_{xv}\delta t^3 + Y_{vv}\delta t^2 + q^{-1}} \begin{bmatrix} Y_{xv}\delta t - Y_{xx}\delta t^2/2 \\ Y_{xx}\delta t^3/2 - 3Y_{xv}\delta t^2/2 + Y_{vv} \end{bmatrix}. \end{aligned}$$

Finally, denoting

$$\mathbf{M}(k) = \begin{bmatrix} M_{xx} & M_{xv} \\ M_{xv} & M_{vv} \end{bmatrix},$$

the propagation gain matrix is found as

$$\begin{aligned} \mathbf{1} - \mathbf{\Omega}(k)\mathbf{G}^T(k) &= \frac{1}{\frac{\delta t^2}{4}M_{xx} - \delta t M_{xv} + M_{vv} + \frac{q^{-1}}{\delta t^2}} \\ &\quad \times \begin{bmatrix} M_{xv}\delta t/2 + M_{vv} + q^{-1}/\delta t^2 & M_{xx}\delta t/2 + M_{xv} \\ M_{xv}\delta t^2/4 + M_{vv}\delta t/2 & M_{xx} + M_{xv}\delta t/2 + q^{-1}/\delta t^2 \end{bmatrix}. \end{aligned}$$

Note also the term

$$\mathbf{F}^{-T}(k)\hat{\mathbf{y}}(k-1 \mid k-1) = \begin{bmatrix} 1 & 0 \\ -\delta t & 1 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} - \delta t y \end{bmatrix}$$

Assume observations are made of the position of the particle at discrete synchronous time intervals according to

$$\mathbf{z}(k) = [\,1 \quad 0\,] \begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + w(k)$$

where $w(t)$ is a zero mean white noise process with $\mathrm{E}\{w^2(t)\} = r$. *The information state is simply given by*

$$\mathbf{i}(k) = \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} r^{-1} z_x = \begin{bmatrix} z_x/r \\ 0 \end{bmatrix},$$

and the information matrix by

$$\mathbf{I}(k) = \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} r^{-1} [\,1 \quad 0\,] = \begin{bmatrix} 1/r & 0 \\ 0 & 0 \end{bmatrix}.$$

A key problem with this information state vector $\hat{\mathbf{y}}(i \mid j)$ *is that it has no obvious metric properties; the difference between two information-state vectors does not relate at all to the difference in state vectors because of scaling by the information matrix. This can make the interpretation of the information filter more difficult than the (state-based) Kalman filter.*

There is a clear duality between the information filter and the conventional Kalman filter, in which the prediction stage of the information filter is related to the update stage of the Kalman filter, and the update stage of the information filter to the prediction stage of the Kalman filter [2]. In particular, identifying the correspondences

$$\mathbf{\Omega}(k) \rightarrow \mathbf{W}(k), \quad \mathbf{\Sigma}(k) \rightarrow \mathbf{S}(k), \quad \mathbf{G}^T(k) \rightarrow \mathbf{H}(k)$$

$\mathbf{\Omega}(k)$ is seen to take on the role of an information prediction gain matrix, $\mathbf{\Sigma}(k)$ an 'information innovation matrix', and $\mathbf{G}^T(k)$ an 'information observation'. This duality is instructive in understanding the relationship of information to state as equivalent representations. It is also of assistance in implementation of the information filtering equations.

With this interpretation, The information-filter form has the advantage that the update Equations 255 and 256 for the estimator are computationally simpler than the corresponding equations for the Kalman Filter, at the cost of increased complexity in prediction. The value of this in decentralized sensing is that estimation occurs locally at each node, requiring partition of the estimation equations which are simpler in their information form. Prediction, which is more complex in this form, relies on a propagation coefficient which is independent of the observations made and so is again simpler to decouple and decentralize amongst a network of sensor nodes. This property is exploited in subsequent sections.

4.2.2 The Information Filter and Bayes Theorem

There is a strong relationship between the information state and the underlying probability distribution. Recall Bayes Theorem

$$P(\mathbf{x}(k) \mid \mathbf{Z}^k) = \frac{P(\mathbf{z}(k) \mid \mathbf{x}(k))P(\mathbf{x}(k) \mid \mathbf{Z}^{k-1})}{P(\mathbf{z}(k) \mid \mathbf{Z}^{k-1})}. \tag{258}$$

If it is assumed that the prior and likelihood are Gaussian as

$$P(\mathbf{x}(k) \mid \mathbf{Z}^k) \propto \exp\left\{-\frac{1}{2}\left[\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k)\right]^T \mathbf{P}^{-1}(k \mid k)\left[\mathbf{x}(k)k - \hat{\mathbf{x}}(k \mid k)\right]\right\}$$

$$P(\mathbf{z}(k) \mid \mathbf{x}(k)) \propto \exp\left\{-\frac{1}{2}\left[\mathbf{z}(k) - \mathbf{H}(k)\mathbf{x}(k)\right]^T \mathbf{R}^{-1}(k)\left[\mathbf{z}(k) - \mathbf{H}(k)\mathbf{x}(k)\right]\right\},$$

then the posterior will also be Gaussian in the form

$$P(\mathbf{x}(k) \mid \mathbf{Z}^{k-1}) \propto \exp\left\{-\frac{1}{2}\left[\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k-1)\right]^T \mathbf{P}^{-1}(k \mid k-1)\left[\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k-1)\right]\right\}.$$

Now, substituting these distributions into Equation 258 and take logs gives

$$\begin{aligned}
&[\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k)]^T \mathbf{P}^{-1}(k \mid k) [\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k)] = \\
&\qquad [\mathbf{z}(k) - \mathbf{H}(k)\mathbf{x}(k)]^T \mathbf{R}^{-1}(k) [\mathbf{z}(k) - \mathbf{H}(k)\mathbf{x}(k)] \\
&\quad + \; [\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k-1)]^T \mathbf{P}^{-1}(k \mid k-1) [\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k-1)] + C(\mathbf{z}(k)) \qquad (259)
\end{aligned}$$

where $C(\mathbf{z}(k))$ is independent of $\mathbf{x}(k)$. This equation relates the log likelihoods of these Gaussian distributions and is essentially a quadratic in $\mathbf{x}(k)$. Now, differentiating this expression once with respect to $\mathbf{x}(k)$ and rearranging gives Equation 255. Differentiating a second time and rearranging gives

$$\mathbf{P}^{-1}(k \mid k) = \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k) + \mathbf{P}^{-1}(k \mid k-1),$$

which is Equation 256. The first derivative of the log-likelihood is known as the score function. The second derivative is, of course, the Fisher information Equation 50.

This analysis suggests that the information filter is, fundamentally, a log likelihood implementation of Bayes Theorem. The first and second derivatives of the log likelihood are essentially moment generating functions (the first derivative is the centre of mass, and the second, the moment of inertia [34]). The information filter is thus a means for recursive calculation of sufficient statistics (the mean and variance) in the case when the distributions of interest our Gaussian.

This interpretation of the information filter shows the relationship between the Kalman filter and the more general probabilistic estimation problem. Indeed, if the distributions were not Gaussian and indeed possibly discrete, the information filter would reduce to the general fusion of log likelihoods as described in Section 2.2.6. This can be exploited in the development of efficient distributed data fusion algorithms for problems which are not linear or Gaussian.

4.2.3 The Information filter in Multi-Sensor Estimation

The information filter is reasonably well known in the literature on estimation [2, 28]. However, its use in data fusion has been largely neglected in favour of conventional state-based Kalman filtering methods. The reasons for this appear to be somewhat spurious, based largely on the incorrect hypothesis that it is “cheaper” to communicate innovation information (of dimension the observation vector) than to communicate information state vectors (of dimension the state vector). The basic problem is that it is generally not possible to extend Equations 230 and 231 in any simple way to deal with multiple observations. The reason for this, as we have seen, is that although the innovation vectors at different times are uncorrelated (by construction), the innovations generated by different sensors at the same time *are* correlated, by virtue of the fact that they use a common prediction. Thus, the innovation covariance matrix $\mathbf{S}(k)$ can never be diagonal, and so can not be simply partitioned and inverted to yield a gain matrix for each individual observation.

Thus, for a set of sensors $i = 1, \cdots, N$, it is not possible to compute the simple sum of innovations as

$$\hat{\mathbf{x}}(k \mid k) \neq \hat{\mathbf{x}}(k \mid k-1) + \sum_{i=1}^{N} \mathbf{W}_i(k) \left[\mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}(k \mid k-1)\right]$$

in which the individual gains are given by

$$\mathbf{W}_i(k) = \mathbf{P}(k \mid k-1)\mathbf{H}_i^T(k)\mathbf{S}_i^{-1}(k).$$

However, the information filter provides a direct means of overcoming these problems. As will be seen, for the same set of sensors, $i = 1, \cdots, N$, and without any additional assumptions, it is the case that the information contributions from all sensors can simply be added to obtain an updated estimate in the form

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \sum_{j=1}^{N} \mathbf{i}_j(k),$$

which is algebraically equivalent to a full (all sensor) state-based Kalman filter estimate.

The reason why this can be done with the information filter is that the information contributions made by the observations are directly related to the underlying likelihood functions for the states rather than to the state estimates themselves. This can be appreciated by considering the interpretation of the information filter directly an implementation of Bayes theorem in terms of log-likelihood rather than in terms of state. Indeed, making the usual assumption that the observations made by the various sensors are conditionally independent given the true state as

$$P(\mathbf{z}_1(k)\cdots, \mathbf{z}_N(k) \mid \mathbf{x}(k)) = \prod_{i=1}^{N} P(\mathbf{z}_i(k) \mid \mathbf{x}(k)), \tag{260}$$

then Bayes Theorem gives (Equation 19)

$$P(\mathbf{x}(k) \mid \mathbf{Z}^n(k)) = P(\mathbf{x}(k) \mid \mathbf{Z}^n(k-1)) \prod_{i=1}^{n} P(\mathbf{z}_i(k) \mid \mathbf{x}(k)).[P(\mathbf{Z}^n(k) \mid \mathbf{Z}^n(k-1))]^{-1}. \tag{261}$$

Taking logs of Equation 261 then gives

$$\begin{aligned} \ln P(\mathbf{x}(k) \mid \mathbf{Z}^n(k)) &= \ln P(\mathbf{x}(k) \mid \mathbf{Z}^n(k-1)) \\ &\quad + \sum_{i=1}^{n} \ln P(\mathbf{z}_i(k) \mid \mathbf{x}(k)) - \ln P(\mathbf{Z}^n(k) \mid \mathbf{Z}^n(k-1)). \end{aligned} \tag{262}$$

Given Equation 259, an identification can be made as

$$\sum_{i=1}^{n} \ln P(\mathbf{z}_i(k) \mid \mathbf{x}(k)) \rightleftharpoons \left\{\textstyle\sum_{i=1}^{n} \mathbf{i}(k), \quad \sum_{i=1}^{n} \mathbf{I}(k)\right\}. \tag{263}$$

This demonstrates why, fundamentally, it is possible to add information states and Information matrices from different sensors while it is not possible to add innovations without accounting for cross-correlations. For this reason also, the information filter is occasionally referred to as the likelihood filter.

Computationally, the information filter thus provides a far more natural means of assimilating information than does the conventional Kalman filter and a far simpler method of dealing with complex multi-sensor data fusion problems.

The linear addition of information in the information filter can also be obtained by direct algebraic manipulation. Consider a system comprising N sensors each taking observations according to

$$\mathbf{z}_i(k) = \mathbf{H}_i(k)\mathbf{x}(k) + \mathbf{v}_i(k) \tag{264}$$

with

$$\mathrm{E}\{\mathbf{v}_p(i)\mathbf{v}_q^T(j)\} \;= \delta_{ij}\delta_{pq}\mathbf{R}_p(i). \tag{265}$$

The observations can be stacked into a composite observation

$$\mathbf{z}(k) = \left[\mathbf{z}_1^T(k), \cdots, \mathbf{z}_N^T(k)\right]^T \tag{266}$$

The observation model can also be stacked into a composite model

$$\mathbf{H}(k) = \left[\mathbf{H}_1^T(k), \cdots, \mathbf{H}_N^T(k)\right]^T, \tag{267}$$

and

$$\mathbf{v}(k) = \left[\mathbf{v}_1^T(k), \cdots, \mathbf{v}_N^T(k)\right]^T \tag{268}$$

to give a composite observation equation in familiar form

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k).$$

Noting that

$$\mathrm{E}\{\mathbf{v}(k)\mathbf{v}^T(k)\} \;= \mathbf{R}(k) = \mathrm{blockdiag}\{\mathbf{R}_1^T(k), \cdots, \mathbf{R}_N^T(k)\}\ , \tag{269}$$

a multiple sensor form of Equation 257 can be obtained as

$$\mathbf{i}(k) = \sum_{i=1}^{N} \mathbf{i}_i(k) = \sum_{i=1}^{N} \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{z}_i(k) \tag{270}$$

and

$$\mathbf{I}(k) = \sum_{i=1}^{N} \mathbf{I}_i(k) = \sum_{i=1}^{N} \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{H}_i(k) \tag{271}$$

where

$$\mathbf{i}_i(k) \triangleq \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{z}_i(k) \tag{272}$$

is the information-state contribution from observation $\mathbf{z}_i(k)$ and

$$\mathbf{I}_i(k) \triangleq \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{H}_i(k) \tag{273}$$

is its associated information matrix.

Equations 270 and 271 show that the total information available to the filter at any time-step is simply the sum of the information contributions from each of the individual sensors. Further, Equations 255 and 256 describing the single sensor information update are easily extended to multiple sensor updates in a straight-forward manner as

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \sum_{i=1}^{N} \mathbf{i}_i(k) \tag{274}$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \sum_{i=1}^{N} \mathbf{I}_i(k). \tag{275}$$

These equations should immediately be compared to the considerably more complex multiple sensor form of the Kalman filter update Equations.

