Introduction to Decentralised Data Fusion

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

September 27, 2002

Version 3.1

This material is BAE SYSTEMS Proprietary Information
This material is for ASCTA use only

©Hugh Durrant-Whyte 2001
Contents

1 Introduction 5
   1.1 What is a Decentralised Data Fusion System? 5
   1.2 Course Summary 6
   1.3 Bibliography 8
   1.4 Acknowledgements 9

2 Probabilistic Data Fusion 10
   2.1 Probabilistic Models 10
   2.2 Probabilistic Methods 12
      2.2.1 Bayes Theorem 12
      2.2.2 Data Fusion using Bayes Theorem 16
      2.2.3 Recursive Bayes Updating 20
      2.2.4 Bayesian Filtering and Tracking 24
      2.2.5 Distributed Data Fusion with Bayes Theorem 28
      2.2.6 Data Fusion with Log-Likelihoods 32
   2.3 Information Measures 35
      2.3.1 Entropic Information 35
      2.3.2 Conditional Entropy 36
      2.3.3 Mutual Information 39
      2.3.4 Fisher Information 43
      2.3.5 The Relation between Shannon and Fisher Measures 44

3 Multi-Sensor Estimation 45
   3.1 The Kalman Filter 45
      3.1.1 State and Sensor Models 46
      3.1.2 The Kalman Filter Algorithm 50
      3.1.3 The Innovation 54
      3.1.4 Asynchronous, Delayed and Asequent Observations 56
      3.1.5 The Extended Kalman Filter 61
      3.1.6 The Covariance Intersect (CI) Algorithm 64
   3.2 The Multi-Sensor Kalman Filter 66
      3.2.1 Observation Models 68
      3.2.2 The Group-Sensor Method 71
      3.2.3 The Sequential-Sensor Method 75
      3.2.4 The Inverse-Covariance Form 77
      3.2.5 Track-to-Track Fusion 84

4 Decentralised Data Fusion Systems 88
   4.1 Data Fusion Architectures 88
      4.1.1 Hierarchical Data Fusion Architectures 89
      4.1.2 Distributed Data Fusion Architectures 92
      4.1.3 Decentralised Data Fusion Architectures 93
4.2 Decentralised Estimation ........................................ 94
  4.2.1 The Information Filter ..................................... 96
  4.2.2 The Information Filter and Bayes Theorem .................. 101
  4.2.3 The Information filter in Multi-Sensor Estimation .......... 102
  4.2.4 The Hierarchical Information Filter ......................... 105
  4.2.5 The Decentralised Information Filter ....................... 107

4.3 Decentralised Multi-Target Tracking ............................ 110
  4.3.1 Decentralised Data Association ............................. 110
  4.3.2 Decentralised Identification and Bayes Theorem ............. 112

5 Communication in Decentralised Sensing Systems .................. 114
  5.1 Decentralised Communication Theory .......................... 114
    5.1.1 Communication Topologies ................................ 114
    5.1.2 Bayesian Communication in Sensor Networks ............... 116
    5.1.3 Identification of Redundant Information in Sensor Networks ... 118
    5.1.4 Communication Channel Filters .................................. 120
    5.1.5 Fully Connected and Broadcast Sensor Networks ............ 124
    5.1.6 General Network Structures .................................. 125
  5.2 Delays and Timing: Temporal Propagation of Information ......... 128
    5.2.1 System Definitions ......................................... 128
    5.2.2 Forward Propagation of State ................................. 129
    5.2.3 Backward Propagation of State ............................... 129
    5.2.4 Notation for Time Invariant Systems ......................... 130
    5.2.5 Definition of Estimates and Information States ............. 131
    5.2.6 Forward Time Propagation .................................... 131
    5.2.7 Backward Propagation ........................................ 132
    5.2.8 Delayed Data ................................................ 133
    5.2.9 Information Observation and Update .......................... 133
  5.3 Delays in Channels and Burst Communication ..................... 134
    5.3.1 Node Algorithms ............................................. 134
    5.3.2 Channel Algorithms .......................................... 135
  5.4 The Communication Channel Algorithm .......................... 136
    5.4.1 Management of Communication ................................. 136
    5.4.2 Structure of the Communication Algorithm .................. 138
    5.4.3 Information Flow ............................................. 140
    5.4.4 Computing the Common Information ........................... 140

6 Management in Decentralised Systems ............................... 141
  6.1 Bayesian Decision Theory ...................................... 142
    6.1.1 Structure of the Problem .................................... 142
    6.1.2 Utility or Loss .............................................. 143
    6.1.3 Expected Utility or Loss ..................................... 143
    6.1.4 Bayes Actions ................................................. 144
6.2 The Utility of Information ................................................. 144
6.2.1 Information as Expected Utility ...................................... 146
6.2.2 Information Filter Metrics ............................................. 147
6.2.3 Discrete State Metrics ................................................ 147
6.2.4 Composite Metrics .................................................... 148
6.3 Decision Making with Multiple Information Sources .................. 148
6.3.1 The Super Bayesian .................................................. 148
6.3.2 Multiple Bayesians .................................................... 148
6.3.3 Practical Bargaining Solutions ........................................ 149
6.4 Decentralised Sensor Management ........................................ 151
6.4.1 Sensor to Target Assignment .......................................... 151
6.4.2 Sensor Hand-off and Cueing .......................................... 155
6.5 Communications Management ............................................ 157
6.6 Organisation and Control of Network Resource .......................... 159

7 Applications and Practical Considerations .................................. 163
7.1 Historical Projects .......................................................... 163
7.1.1 SKIDS ..................................................................... 163
7.1.2 ISSS .................................................................. 163
7.2 OxNav ........................................................................ 164
7.2.1 Organisation and Management .......................................... 166
7.2.2 OxNav System and Demonstrations .................................... 167
7.3 ANSER ........................................................................ 168
7.3.1 Objectives .................................................................. 168
7.3.2 Demonstrations ............................................................. 169
7.3.3 Research .................................................................... 175
7.4 Implementation of DDF Algorithms in ANSER ......................... 175
7.4.1 Preprocessing and Coordinate Transformations .................... 177
7.4.2 Data Association .......................................................... 179
7.4.3 Local Filter ................................................................. 180
7.4.4 Channel Filter .............................................................. 181
7.4.5 Channel Manager .......................................................... 181
7.4.6 Timing ..................................................................... 182
7.4.7 Picture Compilation ........................................................ 183
7.4.8 Terrain Navigation ........................................................ 185
7.4.9 The Communication Layer .............................................. 185
7.5 Large Scale Sensor Networks ............................................... 186
1 Introduction

This course provides a practical introduction to decentralised data fusion (DDF) methods and their application for engineers and scientists wishing to employ and develop DDF systems. The course is mathematically advanced and assumes knowledge of estimation theory and probabilistic model methods. The course emphasizes practical implementations in ground and airborne sensor networks. A key feature of the course is the use of laboratory sessions, based on Matlab, in which DDF methods are implemented and evaluated by students. The intended outcomes of the course are to provide students with the theoretical and practical skills necessary to design, implement and evaluate DDF algorithms.

1.1 What is a Decentralised Data Fusion System?

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 1, 2 and 3 show three possible realisations of a decentralised data fusion system (these are discussed in more detail later in the course). 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 \textit{a priori}, nodes can be constructed and programmed in a \textbf{modular} fashion.

A decentralized system is characterised by being modular, scalable and survivable. Together, these give decentralised systems a substantial advantage over more traditional sensing architectures in a range of defense and civilian data fusion tasks.

Figure 1: A decentralised data fusion system implemented with a point-to-point communication architecture.

Decentralised data fusion methods capture many of the concepts embodied in “network-centric” system architectures. In particular, they immediately impose a requirement for modularity, for scalability and robustness of the network, and for local intelligence at sensor nodes. However, unlike most current work in network centric systems, DDF methods provide a mathematical and algorithmic foundation for the development of such systems.

1.2 Course Summary

This course consists of three main parts.

The first part introduces the problem of data fusion and particularly distributed and decentralised data fusion using probabilistic methods. Section 2 introduces Bayes theorem
Figure 2: 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 3: A decentralised data fusion system implemented with a hybrid, broadcast and point-to-point, communication architecture.
is the primary data fusion mechanism and describes how the algorithm can be distributed and decentralised amongst a number of sensor nodes. The two basic problems in continuous and discrete state estimation are described. Section 2 also introduces important formal measures of Information. These are important in understanding the operation of decentralised sensor networks and in further developing sensor and network management methods. Section 3 then turns attention to the problem of multi-sensor estimation. Continuous state estimation (tracking) is probably the single most important problem in data fusion as it aims to provide a single composite geometric picture of the environment. Section 3 develops both the conventional multi-target multi-sensor Kalman filter and also introduces the Information form of this problem. The Information form subsequently gives rise to the decentralised Kalman filter and related decentralised tracking algorithms.

The second part of this course deals with the essential decentralised data fusion algorithms. Section 4 develops and describes the decentralised Kalman filter algorithm and its application to multi-target multi-sensor tracking problems. The problem of decentralised discrete estimation (classification for example) is also discussed in this context. The operation of the algorithm, communication requirements, and structure of the resulting sensor nodes are described. The modularity, scalability and survivability of the resulting network algorithms is discussed and demonstrated. Section 5 focuses on the issue of communication in decentralised sensor networks. In particular communication algorithms for broadcast, tree-connected, and arbitrary networks. Communication algorithms addressing delayed and asequent data problems are developed. Issues of intermittent and burst communication are also considered.

The third part of this course deals with some advanced principles in decentralised data fusion with the objective of demonstrating how the basic algorithms are applied to more complex and more realistic sensing problems. Section 5 considers the problem of decentralised sensor and network management. This includes issues of sensor-based decision making in a decentralised environment, (sensor-to-target assignment and sensor hand-off), as well as issues in communications and network management. Section 6 finally looks at a number of practical implementations of the decentralised algorithms in both past and current projects. The first project describes the development of a fully modular mobile robot implementing tracking, map-building, platform control and sensor management algorithms. The second project is the current ANSER programme aiming to fly multiple-UAVs cooperating in a fully decentralised form to build ground target pictures and navigation maps. The potential future development of algorithms for very large scale decentralised ground sensor networks is also discussed.

1.3 Bibliography

There is relatively little material on decentralised data fusion methods; indeed these course notes are probably the most comprehensive assembly of such information. A comprehensive bibliography of the citations used in this work is included at the end of these course notes. Here we comment on some general sources:

The general data fusion is well covered in the recent book by Blackman and Popoli...
[11]. 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. The books by Barshalom are also useful sources [5, 3, 4]. There are other notable data fusion books ([63] for example), although these tend to be largely oriented around block diagrams.

There are two books on decentralised data fusion published by some of my previous Ph.D. students. The book by James Manyika [36] focuses on problems in sensor management but also captures some important aspects in the implementation of decentralised systems. The book by Arthur Mutambara [42] is mainly concerned with the problem of decentralised estimation and control. Both books reflect the state-of-the-art in 1995.

A useful source of detailed information are the theses of my previous Ph.D. students. I can often obtain or make available copies of these if required. The theses are generally listed in the citations at the end of these course notes. In addition there are a significant number of papers published in the open literature over the past decade.

Finally there is project documentation, particularly for the current ANSER project. This is usually the most detailed and most current information available. However, detailed project documentation is not normally widely distributed: requests should be made through the programme manager at BAE Systems.

1.4 Acknowledgements

The work described in this course is the result of over a decade of research by myself, my students and research staff, and BAE Systems.

A large number of outstanding graduate students have been involved in this research: Bobby Rao, Stewart Grime, Tim Berg, James Manyika, Mike Stevens, Peter Ho, Tom Burke, Mariano Fernandez, Simukai Utete, Arthur Mutambara, Simon Julier, Jeffrey Uhlmann, Rob Deaves and Eric Nettleton. The course draws substantially on the work described in their theses.

The research would not have been possible without the considerably and continuous support of Phil Greenway at BAE Systems Sowerby Research Centre. A number of staff at Sowerby have also made substantial contributions to the work described here, including: Rob Deaves, Dave Nicholson, Julia Suttcliffe, and Rob Dawkins.
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.

This section begins by briefly introducing the essential elements of probabilistic modeling: probability densities, conditional densities, and Bayes theorem. With these elements, the basic data fusion problem is described and it is shown how information can be combined in a probabilistic manner. From this, the construction of various data fusion architectures is described. It is demonstrated how the mathematics required for data fusion in these architectures can be developed and implemented directly from Bayes theorem. This methodology underlies many of the decentralised data fusion algorithms developed in the remainder of the this course. The idea of information and entropy are then introduced. Entropy measures are a natural way of describing the flow of ‘information’ in different data fusion architectures. It is demonstrated how these information measures can be used to predict, control and manage the data fusion process. These methods, together with the information form of the Kalman filter are the underlying basis for the decentralised 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 $y$, generally written as $P_y(y)$ or simply $P(y)$ 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 $y$; observation or state. The pdf $P(y)$ is considered valid if:

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

$$\int_y P(y)dy = 1.$$ 

The joint distribution $P_{xy}(x,y)$ is defined in a similar manner.
Integrating the pdf $P_{xy}(x,y)$ over the variable $x$ gives the marginal pdf $P_y(y)$ as

$$P_y(y) = \int_x P_{xy}(x,y)dx,$$

and similarly integrating over $y$ gives the marginal pdf $P_x(x)$. The joint pdf over $n$ variables, $P(x_1, \cdots, x_n)$, may also be defined with analogous properties to the joint pdf of two variables.