4.2.4 The Hierarchical Information Filter

It is now shown how the information filter may be partitioned to provide a simple hierarchical estimation architecture based first on the communication of the information terms $\mathbf{i}(\cdot)$ and $\mathbf{I}(\cdot)$ from sensor nodes to a common fusion center, and second on the communication of partial information-state estimates from nodes to a central assimilation point. The latter case corresponds to the algorithm developed in [21]. In Section 2.2.6 it was demonstrated that, because of simple information summation, the log-likelihood form of Bayes theorem can readily be mapped to a number of different architectural forms. For the same reasons, the information additions in Equations 274 and 275 can also be distributed in a simple manner.

One hierarchical architecture that employs this additive property is shown in Figure 34 (this is the information filter form of Figure 8). Each sensor incorporates a full state model and takes observations according to Equation 264. They all calculate an information-state contribution from their observations in terms of $\mathbf{i}_i(k)$ and $\mathbf{I}_i(k)$. These are then communicated to the fusion centre and are incorporated into the global estimate through Equations 274 and 275. The information-state prediction is generated centrally using Equations 250 and 251 and the state estimate itself may be found at any stage from $\hat{\mathbf{x}}(i \mid j) = \mathbf{Y}^{-1}(i \mid j)\hat{\mathbf{y}}(i \mid j)$. To avoid communicating predictions to nodes, any validation or data association should take place at the fusion center.

A second hierarchical system which allows local tracks to be maintained at local sensor sites, is shown in Figure 35 (this is the information filter form Figure 9). In this system, each sensing node produces local information-state estimates on the basis of its own observations and communicates these back to a central fusion center where they are assimilated to provide a global estimate of information-state. Let $\tilde{\mathbf{y}}_i(\cdot \mid \cdot)$ be the information-state estimate arrived at by each sensor site based only on its own observations and local information-state prediction $\hat{\mathbf{y}}_i(k \mid k-1)$. This local estimate can be found from a local form of Equations 255 and 256 as

$$\tilde{\mathbf{y}}_i(k \mid k) = \hat{\mathbf{y}}_i(k \mid k-1) + \mathbf{i}_i(k) \tag{276}$$

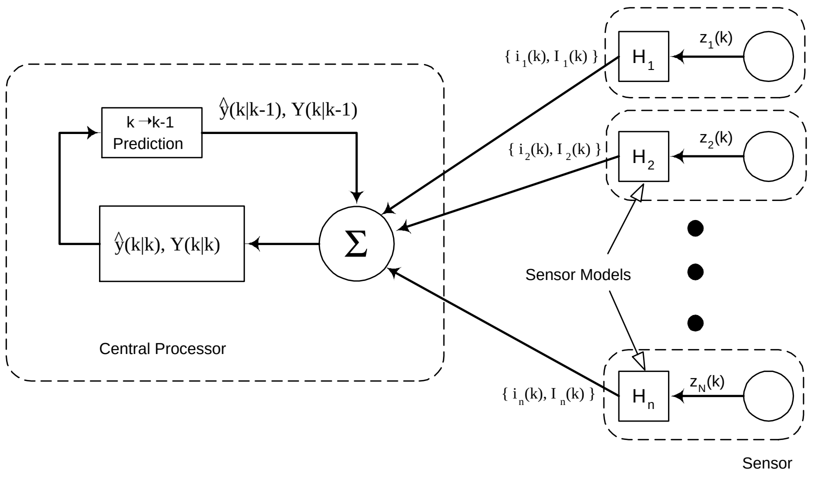


Figure 34: A hierarchical data fusion system where information-state contributions are calculated at each sensor node and transmitted to a central fusion center where a common estimate is obtained by simple summation. All state predictions are undertaken at the central processor.

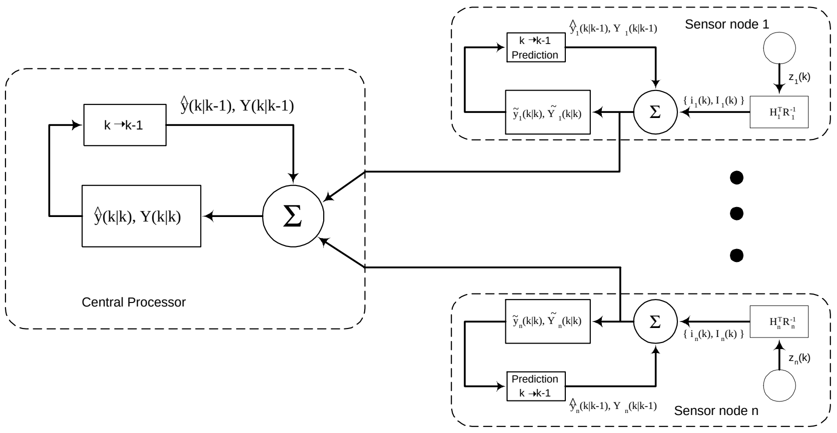


Figure 35: A hierarchical data fusion system where tracks are formed locally and communicated to a central fusion site. Each sensor node undertakes a prediction stage and maintains a track based only on it's local observations. The central processor fuses these tracks.

and

$$\tilde{\mathbf{Y}}_i(k \mid k) = \mathbf{Y}_i(k \mid k-1) + \mathbf{I}_i(k). \tag{277}$$

These partial information-state estimates are communicated to a fusion center where they can be assimilated according to

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \sum_{i=1}^{N} [\tilde{\mathbf{y}}_i(k \mid k) - \hat{\mathbf{y}}(k \mid k-1)] \tag{278}$$

and

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \sum_{i=1}^{N} \left[\tilde{\mathbf{Y}}_i(k \mid k) - \mathbf{Y}(k \mid k-1)\right]. \tag{279}$$

With the assumption that the local predictions at each node are the same as the prediction produced by a central fusion center, it can be seen that these assimilation equations are identical to Equations 274 and 275.

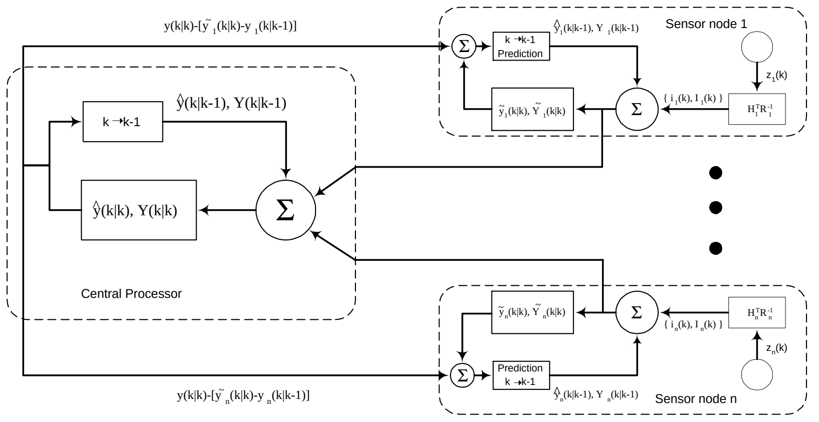


Figure 36: A hierarchical data fusion system where tracks are formed locally and communicated to a central fusion site, then track updates are communicated back to the sensor nodes. Each sensor node undertakes a prediction stage and maintains a global track based on the observations of all sensor nodes.

A third hierarchical system which allows global tracks to be maintained at local sensor sites is shown in Figure 36. This architecture is similar to that shown in Figure 35 except that once a global estimate is obtained at the central fusion site, it is communicated back to the local site where it is assimilated to form a global estimate. The advantage of this architecture is that each site can now act in an "autonomous" manner with access to global track information.

There are a number of other possible implementations of these hierarchical estimation equations, depending on where it is most efficient to generate predictions and where different parts of the system model reside. The important point to note is that the information filter provides a simple and natural method of mapping estimation equations to different architectures.

4.2.5 The Decentralised Information Filter

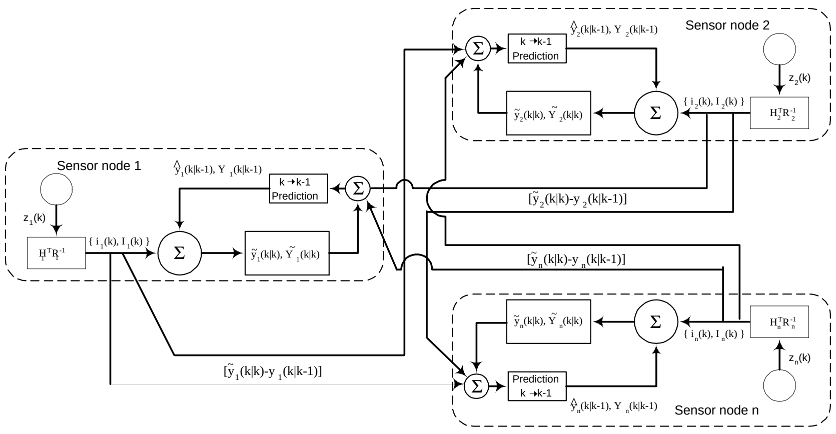

Sensor node 1
Sensor node 2
Sensor node n
k→k-1 Prediction
Prediction k→k-1
$\hat{y}_1(k|k-1), Y_1(k|k-1)$
$\hat{y}_2(k|k-1), Y_2(k|k-1)$
$\hat{y}_n(k|k-1), Y_n(k|k-1)$
$z_1(k)$
$z_2(k)$
$z_n(k)$
$H_1^T R_1^{-1}$
$H_2^T R_2^{-1}$
$H_n^T R_n^{-1}$
$\{ i_1(k), I_1(k) \}$
$\{ i_2(k), I_2(k) \}$
$\{ i_n(k), I_n(k) \}$
$\tilde{y}_1(k|k), \tilde{Y}_1(k|k)$
$\tilde{y}_2(k|k), \tilde{Y}_2(k|k)$
$\tilde{y}_n(k|k), \tilde{Y}_n(k|k)$
$[\tilde{y}_1(k|k)-y_1(k|k-1)]$
$[\tilde{y}_2(k|k)-y_2(k|k-1)]$
$[\tilde{y}_n(k|k)-y_n(k|k-1)]$
Σ

Figure 37: A Decentralized, fully connected, data fusion system where tracks are formed locally and communicated to neighbouring nodes where they are assimilated to provide global track information. Each sensor node undertakes a prediction stage and maintains a global track based on the observations of all sensor nodes. This architecture is also directly equivalent to a broadcast or bus communication system.

It is a relatively simple step to decentralize the assimilation Equations 278 and 279 in systems where there is a fully connected network of sensing nodes as shown in Figure 37. In this type of system, each node generates a prediction, takes an observation and computes a local estimate which is communicated to all neighbouring nodes. Each node receives all local estimates and implements a local form of the assimilation equations to produce a global estimate of information-state, equivalent to that obtained by a central fusion center. In effect, this is the same as *replicating* the central assimilation equations of Figure 36 at every local sensor site, and then simplifying the resulting equations.

It is assumed that each local sensor site or node maintains a state-space model $(\mathbf{F}(k), \mathbf{G}(k), \mathbf{Q}(k))$ identical to an equivalent centralized model so that $\hat{\mathbf{y}}_i(\cdot \mid \cdot) \equiv \hat{\mathbf{y}}(\cdot \mid \cdot)$ for all $i = 1, \cdots, N$. Each node begins by computing a local estimate $\tilde{\mathbf{y}}_i(k \mid k)$ based on a local prediction $\hat{\mathbf{y}}_i(k \mid k-1)$ and the observed local information $\mathbf{i}_i(k)$ according to (Equations 250 and 251)

Prediction:

$$\hat{\mathbf{y}}_i(k \mid k-1) = \left[\mathbf{1} - \mathbf{\Omega}_i(k)\mathbf{G}^T(k)\right] \mathbf{F}^{-T}(k)\hat{\mathbf{y}}_i(k-1 \mid k-1) + \mathbf{Y}_i(k \mid k-1)\mathbf{B}(k)\mathbf{u}(k) \tag{280}$$

$$\mathbf{Y}_i(k \mid k-1) = \mathbf{M}_i(k) - \mathbf{\Omega}_i(k)\mathbf{\Sigma}_i(k)\mathbf{\Omega}_i^T(k) \tag{281}$$

where

$$\mathbf{M}_i(k) = \mathbf{F}^{-T}(k)\mathbf{Y}_i(k-1 \mid k-1)\mathbf{F}^{-1}(k), \tag{282}$$

$$\mathbf{\Omega}_i(k) = \mathbf{M}_i(k)\mathbf{G}(k)\mathbf{\Sigma}_i^{-1}(k), \tag{283}$$

and

$$\mathbf{\Sigma}_i(k) = \left[\mathbf{G}^T(k)\mathbf{M}_i(k)\mathbf{G}(k) + \mathbf{Q}^{-1}(k)\right]. \tag{284}$$

Estimate:

$$\tilde{\mathbf{y}}_i(k \mid k) = \hat{\mathbf{y}}_i(k \mid k-1) + \mathbf{i}_i(k) \tag{285}$$

$$\tilde{\mathbf{Y}}_i(k \mid k) = \mathbf{Y}_i(k \mid k-1) + \mathbf{I}_i(k). \tag{286}$$

these partial information-state estimates are then communicated to neighbouring nodes where they are assimilated according to
Assimilate:

$$\hat{\mathbf{y}}_i(k \mid k) = \hat{\mathbf{y}}_i(k \mid k-1) + \sum_{j=1}^{N} \left[\tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_j(k \mid k-1)\right] \tag{287}$$

$$\mathbf{Y}_i(k \mid k) = \mathbf{Y}_i(k \mid k-1) + \sum_{j=1}^{N} \left[\tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_j(k \mid k-1)\right] \tag{288}$$

If each node begins with a common initial information-state estimate $\hat{\mathbf{y}}_j(0 \mid 0) = \mathbf{0}$, $\mathbf{Y}_j(0 \mid 0) = \mathbf{0}$ and the network is fully connected, then the estimates obtained by each node will be identical.