The conditional pdf $P(x \mid y)$ is defined by

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

and has the usual properties of a pdf with $x$ the dependent variable given that $y$ takes on specific fixed values. The conditional pdf $P(y \mid x)$ 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,

$$P(x,y) = P(x \mid y)P(y).$$

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

$$P(x_1, \cdots, x_n) = P(x_1 \mid x_2 \cdots, x_n) \cdots P(x_{n-1} \mid x_n)P(x_n),$$

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

$$P_y(y) = \int_x P_{x|y}(y \mid x)P_x(x)dx.$$

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

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

$$P(x \mid y) = P(x).$$

With Equation 6 substituted into Equation 3

$$P(x,y) = P(x)P(y).$$

A weaker form of independence can be defined through the important idea of conditional independence. Given three random variables $x$, $y$ and $z$, the conditional distribution of
x given both y and z is defined as \( P(x \mid yz) \). If knowledge of the value of z makes the value of x independent of the value of y then
\[
P(x \mid y, z) = P(x \mid z).
\] (8)

This may be the case for example if z indirectly contains all the information contributed by y to the value of 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 x, y, and z
\[
P(x, y, z) = P(x \mid y, z)P(y \mid z)P(z),
\] (9)
together with the conditional independence result of Equation 8 the intuitive result
\[
P(x, y \mid z) = P(x \mid z)P(y \mid z),
\] (10)
is obtained. That is, if x is independent of y given knowledge of z then the joint probability density function of x and y conditioned on z is simply the product of the marginal distributions of x and y each conditioned on z, analogously to Equation 7.

The idea of conditional independence underlies many data fusion algorithms. Consider the state of a system x and two observations of this state \( z_1 \) and \( z_2 \). It should be clear that the two observations are not independent,
\[
P(z_1, z_2) \neq P(z_1)P(z_2),
\]
as they must both depend on the common state 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
\[
P(z_1, z_2 \mid x) = P(z_1 \mid x)P(z_2 \mid x).
\]
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 x and z on which is defined a joint probability density function \( P(x, z) \). The chain-rule of conditional probabilities can be used to expand this density function in two ways
\[
P(x, z) = P(x \mid z)P(z)
\] = \[
P(z \mid x)P(x).
\] (11)
Rearranging in terms of one of the conditional densities, Bayes theorem is obtained

\[
P(x \mid z) = \frac{P(z \mid x)P(x)}{P(z)}. \tag{12}
\]

The value of this result lies in the interpretation of the probability density functions \(P(x \mid z), P(z \mid x), \text{ and } P(x)\). Suppose it is necessary to determine the various likelihoods of different values of an unknown state of nature \(x \in \mathcal{X}\). There may be prior beliefs about what values of \(x\) might be expect, encoded in the form of relative likelihoods in the prior probability density function \(P(x)\). To obtain more information about the state \(x\), an observation \(z \in \mathcal{Z}\) is made. The observations made are modeled as a conditional probability density function \(P(z \mid x)\) which describes, for each fixed state of nature \(x \in \mathcal{X}\), the likelihood that the observation \(z \in \mathcal{Z}\) will be made; the probability of \(z\) given \(x\). The new likelihoods associated with the state of nature \(x\) must now be computed from the original prior information and the information gained by observation. This is encoded in the posterior distribution \(P(x \mid z)\) which describes the likelihoods associated with \(x\) given the observation \(z\). The marginal distribution \(P(z)\) 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 \(P(z \mid x)\) 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 of \(x = x\) and then asking what pdf in the variable \(z\) results. Thus, in this case, \(P(z \mid x)\) is considered as a distribution on \(z\). For example, if we know the true range to a target \((x)\), then \(P(z \mid x)\) is the distribution on the actual observation of this range. Conversely, once a sensor model exists, observations (numbers, not distributions) are made and \(z = z\) is fixed. From this however, we want to infer the state \(x\). Thus the distribution \(P(z \mid x)\) is now considered as a distribution in \(x\). In this latter, case, the distribution is known as the Likelihood Function and the dependence on \(x\) is made clear by writing \(\Lambda(x) = P(z \mid x)\).

In a practical implementation of Equation 12, \(P(z \mid x)\) is constructed as a function of both variables (or a matrix in discrete form). For each fixed value of \(x\), a distribution in \(z\) is defined. Therefore as \(x\) varies, a family of distributions in \(z\) 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 \(x\), the range to target for example, and an observation \(z\) 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 \(x\) and a variance \(\sigma^2_z\) as

\[
P(z \mid x) = \frac{1}{\sqrt{2\pi}\sigma^2_z} \exp \left( -\frac{1}{2} \frac{(z - x)^2}{\sigma^2_z} \right). \tag{13}
\]

\(^1\)Random variables are denoted by a bold face font, specific values taken by the random variable are denoted by normal fonts.
It should be clear that this is a simple function of both $z$ and $x$. If we know the true value of the state, $x = x$, then the distribution is a function of $z$ only; describing the probability of observing a particular value of range ( Normally distributed around the true range $x$ with variance $\sigma_x^2$). Conversely, if we make a specific observation, $z = z$, then the distribution is a function of $x$ only; describing the probability of the true range value ( Normally distributed around the range observation $z$ with variance $\sigma_z^2$). In this case, the distribution is the Likelihood Function.

Now assume that we have some prior belief about the true state $x$ encoded in a Gaussian prior as

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

Note, this is a function of a single variable $x$ (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 $x$ only, and is sometimes referred to as the posterior likelihood) as

$$P(x | z) = C \frac{1}{\sqrt{2\pi \sigma_z^2}} \exp \left( -\frac{1}{2} \frac{(x - z)^2}{\sigma_z^2} \right) \cdot \frac{1}{\sqrt{2\pi \sigma_x^2}} \exp \left( -\frac{1}{2} \frac{(x - x_p)^2}{\sigma_x^2} \right).$$

(14)

$$= \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left( -\frac{1}{2} \frac{(x - \bar{x})^2}{\sigma^2} \right)$$

(15)

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

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

and

$$\sigma^2 = \frac{\sigma_x^2 \sigma_z^2}{\sigma_x^2 + \sigma_z^2} = \frac{1}{\sigma_x^2 + \frac{1}{\sigma_z^2}}^{-1}.$$

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 $x$ which can take on one of three values:

- $x_1$: $x$ is a type 1 target.
$x_2$: $x$ is a type 2 target.

$x_3$: No visible target.

A single sensor observes $x$ 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 $P_1(z \mid x)$:

\[
\begin{array}{ccc}
   & z_1 & z_2 & z_3 \\
 x_1 & 0.45 & 0.45 & 0.1 \\
 x_2 & 0.45 & 0.45 & 0.1 \\
 x_3 & 0.1 & 0.1 & 0.8 \\
\end{array}
\]

Note, that this likelihood matrix is a function of both $x$ and $z$. 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 $\Lambda(x)$.

The posterior distribution of the true state $x$ after making an observation $z = z_i$ is given by

\[
P(x \mid z_i) = \alpha P_1(z_i \mid x) P(x)
\]

where $\alpha$ is simply a normalizing constant set by requiring the sum, over $x$, 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 $P(x) = (0.333, 0.333, 0.333)$. If we now observe $z = z_1$, then clearly the posterior distribution will be given by $P(x \mid z_1) = (0.45, 0.45, 0.1)$ (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 $P(x) = (0.45, 0.45, 0.1)$, and we again make the observation $z = z_1$, then the new posterior distribution will be given by

\[
P(x \mid z_1) = \alpha P_1(z_1 \mid x) P(x)
= \alpha \times (0.45, 0.45, 0.1) \otimes (0.45, 0.45, 0.1)
= (0.488, 0.488, 0.024).
\]

(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

\[ Z^n \triangleq \{ z_1 \in Z_1, \ldots, z_n \in Z_n \}. \]

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

\[
P(x \mid Z^n) = \frac{P(Z^n \mid x) P(x)}{P(Z^n)} = \frac{P(z_1, \ldots, z_n \mid x) P(x)}{P(z_1, \ldots, z_n)}. \tag{16}
\]

In practice it would be difficult to do this because it requires that the joint distribution \( P(z_1, \ldots, z_n \mid x) \) is known completely; that is, the joint distribution of all possible combinations of observations conditioned on the underlying state. However, it is usually quite reasonable to assume that given the true state \( x \in \mathcal{X} \), 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

\[
P(z_i \mid x, z_{1, \ldots, i-1, i+1, \ldots, n}) = P(z_i \mid x), \tag{17}
\]

and from Equation 10 this gives

\[
P(z_1, \ldots, z_n \mid x) = P(z_1 \mid x) \cdots P(z_n \mid x) = \prod_{i=1}^{n} P(z_i \mid x). \tag{18}
\]

Substituting this back into Equation 16 gives

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

Thus the updated likelihoods in the state, the posterior distribution on \( x \), is simply proportional to the product of prior likelihood and individual likelihoods from each information source. The marginal distribution \( P(Z^n) \) 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 [10]. In practice, the conditional probabilities \( P(z_i \mid x) \) are stored \textit{a priori} as functions of both \( z_i \) and \( x \). When an observation sequence \( Z^n = \{ z_1, z_2, \ldots, z_n \} \) is made, the observed values are instantiated

\footnote{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.}
in this probability distribution and likelihood functions $\Lambda_i(x)$ are constructed, which are functions only of the unknown state $x$. The product of these likelihood functions with the prior information $P(x)$, appropriately normalised, provides a posterior distribution $P(x \mid Z^n)$, which is a function of $x$ only for a specific observation sequence \( \{z_1, z_2, \cdots, z_n\} \). Figure 4 shows the structure of the independent likelihood pool in a centralised architecture.

![Figure 4: 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 $P_i(z_i \mid x)$, together with any prior probabilistic knowledge $P(x)$. On arrival of a measurement set $Z^n$, each sensor model is instantiated with the associated observation to form a likelihood $\Lambda_i(x)$. The normalised product of these yields the posterior distribution $P(x \mid Z^n)$.](image)

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;

$$P(z_1, \cdots, z_n) \neq P(z_1) \cdots P(z_n),$$

because each piece of information depends on a common underlying state $x \in \mathcal{X}$. 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 corre-
spondingly reasonable to assume that the information gathered is conditionally indepen-
dent 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.

**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
matrix $P_2(z_2 \mid x)$ is described by

$$
\begin{array}{ccc}
  z_1 & z_2 & z_3 \\
  x_1 & 0.45 & 0.1 & 0.45 \\
  x_2 & 0.1 & 0.45 & 0.45 \\
  x_3 & 0.45 & 0.45 & 0.1 \\
\end{array}
$$

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 $z = z_1$
with this second sensor, the posterior distribution on possible true states will be
given by $P(x \mid z_1) = (0.45, 0.1, 0.45)$ (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 $P_{12}(z_1, z_2 \mid x) = P_1(z_1 \mid x)P_2(z_2 \mid x)$. Thus if we observe $z_1 = z_1$
using the first sensor, and $z_2 = z_1$ with the second sensor (assuming a uniform prior),
then the posterior likelihood in $x$ is given by

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

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 $z_1, z_2$ observation pair, results in the combined
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 \(z_1 = z_1\)) and again observe target 1 with the second sensor (the first column of this block), then the posterior distribution in the three hypotheses is

\[
P(\mathbf{x} \mid z_1, z_2) = (0.692, 0.154, 0.154),
\]

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,

---

\(^3\)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.
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 \(P(z_k | x)\), 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 \(Z^k \triangleq \{z_k, Z^{k-1}\}\)

\[
P(x, Z^k) = P(x | Z^k)P(Z^k) = P(z_k | Z^{k-1} | x)P(x) \tag{21}
\]

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

\[
P(x | Z^k)P(Z^k) = P(z_k | x)P(Z^{k-1} | x)P(x) \tag{23}
\]

Noting that \(P(Z^k)/P(Z^{k-1}) = P(z_k | Z^{k-1})\) and rearranging gives

\[
P(x | Z^k) = \frac{P(z_k | x)P(x | Z^{k-1})}{P(z_k | Z^{k-1})}. \tag{25}
\]

The advantage of Equation 25 is that we need compute and store only the posterior likelihood \(P(x | Z^{k-1})\) which contains a complete summary of all past information. When
the next piece of information $P(z_k \mid x)$ 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 $x$ under the assumption that the likelihood function for the observations given the true state is Gaussian with known variance $\sigma^2$:

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

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

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

then the posterior distribution in $x$ after the first $k$ observations is given by

$$P(x \mid Z^k) = K \frac{1}{\sqrt{2\pi\sigma^2_k}} \exp \left( -\frac{1}{2} \frac{(z_k - x)^2}{\sigma^2_k} \right) \cdot \frac{1}{\sqrt{2\pi\sigma^2_{k-1}}} \exp \left( -\frac{1}{2} \frac{(x_{k-1} - x)^2}{\sigma^2_{k-1}} \right)$$

$$= \frac{1}{\sqrt{2\pi\sigma^2_k}} \exp \left( -\frac{1}{2} \frac{(x_k - x)^2}{\sigma^2_k} \right)$$

where $K$ is a constant independent of $x$ chosen to ensure that the posterior is appropriately normalized, and $x_k$ and $\sigma^2_k$ are given by

$$x_k = \frac{\sigma^2_{k-1} x_{k-1} + \sigma^2 z_k}{\sigma^2_{k-1} + \sigma^2} + \frac{\sigma^2}{\sigma^2_{k-1} + \sigma^2} x_{k-1},$$

and

$$\sigma^2_k = \frac{\sigma^2 \sigma^2_{k-1}}{\sigma^2 + \sigma^2_{k-1}}.$$
Figure 5: 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.
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 5(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 \( dx_i \cdot dy_j \). The only constraints placed on this distribution are that
\[
P(x_i, y_j) > 0, \quad \forall x_i, y_j
\]
and that
\[
\sum_i \sum_j P(x_i, x_j) dx_i dy_j = 1
\]
(which can easily be satisfied by appropriate normalisation.)