The quantities communicated between sensor nodes; $(\tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_j(k \mid k-1))$ and $(\tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_j(k \mid k-1))$, consist of the difference between the local information at a time k and the prediction based only on information up to time $k-1$. This can be interpreted as the *new* information obtained by that node at the current time step. Indeed, the communicated terms are algebraically equivalent to $\mathbf{i}_j(k)$ and $\mathbf{I}_j(k)$; logically the new information available at a time k is just the information obtained through observation at that time. Thus, the operation of the sensor network can be envisioned as a group of local estimators which communicate new, independent, information between each other and which assimilate both local and communicated information to individual obtain a globally optimal local estimate.

There are three interesting points that can be made about these decentralized equations:

- The additional computation required of each node to assimilate information from adjacent nodes is small; a summation of vectors in Equation 287 and a summation of matrices in Equation 288. This is a direct consequence of the use of the information form of the Kalman filter which places the computational burden on the generation of predictions.

- The amount of communication that needs to take place is actually *less* than is required in a hierarchical organization. This is because each node individually computes a global estimate so that there is no need for estimates or predictions to be communicated prior to an estimation cycle. This results an a halving of required communication bandwidth, which may be further improved if model distribution is incorporated.

- The assimilation equations are the same as those that would be obtained in a system with distributed sensing nodes and a broadcast communication system.

The algorithm defined by Equations 280–288 is appropriate for both fully connected sensor networks or for sensors connected to a broadcast communication facility (such as a bus or Blackboard).

4.3 Decentralised Multi-Target Tracking

4.3.1 Decentralised Data Association

Data association in distributed systems is a complex problem. The reason for this is that hard association decisions made locally, in an optimal manner with respect to local observations, may not be optimal at the global level when all sensor information is made available. Further, an incorrect association decision is almost impossible to undo once data has been fused into a track.

The normal approach to this problem is to maintain both a local and a global track file and periodically to synchronize the two (see for example [9] pp602–605). Alternative approaches involve using either probabilistic data association methods or multiple-hypothesis trackers both of which avoid the need to make hard local association decisions (as described in Section 3.4).

All data association methods require that the normalised innovation is made available at each of the local processor nodes. In a decentralised data fusion architecture the information transmitted from one node to another is in the form of the information-theoretic quantities $\mathbf{i}(k)$ and $\mathbf{I}(k)$. To implement a local validation procedure it is therefore necessary to derive an expression from the prediction and communication terms which allow computation of a normalised innovation by every node on the basis of local information $\hat{\mathbf{y}}(k \mid k-1)$ and $\mathbf{Y}(k \mid k-1)$. The result is *normalised information residual* from which an *information gate*, equivalent to the innovation gate, can be obtained.

To formulate the information gate, the inverse of the information matrix $\mathbf{I}(k)$ is required. Generally, the dimension of the observation vector is less than that of the state vector, so the information matrix $\mathbf{I}(k) = \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k)$ is singular since it has rank n_z, equal to the size of the observation vector, but has dimension $n_x \times n_x$, the size of the state vector; and generally $n_x > n_z$. Consequently the inverse information matrix (the corresponding covariance matrix) is not well defined and instead the generalised-inverse $\mathbf{I}^\dagger(k)$. The generalised-inverse is defined such that

$$\mathbf{I}(k)\mathbf{I}^\dagger(k) = \mathbf{E}, \tag{289}$$

where $\mathbf{E}$ is an idempotent matrix which acts as the identity matrix for the information matrix and its generalised-inverse [25, 36]

$$\mathbf{I}(k)\mathbf{E} = \mathbf{I}(k), \qquad \mathbf{I}^\dagger(k)\mathbf{E} = \mathbf{I}^\dagger(k), \tag{290}$$

so that

$$\mathbf{I}(k)\mathbf{I}^\dagger(k)\mathbf{I}(k) = \mathbf{I}(k), \qquad \mathbf{I}^\dagger(k)\mathbf{I}(k)\mathbf{I}^\dagger(k) = \mathbf{I}^\dagger(k). \tag{291}$$

Amongst all possible generalised inverses that satisfy Equation 289, 290 and 291, the most appropriate definition is that which projects $\mathbf{I}(k)$ into the observation space in the following form;

$$\mathbf{I}^{\dagger}(k) \stackrel{\triangle}{=} \mathbf{H}^T(k)\left[\mathbf{H}(k)\mathbf{I}(k)\mathbf{H}^T(k)\right]^{-1}\mathbf{H}(k) \tag{292}$$

This generalised inverse exploits the role of $\mathbf{H}(k)$ as a projection operation, taking state space to observation space, and $\mathbf{H}^T(k)$ as a back-projection, taking observation space to state space [42]. Both projections do not change the content of the projected matrix so they can be applied without modifying the information contribution. Multiplying Equation 292 by $\mathbf{H}(k)\mathbf{I}(k)$ demonstrates this

$$\mathbf{H}(k)\mathbf{I}(k)\mathbf{I}^{\dagger}(k) = \mathbf{H}(k)\mathbf{I}(k)\mathbf{H}^T(k)\left[\mathbf{H}(k)\mathbf{I}(k)\mathbf{H}^T(k)\right]^{-1}\mathbf{H}(k) = \mathbf{H}(k) \tag{293}$$

The innovation measure is essential for data association and fault detection. The innovation is based on the difference between a predicted and observed measurement

$$\nu(k) = \mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1). \tag{294}$$

In information form, the measurement information is provided by the information vector $\mathbf{i}(k) = \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k)$. The information residual vector is therefore defined analogously

$$\upsilon(k) \stackrel{\triangle}{=} \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\nu(k), \tag{295}$$

which is simply the innovation $\nu(k)$ projected into information space. Substituting Equation 294 into Equation 295 gives

$$\upsilon(k) = \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k) - \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1) \tag{296}$$

or

$$\upsilon(k) = \mathbf{i}(k) - \mathbf{I}(k)\mathbf{Y}^{-1}(k \mid k-1)\hat{\mathbf{y}}(k \mid k-1) \tag{297}$$

The information residual variance is computed from

$$\boldsymbol{\Upsilon}(k) = \mathrm{E}\{\upsilon(k)\upsilon^T(k) \mid \mathbf{Z}^{k-1}(k)\}\ . \tag{298}$$

Substituting in Equation 295 gives

$$\begin{aligned}
\boldsymbol{\Upsilon}(k) &= \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathrm{E}\{\nu(k)\nu^T(k) \mid \mathbf{Z}^{k-1}(k)\}\ \mathbf{R}^{-1}(k)\mathbf{H}(k) \\
&= \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\left[\mathbf{H}^T(k)\mathbf{P}(k \mid k-1)\mathbf{H}(k) + \mathbf{R}(k)\right]\mathbf{R}^{-1}(k)\mathbf{H}(k) \\
&= \mathbf{I}(k) + \mathbf{I}(k)\mathbf{Y}^{-1}(k \mid k-1)\mathbf{I}(k) \\
&= \mathbf{I}(k)\left[\mathbf{I}^{\dagger}(k) + \mathbf{Y}^{-1}(k \mid k-1)\right]^{-1}\mathbf{I}(k)
\end{aligned} \tag{299}$$

The normalised information residual can now be computed from

$$\boldsymbol{\Gamma}(k) = \upsilon^T(k)\boldsymbol{\Upsilon}^{\dagger}(k)\upsilon(k) \tag{300}$$

noting that the pseudo-inverse for $\mathbf{\Upsilon}(k)$ is

$$\begin{aligned}\mathbf{\Upsilon}^{\dagger}(k) &= \mathbf{H}^T(k)\left[\mathbf{H}(k)\mathbf{\Upsilon}(k)\mathbf{H}^T(k)\right]^{-1}\mathbf{H}(k)\\ &= \mathbf{H}^T(k)\left[\mathbf{H}(k)\mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{S}(k)\mathbf{R}^{-1}(k)\mathbf{H}(k)\mathbf{H}^T(k)\right]^{-1}\mathbf{H}(k)\\ &= \mathbf{H}^T(k)\left[\mathbf{H}(k)\mathbf{H}^T(k)\right]^{-1}\mathbf{R}(k)\mathbf{S}^{-1}(k)\mathbf{R}(k)\left[\mathbf{H}(k)\mathbf{H}^T(k)\right]^{-1}\mathbf{H}(k) \qquad (301)\end{aligned}$$

and substituting Equations 295 and 301 into Equation 300 gives

$$\begin{aligned}\upsilon^T(k)\mathbf{\Upsilon}^{\dagger}(k)\upsilon(k) &= \nu^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k)\mathbf{H}^T(k)\left[\mathbf{H}(k)\mathbf{H}^T(k)\right]^{-1}\mathbf{R}(k)\\ &\quad\times\mathbf{S}^{-1}(k)\mathbf{R}(k)\left[\mathbf{H}(k)\mathbf{H}^T(k)\right]^{-1}\mathbf{H}(k)\mathbf{H}^T(k)\mathbf{R}^{-1}(k)\nu(k)\\ &= \nu^T(k)\mathbf{S}^{-1}(k)\nu(k). \qquad (302)\end{aligned}$$

That is, the normalised information residual is identically the conventional (observation) residual.

Thus to implement a data association or gating policy for a decentralised sensing network using the information filter, a normalised gate (or modified log-likelihood) can be constructed using Equations 297 and 299. This gate will be identical to the gate used in conventional multi-target tracking algorithms. Once the gate is established, it becomes possible to use any of the data association methods described in Section 3.4.

4.3.2 Decentralised Identification and Bayes Theorem

Decentralised data fusion principles can be easily extended to situations in which the underlying probability densities are not Gaussian and indeed where the densities are discrete. This provides a means of implementing decentralised (discrete) identification algorithms. The method is based on the use of log-likelihoods and extends the hierarchical log-likelihood architectures described in Section 2.2.6.

Recall the recursive form of Bayes theorem

$$P(\mathbf{x} \mid \mathbf{Z}^k) = \frac{P(\mathbf{z}(k) \mid \mathbf{x})P(\mathbf{x} \mid \mathbf{Z}^{k-1})}{P(\mathbf{z}(k) \mid \mathbf{Z}^{k-1})}, \qquad (303)$$

which may be written in terms of log-likelihoods as

$$\ln P(\mathbf{x} \mid \mathbf{Z}^k) = \ln P(\mathbf{x} \mid \mathbf{Z}^{k-1}) + \ln\frac{P(\mathbf{z}(k) \mid \mathbf{x})}{P(\mathbf{z}(k) \mid \mathbf{Z}^{k-1})}, \qquad (304)$$

where $\mathbf{x}$ is the state to be estimated and $\mathbf{Z}^k$ is the set of observations up to the k^{th} timestep. As has been previously demonstrated, the information form of the Kalman filter, for example, can be derived directly from this expression.

In Equation 304, the term $\ln P(\mathbf{x} \mid \mathbf{Z}^{k-1})$ corresponds to information accumulated about the state up to time $k-1$. The term

$$\ln\frac{P(\mathbf{z}(k) \mid \mathbf{x})}{P(\mathbf{z}(k) \mid \mathbf{Z}^{k-1})}$$

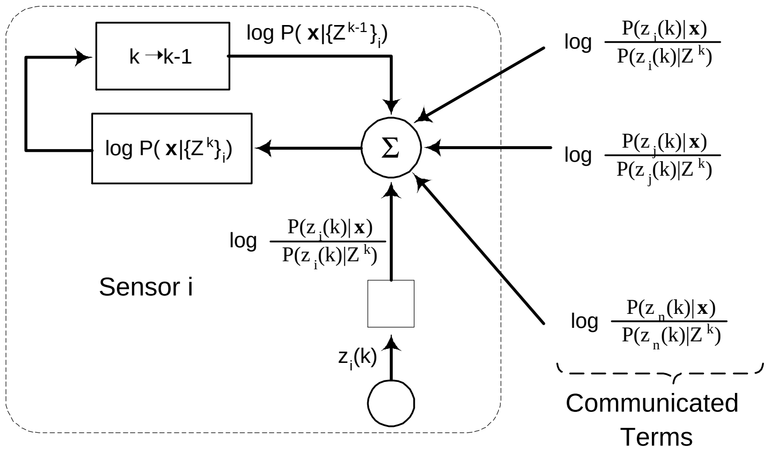

k → k-1
log P(**x**|{Z^{k-1}}_i)
log P(**x**|{Z^k}_i)
Σ
log P(z_i(k)|**x**) / P(z_i(k)|Z^k)
Sensor i
z_i(k)
log P(z_i(k)|**x**) / P(z_i(k)|Z^k)
log P(z_j(k)|**x**) / P(z_j(k)|Z^k)
log P(z_n(k)|**x**) / P(z_n(k)|Z^k)
Communicated Terms

Figure 38: A Decentralized, fully connected, Bayesian data fusion system appropriate for decentralised identification tasks.

corresponds to the new information generated at time k. Equation 304 exactly represents the communication requirements of a fully connected decentralised system, in which each node communicates a likelihood based on its own observation to all others and receives the likelihoods of all its neighbours which are then fused to form a global estimate. Rewriting Equation 304 in terms of an individual node i, this global estimate is computed as

$$\ln P(\mathbf{x}_i \mid \mathbf{Z}^k) = \ln P(\mathbf{x}_i \mid \mathbf{Z}^{k-1}) + \sum_j \ln \frac{P(\mathbf{z}_j(k) \mid \mathbf{x}_j)}{P(\mathbf{z}_j(k) \mid \mathbf{Z}^{k-1})}, \tag{305}$$

where the summation represents the communicated terms. This is illustrated for a node i in Figure 38.

4.4 Communication in Decentralised Sensing Systems

The decentralised data fusion algorithms described so far are implicitly limited in requiring full communication, either as a fully connected sensing network or as a broadcast system. The reason for this is that it is assumed that all new information in the network is made available to all sensors at observation time.