A sensor (sensor 1) now takes observations of the target from a sensor located at \( x = 15, y = 0 \) km. The likelihood function generated from this sensor following an observation \( z_1 \) is shown in Figure 5(b). This likelihood \( P_1(z_1 | x_i, y_j) \) 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 5(c) and is computed from the point-wise product of prior and likelihood,
\[
P(x_i, y_j | z_1) = \alpha \times P_1(z_1 | x_i, y_j) \otimes P(x_i, y_j),
\]
where \( \alpha \) 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 5(d) shows the posterior \( P(x_i, y_j | z_1, z_2) \) following a second observation \( z_2 \) by the same sensor. This is again computed by point-wise multiplication of the likelihood \( P_1(z_2 | x_i, y_j) \) with the new prior (the posterior from the previous observation \( P(x_i, y_j | z_1) \)). 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.

A second sensor (sensor 2) now takes observations of the target from a location \( x = 50, y = 20 \). Figure 5(e) shows the target likelihood \( P_2(z_3 | x_i, y_j) \) 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 \( P(x_i, y_j | z_1, z_2) \) from the previous two observations of sensor 1), we obtain the posterior \( P(x_i, y_j | z_1, z_2, z_3) \) shown in Figure 5(f) which shows all probability mass highly concentrated around a single target location.
2.2.4 Bayesian Filtering and Tracking

The general filtering and tracking problem can be formulated in Bayesian form. This is significant because it provides a common representation for a range of discrete and continuous data fusion problems without recourse to specific target or observation models. This general representation forms a "base class" from which specific data fusion, tracking and identification tasks can be derived. Distributed data fusion algorithms for specific network architectures will subsequently be formulated with respect to this general representation. From these, specific decentralised data fusion algorithms can be derived.

Consider a state $x_t$ which is generally taken to be a function of time $k$. This may, for example, describe a target to be tracked or the location of a platform for which navigation data is required. For convenience and without loss of generality, time is defined at discrete (asynchronous) times $t_k \triangleq k$. At a time instant $k$, the following quantities are defined:

- **$x_k$**: The state vector to be estimated at time $k$.
- **$u_k$**: A control vector, assumed known, and applied at time $k - 1$ to drive the state from $x_{k-1}$ to $x_k$ at time $k$.
- **$z_k$**: An observation taken of the state $x_k$ at time $k$.

In addition, the following sets are also defined:

- The history of states: $X^k = \{x_0, x_1, \ldots, x_k\} = \{X^{k-1}, x_k\}$.
- The history of control inputs: $U^k = \{u_1, u_2, \ldots, u_k\} = \{U^{k-1}, u_k\}$.
- The history of state observations: $Z^k = \{z_1, z_2, \ldots, z_k\} = \{Z^{k-1}, z_k\}$.

In probabilistic form, the general data fusion problem aims to find the posterior density

$$P(x_k \mid Z^k, U^k, x_0).$$

for all times $k$. This is the posterior density of the state (at time $k$) given the recorded observations and control inputs up to and including time $k$ together (possibly) with knowledge of the initial state $x_0$.

In general, a recursive solution to this data fusion problem is desirable. Starting with an estimate for the distribution $P(x_{k-1} \mid Z^{k-1}, U^{k-1})$ at time $k - 1$, the joint posterior, following a control $u_k$ and observation $z_k$, is to be computed using Bayes Theorem. This computation requires that a state transition model and an observation model are defined describing the effect of the control input and observation respectively.

The observation model describes the probability of making an observation $z_k$ when the true state $x(k)$ is known, and is generally described in the form

$$P(z_k \mid x_k).$$
It is reasonable to assume that observations are conditionally independent given knowledge of the current state so that

$$P(Z^k \mid X^k) = \prod_{i=1}^{k} P(z_i \mid X^k) = \prod_{i=1}^{k} P(z_i \mid x_i).$$

(32)

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

$$P(x_k \mid x_{k-1}, u_k).$$

(33)

That is, the state transition may reasonably be assumed to be a Markov process in which the next state $x_k$ depends only on the immediately proceeding state $x_{k-1}$ and the applied control $u_k$, and is independent of both the observations and preceding states. With these definitions and models, Bayes Theorem may be employed to define a recursive solution to Equation 30.

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

$$P(x_k, z_k \mid Z^{k-1}, U^k, x_0) = P(x_k \mid z_k, Z^{k-1}, U^k, x_0)P(z_k \mid Z^{k-1}, U^k, x_0)$$

(34)

and then in terms of the observation

$$P(x_k, z_k \mid Z^{k-1}, U^k, x_0) = P(z_k \mid x_k, Z^{k-1}, U^k, x_0)P(x_k \mid Z^{k-1}, U^k, x_0)$$

(35)

where the last equality employs the assumptions established for the sensor model in Equation 31.

Equating Equations 34 and 35 and rearranging gives

$$P(x_k \mid Z^k, U^k, x_0) = \frac{P(z_k \mid x_k)P(x_k \mid Z^{k-1}, U^k, x_0)}{P(z_k \mid Z^{k-1}, U^k)}.$$
Figure 6: Observation update for the full Bayes filter. Prior to observation, an observation model in the form of the conditional density $P(z_k \mid x_k)$ is established. For a fixed value of $x_k$, equal to $x_1$ or $x_2$ for example, a density function $P(z_k \mid x_k = x_1)$ or $P(z_k \mid x_k = x_2)$ is defined describing the likelihood of making the observation $z_k$. Together the density $P(z_k \mid x_k)$ is then a function of both $z_k$ and $x_k$. This conditional density then defines the observation model. Now, in operation, a specific observation $z_k = x_1$ is made and the resulting distribution $P(z_k = x_1 \mid x_k)$ defines a density function (now termed the likelihood function) on $x_k$. This density is then multiplied by the prior density $P(x_k)$ and normalised to obtain the posterior distribution $P(x_k \mid z_k)$ describing knowledge in the state after observation.
Figure 7: Time update step for the full Bayes filter. At a time $k-1$, knowledge of the state $x_{k-1}$ is summarised in a probability distribution $P(x_{k-1})$. A vehicle model, in the form of a conditional probability density $P(x_k | x_{k-1})$, then describes the stochastic transition of the vehicle from a state $x_{k-1}$ at a time $k-1$ to a state $x_k$ at a time $k$. Functionally, this state transition may be related to an underlying kinematic vehicle model in the form $x_k = f(x_{k-1}, u_k)$. The figure shows two typical conditional probability distributions $P(x_k | x_{k-1})$ on the state $x_k$ given fixed values of $x_{k-1}$. The product of this conditional distribution with the marginal distribution $P(x_{k-1})$, describing the prior likelihood of values of $x_{k-1}$, gives the the joint distribution $P(x_k, x_{k-1})$ shown as the surface in the figure. The total marginal density $P(x_k)$ describes knowledge of $x_k$ after state transition has occurred. The marginal density $P(x_k)$ is obtained by integrating (projecting) the joint distribution $P(x_k, x_{k-1})$ over all $x_{k-1}$. Equivalently, using the total probability theorem, the marginal density can be obtained by integrating (summing) all conditional densities $P(x_k | x_{k-1})$ weighted by the prior probability $P(x_{k-1})$ of each $x_{k-1}$. The process can equally be run in reverse (a retroverse motion model) to obtain $P(x_{k-1})$ from $P(x_k)$ given a model $P(x_{k-1} | x_k)$.
where the last equality follows from the assumed independence of vehicle motion from map
and observations, and from the causality of the vehicle control input on vehicle motion.

Equation 37 is the time update step for the full Bayes data fusion algorithm. A
graphical description of this equation is shown in Figure 7.

The final step is simply to substitute Equation 37 into Equation 36 to yield

\[
P(x_k | Z^k, U^k, x_0) = \mathcal{K} P(z_k | x_k) \int P(x_k | x_{k-1}, u_k) P(x_{k-1} | Z^{k-1}, U^{k-1}, x_0) \, dx_{k-1}.
\]

Equation 38 provides a recursive expression for the calculation of the joint posterior
\( P(x_j | Z^j, U^j, x_0) \) for the state \( x_j \) at a time \( j \) based on all observations \( Z^j \) and all control
inputs \( U^j \) up to and including time \( j \). The recursion is a function of a state transition
model \( P(x_k | x_{k-1}, u_k) \) and an observation model \( P(z_k | x_k) \). A related problem is to in-
clude the complete history of states (a smoothed trajectory estimates) as \( P(X^k | Z^k, U^k) \).
However this is not usually required.

### 2.2.5 Distributed Data Fusion with Bayes Theorem

![Figure 8: The distributed implementation of the independent likelihood pool. Each sensor maintains it’s own model in the form of a conditional probability distribution \( P_i(z_i | x) \). On arrival of a measurement, \( z_i \), the sensor model is instantiated with the associated observation to form a likelihood \( \Lambda_i(x) \). This is transmitted to a central fusion centre were the normalised product of likelihoods and prior yields the posterior distribution \( P(x | Z^n) \).](image-url)
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 8 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 $\Lambda_i(x)$ 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 4). 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 as

$$P(x \mid Z^n) = P(x) \prod_i \Lambda_i(x)$$

**Example 6**

Continuing with example 3, the two sensors are now distributed in the form of Figure 8. The two sensor likelihood matrices $P_1(z_1 \mid x)$;

<table>
<thead>
<tr>
<th></th>
<th>$z_1$</th>
<th>$z_2$</th>
<th>$z_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.45</td>
<td>0.45</td>
<td>0.1</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0.45</td>
<td>0.45</td>
<td>0.1</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0.1</td>
<td>0.45</td>
<td>0.8</td>
</tr>
</tbody>
</table>

and $P_2(z_2 \mid x)$;

<table>
<thead>
<tr>
<th></th>
<th>$z_1$</th>
<th>$z_2$</th>
<th>$z_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.45</td>
<td>0.1</td>
<td>0.45</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0.1</td>
<td>0.45</td>
<td>0.45</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0.45</td>
<td>0.45</td>
<td>0.1</td>
</tr>
</tbody>
</table>

respectively, are stored locally with each sensor. As observations occur, these likelihood functions are instantiated with observations to produce likelihood functions on $x$. For example, if $z_1 = z_1$ is observed with sensor 1, $z_2 = z_1$ is observed with sensor 2, the instantiated likelihood functions returned to the central processor are $\Lambda_1(x) = (0.45, 0.45, 0.1)$ and $\Lambda_2(x) = (0.45, 0.1, 0.45)$ respectively. Note again, these are probability densities defined on the state. The communicated likelihoods are fused in the central processor by simple multiplication with the prior. On successive observations, likelihoods continue to be communicated by the local sensors.

A second approach to distributed data fusion using Bayes Theorem is shown in Figure 9. In this case, each sensor computes a likelihood but then combines this, locally, with
Figure 9: A distributed implementation of the independent opinion pool in which each sensor maintains both its 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.
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.

Example 7

Consider the same two sensors as example 6 now distributed as Figure 9. Assuming an existing prior, \( P(x) = (1/3, 1/3, 1/3) \), communicated to the two sensors. After observation of, say, \( z_1 = z_1 \) by sensor 1 and \( z_2 = z_1 \) by sensor 2, local likelihoods are instantiated but then multiplied by the prior to yield local posteriors \( P_1(x | z_1) = (0.45, 0.45, 0.1) \) and \( P_2(x | z_2) = (0.45, 0.1, 0.45) \) respectively. These posteriors are communicated to the central fusion site. At the fusion center, care must be taken to avoid "double counting" the common prior information communicated in the posteriors from each sensor site. This is achieved by dividing the posterior by the prior before fusion as

\[
P_{12}(x | z_1, z_2) = P(x) \left[ \frac{P_1(x | z_1)}{P(x)} \right] \left[ \frac{P_2(x | z_2)}{P(x)} \right] = (0.692, 0.154, 0.154)
\]

where the division is taken element wise. In this form of distributed architecture, the combined posterior is now communicated back to the local sensor sites. This information can be used locally at the sensor site. If a second observation is made locally at the site, the local likelihood is combined with this global posterior to yield a new local posterior. For example, if sensor 2 now detects a target type 1, then the local posterior is computed from

\[
P_{12}(x | z_1, z_2z_2) = (0.108, 0.024, 0.868) = (0.692, 0.154, 0.154) \otimes (0.45, 0.1, 0.45).
\]

This new posterior is then communicated to the central fusion site and the process repeats.

Almost any data fusion structure is possible as long as the essential rules of conditional probability and Bayes theorem are followed. Figure 10 shows a Bayes filter in feedback form. The filter outputs state predictions in the form of a pdf \( P(x_k^- | Z^{k-1}) \) of the state \( x_k^- \) at time \( k \) given all observations \( Z^{k-1} \) up to time \( k - 1 \) (typically using Equation 37). 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 \( \Lambda(x) \) associated with an incoming observation. This multiplication essentially implements Bayes
Figure 10: A Bayesian formulation of the classical feedback aided navigation system, fusing rate data with external observation information (see text for details).