Fully connected, "complete information" networks are not practically viable: Any realisable distributed sensing network should be able to cater for a variety of communication topologies, variable communication delays and insertion of new sensors into the network. The key to providing these abilities lies in the algorithms used to decide what information should be communicated between different sensing nodes. This is the focus of this section.

It is first demonstrated that the need for fully-connectedness is a result of the assumption that all nodes share a common prediction. Implementation of the straight-forward decentralised data fusion equations consequently results in a nearest-neighbour communication policy in which information is not propagated through the network.

To overcome this, the idea of a channel filter is introduced. A channel filter records the information that is communicated between nodes. In a tree-connected network the information communicated between nodes is clearly also the information they have in common. By subtracting this common information from any future communication, only "new" information is communicated. However, it is also demonstrated that in general sensor network topologies, the channel filter is impossible to implement in practice. This is because the common information between two nodes can not be established uniquely. Three (sub-optimal) methods of overcoming this problem are described.

The channel filter provides a valuable buffer between a sensor node and the remainder of the sensing network. The channel filter can be used to implement many practical aspects of network operation including intermittent communication, insertion of new and removal of old communication links. An algorithm is developed to dealing with information that is delayed or asequent (out of temporal order). Using this algorithm, general channel operations are described.

4.4.1 Fully Connected and Broadcast Sensor Networks

The assumption of a fully connected topology is, in general, unrealistic within the node-to-node communication constraints imposed. This is because as the number of nodes grows, the number of communication links required by *each* node also increases. In addition, the loss of any one communication channel will result in the fully-connected assumption being violated.

Recall the assimilation equations

$$\begin{aligned}\hat{\mathbf{y}}_i(k \mid k) &= \hat{\mathbf{y}}_i(k \mid k-1) + \sum_j \left[\tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_j(k \mid k-1)\right] \\ &= \hat{\mathbf{y}}_i(k \mid k-1) + \sum_{j \in N_i} \mathbf{i}_j(k) \end{aligned} \qquad (306)$$

and

$$\begin{aligned}\mathbf{Y}_i(k \mid k) &= \mathbf{Y}_i(k \mid k-1) + \sum_j \left[\tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_j(k \mid k-1)\right] \\ &= \mathbf{Y}_i(k \mid k-1) + \sum_{j \in N_i} \mathbf{I}_j(k). \end{aligned} \qquad (307)$$

Equations 306 and 307 make it clear that the estimate arrived at locally by a node i is based on the observation information $\mathbf{i}_j(k)$ and $\mathbf{I}_j(k)$ communicated to it by nodes j. If node i only communicates with a local neighbourhood N_i, a subset of the complete network, then the estimate arrived at locally will be based only on this information and will not be equivalent to a centralised estimate.

In effect, each time a new local estimate $\tilde{\mathbf{y}}_j(k \mid k)$ is obtained at a node j, the prior information at this node $\hat{\mathbf{y}}_j(k \mid k-1)$ is subtracted to provide the 'new' information to be communicated to a node i. The assumption here is that the prior information $\hat{\mathbf{y}}_j(k \mid k-1)$ at node j is the information that nodes i and j have in common up to time

$k-1$. Subtracting this from the local estimate should then give the new information to be communicated from node j to node i. In the fully-connected case, the prior information at node j is indeed the common information between nodes i and j and so only the information $\mathbf{i}_j(k)$ is communicated. In the non-fully connected case, the prior information at node j includes not only information common to node i but also information from other branches in the network. Subtracting this from the local estimate however, again results in only the information $\mathbf{i}_j(k)$ being communicated to node i. This is because it assumes that node i already has the information communicated to node j through other branches of the network. The net result of this is that nodes only exchange information in their immediate neighbourhoods and do not propagate information from distant branches.

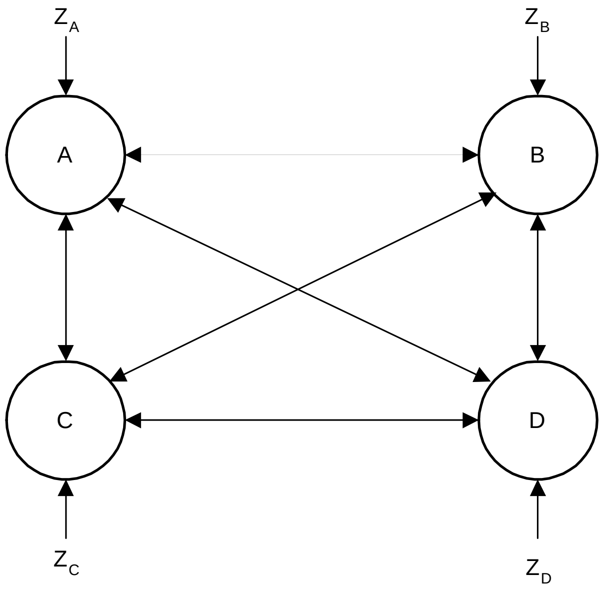


Figure 39: A fully connected sensor network consisting of four nodes, A, B, C, and D. All nodes in this network can generate estimates, equal to a centralised estimate, using only Equations 306 and 307.

Figure 39 shows a fully connected network in which local estimates will be equal to global estimates using only Equations 306 and 307 for assimilation. Figure 40 shows a linear connected sensor network and Figure 41 shows a tree connected sensor network. Here, the estimates arrived at by each node, using Equations 306 and 307 for assimilation, will be based only on observations made by sensors in the immediate neighbourhood.

4.4.2 Identification of Redundant Information in Sensor Networks

The key step in deriving fusion equations for decentralised sensing networks is to identify the common information among estimates so that it is not used redundantly. The problem of accounting for redundant information in decentralised communication structures is en-

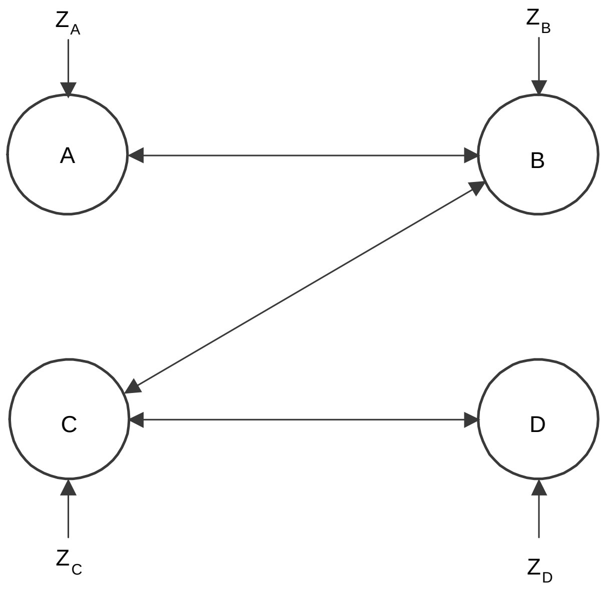


Figure 40: A linear sensor network consisting of four nodes, A, B, C, and D. Nodes in this network, using only Equations 306 and 307, can only generate estimates based on information available in their immediate neighbourhood.

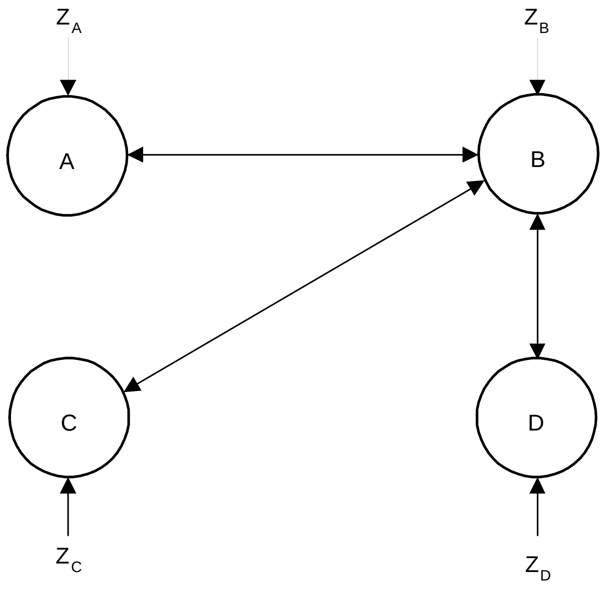


Figure 41: A tree connected sensor network consisting of four nodes, A, B, C, and D. Nodes in this network, using only Equations 306 and 307, generate estimates based on information available in their immediate neighbourhood.

countered in many applications[13]. In decentralised data fusion systems, the incorporation of redundant information may lead to bias, over-confidence and divergence in estimates.

The problem of identifying common information is most generally considered in terms of information sets. A neighbourhood of a node is the set of nodes to which it is linked directly. A complete neighbourhood includes the node itself. A node i forms an information set $\mathbf{Z}_i^k$ at time k based on a local sensor observation $\mathbf{z}_i(k)$ and the information communicated by its neighbours. The objective of each node is to form the *union* of the information sets in the complete neighbourhood $[N_i]$: $\bigcup_{j\in[N_i]}\mathbf{Z}_j^k$. In particular, consider communications between node i and a neighbour j. The union of information sets, on which estimates are to be based, may be written as

$$\mathbf{Z}_i^k \cup \mathbf{Z}_j^k = \mathbf{Z}_i^k + \mathbf{Z}_j^k - \mathbf{Z}_{i\cap j}^k, \tag{308}$$

that is, the union is equal to the sum of information sets communicated minus the intersection of, or common information between, these information sets. A fully decentralised solution to the estimation problem is only possible where the intersection or common communicated information $\mathbf{Z}_{i\cap j}^k$ can be determined from information which is available locally.

The topology of the sensing network is the most important factor in determining common communicated information. Consider the following three cases:

- **Full Connection:** Consider the case in which each node is connected to every other in a fully connected, or completely connected topology (Figure 39). In this case, the sensor nodes may acquire observation information from the entire network through direct communication and the neighbourhood of any node is the full network. In a fully connected network, the problem of locally detecting and eliminating redundant information is considerably simplified as every node has access to the same information. In this situation, the estimates formed by the nodes are identical and new information is immediately communicated after the removal of the estimate from the previous time-step as

$$\bigcup_{j\in[N_i]} \mathbf{Z}_j^k = \left[\sum_{j\in[N_i]} \mathbf{Z}_j^k - \bigcup_{j\in[N_i]} \mathbf{Z}_j^{k-1}\right]. \tag{309}$$

  This gives rise to a communication strategy where each node subtracts the estimate formed at the previous timestep prior to communicating its current observation information. Thus, Equation 309 is an information-set equivalent of Equations 306 and 307 which explicitly subtract the common prediction from the local partial estimates. Since the fully connected condition means that global and local information are in fact the same, this must be regarded as a special case of the common information problem.

[13] In human communication decentralised and cyclic communication structures give rise to “rumour propagation” or the “chicken licken” problem.

- **Tree Connection:** In a tree connected topology, there is only one path between each pair of nodes (see Figures 40 and 41). Therefore, a receiving node can be certain that the only redundant information communicated by a neighbour is the information that they have exchanged in the past. Thus, the observation information history that is required for a tree communication system extends only to the previous timestep. The only common information between node i and a neighbour j at time k is the information which they exchanged in the last time interval $k-1$. This results in a pair-wise communication algorithm. For a node i on link (i,j),

$$\mathbf{Z}_i^k \cup \mathbf{Z}_j^k = \mathbf{Z}_i^k + \mathbf{Z}_j^k - \mathbf{Z}_{i\cap j}^k,$$

where, crucially, the intersection can be found from information sets at the previous timestep,

$$\mathbf{Z}_{i\cap j}^{k-1} = \mathbf{Z}_i^{k-1} \cup \mathbf{Z}_j^{k-1}. \tag{310}$$

This generalises to a communication strategy over the neighbourhood as

$$\bigcup_{j\in[N_i]} \mathbf{Z}_j^k = \mathbf{Z}_i^k + \left[\sum_{j\in N_i} \mathbf{Z}_j^k - \bigcup_{j\in N_i} \mathbf{Z}_j^{k-1}\right], \tag{311}$$

where $[N_i]$ is the complete neighbourhood, and $i \notin N_i$.

- **Arbitrary Networks:** In an arbitrary network, the connectedness of nodes is unknown and may be partially connected, or non fully connected non-trees. Removal of common information in arbitrary network topologies is complex because the pattern of communication varies from node to node, yet each node is required to implement the same algorithm. Consider again two communicating nodes

$$\mathbf{Z}_i^k \cup \mathbf{Z}_j^k = \mathbf{Z}_i^k + \mathbf{Z}_j^k - \mathbf{Z}_{i\cap j}^k.$$

In the arbitrary network case, the intersection term $\mathbf{Z}_{i\cap j}^k$ can not be simply determined from past communication on a single link. In the tree case, the common information term depends only on the observation information of i and j. In a non-tree network, the information common to i and j may contain terms from other nodes outside of the neighbourhood. This is because multiple propagation paths are possible. Figure 42 illustrates the information that is communicated to nodes in three cases. The communicated information terms at i and j are denoted T_i and T_j respectively. The information T_j is integrated into the observation information sets $\mathbf{Z}_j$ upon arrival at j. This information must then be acquired by i through some operation of union or intersect of information sets. To main the constraints imposed by full decentralisation, it must be possible to eliminate common communicated information terms between any pair of communicating nodes, on the basis of local information only. In a fully connected network, the solution is immediate since $T_i = T_j$ at every time-step and Equation 309 follows. A tree network is partitioned about any pair of connected nodes (i,j) such that the information T_i held in the subtree from i and that held in the subtree from j are disjoint: $T_i \cap T_j = \emptyset$. Therefore, i acquires the terms from the subtree T_j only through j. In an arbitrary network, it may be possible for i to acquire the information known to j along other routes. The problem is that the communicated terms are not necessarily disjoint, therefore, each node must be able to determine $T_i \cap T_j$ locally. As shown in Figure 42, information in the region of intersection arrives at both i and j. This multiple propagation must be accounted for. The problem of arbitrary networks can be alternately be considered as estimation in networks which admit multiple cycles.