Theorem to produce the posterior $P(x^+_k \mid Z^k)$, where the superscript ‘+’ denotes ‘after observation’. The likelihood itself is generated from the sensor model $P(z \mid x)$ in the normal manner. The posterior is fed-back and multiplied by another distribution $P(x^- \mid x^+)$ 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 feedback aided navigation system. The state prediction distribution $P(x^- \mid x^+)$ 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 $P(z \mid x)$. 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:

$$l(x) \triangleq \log P(x), \quad l(x \mid y) \triangleq \log P(x \mid y).$$  \hspace{1cm} (39)

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 $x$; $l(x) \leq 0$. 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

$$l(x \mid z) = l(z \mid x) + l(x) - l(z).$$  \hspace{1cm} (40)

**Example 8**
Consider again the two-sensor discrete target identification example (Example 3). The log-likelihood matrix for the first sensor (using natural logs) is

\[
\begin{array}{ccc}
z_1 & z_2 & z_3 \\
 x_1 & -0.799 & -2.303 & -0.799 \\
x_2 & -2.303 & -0.799 & -0.799 \\
x_3 & -0.799 & -0.799 & -2.303 \\
\end{array}
\]

and for the second

\[
\begin{array}{ccc}
z_1 & z_2 & z_3 \\
 x_1 & -0.799 & -2.303 & -0.799 \\
x_2 & -2.303 & -0.799 & -0.799 \\
x_3 & -0.799 & -0.799 & -2.303 \\
\end{array}
\]

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

\[
l(x \mid z_1, z_1) = l_1(z_1 \mid x) + l_2(z_1 \mid x) + C
= (-0.7985, -0.7985, -2.3026) + (-0.7985, -2.3026, -0.7985) + C
= (-1.5970, -3.1011, -3.1011) + C
= (-0.3680, -1.8721, -1.8721)
\]

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).

\[\text{Example 9}\]

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

\[
l(x \mid Z^k) = -\frac{1}{2} \frac{(x_k - x)^2}{\sigma_k^2}
= -\frac{1}{2} \frac{(z_k - x)^2}{\sigma^2} - \frac{1}{2} \frac{(x_{k-1} - x)^2}{\sigma_{k-1}^2} + C. \quad (41)
\]

as before, completing squares gives

\[
x_k = \frac{\sigma_{k-1}^2}{\sigma_{k-1}^2 + \sigma^2} z_k + \frac{\sigma^2}{\sigma_{k-1}^2 + \sigma^2} x_{k-1},
\]
and
\[
\sigma_k^2 = \frac{\sigma_k^2 \sigma_{k-1}^2}{\sigma_k^2 + \sigma_{k-1}^2},
\]
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 11: A log-likelihood implementation of a fully centralised data fusion architecture.

Figure 12: A log-likelihood implementation of the independent likelihood pool architecture.

Log-likelihoods are a convenient way of implementing distributed data fusion architectures. Figure 11 shows a log-likelihood implementation of a fully centralised data fusion architecture (equivalent to Figure 4) 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 12 shows a log-likelihood implementation of the independent likelihood architecture (equivalent to Figure 8). 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 13 shows a log-likelihood implementation of the independent opinion pool (equivalent to Figure 9) in which each local sensor maintains a posterior likelihood which is communicated 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 its 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\(^4\) \(H_P(x)\) associated with a probability distribution \(P(x)\), defined on a random variable \(x\), is defined as the expected value of minus the

\[^4\]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 maximization.
log-likelihood. For continuous-valued random variables this is given by (see [46] Chapter 15)
\[
H_P(x) \triangleq - E\{\log P(x)\} = - \int_{-\infty}^{\infty} P(x) \log P(x) dx
\]
(42)
and for discrete random variables
\[
H_P(x) \triangleq - E\{\log P(x)\} = - \sum_{x \in \mathcal{X}} P(x) \log P(x).
\]
(43)

Note that following convention, we have used \(x\) as an argument for \(H_P(\cdot)\) even though the integral or sum is taken over values of \(x\) so \(H_P(\cdot)\) is not strictly a function of \(x\) 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 \(x\); 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 \(x\) can take on at most \(n\) discrete values in the set \(\mathcal{X}\), then the least informative (maximum entropy) distribution on \(x\) is one which assigns a uniform probability \(1/n\) to each value. This distribution will clearly have an entropy of \(\log n\). When \(x\) is continuous-valued, the least informative distribution is also uniform. However, if the range of \(x\) 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 \(P(x)\) can be set to any convenient constant value over the whole range of \(x\):
\[
P(x) = 1, \forall x,
\]
for example, without affecting the values computed for the posterior \(P(x | z)\).

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 ‘informativeness’. An excellent proof of this quite remarkable result (first shown by Shannon) can be found in [15]. 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
\[
H_P(x | z_j) \triangleq - E\{\log P(x | z_j)\} = - \int_{-\infty}^{\infty} P(x | z_j) \log P(x | z_j) dx
\]
(44)
\footnote{A distribution such as this which clearly violates the constraint that \(\int_{-\infty}^{+\infty} P(x) dx = 1\) is termed an \textit{improper} distribution, or in this case an improper prior.}

\(H_P(x | z_j)\) measures the compactness of a conditional density \(P(x | z_j)\). It achieves a minimum of zero when all probability mass is assigned to a single value of \(x\); this agrees with an intuitive notion of the ‘most informative’ conditional distribution. Conversely, when probability mass is uniformly distributed over states, the conditional entropy in the distribution is a maximum; this too agrees with the idea of least informative (maximum entropy) conditional distributions. Maximum entropy conditional distributions are often used as prior distributions when no useful prior information is available. For example, if the random variable \(x\) can take on at most \(n\) discrete values in the set \(\mathcal{X}\), then the least informative (maximum entropy) distribution on \(x\) is one which assigns a uniform probability \(1/n\) to each value. This distribution will clearly have an entropy of \(\log n\). When \(x\) is continuous-valued, the least informative conditional distribution is also uniform. However, if the range of \(x\) 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 \(P(x)\) can be set to any convenient constant value over the whole range of \(x\):
\[
P(x) = 1, \forall x,
\]
for example, without affecting the values computed for the posterior \(P(x | z)\).

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 ‘informativeness’. An excellent proof of this quite remarkable result (first shown by Shannon) can be found in [15]. 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.
and for discrete random variables

\[ H_P(x \mid z_j) \triangleq -\mathbb{E}\{\log P(x \mid z_j)\} = -\sum_x P(x \mid z_j) \log P(x \mid z_j). \]  \hspace{1cm} (45)

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

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

\[
\overline{H}(x \mid z) \triangleq \mathbb{E}\{H(x \mid z)\} = \int_{-\infty}^{+\infty} P(z)H(x \mid z)dz = -\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} P(z)P(x \mid z) \log P(x \mid z) dx dz = -\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} P(x, z) \log P(x \mid z) dx dz, \]  \hspace{1cm} (46)

for continuous random variables and

\[
\overline{H}(x \mid z) \triangleq \sum_z H(x \mid z)P(z) = -\sum_z \sum_x P(x, z) \log P(x \mid z). \]  \hspace{1cm} (47)

for discrete random variables. Note that \( \overline{H}(x \mid z) \) is not a function of either \( x \) or \( z \). It is 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 10**

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 \( z_1 \) has been made (first column of the likelihood matrix for sensor 1), is simply given by (using natural logs)

\[
H_{P_1}(x \mid z_1) = -\sum_{i=1}^3 P_1(x_i \mid z_1) \log P_1(x_i \mid z_1) = 0.9489. \]  \hspace{1cm} (48)

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

\[
H_{P_1}(x \mid z) = -\sum_{i=1}^3 P_1(x_i \mid z) \log P_1(x_i \mid z) = (0.9489, 0.9489, 0.6390). \]  \hspace{1cm} (49)

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 den-
sity on \( x \) 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
\[
H_{P_2}(x | z) = (0.948, 0.948, 0.948);
\]
that is any observation is equally
as informative. This is because, in the likelihood function for sensor 2, the relative distri-
bution of probability mass is the same for any observation, even though the mass itself is
placed on different states. Successive observation of any of \( z_1 \) or \( z_2 \), or \( z_3 \) results in probabil-
ity being assigned to only two of three states and thus for the posterior entropy to achieve
a minimum of \( \log(2) = 0.6931 \). 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
\[
P(x, z) = P(z | x)P(x)
\]
\[
= \begin{bmatrix} 0.45 & 0.45 & 0.1 \\ 0.45 & 0.45 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \otimes \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}
\]
\[
= \begin{bmatrix} 0.1500 & 0.1500 & 0.0333 \\ 0.1500 & 0.1500 & 0.0333 \\ 0.0333 & 0.0333 & 0.2667 \end{bmatrix},
\]
(Note the sum of elements is equal to 1). Substituting these values into Equation 47 and
summing over both \( x \) and \( z \) gives the mean conditional entropy as 0.8456.

**Example 11**

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:
\[
P(x) = N(\bar{x}, \mathbf{P}) = |2\pi \mathbf{P}|^{-1/2} \exp \left[ -\frac{1}{2} (x - \bar{x})^T \mathbf{P}^{-1} (x - \bar{x}) \right],
\]
where \( \bar{x} \) is the mean of the distribution and \( \mathbf{P} \) the covariance, and where \(| \cdot |\) denotes
matrix determinant. The entropy for this distribution is obtained as follows
\[
H_P(x) = -E\{\log P(x)\}
\]
\[
= \frac{1}{2} E\{(x - \bar{x})^T \mathbf{P}^{-1} (x - \bar{x}) + \log[(2\pi)^n | \mathbf{P} |)\}
\]
\[
= \frac{1}{2} \sum_{ij} E\{(x_i - \bar{x}_i)(x_j - \bar{x}_j)\} \mathbf{P}^{-1}_{ij} - \frac{1}{2} \log[(2\pi)^n | \mathbf{P} |]
\]
\[
= \frac{1}{2} \sum_{ij} E\{(x_j - \bar{x}_j)(x_i - \bar{x}_i)\} \mathbf{P}^{-1}_{ij} - \frac{1}{2} \log[(2\pi)^n | \mathbf{P} |]
\]
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 $\sigma^2$, then the entropy is simply given by

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

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

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

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

### 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 40 with respect to both the state $x$ and the observation $z$ gives (we will now drop the suffix $P$ when the context of the distribution is obvious)

$$\bar{H}(x \mid z) = \bar{H}(z \mid x) + H(x) - H(z).$$  

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

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 $I(x,z)$ obtained about a random variable $x$ with respect to a second random variable $z$ is now defined as

$$I(x,z) = -E\{\log \frac{P(x,z)}{P(x)P(z)}\}$$

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 $P(x | z)$ following an observation to the prior density $P(x)$. The expectation is taken over $z$ and $x$, 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

$$I(x,z) = -\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} P(x,z) \log \frac{P(x,z)}{P(x)P(z)} dx dz,$$

and for discrete distributions

$$I(x,z) = \sum_{z \in Z} \sum_{x \in X} P(x,z) \log \frac{P(x,z)}{P(x)P(z)}.$$  

Equation 53 can be written in terms of the component entropies as

$$I(x,z) = H(x) + H(z) - H(x,z)$$

Equation 57 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 $x$ if the value of $z$ where known. Most importantly mutual information provides a pre-experimental measure of the usefulness of obtaining information (through observation) about the value of $z$.

**Example 12**

Continuing the two sensors target identification example (Example 10), 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 $x$ is uniform so that

$$P(x) = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}^T.$$

Consider taking an observation with sensor 1. The total probability of observation $P(z)$ can be found by summing the joint probability $P(x, z)$ (computed in Example 10) to obtain

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

$$I(x, z) = \sum_j P(z_j) \log P(z_j) - \sum_i \sum_j P(x_i, z_j) \log P(z_j | x_i)$$

$$= 1.0986 - 0.8456 = 0.2530.$$

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)

$$P(x) = P_1(x \mid z_1) = \begin{bmatrix} 0.45 \\ 0.1 \\ 0.45 \end{bmatrix}.$$ 

The mutual information gain from using sensor 1 given this new prior is $I(x, z) = 0.1133$. 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.

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 13**

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

\[ H(x, y) = -\mathbb{E}\{\log P(x, y)\} \]

\[ = \int \int P(x, y) \log P(x, y) dx dy \]

\[ \approx \sum \sum P(x_i, y_j) \log P(x_i, y_j) \delta x_i \delta y_j \quad (58) \]

The total entropy associated with the prior shown in Figure 5(a), computed with Equation 58 is \(H(x) = 12.80\). The total entropy associated with the sensor 1 likelihood shown in Figure 5(b) is \(\overline{H}(z \mid x) = 4.2578\), and the entropy associated with the resulting posterior of Figure 5(c) is \(\overline{H}(x \mid z) = 3.8961\). The predicted mutual information gain from this observation is therefore \(I(x, z) = 8.9039\) (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 5(c) to Figure 5(d). Conversely, the predicted
mutual information gain from sensor 2, with likelihood shown in Figure 5(e), is 2.8598 (a relatively large gain). This is reflected in the posterior shown in Figure 5(f) which has an entropy of only 0.5138.

This example shows that it is straightforward 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 $J(x)$ is defined as the second derivative of the log-likelihood:

$$J(x) = \frac{d^2}{dx^2} \log P(x).$$  

(59)

In general, if $x$ is a vector, then $J(x)$ will be a matrix, usually called the Fisher Information Matrix. The Fisher information describes the information content about the values of $x$ contained in the distribution $P(x)$. 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 $P(x)$ describes all the (probabilistic) information we have about the quantity $x$, then the smallest variance that we can have on an estimate of the true value of $x$ is known as the Cramer-Rao lower bound, and is equal to the Fisher information $J(x)$. An estimator that achieves this lower bound is termed ‘efficient’.  