Determination of the communication requirements for non-fully connected decentralised networks therefore hinges on the extraction of common information.

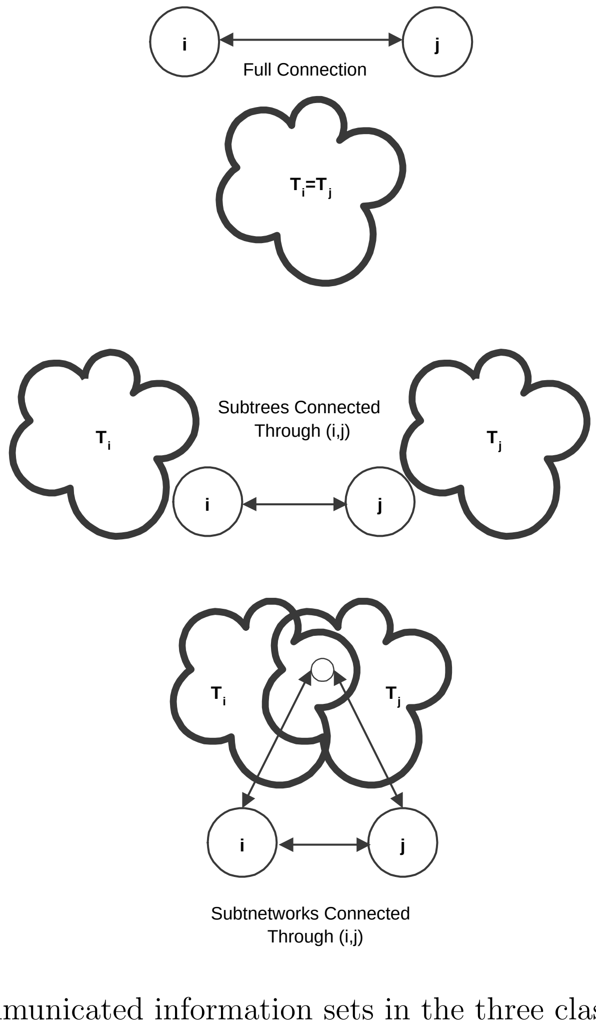

i
j
Full Connection
Ti=Tj
Ti
Subtrees Connected Through (i,j)
Tj
i
j
Ti
Tj
i
j
Subtnetworks Connected Through (i,j)

Figure 42: Communicated information sets in the three classes of topology.

4.4.3 Bayesian Communication in Sensor Networks

Expressions for the common information between two nodes are now derived from Bayes theorem. This in turn establishes the relationships between log-likelihoods from which common information filters can be derived.

The interaction of *pairs* of nodes are considered. For each communicating pair (i, j), the required probability is $P(\mathbf{Z}_i \cup \mathbf{Z}_j)$. Let the union of the individual observation information sets be partitioned into disjoint sets as

$$\mathbf{Z}_i \cup \mathbf{Z}_j = \mathbf{Z}_{i\backslash j} \cup \mathbf{Z}_{j\backslash i} \cup \mathbf{Z}_{ij} \tag{312}$$

where

$$\mathbf{Z}_{i\backslash j} = \mathbf{Z}_i \backslash \mathbf{Z}_{ij}, \quad \mathbf{Z}_{j\backslash i} = \mathbf{Z}_j \backslash \mathbf{Z}_{ij}, \quad \mathbf{Z}_{ij} = \mathbf{Z}_i \cap \mathbf{Z}_j,$$

and where the notation $p\backslash r$ (the restriction operation) means elements of the set p excluding those elements that are also in set r. Note also that

$$\mathbf{Z}_{i\backslash j} \cup \mathbf{Z}_{ij} = \mathbf{Z}_i, \qquad \mathbf{Z}_{j\backslash i} \cup \mathbf{Z}_{ij} = \mathbf{Z}_j.$$

Then,

$$\begin{aligned}
P(\mathbf{Z}_i \cup \mathbf{Z}_j \mid \mathbf{x}) &= P(\mathbf{Z}_{i\backslash j} \cup \mathbf{Z}_{j\backslash i} \cup \mathbf{Z}_{ij} \mid \mathbf{x}) \\
&= P(\mathbf{Z}_{i\backslash j} \mid \mathbf{Z}_{j\backslash i} \cup \mathbf{Z}_{ij}, \mathbf{x}) P(\mathbf{Z}_{j\backslash i} \cup \mathbf{Z}_{ij} \mid \mathbf{x}) = P(\mathbf{Z}_{i\backslash j} \mid \mathbf{Z}_{ij}, \mathbf{x}) P(\mathbf{Z}_j \mid \mathbf{x}) \\
&= \frac{P(\mathbf{Z}_{i\backslash j} \cup \mathbf{Z}_{ij} \mid \mathbf{x})}{P(\mathbf{Z}_{ij} \mid \mathbf{x})} P(\mathbf{Z}_j \mid \mathbf{x}) = \frac{P(\mathbf{Z}_i \mid \mathbf{x})}{P(\mathbf{Z}_{ij} \mid \mathbf{x})} P(\mathbf{Z}_j \mid \mathbf{x}) \\
&= \frac{P(\mathbf{Z}_i \mid \mathbf{x}) P(\mathbf{Z}_j \mid \mathbf{x})}{P(\mathbf{Z}_i \cap \mathbf{Z}_j \mid \mathbf{x})}.
\end{aligned} \tag{313}$$

Substituting Equation 313 into Bayes theorem gives

$$\begin{aligned}
P(\mathbf{x} \mid \mathbf{Z}_i \cup \mathbf{Z}_j) &= \frac{P(\mathbf{Z}_i \cup \mathbf{Z}_j \mid \mathbf{x}) P(x)}{P(\mathbf{Z}_i \cup \mathbf{Z}_j)} \\
&= \frac{P(\mathbf{Z}_i \mid \mathbf{x}) P(\mathbf{Z}_j \mid \mathbf{x})}{P(\mathbf{Z}_i \cap \mathbf{Z}_j \mid \mathbf{x})} \frac{P(\mathbf{x})}{P(\mathbf{Z}_i \cup \mathbf{Z}_j)} \\
&= \frac{P(\mathbf{x} \mid \mathbf{Z}_i) P(\mathbf{x} \mid \mathbf{Z}_j)}{P(\mathbf{x} \mid \mathbf{Z}_i \cap \mathbf{Z}_j)} \frac{P(\mathbf{Z}_i) P(\mathbf{Z}_j)}{P(\mathbf{Z}_i \cap \mathbf{Z}_j)} P(\mathbf{Z}_i \cup \mathbf{Z}_j) \\
&= c . \frac{P(\mathbf{x} \mid \mathbf{Z}_i) P(\mathbf{x} \mid \mathbf{Z}_j)}{P(\mathbf{x} \mid \mathbf{Z}_i \cap \mathbf{Z}_j)}.
\end{aligned} \tag{314}$$

This shows that the relation between the posterior probability in the unknown state given information from both nodes, $P(\mathbf{x} \mid \mathbf{Z}_i \cup \mathbf{Z}_j)$, as a function of the posteriors based only on locally available information, $P(\mathbf{x} \mid \mathbf{Z}_i)$ and $P(\mathbf{x} \mid \mathbf{Z}_j)$, and the information the two nodes have in common $P(\mathbf{x} \mid \mathbf{Z}_i \cap \mathbf{Z}_j)$.

Taking logs of Equation 314, gives

$$\ln P(\mathbf{x} \mid \mathbf{Z}_i \cup \mathbf{Z}_j) = \ln P(\mathbf{x} \mid \mathbf{Z}_i) + \ln P(\mathbf{x} \mid \mathbf{Z}_j) - \ln P(\mathbf{x} \mid \mathbf{Z}_i \cap \mathbf{Z}_j). \tag{315}$$

Equation 315 simply states that the fused information is constructed from the sum of the information from each of the nodes minus the information they have in common. The term $\ln P(\mathbf{x} \mid \mathbf{Z}_i \cap \mathbf{Z}_j)$ describes the common information between two nodes which must be removed before fusion.

Equation 315 serves as the basis for developing information communication policies for non-fully connected sensor networks.

4.4.4 The Channel Filter

The probability density functions in Equation 315 can represent four different information or Kalman filter estimates:

- The local estimate at node i:
$$\hat{\mathbf{x}}_i(k \mid k) \stackrel{\triangle}{=} \mathrm{E}\{\mathbf{x}(k) \mid \mathbf{Z}_i^k\}$$
with covariance $\mathbf{P}_i(k \mid k)$.
- The local estimate at node j:
$$\hat{\mathbf{x}}_j(k \mid k) \stackrel{\triangle}{=} \mathrm{E}\{\mathbf{x}(k) \mid \mathbf{Z}_j^k\}$$
with covariance $\mathbf{P}_j(k \mid k)$.
- The estimate based on the union of all information possessed by nodes i and j (in effect the global estimate):
$$\hat{\mathbf{x}}_{i\cup j}(k \mid k) \stackrel{\triangle}{=} \mathrm{E}\{\mathbf{x}(k) \mid \mathbf{Z}_i^k \cup \mathbf{Z}_j^k\}$$
with covariance $\mathbf{P}_{i\cup j}(k \mid k)$.
- The estimate based on the common information between nodes i and j:
$$\hat{\mathbf{x}}_{i\cap j}(k \mid k) \stackrel{\triangle}{=} \mathrm{E}\{\mathbf{x}(k) \mid \mathbf{Z}_i^k \cap \mathbf{Z}_j^k\}$$
with covariance $\mathbf{P}_{i\cap j}(k \mid k)$.

Following Equation 259; Substituting Gaussian distributions for the probability density functions in Equation 315 and taking natural logarithms immediately gives the information filter equivalent of Equation 315 as

$$\hat{\mathbf{y}}_{i\cup j}(k \mid k) = \tilde{\mathbf{y}}_i(k \mid k) + \tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_{i\cap j}(k \mid k) \tag{316}$$

$$\mathbf{Y}_{i\cup j}(k \mid k) = \tilde{\mathbf{Y}}_i(k \mid k) + \tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_{i\cap j}(k \mid k) \tag{317}$$

where

$$\tilde{\mathbf{y}}_i(k \mid k) = \hat{\mathbf{y}}_i(k \mid k-1) + \mathbf{i}_i(k), \qquad \tilde{\mathbf{y}}_j(k \mid k) = \hat{\mathbf{y}}_j(k \mid k-1) + \mathbf{i}_j(k) \tag{318}$$

$$\tilde{\mathbf{Y}}_i(k \mid k) = \mathbf{Y}_i(k \mid k-1) + \mathbf{I}_i(k), \qquad \tilde{\mathbf{Y}}_j(k \mid k) = \mathbf{Y}_j(k \mid k-1) + \mathbf{I}_j(k) \tag{319}$$

It remains to evaluate the common information terms

$$\hat{\mathbf{y}}_{ij}(k \mid k) \stackrel{\triangle}{=} \hat{\mathbf{y}}_{i\cap j}(k \mid k), \qquad \mathbf{Y}_{ij}(k \mid k) \stackrel{\triangle}{=} \mathbf{Y}_{i\cap j}(k \mid k).$$

There are three cases

- **Fully Connected:** When the network is fully connected, the common information between two nodes at a time k is exactly the information communicated up to time $k-1$. This is simply the common prediction

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_i(k \mid k-1) = \hat{\mathbf{y}}_j(k \mid k-1) \\ \mathbf{Y}_{ij}(k \mid k) &= \mathbf{Y}_i(k \mid k-1) = \mathbf{Y}_j(k \mid k-1). \end{aligned} \tag{320}$$

  Substitution of Equations 320, 318 and 319 into Equations 316 and 317, yields the previously derived decentralised data fusion Equations 306 and 307.

- **Tree Connected:** When there is only one pathway joining any two sensor nodes, the common information between these nodes can be obtained locally by simply adding up the information that has previously been communicated on the channel connecting the two nodes. Equations 287 and 288 provide a recursive expression to for the total information communicated between the two nodes as

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_{ij}(k \mid k-1) \\ &\quad +[\tilde{\mathbf{y}}_i(k \mid k) - \hat{\mathbf{y}}_{ij}(k \mid k-1)] \\ &\quad +[\tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_{ij}(k \mid k-1)] \\ &= \tilde{\mathbf{y}}_i(k \mid k) + \tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_{ij}(k \mid k-1) \end{aligned} \tag{321}$$

  and

$$\begin{aligned} \mathbf{Y}_{ij}(k \mid k) &= \mathbf{Y}_{ij}(k \mid k-1) \\ &\quad +[\tilde{\mathbf{Y}}_i(k \mid k) - \mathbf{Y}_{ij}(k \mid k-1)] \\ &\quad +[\tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_{ij}(k \mid k-1)] \\ &= \tilde{\mathbf{Y}}_i(k \mid k) + \tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_{ij}(k \mid k-1) \end{aligned} \tag{322}$$

  This estimate of common information replaces the prior information terms in Equations 287 and 288. The local updates at each node remain unchanged

$$\tilde{\mathbf{y}}_i(k \mid k) = \hat{\mathbf{y}}_i(k \mid k-1) + \mathbf{i}_i(k) \tag{323}$$

$$\tilde{\mathbf{Y}}_i(k \mid k) = \mathbf{Y}_i(k \mid k-1) + \mathbf{I}_i(k) \tag{324}$$

  and the assimilation stage becomes

$$\hat{\mathbf{y}}_i(k \mid k) = \tilde{\mathbf{y}}_i(k \mid k) + \sum_{j \in N_i} [\tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_{ji}(k \mid k-1)] \tag{325}$$

$$\mathbf{Y}_i(k \mid k) = \tilde{\mathbf{Y}}_i(k \mid k) + \sum_{j \in N_i} \left[\tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_{ji}(k \mid k-1)\right]. \tag{326}$$

  This filter is clearly symmetric, $\hat{\mathbf{y}}_{ij}(\cdot \mid \cdot) = \hat{\mathbf{y}}_{ji}(\cdot \mid \cdot)$, as two nodes have the same common information, so need only be computed once for each channel.

- **General Networks:** In networks which admit multiple cycles, it is possible for information from a node j to be communicated to a node i both directly through the link (i, j) and indirectly through other nodes connected to both i and j. In such cases, simply adding up information transmitted through the direct connection (i, j) is *not* the same as determining the common information $\hat{\mathbf{y}}_{ij}(k \mid k)$ and $\mathbf{Y}_{ij}(k \mid k)$ between these two nodes. Indeed, in general it is *not* possible to determine this common information on the basis of local information only. A number of alternative approaches to fusion in general networks are considered in Section 4.4.7.

Equations 321 and 322 define an information filter which estimates the common information between nodes. The filter is completed by addition of a corresponding prediction stage as

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k-1) &= \left[\mathbf{1} - \mathbf{\Omega}_{ij}(k)\mathbf{G}^T(k)\right]\mathbf{F}^{-T}(k)\hat{\mathbf{y}}_{ij}(k-1 \mid k-1) \\ &\quad + \mathbf{Y}_{ij}(k \mid k-1)\mathbf{B}(k)\mathbf{u}(k) \\ \mathbf{Y}_{ij}(k \mid k-1) &= \mathbf{M}_{ij}(k) - \mathbf{\Omega}_{ij}(k)\mathbf{\Sigma}_{ij}(k)\mathbf{\Omega}_{ij}^T(k) \end{aligned} \qquad (327)$$

where

$$\mathbf{M}_{ij}(k) = \mathbf{F}^{-T}(k)\mathbf{Y}_{ij}(k-1 \mid k-1)\mathbf{F}^{-1}(k), \qquad \mathbf{\Omega}_{ij}(k) = \mathbf{M}_{ij}(k)\mathbf{G}(k)\mathbf{\Sigma}_{ij}^{-1}(k), \qquad (328)$$

and

$$\mathbf{\Sigma}_{ij}(k) = \left[\mathbf{G}^T(k)\mathbf{M}_{ij}(k)\mathbf{G}(k) + \mathbf{Q}^{-1}(k)\right]. \qquad (329)$$

The channel filter is simply an information-state estimator which generates estimates on the basis of information communicated through a channel joining two adjacent nodes. The channel filter The effect of the channel filter is to provide information to nodes from further afield in the network with no extra communication cost. If the network is strictly synchronous (all nodes cycle at the same speed) then the information arriving at a specific node will be time delayed in proportion to the number of nodes through which it must pass. This gives rise to a characteristic triangular 'wave-front' in the information map which describes the way information is used at any one node to obtain an estimate.

4.4.5 Delayed, Asequent and Burst Communication

The channel filter provides a mechanism for handling the practical problems of communication between sensor nodes. A communication medium for a sensor network will always be subject to bandwidth limitations, and possibly intermittent or jammed communications between nodes. This in turn gives rise to data that is delayed prior to fusion and possibly data that is asequent (arriving out of temporal order). If problems of data delay and order can be resolved, then efficient strategies can be developed for managing communications in sensor networks.

At the heart of practical communication issues is the delayed data problem. The solution to the delayed data problem requires an ability to propagate information both forward and backward in time using the information prediction Equations 250 and 251.

Consider the information matrix $\mathbf{Y}(k \mid k)$ obtained locally at a node at time k. Following Equation 251, this information matrix can be propagated forward in time n steps according to

$$\mathbf{Y}(k+n \mid k) = \mathbf{M}_n(k) - \mathbf{M}_n(k)\mathbf{G}_n(k)\mathbf{\Sigma}_n^{-1}(k)\mathbf{G}_n^T(k)\mathbf{M}_n(k) \tag{330}$$

where

$$\mathbf{M}_n(k) = \left[\mathbf{F}^{-T}(k)\right]^n \mathbf{Y}(k \mid k)\left[\mathbf{F}^{-1}(k)\right]^n,$$
$$\mathbf{\Sigma}_n(k) = \mathbf{G}_n^T(k)\mathbf{M}_n(k)\mathbf{G}_n(k) + \mathbf{Q}_n^{-1}(k)$$

and where the subscript n denotes that the associated transition matrices are evaluated from the integrals in Equation 73 over the interval $n\Delta T$. Likewise, from Equation 250 the information-state vector can be propagated forward according to

$$\begin{aligned}\hat{\mathbf{y}}(k+n \mid k) &= \left[\mathbf{1} - \mathbf{M}_n(k)\mathbf{G}_n(k)\mathbf{\Sigma}_n^{-1}(k)\mathbf{G}_n^T(k)\right]\left[\mathbf{F}^{-T}(k)\right]^n \hat{\mathbf{y}}(k \mid k) \\ &\quad + \mathbf{Y}(k+n \mid k)\mathbf{B}_n(k)\mathbf{u}_n(k).\end{aligned} \tag{331}$$

An equivalent expression can be obtained for propagating information backwards in time as

$$\mathbf{Y}(k-n \mid k) = \mathbf{M}_{-n}(k) + \mathbf{M}_{-n}(k)\mathbf{G}_{-n}(k)\mathbf{\Sigma}_{-n}^{-1}(k)\mathbf{G}_{-n}^T(k)\mathbf{M}_{-n}(k) \tag{332}$$

where

$$\mathbf{M}_{-n}(k) = \left[\mathbf{F}^{T}(k)\right]^n \mathbf{Y}(k \mid k)\left[\mathbf{F}(k)\right]^n,$$
$$\mathbf{\Sigma}_{-n}(k) = \mathbf{Q}_n^{-1}(k) - \mathbf{G}_{-n}^T(k)\mathbf{M}_{-n}(k)\mathbf{G}_{-n}(k)$$

and where the subscript $-n$ denotes that the associated transition matrices are evaluated from the integrals in Equation 73 over the interval $-n\Delta T$. Similarly, the information-state vector can be propagated backward according to

$$\begin{aligned}\hat{\mathbf{y}}(k-n \mid k) &= \left[\mathbf{1} - \mathbf{M}_{-n}(k)\mathbf{G}_{-n}(k)\mathbf{\Sigma}_{-n}^{-1}(k)\mathbf{G}_{-n}^T(k)\right]\left[\mathbf{F}^{T}(k)\right]^n \hat{\mathbf{y}}(k \mid k) \\ &\quad + \mathbf{Y}(k-n \mid k)\mathbf{B}_{-n}(k)\mathbf{u}_n(k).\end{aligned} \tag{333}$$

The delayed data problem can now be solved using the following algorithm:

1. Back-propagate the estimate $\mathbf{Y}(k \mid k)$ to the time, $k-n$, at which the information $\mathbf{I}(k-n)$ was obtained, using Equations 332 and 333.

2. Add the delayed data $\mathbf{I}(k-n)$ to the estimate $\mathbf{Y}(k-n \mid k)$ in the normal manner.

3. Propagate the new fused estimate back to time k using Equations 330 and 331.

It can be demonstrated that the net effect of this algorithm is to produce an estimate in the form

$$\mathbf{Y}(k \mid k) = \mathbf{Y}_Y(k \mid k) + \mathbf{Y}_I(k \mid k) + \mathbf{Y}_{YI}(k \mid k)$$

where $\mathbf{Y}_Y(k \mid k)$ is the estimate obtained without the delayed information, $\mathbf{Y}_I(k \mid k)$ is the estimate obtained using only the delayed information, and $\mathbf{Y}_{YI}(k \mid k)$ is a cross-information term (uniquely defined $\mathbf{Y}_Y(k \mid k)$ and $\mathbf{Y}_I(k \mid k)$) describing the cross-correlation between the information states caused by propagation through a common process model. It should be noted that $\mathbf{Y}_{YI}(k \mid k)$ is additive, so the obvious approximation

$$\mathbf{Y}(k \mid k) = \mathbf{Y}_Y(k \mid k) + \mathbf{Y}_I(k \mid k)$$

is conservative.

Recall that the channel filter is simply a standard information filter used to maintain an estimate of common data passed through a particular channel. A channel filter on node i connected to node j maintains the common information vector $\hat{\mathbf{y}}_{ij}(k \mid k)$ and the common information matrix $\mathbf{Y}_{ij}(k \mid k)$.

The prediction equations (for both forward and backward propagation) used in the channel filter are the same as Equations 330–333 described above. For the update stage a channel filter receives information estimates from other nodes. When this happens, the channel filter predicts the received information to a local time horizon and then determines the new information at that time. The new information at the channel filter at node i when data arrives through the channel connected to node j is the information gain from node j

$$\begin{aligned} \mathbf{m}_j(k) &= \hat{\mathbf{y}}_j(k \mid k-n) - \hat{\mathbf{y}}_{ij}(k \mid k-m) \\ \mathbf{M}_j(k) &= \mathbf{Y}_j(k \mid k-n) - \mathbf{Y}_{ij}(k \mid k-m) \end{aligned} \tag{334}$$

Note now that the time indices $\mathbf{Y}_j(k \mid k-n)$ and $\mathbf{Y}_{ij}(k \mid k-m)$ are different as the different nodes are asynchronous and the two information sets can be predicted over different time horizons. This information gain $\mathbf{m}(k)$ and $\mathbf{M}(k)$ is analogous to the observation information vector $\mathbf{i}(k)$ and $\mathbf{I}(k)$, and is merely added in the update stage. The update for the channel filter between nodes i and j when new information has been received from node j can then be written as

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_{ij}(k \mid k-m) + \mathbf{m}_j(k) \\ \mathbf{Y}_{ij}(k \mid k) &= \mathbf{Y}_{ij}(k \mid k-m) + \mathbf{M}_j(k) \end{aligned} \tag{335}$$

This can be further simplified by substituting Equation 334

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_{ij}(k \mid k-m) + \mathbf{m}_j(k) \\ \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_{ij}(k \mid k-m) + \hat{\mathbf{y}}_j(k \mid k-n) - \hat{\mathbf{y}}_{ij}(k \mid k-m) \\ \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_j(k \mid k-n) \end{aligned} \tag{336}$$

This reveals a trivial update stage where the channel filter merely overwrites the previous state with the newer one. When a new information set arrives at node i from node j, the channel filter is updated by

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_j(k \mid k-n) \\ \mathbf{Y}_{ij}(k \mid k) &= \mathbf{Y}_j(k \mid k-n) \end{aligned} \tag{337}$$

Alternatively, if the new information at the channel filter of node i is from the local filter at node i, the update becomes

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_i(k \mid k) \\ \mathbf{Y}_{ij}(k \mid k) &= \mathbf{Y}_i(k \mid k) \end{aligned} \tag{338}$$

While the normal information filter update of Equation 335 is perfectly valid, implementation in the form of Equation 337 and 338 is much simpler as there is no computation, just the overwriting of the previous state.

Together, the channel update equations allow data delayed from neighbouring nodes to be fused in an optimal manner. This together with local assimilation equations permits burst communication of estimates accumulated over a time period, in a manner that ensures no double counting of information. This also means that the channel filter algorithms are robust to intermittent communication failure.

4.4.6 Management of Channel Operations

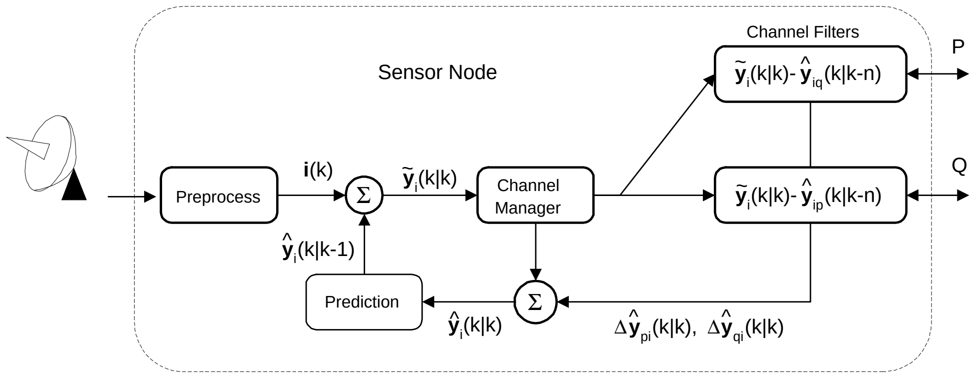

Sensor Node
Channel Filters
$\tilde{\mathbf{y}}_i(k|k)$- $\hat{\mathbf{y}}_{iq}(k|k\text{-}n)$
P
Q
Preprocess
$\mathbf{i}(k)$
Σ
$\tilde{\mathbf{y}}_i(k|k)$
Channel Manager
$\tilde{\mathbf{y}}_i(k|k)$- $\hat{\mathbf{y}}_{ip}(k|k\text{-}n)$
$\hat{\mathbf{y}}_i(k|k\text{-}1)$
Prediction
Σ
$\hat{\mathbf{y}}_i(k|k)$
$\Delta\hat{\mathbf{y}}_{pi}(k|k)$, $\Delta\hat{\mathbf{y}}_{qi}(k|k)$

Figure 43: Algorithmic structure of a decentralised sensing node.

Figure 43 shows a typical sensor node, i, in a decentralised data fusion system. The node generates information measures $\hat{\mathbf{y}}_i(k \mid k)$ at a time k given observations made locally and information communicated to the node up to time k. The node implements a local prediction stage to produce information measure predictions $\hat{\mathbf{y}}_i(k \mid k-1)$ at time k given all local and communicated data up to time $k-1$ (this prediction stage is often the same on each node and may, for example, correspond to the path predictions of a number of common targets). At this time, local observations produce local information measures $\mathbf{i}(k)$ on the basis of local observations. The prediction and local information measures are combined, by simple addition, into a total local information measure $\tilde{\mathbf{y}}_i(k \mid k)$ at time k. This measure is handed down to the communication channels for subsequent communication to other nodes in the decentralised network. Incoming information from other nodes $\hat{\mathbf{y}}_{ji}(k \mid k)$ is extracted from appropriate channels and is assimilated with the total local information by simple addition. The result of this fusion is a locally available global information measure $\hat{\mathbf{y}}_i(k \mid k)$. The algorithm then repeats recursively.