**Example 14**

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

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

(60)

Taking logs of this distribution, and differentiating twice with respect to $x$ gives $J(x) = P^{-1}$. 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. $P^{-1}$ describes the surface of this ellipsoid, the square root of it’s eigenvalues being the dimensions of each axis of the ellipsoid.

---

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

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 Equipartition Property (AEP) is given in Cover [17].

Example 15

If the pdf associated with a random variable $x$ is known to be Gaussian distributed with mean $\mu$ and covariance $\Sigma$ (as Equation 60), then an explicit relation for the relation between Fisher and Shannon information can be obtained, and indeed is given in Equation 51 (Example 11) as:

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

This clearly shows the relation between the Fisher surface measure of information $\Sigma^{-1}$ and the entropic volume measure through the determinant of $\Sigma$. 
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 [20] or the numerous excellent books on Kalman filtering [12, 5, 37]). 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 $x(t)$ evolves over time and an explicit statistical model of how the observations $z(t)$ 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 $\hat{x}(t)$ minimises mean-squared error

$$L(t) = \int_{-\infty}^{\infty} (x(t) - \hat{x}(t))^T (x(t) - \hat{x}(t)) P(x(t) \mid Z^t) dx.$$

Differentiation of Equation 61 with respect to $x(t)$ and setting equal to zero gives

$$\hat{x}(t) = \int_{-\infty}^{\infty} x(t) P(x(t) \mid Z^t) dx,$$

which is simply the conditional mean $\hat{x}(t) = E\{x(t) \mid Z^t\}$. 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 [53, 3, 5, 37].

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;

$$\dot{x}(t) = F(t)x(t) + B(t)u(t) + G(t)v(t), \quad (63)$$

where

- $x(t) \in \mathbb{R}^n$ is the state vector of interest,
- $u(t) \in \mathbb{R}^s$ is a known control input,
- $v(t) \in \mathbb{R}^q$ is a random variable describing uncertainty in the evolution of the state,
- $F(t)$ is the $n \times n$ state (model) matrix,
- $B(t)$ is the $n \times s$ input matrix, and
- $G(t)$ is the $n \times q$ noise matrix.

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

$$z(t) = H(t)x(t) + D(t)w(t), \quad (64)$$

where

- $z(t) \in \mathbb{R}^m$ is the observation vector,
- $w(t) \in \mathbb{R}^r$ is a random variable describing uncertainty in the observation,
- $H(t)$ is the $m \times n$ observation (model) matrix,
- $D(t)$ 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 63 and 64.
First, a discrete-time set \( t = \{ t_0, t_1, \ldots, t_k, \ldots \} \) is defined. Equation 64 can be written in discrete time as
\[
z(t_k) = H(t_k)x(t_k) + D(t_k)w(t_k), \quad \forall t_k \in t
\] (65)
where \( z(t_k), x(t_k) \) and \( w(t_k) \) are the discrete-time observation, state and noise vectors respectively, and \( H(t_k) \) and \( D(t_k) \) 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 63 over the interval \((t_k, t_{k-1})\) as
\[
x(t_k) = \Phi(t_k, t_{k-1})x(t_{k-1}) + \int_{t_{k-1}}^{t_k} \Phi(t_k, \tau)B(\tau)u(\tau)\,d\tau + \int_{t_{k-1}}^{t_k} \Phi(t_k, \tau)G(\tau)v(\tau)\,d\tau.
\] (66)
where \( \Phi(\cdot, \cdot) \) is the state transition matrix satisfying the matrix differential equation
\[
\dot{\Phi}(t_k, t_{k-1}) = F(t_k)\Phi(t_k, t_{k-1}), \quad \Phi(t_{k-1}, t_{k-1}) = 1.
\] (67)
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) \( \Phi(t_3, t_1) = \Phi(t_3, t_2)\Phi(t_2, t_1) \).
3. \( \Phi(t_k, t_{k-1}) \) is non singular and \( \Phi^{-1}(t_k, t_{k-1}) = \Phi(t_{k-1}, t_k) \).

When \( F(t) = F \) is a constant matrix, the state transition matrix is given by
\[
\Phi(t_k, t_{k-1}) = \Phi(t_k - t_{k-1}) = \exp F(t_k - t_{k-1}).
\] (68)
which clearly satisfies these three properties.

If \( u(t) = u(t_k) \) and \( v(t) = v(t_k) \) remain approximately constant over the interval \((t_{k-1}, t_k)\) then the following discrete-time models can be defined:
\[
F(t_k) \triangleq \Phi(t_k, t_{k-1})
\]
\[
B(t_k) \triangleq \int_{t_{k-1}}^{t_k} \Phi(t_k, \tau)B(\tau)\,d\tau
\] (69)
\[
G(t_k) \triangleq \int_{t_{k-1}}^{t_k} \Phi(t_k, \tau)G(\tau)\,d\tau.
\]

Equation 66 can then be written in a discrete-time form equivalent to Equation 63 as
\[
x(t_k) = F(t_k)x(t_{k-1}) + B(t_k)u(t_k) + G(t_k)v(t_k).
\] (70)
The accuracy of this model could be improved by taking mean values for both \( u(t) \) and \( v(t) \) over the sampling interval.
In almost all cases the time interval $\Delta t(k) \triangleq t_k - t_{k-1}$ 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 70 is written as

$$x(k) = F(k)x(k-1) + B(k)u(k) + G(k)v(k),$$ (71)

and Equation 65 as

$$z(k) = H(k)x(k) + D(k)w(k)$$ (72)

Equations 71 and 72 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 $v(k)$ and $w(k)$ describing process and observation noise are all Gaussian, temporally uncorrelated and zero-mean

$$E\{v(k)\} = E\{w(k)\} = 0, \quad \forall k,$$ (73)

with known covariance

$$E\{v(i)v^T(j)\} = \delta_{ij}Q(i), \quad E\{w(i)w^T(j)\} = \delta_{ij}R(i).$$ (74)

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

$$E\{v(i)w^T(j)\} = 0, \quad \forall i, j.$$ (75)

These are effectively equivalent to a Markov property requiring observations and successive states to be conditionally independent. If the sequences $v(k)$ and $w(k)$ are temporally correlated, a shaping filter can be used to ‘whiten’ the observations; again making the assumptions required for the Kalman filter valid [37]. If the process and observation noise sequences are correlated, then this correlation can also be accounted for in the Kalman filter algorithm [1]. 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 [55]. Problems in which the process and observation models are non-linear are dealt in Section 3.1.5

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 16**

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

$$
\begin{bmatrix}
\dot{x}(t) \\
\ddot{x}(t)
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
x(t) \\
\dot{x}(t)
\end{bmatrix} +
\begin{bmatrix}
0 \\
v(t)
\end{bmatrix}.
$$
In this case the state-transition matrix from Equation 68 over a time interval $\Delta T$ is given by

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

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

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

With the definitions given in Equation 69, 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

$$z_x = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + w(k), \quad E\{w^2(k)\} = \sigma_r^2$$
Figure 14 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:

$$ \begin{bmatrix} x(k) \\ \dot{x}(k) \\ y(k) \\ \dot{y}(k) \end{bmatrix} = \begin{bmatrix} 1 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(k-1) \\ \dot{x}(k-1) \\ y(k-1) \\ \dot{y}(k-1) \end{bmatrix} + \begin{bmatrix} \Delta T^2/2 & 0 \\ 0 & \Delta T^2/2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_x(k) \\ v_y(k) \end{bmatrix}, \quad (76) $$

and

$$ \begin{bmatrix} z_x \\ z_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \dot{x}(k) \\ y(k) \\ \dot{y}(k) \end{bmatrix} + \begin{bmatrix} w_x(k) \\ w_y(k) \end{bmatrix}, \quad (77) $$

with

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

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 15 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 15 (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 14. 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

$$ \hat{x}(i \mid j) = \arg \min_{\hat{x}(i \mid j) \in \mathbb{R}^n} \mathbb{E}\{(x(i) - \hat{x})(x(i) - \hat{x})^T \mid z(1), \ldots, z(j)\}. \quad (78) $$

As has been previously demonstrated (Equation 62) the estimate obtained is simply the expected value of the state at time $i$ conditioned on the observations up to time $j$. The
Figure 15: True target track for linear the uncoupled linear target model of Example 16
(a) $x - y$ track; (b) Detail of track and corresponding observations; (c) Deduced target velocity; (d) Deduced target heading.
The estimate is thus defined as the conditional mean
\[ \hat{x}(i | j) \triangleq E\{x(i) | z(1), \ldots, z(j)\} = E\{x(i) | Z^j\} . \] (79)

The estimate variance is defined as the mean squared error in this estimate
\[ P(i | j) \triangleq E\{(x(i) - \hat{x}(i | j))(x(i) - \hat{x}(i | j))^T | Z^j\} . \] (80)

The estimate of the state at a time \( k \) given all information up to time \( k \) will be written as \( \hat{x}(k | k) \). 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 \( \hat{x}(k | k - 1) \).

The Kalman filter algorithm is now stated without proof. Detailed derivations can be found in many books on the subject, [37, 5] for example (see also [20]). The state is assumed to evolve in time according to Equation 71. Observations of this state are made at regular time intervals according to Equation 72. The assumptions about the noise processes entering the system, as described by Equations 73, 74 and 75, are assumed true. It is also assumed that an estimate \( \hat{x}(k - 1 | k - 1) \) of the state \( x(k - 1) \) at time \( k - 1 \) is available, and that this estimate is equal to the conditional mean of the true state \( x(k - 1) \) conditioned on these observations. The conditional variance \( P(k - 1 | k - 1) \) in this estimate is also assumed known. The Kalman filter then proceeds recursively in two stages:

**Prediction:** A prediction \( \hat{x}(k | k - 1) \) of the state at time \( k \) and its covariance \( P(k | k - 1) \) is computed according to
\[
\hat{x}(k | k - 1) = F(k)\hat{x}(k - 1 | k - 1) + B(k)u(k) \tag{81}
\]
\[
P(k | k - 1) = F(k)P(k - 1 | k - 1)F^T(k) + G(k)Q(k)G^T(k). \tag{82}
\]

**Update:** At time \( k \) an observation \( z(k) \) is made and the updated estimate \( \hat{x}(k | k) \) of the state \( x(k) \), together with the updated estimate covariance \( P(k | k) \) is computed from the state prediction and observation according to
\[
\hat{x}(k | k) = \hat{x}(k | k - 1) + W(k)(z(k) - H(k)\hat{x}(k | k - 1)) \tag{83}
\]
\[
P(k | k) = (1 - W(k)H(k))P(k | k - 1)(1 - W(k)H(k))^T + W(k)R(k)W^T(k) \tag{84}
\]
where the gain matrix \( W(k) \) is given by
\[
W(k) = P(k | k - 1)H(k)[H(k)P(k | k - 1)H^T(k) + R(k)]^{-1} \tag{85}
\]

The Kalman filter is recursive or cyclic (see Figure 16). 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
observation model in generating an update. The update stage of the filter is clearly linear, with a weight $W(k)$ being associated with the observation $z(k)$ and a weight $1 - W(k)H(k)$ being associated with the prediction $\hat{x}(k | k - 1)$. The Kalman filter also provides a propagation equation for the covariance in the prediction and estimate.