The communication channels exploit the associativity property of information measures. The channels take the total local information $\tilde{\mathbf{y}}_i(k \mid k)$ and subtract out all information that has previously been communicated down the channel, $\hat{\mathbf{y}}_{ij}(k \mid k)$, thus transmitting only new information obtained by node i since the last communication. Intuitively, communicated data from node i thus consists only of information not previously transmitted to a node j; because common data has already been removed from the communication, node j can simply assimilate incoming information measures by addition. As these channels essentially act as information assimilation processes, they are usually referred to as channel filters

Channel filters have two important characteristics:

1. Incoming data from remote sensor nodes is assimilated by the local sensor node *before* being communicated on to subsequent nodes. Therefore, no matter the number of incoming messages, there is only a single outgoing message to each node. Consequently, as the sensor network grows in size, the amount of information sent down any one channel remains constant. This leads to an important conclusion: *A decentralised data fusion system can scale up in size indefinitely without any increase in node-to-node communication bandwidth.*

2. A channel filter compares what has been previously communicated with the total local information at the node. Thus, if the operation of the channel is suspended, the filter simply accumulates information in an additive fashion. When the channel is re-opened, the total accumulated information in the channel is communicated in one single message. The consequences of this are many-fold; burst transmission of accumulate data can be employed to substantially reduce communication bandwidth requirements (and indeed be used to manage communications); if a node is disconnected from the communications network, it can be re-introduced and information synchronised in one step (the same applies to new nodes entering the system, dynamic network changes and signal jamming). This leads to a second important conclusion: *A decentralised data fusion system is robust to changes and loss of communication media.*

Together, the node and channel operation define a system which is fully modular in algorithmic construction, is indefinitely scalable in numbers of nodes and communication topology, and is survivable in the face of failures of both sensor nodes and communication facility.

To summarise, this algorithm provides a framework that enables information to flow through the network with a fixed communication overhead, independent of network size. Information flow is achieved without any additional communication over that required for the simple nearest neighbour fusion algorithm described in Equations 306 and 307. This involves the transmission and reception of one information matrix and one information-state vector to and from each neighbouring node. The algorithm yields estimates equivalent to those of a centralised system with some data delays (due to the time taken for information to travel across the network).

4.4.7 Communication Topologies

The sensor networks considered in preceding sections were restricted to having only a single path between any two nodes. Clearly such networks are not robust as failure of any one channel or node will divide the network into two non-communicating halves. Networks which are to be made reliable must ensure that there are multiple, redundant, paths between sensor nodes.

However, the algorithms described in the previous section will not produce consistent information-state estimates for networks with multiple paths. The reason for this is that the channel filter $\hat{\mathbf{y}}_{ij}(\cdot \mid \cdot)$, which is intended to determine information common to nodes i and j, does so only by summing up information communicated through the channel linking these two nodes. The assumption is that there is only one path between two neighbouring nodes so the information they have in common must have passed through the communication link connecting them. If information is communicated to both nodes i and j directly by some third node then this information will clearly be common information held by both i and j, but will not appear in the channel filter $\hat{\mathbf{y}}_{ij}(\cdot \mid \cdot)$ because it has not passed through the channel between them.

There are three possible solutions to this problem

- **Data Tagging:** An obvious solution is to tag information as passes through the network so that nodes can determine which comments are "new" and which have already been communicated through other links. The problem with this method is that it will not scale well with large data sets and with large sensor networks.

- **Spanning Trees:** A viable solution is to use an algorithm which dynamically selects a minimal spanning tree for the network. This, in effect would allow redundant links to be artificially "broken" at run-time so allowing tree-based communication algorithms to function in a consistent manner. Links can be artificially broken by setting $\hat{\mathbf{y}}_{ij}(\cdot \mid \cdot) = \tilde{\mathbf{y}}_j(\cdot \mid \cdot)$, ensuring no information is communicated across the cut. The distributed Belman-Ford algorithm is one algorithm which allows such a tree to be constructed in a fully distributed or decentralised manner (it is commonly used for internet routing). Overall system performance depends on reducing data delays by choosing a topology with short paths between nodes. The best topology can then be implemented by artificially breaking the necessary links and using the channel filter algorithm. The network remains scalable although reconfiguration may be necessary if nodes are to be added or removed. This may, at first, seem to be a disadvantage but, should the network sustain damage to links or nodes such a strategy will enable the surviving nodes to reconfigure to form a new topology, bringing formerly disused links into service to bypass damaged nodes or links. Such a system will be more robust than a purely tree connected system which will be divided into two non-communicating parts by the loss of a node or link.

- **Dynamic Determination of Cross-Information:** A promising approach in highly dynamic large scale networks is to use a local algorithm which attempts to determine how the information in two incoming channels is correlated. One such method is the covariance intersect filter. This computes the relative alignment between information matrices and produces a conservative local update based on the worst-case correlation between incoming messages. The advantage of this method is that it is fully general and will work in any network topology. The disadvantage with this method is that the estimates arrived at, while consistent, are often too conservative and far from optimal.

4.5 Advanced Methods in Decentralised Data Fusion

Essential decentralised data fusion algorithms have been developed and extended in a number of ways. These include model distribution methods [31], in which each local sensor node maintains only a part of the global state-space model, in sensor management methods [27], using natural entropic measures of information arising from the log-likelihood definitions intrinsic to the channel filter, and in system organisation. These are not addressed further in this course.

5 Making Decisions

Low-level data fusion tasks are concerned with the problem of modeling and combining information. Having obtained this information it is usually necessary to make some kind of decision; to make an estimate of the true state, to perform a control action, or to decide to obtain more information. Higher level data fusion problems involve significant *decision making* processes.

A decision model is a function which takes as an argument all the information currently available to the system and which produces an action corresponding to the required decision. Central to the decision model is the idea of loss or utility. A loss function (or equivalently a utility function) provides a means of evaluating different actions, allowing for direct comparison of alternative decision rules. Given a loss function, it becomes possible to model the decision making process and to evaluate different data-fusion strategies. This section briefly defines the concepts of action loss or utility[14] and by describing the decision making process that results. Some important elements of statistical decision theory are then introduced with particular emphasis on multiple-person decision making problems. This provides some powerful techniques for modeling the organization and control of information in complex data fusion systems.

5.1 Actions, Loss and Utility

Decision making begins with an unknown state of nature $\mathbf{x} \in \mathcal{X}$. An experiment is made which generates an outcome or observation $\mathbf{z} \in \mathcal{Z}$. On the basis of this observation an action $\mathbf{a} \in \mathcal{A}$ must be computed. The action may be to make an estimate of the true state, in which case $\mathbf{a} \in \mathcal{X}$, but in general it is simply to choose one of a specified set of possible actions in $\mathcal{A}$. A decision rule δ is simply a function taking observations into actions; $\delta : \mathcal{Z} \longrightarrow \mathcal{A}$, so that for every possible observation $\mathbf{z} \in \mathcal{Z}$, δ defines what action $\mathbf{a} \in \mathcal{A}$ must be taken; $\mathbf{a} = \delta(\mathbf{z})$.

5.1.1 Utility or Loss

A loss function $L(\cdot,\cdot)$ (or corresponding utility function $U(\cdot,\cdot)$) is defined as a mapping from pairs of states and actions to the real line;

$$L : \mathcal{X} \times \mathcal{A} \longrightarrow \Re, \qquad U : \mathcal{X} \times \mathcal{A} \longrightarrow \Re.$$

The interpretation of a loss function $L(\mathbf{x}, \mathbf{a})$ is that L is the loss incurred in taking the action $\mathbf{a}$ when the true state of nature is $\mathbf{x}$. Similarly, the interpretation of a utility function $U(\mathbf{x}, \mathbf{a})$ is that U is the gain obtained in taking the action $\mathbf{a}$ when the true state of nature is $\mathbf{x}$. In principle, it should be the case that for a fixed state of nature, both

[14]Engineers, being pessimists by nature, tend to use the term 'Loss' to compare different decisions. Economists, being optimists by necessity, use the term 'Utility'. With the obvious sign change, the terms can be used interchangeably and results derived with respect to one can always be interpreted in terms of the other.

utility and loss will induce a *preference ordering* on $\mathcal{A}$. To ensure this, loss and utility functions must obey four rules called the utility or 'rationality axioms' which guarantee a preference pattern. We state these rules here in terms of utility, following convention. An analogous set of rules applies to loss functions. For fixed $\mathbf{x} \in \mathcal{X}$:

1. Given any $\mathbf{a}_1, \mathbf{a}_2 \in \mathcal{A}$, either $U(x, \mathbf{a}_1) < U(x, \mathbf{a}_2)$, $U(x, \mathbf{a}_1) = U(x, \mathbf{a}_2)$, or $U(x, \mathbf{a}_1) > U(x, \mathbf{a}_2)$. That is, given any two actions we can assign real numbers which indicate our preferred alternative.

2. If $U(x, \mathbf{a}_1) < U(x, \mathbf{a}_2)$ and $U(x, \mathbf{a}_2) < U(x, \mathbf{a}_3)$ then $U(x, \mathbf{a}_1) < U(x, \mathbf{a}_3)$. That is, if we prefer action $\mathbf{a}_2$ to $\mathbf{a}_1$ and action $\mathbf{a}_3$ to $\mathbf{a}_2$, then we must prefer action $\mathbf{a}_3$ to action $\mathbf{a}_1$; the preference ordering is transitive.

3. If $U(\mathbf{x}, \mathbf{a}_1) < U(\mathbf{x}, \mathbf{a}_2)$, then $\alpha U(\mathbf{x}, \mathbf{a}_1) + (1 - \alpha) U(\mathbf{x}, \mathbf{a}_3) < \alpha U(\mathbf{x}, \mathbf{a}_2) + (1 - \alpha) U(\mathbf{x}, \mathbf{a}_3)$, for any $0 < \alpha < 1$. That is, if $\mathbf{a}_2$ is preferred to $\mathbf{a}_1$ then, in a choice between two random situations which are identical except that both $\mathbf{a}_1$ and $\mathbf{a}_2$ occur with probability α, the situation involving $\mathbf{a}_2$ will be preferred.

4. If $U(\mathbf{x}, \mathbf{a}_1) < U(\mathbf{x}, \mathbf{a}_2) < U(\mathbf{x}, \mathbf{a}_3)$, there are numbers $0 < \alpha < 1$ and $0 < \beta < 1$ such that $\alpha U(\mathbf{x}, \mathbf{a}_1) + (1 - \alpha) U(\mathbf{x}, \mathbf{a}_3) < U(\mathbf{x}, \mathbf{a}_2)$ and $\alpha U(\mathbf{x}, \mathbf{a}_1) + (1 - \alpha) U(\mathbf{x}, \mathbf{a}_3) < U(\mathbf{x}, \mathbf{a}_2)$. That is, there is no infinitely desirable or infinitely bad reward (no heaven or hell).

A proof that these axioms imply the existence of a utility function can be found in [8]. The first three axioms seem intuitively reasonable. The final axiom is rather more difficult to justify. Indeed, the well-known least-squares loss function does not satisfy this final axiom because it is not bounded from above. However, it turns out that the need for boundedness rarely affects a decision problem. One exception to this, in group decision making problems, will be discussed in later sections.

It is well known and much discussed by economists that people do not always act rationally according to the 'rationality axioms' and that indeed they normally have considerable problems constructing any kind of consistent utility function by which to judge decisions. The definition of rationality does not really concern us when dealing with a data fusion system that consists of deterministic algorithms; we can always enforce the definition of rationality chosen. What is of importance is the construction of the loss function itself. It is, in principle, important to ensure that the loss (or utility) function employed truly represents the value we place on different decisions. Constructing such a loss function can be quite difficult (see [8] for a systematic approach to the construction of utility functions). Fortunately however, many of the decision rules we will have cause to consider are insensitive to the exact form of the loss function employed provided it is restricted to a certain class; all symmetric loss functions for example.

5.1.2 Expected Utility or Loss

Except in case of perfect knowledge, a loss function in the form of $L(\mathbf{x}, \mathbf{a})$ is not very useful because the true state of nature $\mathbf{x} \in \mathcal{X}$ will not be known with precision and so the true loss incurred in taking an action will not be known. Rather, there will be a probability distribution $P(\mathbf{x})$ summarizing all the (probabilistic) information we have about the state at the time of decision making. With this information, one natural method of defining loss is as an expected loss (or Bayes expected loss) which for continuous-valued state variables is simply

$$\beta(\mathbf{a}) \stackrel{\triangle}{=} \mathrm{E}\{L(\mathbf{x},\mathbf{a})\} \ = \int_{-\infty}^{\infty} L(\mathbf{x},\mathbf{a})P(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{339}$$

and for discrete-valued state variables is given by

$$\rho(\mathbf{a}) = \ \stackrel{\triangle}{=} \mathrm{E}\{L(\mathbf{x},\mathbf{a})\} \ = \sum_{\mathbf{x}\in\mathcal{X}} L(\mathbf{x},\mathbf{a})P(\mathbf{x}). \tag{340}$$

Clearly, Bayes expected loss simply weights the loss incurred by the probability of occurrence (an average loss). More pessimistic approaches, such as game-theoretic methods, could also be used (this depends also on the *risk*).