**Example 17**

It is straight-forward to build a state-estimator for the target and observation model of example Example 16. The Kalman filter Equations 83–85 are implemented with $F(k)$, $Q(k)$ as defined in Equation 76 and with $H(k)$ and $R(k)$ as defined in Equation 77. Initial conditions for $\hat{x}(0 | 0)$ are determined from the first few observations and with $P(0 | 0) = 10Q(k)$ (see [20] for details). The results of the filter implementation are shown in Figure 17. 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.
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 72 conditioned on previous observations:

\[
\hat{z}(k | k - 1) \triangleq E\{z(k) | Z^{k-1}\} = E\{H(k)x(k) + W(k) | Z^{k-1}\} = H(k)\hat{x}(k | k - 1) \tag{86}
\]

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

\[
\nu(k) = z(k) - H(k)\hat{x}(k | k - 1) \tag{87}
\]

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,

\[
E\{\nu(k) | Z^{k-1}\} = 0, \quad E\{\nu(i)\nu^T(j)\} = S(i)\delta_{ij}, \tag{88}
\]

where

\[
S(k) = R(k) + H(k)P(k | k - 1)H(k) \tag{89}
\]

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:

\[
\hat{x}(k | k) = \hat{x}(k | k - 1) + W(k)\nu(k) \tag{90}
\]

\[
P(k | k) = P(k | k - 1) - W(k)S(k)W^T(k) \tag{91}
\]

and from Equation 85

\[
W(k) = P(k | k - 1)H(k)S^{-1}(k) \tag{92}
\]

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

Example 18

Continuing Example 17, the innovation and innovation covariance can be calculated from Equations 87 and 89. These are shown in Figure 18(a). The most important points to note are that the innovation sequence is zero mean and white, and that approximately
Figure 17: Estimated target track for the linear uncoupled target model of Examples 16 and 17 (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 18(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 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 70 and 65 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 \( \hat{x}(t_{k-1} | t_{k-1}) \) 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 65. The first step is simply to generate a prediction of the state at the time the observation was made according to

\[
\dot{x}(t_k | t_{k-1}) = F(t_k)\hat{x}(t_{k-1} | t_{k-1}) + B(t_k)u(t_k),
\]

(93)

Together with an associated prediction covariance

\[
P(t_k | t_{k-1}) = F(t_k)P(t_{k-1} | t_{k-1})F^T(t_k) + G(t_k)Q(t_k)G^T(t_k).
\]

(94)

An innovation and innovation covariance are then computed according to

\[
\nu(t_k) = z(t_k) - H(t_k)\hat{x}(t_k | t_{k-1}),
\]

(95)

\[
S(t_k) = H(t_k)P(t_k | t_{k-1})H^T(t_k) + R(t_k),
\]

(96)

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

\[
\dot{x}(t_k | t_k) = \dot{x}(t_k | t_{k-1}) + W(t_k)\nu(t_k),
\]

(97)
Figure 18: Track Errors for the linear uncoupled target model of Examples 16 and 17: (a) Innovation and innovation variance in $x$ estimate; (b) Estimated error (standard deviation) in $x$ estimate and $x$ prediction.
\[
\mathbf{P}(t_k \mid t_k) = \mathbf{P}(t_k \mid t_{k-1}) - \mathbf{W}(t_k)\mathbf{S}(t_k)\mathbf{W}^T(t_k),
\]

(98)

where
\[
\mathbf{W}(t_k) = \mathbf{P}(t_k \mid t_{k-1})\mathbf{H}^T(t_k)\mathbf{S}(t_k)
\]

(99)
is the associated gain matrix. The main point to note here is that the injected process noise variance \(Q(t_k)\) 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 19**

Consider again the example of the constant-velocity particle motion described by the continuous-time process model
\[
\begin{bmatrix}
\dot{x}(t) \\
\ddot{x}(t)
\end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \delta t \end{bmatrix} v(t),
\]

We assume observations of the position of the particle are made asynchronously at times \(t_k\) according to
\[
z_x(t_k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x(t_k) \\ \dot{x}(t_k) \end{bmatrix} + w(t_k), \quad \mathbb{E}\{w^2(t_k)\} = \sigma_r^2.
\]

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 69, the discrete-time process model becomes
\[
\begin{bmatrix} x(t_k) \\ \dot{x}(t_k) \end{bmatrix} = \begin{bmatrix} 1 & \delta t_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(t_{k-1}) \\ \dot{x}(t_{k-1}) \end{bmatrix} + \begin{bmatrix} \delta t^2/2 \\ \delta t \end{bmatrix} v(t_k),
\]

with
\[
Q(t_k) = \mathbb{E}[v(t_k)v^T(t_k)] = \int_0^{\delta t} \left[ \begin{array}{c} \delta t \\ 1 \end{array} \right] q d\tau = \begin{bmatrix} \delta t^3/3 & \delta t^2/2 \\ \delta t^2/2 & \delta t \end{bmatrix} q.
\]

The implementation of an asynchronous filter looks similar to the implementation of a synchronous filter except that the matrices defined in Equation 69 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 19 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 19(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 19(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.
Figure 19: 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).
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 $\hat{x}(t_{k-1} | t_{k-1})$ and an estimate (strictly a prediction) $\hat{x}(t_p | t_{k-1})$ and that we acquire an observation $z(t_k)$ taken at time $t_k$. We start by simply discarding the estimate $\hat{x}(t_p | t_{k-1})$, and generating a new prediction $\hat{x}(t_k | t_{k-1})$ and prediction covariance $P(t_k | t_{k-1})$ from Equations 93 and 94. These together with the delayed observation are used to compute a new estimate $\hat{x}(t_k | t_k)$, at the time the delayed observation was made, from Equations 97, 98, and 99. This estimate is then predicted forward to the current time to produce an estimate $\hat{x}(t_c | t_k)$ and its associated covariance $P(t_c | t_k)$ according to

$$\hat{x}(t_c | t_k) = F(t_p)\hat{x}(t_k | t_k) + B(t_p)u(t_p),$$  \hspace{1cm} (100)$$

$$P(t_c | t_k) = F(t_p)P(t_k | t_k)F^T(t_p) + G(t_p)Q(t_p)G^T(t_p).$$  \hspace{1cm} (101)$$

Both estimates $\hat{x}(t_k | t_k)$ and $\hat{x}(t_p | t_k)$ 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.
3.1.5 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

\[ \dot{x}(t) = f[x(t), u(t), v(t), t], \]  

where

- \( x(t) \in \mathbb{R}^n \) is the state of interest,
- \( u(t) \in \mathbb{R}^r \) is a known control input,
- \( f[\cdot, \cdot, \cdot] \) a mapping of state and control input to state ‘velocity’, and
- \( v(t) \) a random vector describing both dynamic driving noise and uncertainties in the state model itself (\( v(t) \) 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

\[ z(t) = h[x(t), u(t), w(t), t] \]

where

- \( z(t) \in \mathbb{R}^m \) is the observation made at time \( t \),
- \( h[\cdot, \cdot, \cdot] \) is a mapping of state and control input to observations, and
- \( w(t) \) a random vector describing both measurement corruption noise and uncertainties in the measurement model itself (\( w(t) \) 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 102 and 103 are required. First, a discrete-time set \( t = \{t_0, t_1, \cdots, t_k, \cdots\} \) is defined. Equation 103 can then be written in discrete time as

\[ z(t_k) = h[x(t_k), u(t_k), w(t_k), t_k], \quad \forall t_k \in t \]

where \( z(t_k), x(t_k) \) and \( w(t_k) \) 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 102 over the interval \( (t_k, t_{k-1}) \) as

\[ x(t_k) = x(t_{k-1}) + \int_{t_{k-1}}^{t_k} f[x(\tau), u(\tau), v(\tau), \tau]d\tau. \]
In practice, this integration is normally solved using a simple Euler (backward difference) approximation as

\[ x(t_k) = x(t_{k-1}) + \Delta T_k f[x(t_{k-1}), u(t_{k-1}), v(t_{k-1}), t_{k-1}], \quad (106) \]

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 106 and 104 are written as

\[ x(k) = x(k-1) + \Delta T f[x(k-1), u(k-1), v(k-1), k], \quad (107) \]

and

\[ z(k) = h[x(k), u(k), w(k), k]. \quad (108) \]

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 102, a nominal state trajectory is described using the same process model (assuming \( v(t) \) is zero mean),

\[ \dot{x}_n(t) = f[x_n(t), u_n(t), t]. \quad (109) \]

Then, Equation 102 is expanded about this nominal trajectory as a Taylor series;

\[
\begin{align*}
\dot{x}(t) &= f[x_n(t), u_n(t), t] \\
&+ \nabla f_x(t) \delta x(t) + O\left[ (\delta x(t))^2 \right] \\
&+ \nabla f_u(t) \delta u(t) + O\left[ (\delta u(t))^2 \right] \\
&+ \nabla f_v(t) v(t),
\end{align*}
\]

(110)

where

\[
\begin{align*}
\nabla f_x(t) \triangleq \frac{\partial f}{\partial x} \bigg|_{x=x_n(t), u=u_n(t)} \\
\nabla f_u(t) \triangleq \frac{\partial f}{\partial u} \bigg|_{x=x_n(t), u=u_n(t)} \\
\nabla f_v(t) \triangleq \frac{\partial f}{\partial v} \bigg|_{x=x_n(t), u=u_n(t)}
\end{align*}
\]

(111)

and

\[
\delta x(t) \triangleq (x(t) - x_n(t)), \quad \delta u(t) \triangleq (u(t) - u_n(t)).
\]

(112)

Subtracting Equation 109 from Equation 110 provides a linear error model in the form

\[ \delta \dot{x}(t) = \nabla f_x(t) \delta x(t) + \nabla f_u(t) \delta u(t) + \nabla f_v(t) v(t). \]

(113)

Identifying \( F(t) = \nabla f_x(t), B(t) = \nabla f_u(t), \) and \( G(t) = \nabla f_v(t) \), Equation 113, is now in the same form as Equation 63 and may be solved for the perturbed state vector \( \delta x(t) \) in closed form through Equation 66, 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 \( f(\cdot) \) 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 110 and so improve the validity of the approximation.

Similar arguments can be applied to linearise Equation 64 to provide an observation error equation in the form

$$\delta \dot{z}(t) = \nabla h_x(t) \delta x(t) + \nabla f_{w}(t) \delta w(t).$$

(114)

Identifying $H(t) = \nabla f_{x}(t)$, and $D(t) = \nabla f_{w}(t)$, Equation 114, is now in the same form as Equation 64.

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

$$x(k) = f(x(k-1), u(k), v(k), k),$$

(115)

and the observation model as

$$z(k) = h(x(k), w(k)).$$

(116)

Like the Kalman filter, it is assumed that the noises $v(k)$ and $w(k)$ are all Gaussian, temporally uncorrelated and zero-mean with known variance as defined in Equations 73–75. 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,

$$\hat{x}(k - 1 | k - 1) \approx E\{x(k - 1) | Z^{k-1}\}.$$

(117)

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 $\hat{x}(k | k - 1)$ of the state at time $k$ and its covariance $P(k | k - 1)$ is computed according to

$$\hat{x}(k | k - 1) = f(\hat{x}(k - 1 | k - 1), u(k))$$

(118)

$$P(k | k - 1) = \nabla f_{x}(k) P(k - 1 | k - 1) \nabla^{T} f_{x}(k) + \nabla f_{v}(k) Q(k) \nabla^{T} f_{v}(k)$$

(119)

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

$$\hat{x}(k | k) = \hat{x}(k | k - 1) + W(k)[z(k) - h(\hat{x}(k | k - 1))]$$

(120)

$$P(k | k) = P(k | k - 1) - W(k) S(k) W^{T}(k)$$

(121)
where
\[ W(k) = P(k | k - 1)\nabla^T h_x(k)S^{-1}(k) \] (122)
and
\[ S(k) = \nabla h_x(k)P(k | k - 1)\nabla^T h_x(k) + \nabla h_w(k)R(k)\nabla^T h_w(k). \] (123)
and where the Jacobian \( \nabla f \cdot (k) \) is evaluated at \( x(k - 1) = \hat{x}(k - 1 | k - 1) \) and \( \nabla h \cdot (k) \) is evaluated at and \( x(k) = \hat{x}(k | k - 1) \).

A comparison of Equations 81–92 with Equations 118–123 makes it clear that the extended Kalman filter algorithm is very similar to the linear Kalman filter algorithm, with the substitutions \( F(k) \rightarrow \nabla f_x(k) \) and \( H(k) \rightarrow \nabla h_x(k) \) 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 72.

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

- The Jacobians \( \nabla f_x(k) \) and \( \nabla h_x(k) \) 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.

### 3.1.6 The Covariance Intersect (CI) Algorithm

The Covariance Intersection (CI) algorithm provides a solution to the problem of combining two random vectors in the case where the correlation between these vectors is unknown [44]. The algorithm is based on a geometric interpretation of the normal covariance or information matrices involved in filtering problems. Figure 20 demonstrates the principle. Consider two estimates \( a \) and \( b \) with covariances \( P_a \) and \( P_b \) respectively. For simplicity and future compatability, denote \( Y_a = P_a^{-1} \) and \( Y_a = P_a^{-1} \). Figure 20(a) shows
Figure 20: Operation of the covariance intersection (CI) algorithm: (a) The initial two estimate covariances and the updated covariance computed by the Kalman filter algorithm; (b) The possible updated covariances computed from a convex combination of the two initial covariances parameterized by the scalar $\omega$; (c) The updated covariance as the convex combination of the two initial covariances which maximises information (minimises enclosed volume); (d) The case where one initial covariance dominates the other.
two (equiprobable) ellipsoids generated from these covariance matrices together with the
combined estimate covariance ellipse calculated by the Kalman filter as:

\[ P_c^{-1} = P_a^{-1} + P_b^{-1} \]

or equivalently

\[ Y_c = Y_a + Y_b. \]

The assumption in the Kalman filter is that the two estimates are independent. If this is
not the case then the updated covariance computed by the Kalman filter will be overly
optimistic and the filter will become inconsistent.

Instead, the CI algorithm computes an updated covariance matrix as a convex com-
bination of the two initial covariance matrices in the form

\[ Y_c = (1 - \omega)Y_a + \omega Y_b \]  \hspace{1cm} (124)

\[ y_c = (1 - \omega)y_a + \omega y_b. \]  \hspace{1cm} (125)

Figure 20(b) shows these different ellipses as \( \omega \) varies between 0 and 1. This family of
ellipses all enclose the intersection of the two initial covariance ellipsoids. Any of these
ellipses would provide a consistent estimate of the updated covariance. However, a value
of \( \omega \) is normally chosen which minimizes the volume of the updated covariance (this is
equivalent to minimizing the determinant of \( Y_c \)). This minimum is shown in Figure 20(c).

It is clear that the CI algorithm can be quite conservative. Indeed, in many cases, the CI
algorithm will set \( \omega \) to an extrema of 1 or 0 when one of the initial covariances dominates
the other. This results in no new information being fused as is shown in Figure 20(d).
This must be weighed against the fact that the CI algorithm will guarantee consistency
for cases when the correlation between two estimates is unknown.