5.1.3 Bayesian Decision Theory

It is most likely that probabilistic information concerning $\mathbf{x}$ is obtained after taking a number of observations, in the form of a posterior distribution $P(\mathbf{x} \mid \mathbf{Z}^n)$. An *expected utility* (or loss) β following an action $\mathbf{a}$ may then be defined with respect to a specified posterior distribution in the true state, as

$$\begin{aligned}\beta(P(\mathbf{x} \mid \mathbf{Z}^n),\mathbf{a}) &\stackrel{\triangle}{=} \mathrm{E}\{U(\mathbf{x},\mathbf{a})\} \\ &= \int_{\mathbf{x}} U(\mathbf{x},\mathbf{a})\ P(\mathbf{x} \mid \mathbf{Z}^n)\ \mathrm{d}\mathbf{x}.\end{aligned} \tag{341}$$

The Bayes action $\hat{\mathbf{a}}$ is defined as the strategy which maximises the posterior expected utility

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{a}} \beta(P(\mathbf{x} \mid \mathbf{Z}^n),\mathbf{a}) \tag{342}$$

It can be shown that this is equivalent to maximising

$$\int_{\mathbf{x}} U(\mathbf{x},\mathbf{a})\ P(\mathbf{Z}^n \mid \mathrm{x})\ P(\mathbf{x})\ \mathrm{d}\mathbf{x}. \tag{343}$$

Many well-known decision theoretic principles can be stated in terms of a Bayes action. For example, in estimation problems, the action set is made equal to the set of possible states of nature ($\mathcal{A} = \mathcal{X}$). In particular, the MMSE estimate defined by

$$\begin{aligned}\hat{\mathbf{x}} &= \arg\min_{\mathbf{a}\in\mathcal{X}} \int_{\mathbf{x}} (\mathbf{x}-\mathbf{a})^T(\mathbf{x}-\mathbf{a})P(\mathbf{x} \mid \mathbf{Z}^n)\mathrm{d}\mathbf{x} \\ &= \arg\min_{\mathbf{a}\in\mathcal{X}} \int_{\mathbf{x}} L(\mathbf{x},\mathbf{a})P(\mathbf{x} \mid \mathbf{Z}^n)\mathrm{d}\mathbf{x},\end{aligned} \tag{344}$$

is clearly a Bayes action with respect to a squared error loss function defined by $L(\mathbf{x},\hat{\mathbf{x}}) = (\mathbf{x}-\hat{\mathbf{x}})^T(\mathbf{x}-\hat{\mathbf{x}})$.

5.2 Decision Making with Multiple Information Sources

Decision-making with multiple sources of information is fundamentally more complex than the problem of single source decision making. The essential reason for this is that it is difficult to provide a consistent measure of utility or loss for each decision maker. In particular, two basic problems exist. First, how can the utilities of two different decision makers be compared unless there is a common measure of value. This is known as the problem of inter-personal utility comparison. Second, should decision makers aim to maximize a utility which expresses only local preferences, or should each decision maker evaluate its actions with respect to some common or *group* utility function. In the literature on the subject, no single agreed solution exists for these two problems. However, for specific decision making problems in which concepts of utility may be simplified and made precise, it is possible to arrive at consistent and useful solutions to both the utility comparison and the group utility problems. These are briefly discussed here.

The problem of making decisions involving multiple information sources has been formulated under a variety of assumptions in the literature [10][26]. We shall consider the following representative cases:

5.2.1 The Super Bayesian

Case 1: Consider a system consisting of N information sources and a single overall decision maker. The task of the decision maker is, in the first instance, to combine probabilistic information from all the sources and then to make decisions based on the global posterior. Given that the global posterior is $P(\mathbf{x} \mid \mathbf{Z}^k)$, the Bayes group action is given by

$$\begin{aligned}\hat{\mathbf{a}} &= \arg\max_{\mathbf{a}} \beta(P(\mathbf{x} \mid \mathbf{Z}^k), \mathbf{a}) \\ &= \arg\max_{a} \mathrm{E}\{U(\mathbf{x}, \mathbf{a}) \mid \mathbf{Z}^k\} \ , \end{aligned} \tag{345}$$

where $U(\mathbf{x}, \mathbf{a})$ is a group utility function. The solution in this case is well defined in terms of classical Bayesian analysis and is often termed the "super Bayesian" approach [44].

5.2.2 Multiple Bayesians

Case 2: Consider a system consisting of N Bayesians each able to obtain its own probabilistic information which it shares with *all* the other Bayesians before computing a posterior PDF. From the previous discussion, the posterior PDF obtained by each Bayesian i is identical and is given by $P(\mathbf{x}_i \mid \mathbf{Z}_i^k)$. Each Bayesian is then required to compute an optimal action which is consistent with those of the other Bayesians.

This second case describes a fully decentralised system. It presents a considerable challenge for which a universally prescribed solution is not possible. This is partly because of the lack of generally applicable criteria for determining rationality and optimality, coupled with the question of whether to pursue group or individual optimality. The trivial case exists where the optimal action $\mathbf{a}_i$ at each Bayesian i happens to be the same for all the Bayesians and thus becomes the group action. In general local decisions are not the same, and so a solution proceeds as follows; each Bayesian i computes an acceptable (admissible) set of actions $A_i \subseteq \mathcal{A}$ and if the set $A = \cap_j A_j$ is non-empty, then the group action is selected from the set A. An acceptable class of actions can be obtained by maximising

$$\beta(P(\mathbf{x} \mid \mathbf{Z}^k), \mathbf{a}) = \sum_j w_j \ \mathrm{E}\{U_j(\mathbf{x}, \mathbf{a}) \mid \mathbf{Z}^k\} \tag{346}$$

where $0 \leq w_j \leq 1$ and $\sum_j w_j = 1$. Equivalently, from the Likelihood Principle (Equation 343), this can be written as a maximisation of

$$\sum_j w_j \int U_j(\mathbf{x}, \mathbf{a}) \ P(\mathbf{Z}_j^k \mid \mathbf{x}) \ P(\mathbf{x}) \ \mathrm{d}\mathbf{x}. \tag{347}$$

Although it is clear that Equation 346 and Equation 347 represent acceptable actions, it is extremely difficult to choose a single optimal action $\hat{\mathbf{a}}$ from this acceptable set. The real problem is that the individual utilities U_i can not, in general, be compared as they may correspond to completely different scales of value. If each local utility function U_i were described on a common scale, then maximisation of Equation 346 or Equation 347 clearly gives the optimal group action. The general problem of comparing local utilities to arrive at an optimal group decision is known to be unsolvable (Arrow's impossibility theorem [3]). However it is also known that with a few additional restrictions on the form of U_i, such comparisons can be made to give sensible results.

In the case where comparison between local utilities can be justified, a solution to the group decision problem may be obtained by maximising Equation 346. A rather more general solution to the group decision problem in this case has been suggested in [44]. The optimal group decision is obtained from

$$\hat{\mathbf{a}} = \arg\max_a \left\{ \sum_j w_j \left[\mathrm{E}\{U_j(\mathbf{x}, \mathbf{a})\} \ - c(j)\right]^\gamma \right\}^{1/\gamma}, \tag{348}$$

where $c(j)$ is regarded as decision maker j's security level which plays the role of "safeguarding j's interests". The weight w_j is as given in Equation 346 and $-\infty \leq \gamma \leq \infty$. The case when $\gamma = 1$, gives a solution which minimises Equation 346 and $\gamma = -\infty$ and $\gamma = \infty$ give the so-called Bayesian max-min and min-max solutions respectively [8][44]. The solution arising from using $\gamma = 1$ and $\gamma = \infty$ can result in an individual decision-maker ending up with a net loss of expected utility, depending on the value of $c(j)$.

5.2.3 Nash Solutions

One soloution to the case when direct comparisons of utility cannot be justified is the well known Nash product [32] . The Nash product consists simply of a product of individual utilities. In this case, the absolute scale of each individual utility has no effect on the optimal action chosen. The Nash solution may be obtained from

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{a}} \prod_j \left[\mathrm{E}\{U_j(\mathbf{x}, \mathbf{a})\} \ - c(j) \right], \tag{349}$$

in which the value of $c(j)$ can be used to maintain some level of individual optimality. The value of $c(j)$ plays no part in the actual derivation of the Nash solution and is thus arbitrary.

In many situations, it may be that the various decision makers do not come to a single unified opinion on what the global decision should be. Such a situation occurs when the local utility of a global decision is much poorer than the value that would be obtained by "agreeing to disagree". This leads to a range of further issues in cooperative game playing and bargaining.

References

[1] D.L. Alspace and H.W. Sorenson. Gaussian filters for nonlinear filtering problems. *IEEE Trans. Automatic Control*, 17(2):439–448, 1972.

[2] B.D.O. Anderson and J.B. Moore. *Optimal Filtering.* Prentice Hall, 1979.

[3] K.J. Arrow. *Social Choice and Individual Values.* Wiley, 1966.

[4] Y. Bar-Shalom. On the track to track correlation problem. *IEEE Trans. Automatic Control*, 25(8):802–807, 1981.

[5] Y. Bar-Shalom. *Multi-Target Multi-Sensor Tracking.* Artec House, 1990.

[6] Y. Bar-Shalom. *Multi-Target Multi-Sensor Tracking II.* Artec House, 1990.

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

[8] J.O. Berger. *Statistical Decision Theory and Bayesian Analysis.* Springer Verlag, 1985.

[9] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems.* Artec House, 1999.

[10] D. Blackwell and M. A. Girshick. *Theory of Games and Statistical Decisions.* Dover Publications, 1954.

[11] P. L. Bogler. Shafter-dempster reasoning with applications to multisensor target identification systems. *IEEE Trans. Systems Man and Cybernetics*, 17(6):968–977, 1987.

[12] R.G. Brown and P.Y.C. Hwang. *Introduction to Applied Kalman Filtering.* John Wiley, 1992.

[13] D.M. Buede. Shafer-Dempster and Bayesian reasoning: a response to Shafer-Dempster reasoning with applications to multisensor target identification systems. *IEEE Trans. Systems Man and Cybernetics*, 18(6):1009–1011, 1988.

[14] D.E. Catlin. *Estimation, Control and the Discrete Kalman Filter.* Springer Verlag, 1984.

[15] C. Chong, K. Chang, and Y. Bar-Shalom. Joint probabilistic data association in distributed sensor networks. *IEEE Trans. Automatic Control*, 31(10):889–897, 1986.

[16] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* John Wiley, 1991.

[17] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications.* Academic Press, 1980.

[18] H.F. Durrant-Whyte. *Introduction to Estimation and The Kalman Filter.* Australian Centre for Field Robotics, 2000.

[19] S.Y. Harmon. Sensor data fusion through a blackboard. In *Proc. IEEE Int. Conf. Robotics and Automation*, page 1449, 1986.

[20] C.J. Harris and I. White. *Advances in Command, Control and Communication Systems.* IEE press, 1987.

[21] H.R. Hashemipour, S. Roy, and A.J. Laub. Decentralized structures for parallel Kalman filtering. *IEEE Trans. Automatic Control*, 33(1):88–93, 1988.

[22] K. Ito and K. Xiong. Gaussian filters for nonlinear filtering problems. *IEEE Trans. Automatic Control*, 45(5):910–927, 2000.

[23] S. Julier, J. Uhlmann, and H.F. Durrant-Whyte. A new method for the non-linear approximation of means and covariances in filters and estimators. *IEEE Trans. Automatic Control*, 45(3):477–481, 2000.

[24] T. Kilath. *Linear Systems.* John Wiley, 1980.

[25] P. Lancaster and M. Tismenetsky. *The Theory of Matrices.* Academic Press, 1985.

[26] R.D. Luce and H. Raiffa. *Games and Decisions.* Wiley, 1957.

[27] J. Manyika and H.F. Durrant-Whyte. *Data Fusion and Sensor Management: An Information-Theoretic Approach.* Prentice Hall, 1994.

[28] P.S. Maybeck. *Stochastic Models, Estimaton and Control, Vol. I.* Academic Press, 1979.

[29] V. Mazya and G. Schmidt. On approximating approximations using gaussian kernels. *IMA J. Numerical Anal.*, 16:13–29, 1996.

[30] R.E. Moore. *Interval Analysis.* Prentice Hall, 1966.

[31] A.G.O Mutambura. *Decentralised Estimation and Control for Multisensor Systems.* CRC Press, 1998.

[32] J.F. Nash. The bargaining problem. *Econometrica*, page 155, 1950.

[33] H.P. Nii. Blackboard systems. *AI Magazine*, 1986.

[34] A. Papoulis. *Probability, Random Variables, and Stochastic Processes; Third Edition.* McGraw-Hill, 1991.

[35] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausable Inference.* Morgan Kaufmann Publishers Inc., 1988.

[36] C.R. Rao. *Linear Statistical Inference and its Applications.* John Wiley, 1965.

[37] N.R. Sandell, P. Varaiya, M. Athans, and M.G. Safonov. Survey of decentralized control methods for large scale systems. *IEEE Trans. Automatic Control*, 23(2):108–128, 1978.

[38] H.W. Sorenson. Special issue on applications of Kalman filtering. *IEEE Trans. Automatic Control*, 28(3), 1983.

[39] J.L. Speyer. Communication and transmission requirments for a decentralized linear-quadratic-gaussian control problem. *IEEE Trans. Automatic Control*, 24(2):266–269, 1979.

[40] L.D. Stone, C.A. Barlow, and T.L. Corwin. *Bayesian Multiple Target Tracking.* Artech House, 1999.

[41] M. Stone. *Coordinate Free Multivaribale Statistics.* Oxford University Press, 1989.

[42] G. Strang. *Linear Algebra and its Applications, Third Edition.* Harcourt Brace Jovanovich, 1988.

[43] E.L. Waltz and J. Llinas. *Sensor Fusion.* Artec House, 1991.

[44] S. Weerahandi and J.V. Zidek. Elements of multi-Bayesian decision theory. *The Annals of Statistics*, 11(4):1032–1046, 1983.