### 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 20**

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 \(\kappa\) is a constant minimum instantaneous turn radius for the target. Then \(x(t) = [x(t), y(t), \phi(t)]^T\) is defined as the state of the target and \(u(t) = [V(t), \gamma(t)]^T\) 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

\[
\begin{bmatrix}
x(k) \\
y(k) \\
\phi(k)
\end{bmatrix} = \begin{bmatrix}
x(k-1) + \Delta TV(k) \cos(\phi(k-1) + \gamma(k)) \\
y(k-1) + \Delta TV(k) \sin(\phi(k-1) + \gamma(k)) \\
\Delta T \phi(k-1) + \frac{V(k)}{\kappa} \sin(\gamma(k))
\end{bmatrix}
\]
which is in the standard form \( x(k) = f(x(k-1), u(k)) \). For the purposes of simulation, the model is assumed driven by Brownian models for velocity and heading,

\[
\dot{V}(t) = V(t) + v_V(t), \quad \dot{\gamma}(t) = \gamma(t) + v_\gamma(t),
\]

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

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

The discrete-time model for this is simply

\[
V(k) = V(k-1) + \Delta T_k v_V(k), \quad \gamma(k) = \gamma(k-1) + \Delta T_k v_\gamma(k).
\]

A group of typical trajectories generated by this model is shown in Figure 21(a).

The targets are observed using sensors (tracking stations) \( i = 1, \ldots, N \) whose location and pointing angles \( T_i(t) = [X_i(t), Y_i(t), \psi_i(t)]^T \) 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

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

where the random vector \( w_{ij}(k) = [r^r_{ij}(k), r^\theta_{ij}(k)]^T \) 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

\[
E\{w_{ij}(k)w_{ij}^T(k)\} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}.
\]

A typical set of observations generated by this model for a track is shown in Figure 21(b).

### 3.2.1 Observation Models

Figure 22 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;

\[
x(k) = F(k)x(k-1) + G(k)u(k) + v(k),
\]

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

\[
E\{v(k)\} = 0, \quad \forall k,
\]

\[
E\{v(i)v^T(j)\} = \delta_{ij} Q(i).
\]
Figure 21: Typical tracks and observations generated by the ‘standard’ multi-target tracking example of Example 20: (a) true $x$–$y$ tracks; (b) detail of track observations.
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

\[ z_s(k) = H_s(k)x(k) + w_s(k), \quad s = 1, \cdots, S. \]  

(128)

where \( z_s(k) \) is the observation made at time \( k \) by sensor \( s \) of a common state \( x(k) \) according to the observation model \( H_s(k) \) in additive noise \( w_s(k) \). 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 \( H_s(t_k) \).

It is assumed that the observation noise models \( w_s(k) \) are all zero mean, uncorrelated between sensors and also temporally uncorrelated:

\[ \mathbb{E}\{w_s(k)\} = 0, \quad s = 1, \cdots, S, \quad \forall k \]

\[ \mathbb{E}\{w_p(i)w_q(j)\} = \delta_{ij}\delta_{pq}R_p(i). \]  

(129)
It is also assumed that the process and observation noises are uncorrelated

\[ E\{v(i)w_s^T(j)\} = 0, \quad \forall i, j, s. \]  

(130)

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 [28].)

### 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 into 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

\[ z(k) \triangleq [z_1^T(k), \cdots, z_s^T(k)]^T, \]  

(131)

and a composite observation model by

\[ H(k) \triangleq [H_1^T(k), \cdots, H_s^T(k)]^T, \]  

(132)

with

\[ w(k) \triangleq [w_1^T(k), \cdots, w_s^T(k)]^T, \]  

(133)

where from Equation 129

\[ R(k) = E\{w(k)w^T(k)\} \]

\[ = E\{[w_1^T(k), \cdots, w_s^T(k)]^T [w_1^T(k), \cdots, w_s^T(k)]\} \]

\[ = \text{blockdiag}\{R_1(k), \cdots, R_s(k)\}, \]  

(134)

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 128 may now be rewritten as a single ‘group’ observation model in standard form

\[ z(k) = H(k)x(k) + w(k). \]  

(135)

With a process model described by Equation 126 and a group observation model defined by Equation 135, estimates of state can in principle be computed using the standard Kalman filter algorithm given by Equations 90, 91, and 92.

**Example 21**

*Consider again the tracking of a particle moving with constant velocity with process model as defined in Example 16. 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

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

The composite group sensor model is simply given by

\[
\begin{bmatrix}
z_1 \\
z_2 \\
z_3
\end{bmatrix} =
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
x(k) \\
\dot{x}(k) \\
\dot{x}(k)
\end{bmatrix}
+ 
\begin{bmatrix}
w_1 \\
w_2 \\
w_3
\end{bmatrix}, \quad \text{E}\{
\begin{bmatrix}
w_1 \\
w_2 \\
w_3
\end{bmatrix}
\begin{bmatrix}
w_1 \\
w_2 \\
w_3
\end{bmatrix}\} =
\begin{bmatrix}
\sigma_{r_1}^2 & 0 & 0 \\
0 & \sigma_{r_2}^2 & 0 \\
0 & 0 & \sigma_{r_3}^2
\end{bmatrix}.
\]

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

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

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

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

\[
\text{E}\{
\begin{bmatrix}
w_1 \\
w_2 \\
w_3
\end{bmatrix}
\begin{bmatrix}
w_1 \\
w_2 \\
w_3
\end{bmatrix}\} =
\begin{bmatrix}
\sigma_{r_1}^2 & 0 & 0 \\
0 & \sigma_{r_2}^2 & 0 \\
0 & 0 & \sigma_{r_3}^2
\end{bmatrix}.
\]

It 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 \(x(k)\) of dimension \(n\) and \(S\) sensors each with an observation vector \(z_s(k), s = 1, \cdots, S\) of dimension \(m_s\) together with an observation model \(H_s(k)\) of dimension \(m_s \times n\) then the group observation vector \(z(k)\) will have dimension \(m = \sum_{s=1}^S m_s\) and the group observation model \(H(k)\) will have dimension \(m \times n\). The consequence of this lies in Equations 90, 91 and 92. Clearly the group-sensor innovation \(\nu(k) \triangleq [\nu_1^T(k), \cdots, \nu_m^T(k)]^T\) will now have dimension \(m\), and the group-sensor innovation covariance matrix \(S(k)\) will have dimension \(m \times m\). 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.
$W(k)$ in Equation 92. 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 22**

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 20. The range and bearing observation vector $z_{ij}(k) = [z^r_{ij}(k), z^\theta_{ij}(k)]^T$ 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

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

where $\text{Rot}(\theta)$ is the rotation matrix

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

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 16 with process model defined in Equation 76 and with an observation model (now not constant) defined in Equation 77, can be used in the filter defined by Equations 83–85 in Example 17.

Figure 23 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 23(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.
Figure 23: 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).
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 126. Observations are made of this common state by \( S \) sensors according to Equation 128. It 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 \( z_p(k) \), \( 1 \leq p \leq S \), from the \( p^{th} \) sensor will be before the observation \( z_{p+1}(k) \) 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)

\[
\mathcal{Z}_p(k) \triangleq \{z_1(k), \cdots, z_p(k)\},
\]

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

\[
\mathcal{Z}^k_p \triangleq \{z_p(1), \cdots, z_p(k)\},
\]

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

\[
\mathcal{Z}^k_p \triangleq \{Z_1^k, \cdots, Z_p^k\}, \quad p \leq S.
\]

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 \( j \)

\[
\mathcal{Z}_{p,q}^{i,j} = \{Z_p^i \cup Z_q^j\} = \{Z_1^i, \cdots, Z_p^i, Z_{p+1}^i, \cdots, Z_q^j\}, \quad p < q, i \geq j.
\]

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 \( k \)

\[
\mathcal{Z}_{p,S}^{k,k-1} = \{Z_p^k \cup Z_S^{k-1}\} = \{Z_1^k, \cdots, Z_p^k, Z_{p+1}^k, \cdots, Z_S^{k-1}\} = \{Z_p(k) \cup Z_S^{k-1}\}, \quad p < S.
\]
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

\[
\hat{x}(k \mid k, p) = E\{x(k) \mid Z_{p,S}^{k,k-1}\},
\]

where in particular

\[
\hat{x}(k \mid k, 0) = \hat{x}(k \mid k - 1) = E\{x(k) \mid Z_{S}^{k-1}\},
\]

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

\[
\hat{x}(k \mid k, S) = \hat{x}(k \mid k) = E\{x(k) \mid Z_{S}^{k}\},
\]

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

It is assumed that Equations 127, 129, and 130 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

\[
\hat{x}(k \mid k, 0) = \hat{x}(k \mid k - 1)
\]

\[
= F(k)\hat{x}(k - 1 \mid k - 1) + G(k)u(k)
\]

\[
= F(k)\hat{x}(k - 1 \mid k - 1, S) + G(k)u(k),
\]

with corresponding covariance

\[
P(k \mid k, 0) = P(k \mid k - 1)
\]

\[
= F(k)P(k - 1 \mid k - 1)F^T(k) + Q(k)
\]

\[
= F(k)P(k - 1 \mid k - 1, S)F^T(k) + Q(k).
\]

The new estimate found by integrating the observation \(z_1(k)\) made by the first sensor at time \(k\) is then simply given by

\[
\hat{x}(k \mid k, 1) = \hat{x}(k \mid k, 0) + W_1(k) [z_1(k) - H_1(k)\hat{x}(k \mid k, 0)]
\]

with covariance

\[
P(k \mid k, 1) = P(k \mid k, 0) - W_1(k)S_1(k)W_1^T(k),
\]

where

\[
W_1(k) = P(k \mid k, 0)H_1^T(k)S_1^{-1}(k)
\]

and

\[
S_1(k) = H_1(k)P(k \mid k, 0)H_1^T(k) + R_1(k)
\]

To now integrate the observation made by the second sensor we need to employ the estimate \(\hat{x}(k \mid k, 1)\) 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 \( F(k) = 1 \) and \( Q(k) = 0 \) (this is not true in the asynchronous case). This means that the estimate \( \hat{x}(k | k, 1) \) is itself the prediction for the next update. In general, the estimate of the state \( \hat{x}(k | k, p) \) 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 \( \hat{x}(k | k, p - 1) \) as

\[
\hat{x}(k | k, p) = \hat{x}(k | k, p - 1) + W_p(k) [z_p(k) - H_p(k)\hat{x}(k | k, p - 1)]
\]

with covariance

\[
P(k | k, p) = P(k | k, p - 1) - W_p(k)S_p(k)W_p^T(k),
\]

where

\[
W_p(k) = P(k | k, p - 1)H_p^T(k)S_p^{-1}(k)
\]

and

\[
S_p(k) = H_p(k)P(k | k, p - 1)H_p^T(k) + R_p(k)
\]

The state update at each time-step could be computed in a batch-mode by explicitly expanding Equation 150 to become

\[
\hat{x}(k | k) = \left[ \prod_{i=1}^{S_p} (1 - W_i(k)H_i(k)) \right] \hat{x}(k | k - 1)
\]

\[
+ \sum_{i=1}^{S_p} \left[ \prod_{j=i+1}^{S_p} (1 - W_j(k)H_j(k)) \right] W_i(k)z_i(k)
\]

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\(^7\). 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.

\(^7\)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.
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

$$\hat{x}(k \mid k) \neq \hat{x}(k \mid k - 1) + \sum_{i=1}^{S} W_i(k) \left[ z_i(k) - H_i(k) \hat{x}(k \mid k - 1) \right], \quad (155)$$

with

$$W_i(k) = P(k \mid k - 1)H_i^T(k)S^{-1}(k), \quad i = 1, \ldots, S,$$

and

$$S_i(k) = H_i(k)P(k \mid k - 1)H_i^T(k) + R_i(k), \quad i = 1, \ldots, S.$$  

If the equality in Equation 155 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 92 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

$$z(k) = [z_1^T(k), z_2^T(k)]^T,$$

doing the group-sensor observation matrix is

$$H(k) = [H_1^T(k), H_2^T(k)]^T,$$

and the group-sensor observation noise covariance matrix

$$R(k) = blockdiag\{R_1(k), R_2(k)\}.$$ 

The group sensor innovation covariance is thus

$$S(k) = H(k)P(k \mid k - 1)H^T(k) + R(k)$$

$$= \begin{bmatrix} H_1(k) & \cdots & H_2(k) \end{bmatrix} \begin{bmatrix} P(k \mid k - 1) & H_2^T(k) \end{bmatrix}^T + \begin{bmatrix} R_1(k) & 0 \\ 0 & R_2(k) \end{bmatrix} \quad (156)$$

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

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

will also not be block-diagonal.

Fundamentally, the reason why Equation 155 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\(^8\). This correlation is due to the fact that the

---

\(^8\)the innovations from one time-step to the next are uncorrelated but the innovations generated by many sensors at a single time are correlated.
innovations at a single time-step have a common prediction. This correlation is reflected in the off-diagonal terms of Equation 156 of the form $H_i(k)P(k | k - 1)H_j^T(k)$.

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 155 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.

**Example 23**

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 [30]

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

(157)

where

$$\Delta = S_{11} - S_{12}S_{22}^{-1}S_{12}^T.$$

Making appropriate substitutions from Equation 156, and employing the matrix inversion lemma twice, we have

$$\Delta^{-1} = \begin{bmatrix} \Delta^{-1} \end{bmatrix}$$

$$= \begin{bmatrix} H_1PH_1^T + R_1 - H_1PH_2^T[H_2PH_2^T + R_2]^{-1}H_2PH_1^T \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} R_1 + H_1\left(P - PH_2^T[H_2PH_2^T + R_2]^{-1}H_2P\right)H_1^T \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} R_1 + H_1\left[P^{-1} + H_2^TR_2^{-1}H_2^{-1}H_1^T \right]^{-1}$$

$$= \begin{bmatrix} R_1^{-1} - R_1^{-1}H_1\left[P^{-1} + H_1^TR_2^{-1}H_2^{-1}H_1^T \right]^{-1}H_1^TR_1^{-1}.$$  

The two gain matrices may now be computed from

$$[W_1 \ W_2] = P \begin{bmatrix} H_1^T \\ H_2^T \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} \\ S_{12}^T & S_{22} \end{bmatrix}^{-1}.$$
Substituting in from Equations 156 and 157 we have for the first gain

\[
W_1 = PH_1^T \Delta^{-1} - PH_2^T \left[ H_2PH_2^T + R_2 \right]^{-1} H_2PH_1^T \Delta^{-1}
\]

\[
= \left( P - PH_2^T \left[ H_2PH_2^T + R_2 \right]^{-1} H_2P \right) H_1^T \Delta^{-1}
\]

\[
= \left[ P^{-1} + H_2^T R_2^{-1} H_2 \right]^{-1} H_1^T \Delta^{-1}
\]

\[
\times \left( 1 - H_1^T R_1^{-1} H_1 \left[ P^{-1} + H_1^T R_1^{-1} H_1 + H_2^T R_2^{-1} H_2 \right]^{-1} \right) H_1^T R_1^{-1}
\]

\[
= \left[ P^{-1} + H_2^T R_2^{-1} H_2 \right]^{-1} H_1^T R_1^{-1},
\]

and similarly for the second gain

\[
W_2 = \left[ P^{-1} + H_2^T R_2^{-1} H_1 + H_2^T R_2^{-1} H_2 \right]^{-1} H_2^T R_2^{-1}
\]

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 156, 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.

\[
I - W(k)H(k) = \left[ P(k \mid k - 1) - W(k)H(k)P(k \mid k - 1) \right] P^{-1}(k \mid k - 1)
\]

\[
= \left[ P(k \mid k - 1) - W(k)S(k) \left( S^{-1}(k)H(k)P(k \mid k - 1) \right) \right]
\]

\[
\times P^{-1}(k \mid k - 1)
\]

\[
= \left[ P(k \mid k - 1) - W(k)S(k)W^T(k) \right] P^{-1}(k \mid k - 1)
\]

\[
= P(k \mid k)P^{-1}(k \mid k - 1).
\]

Similarly,

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

\[
W(k) \left[ H(k)P(k \mid k - 1)H^T(k) + R(k) \right] = P(k \mid k - 1)H^T(k)
\]
\[
W(k)R(k) = [1 - W(k)H(k)]P(k | k - 1)H^T(k)
\]

so
\[
W(k) = P(k | k)H^T(k)R^{-1}(k). \tag{159}
\]

Substituting Equations 158 and 159 into Equation 90 gives the state update equation as
\[
\hat{x}(k | k) = P(k | k) \left[ P^{-1}(k | k - 1)\hat{x}(k | k - 1) + H^T(k)R^{-1}(k)z(k) \right]. \tag{160}
\]

From Equations 91, 92 and 158 we have
\[
P(k | k) = [I - W(k)H(k)]P(k | k - 1)[I - W(k)H(k)]^T + W(k)R(k)W^T(k). \tag{161}
\]

Substituting in Equations 158 and 159 gives
\[
P(k | k) = [P^{-1}(k | k - 1) + H^T(k)R^{-1}(k)H(k)]^{-1}. \tag{163}
\]

Thus, in the inverse-covariance filter\(^9\), the state covariance is first obtained from Equation 163 and then the state itself is found from Equation 160.

Consider now \(S\) sensors, each observing a common state according to
\[
z_s(k) = H_s(k)x(k) + v_s(k), \quad s = 1, \ldots, S, \tag{164}
\]
where the noise \(v_s(k)\) is assumed to be white and uncorrelated in both time and between sensors;
\[
E\{v_s(k)\} = 0,
E\{v_s(i)v_p(j)\} = \delta_{ij}\delta_{sp}R_s(k) \quad s, p = 1, \ldots, S; \quad i, j = 1, \ldots. \tag{165}
\]

A composite observation vector \(z(k)\) comprising a stacked vector of observations from the \(S\) sensors may be constructed in the form
\[
z(k) = \begin{bmatrix} z_1(k) \\ \cdots \\ z_S(k) \end{bmatrix}, \tag{166}
\]
a composite observation matrix \(H(k)\) comprising the stacked matrix of individual sensor observation matrices
\[
H(k) = \begin{bmatrix} H_1(k) \\ \cdots \\ H_S(k) \end{bmatrix}, \tag{167}
\]

\(^9\)Strictly, the inverse covariance filter is defined using the inverse of Equation 163 [37].
and a composite noise vector $v(k)$ comprising a stacked vector of noise vectors from each of the sensors

$$v(k) = \begin{bmatrix} v_1(k) \\ \vdots \\ v_T^S(k) \end{bmatrix}. \quad (168)$$

From Equation 165, the covariance in this composite noise vector is a block diagonal matrix

$$R(k) = E\{v(k)v^T(k)\} = \text{blockdiag}(R_1(k), \ldots, R_S(k)). \quad (169)$$

With these definitions, we have

$$H^T(k)R^{-1}(k)z(k) = \begin{bmatrix} H^T_1(k) & H^T_2(k) & \cdots & H^T_S(k) \end{bmatrix} \begin{bmatrix} R_1^{-1}(k) & 0 & \cdots & 0 \\ 0 & R_2^{-1}(k) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_S^{-1}(k) \end{bmatrix} \begin{bmatrix} z_1(k) \\ z_2(k) \\ \vdots \\ z_S(k) \end{bmatrix} = \sum_{i=1}^S H_i^T(k)R_i^{-1}(k)z_i(k), \quad (170)$$

and

$$H^T(k)R^{-1}(k)H(k) = \begin{bmatrix} H^T_1(k) & H^T_2(k) & \cdots & H^T_S(k) \end{bmatrix} \begin{bmatrix} R_1^{-1}(k) & 0 & \cdots & 0 \\ 0 & R_1^{-1}(k) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_S^{-1}(k) \end{bmatrix} \begin{bmatrix} H_1(k) \\ H_2(k) \\ \vdots \\ H_S(k) \end{bmatrix} = \sum_{i=1}^S H_i^T(k)R_i^{-1}(k)H_i(k). \quad (171)$$

Substituting Equation 170 into Equation 160, the state update equation becomes

$$\hat{x}(k \mid k) = P(k \mid k) \left[ P^{-1}(k \mid k - 1) \hat{x}(k \mid k - 1) + \sum_{i=1}^S H_i^T(k)R_i^{-1}(k)z_i(k) \right]. \quad (172)$$

Substituting Equation 171 into Equation 163, the state-covariance update equation becomes

$$P(k \mid k) = \left[ P^{-1}(k \mid k - 1) + \sum_{i=1}^S H_i^T(k)R_i^{-1}(k)H_i(k) \right]^{-1}. \quad (173)$$

The multi-sensor inverse-covariance filter thus consists of a conventional state and state-covariance prediction stage given by Equations 81 and 82, followed by a state and state-covariance update from Equations 172 and 173.
Example 24

Consider again the multi-sensor tracking problem of Example 21 with three sensors:

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

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

and

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

Thus the group sensor model is given by

\[
H = \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix},
\]

\[
R = \begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_3 \end{bmatrix} = \begin{bmatrix} \sigma_{r_1}^2 & 0 & 0 \\ 0 & \sigma_{r_2}^2 & 0 \\ 0 & 0 & \sigma_{r_3}^2 \end{bmatrix}.
\]

It follows that

\[
H^T R^{-1} z = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_{r_1}^2} & 0 & 0 \\ 0 & \frac{1}{\sigma_{r_2}^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_{r_3}^2} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} \frac{z_1}{\sigma_{r_1}} + \frac{z_2}{\sigma_{r_2}} \\ \frac{z_2}{\sigma_{r_2}} + \frac{z_3}{\sigma_{r_3}} \end{bmatrix} = \sum_{i=1}^{3} H_i^T R_i^{-1} z_i,
\]

and

\[
H^T R^{-1} H = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_{r_1}^2} & 0 & 0 \\ 0 & \frac{1}{\sigma_{r_2}^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_{r_3}^2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}
\]

\[
= \begin{bmatrix} \frac{1}{\sigma_{r_1}^2} & \frac{1}{\sigma_{r_3}^2} \\ \frac{1}{\sigma_{r_2}^2} & \frac{1}{\sigma_{r_3}^2} \end{bmatrix}
\]

\[
= \sum_{i=1}^{3} H_i^T R_i^{-1} H_i
\]

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 $H_i^T R_i^{-1} z_i(k)$ and $H_i^T (k) R_i^{-1}(k) H_i(k)$ 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 173 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 Architecture](image)

**Figure 24:** 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.

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 generate a global track estimate. A typical track-to-track fusion architecture is shown in Figure 24. In some configurations, the global estimate is then communicated back to the local sensor sites (called a feedback 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 [2, 11].

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

$$\mathbf{x}(k) = \mathbf{F}(k)\mathbf{x}(k-1) + \mathbf{B}(k)\mathbf{u}(k) + \mathbf{G}(k)\mathbf{v}(k).$$  \hspace{1cm} (174)

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

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

but where observation noises are assumed independent.

$$\mathbb{E}\{\mathbf{w}_i(k)\mathbf{w}_j(k)\} = \delta_{ij}\mathbf{R}_i(k).$$  \hspace{1cm} (176)

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

$$\hat{\mathbf{x}}_i(k \mid k) = \hat{\mathbf{x}}_i(k \mid k-1) + \mathbf{W}_i(k) [\mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}_i(k \mid k-1)],$$  \hspace{1cm} (177)

and

$$\mathbf{P}_i(k \mid k) = \mathbf{P}_i(k \mid k-1) - \mathbf{W}_i(k)\mathbf{S}_i(k)\mathbf{W}_i^T(k),$$  \hspace{1cm} (178)

where

$$\mathbf{W}_i(k) = \mathbf{P}_i(k \mid k-1)\mathbf{H}_i^T(k)\mathbf{S}_i^{-1}(k),$$  \hspace{1cm} (179)

and

$$\mathbf{S}_i(k) = \left[\mathbf{H}_i(k)\mathbf{P}_i(k \mid k-1)\mathbf{H}_i^T(k) + \mathbf{R}_i(k)\right].$$  \hspace{1cm} (180)

Local state predictions are generated from a common state model

$$\hat{\mathbf{x}}_i(k \mid k-1) = \mathbf{F}(k)\hat{\mathbf{x}}_i(k-1 \mid k-1) + \mathbf{B}(k)\mathbf{u}(k)$$  \hspace{1cm} (181)

and

$$\mathbf{P}_i(k \mid k-1) = \mathbf{F}(k)\mathbf{P}_i(k-1 \mid k-1)\mathbf{F}^T(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^T(k)$$  \hspace{1cm} (182)

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

$$\hat{\mathbf{x}}_T(k \mid k) = \mathbf{P}_T(k \mid k)\sum_{i=1}^{N} \mathbf{P}_i^{-1}(k \mid k)\hat{\mathbf{x}}_i(k \mid k)$$  \hspace{1cm} (183)

$$\mathbf{P}_T(k \mid k) = \left[\sum_{i=1}^{N} \mathbf{P}_i^{-1}(k \mid k)\right]^{-1}.$$  \hspace{1cm} (184)

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 $\hat{x}_i(k \mid k)$ and $\hat{x}_j(k \mid k)$ 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 $\hat{x}_T(k \mid k)$ 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

$$\hat{x}_i(k \mid k) = x(k) - \hat{x}_i(k \mid k)$$

$$= x(k) - \hat{x}_i(k \mid k - 1) - W_i(k)\left[z_i(k) - H_i(k)\hat{x}_i(k \mid k - 1)\right]$$

$$= x(k) - \hat{x}_i(k \mid k - 1) - W_i(k)\left[H_i(k)x(k) + w_i(k) - H_i(k)\hat{x}_i(k \mid k - 1)\right]$$

$$= [1 - W_i(k)H_i(k)] \hat{x}_i(k \mid k - 1) - W_i(k)w_i(k) \tag{185}$$

and

$$\hat{x}_i(k \mid k - 1) = x(k) - \hat{x}_i(k \mid k - 1)$$

$$= F(k)x(k - 1) - F(k)\hat{x}_i(k - 1 \mid k - 1) + G(k)v(k)$$

$$= F(k)\hat{x}_i(k - 1 \mid k - 1) + G(k)v(k) \tag{186}$$

Squaring and taking expectations of Equation 186 yields an expression for the predicted track-to-track cross-correlation

$$P_{ij}(k \mid k - 1) = E\{\hat{x}_i(k \mid k - 1)\hat{x}_j^T(k \mid k - 1) \mid Z^{k-1}\}$$

$$= E\{[F(k)\hat{x}_i(k - 1 \mid k - 1) + G(k)v(k)] \times [F(k)\hat{x}_i(k - 1 \mid k - 1) + G(k)v(k)]^T\}$$

$$= F(k)E\{\hat{x}_i(k - 1 \mid k - 1)\hat{x}_j^T(k - 1 \mid k - 1) \mid Z^{k-1}\} F^T(k)$$

$$+ G(k)E\{v(k)v^T(k)\} \quad G^T(k)$$

$$= F(k)P_{ij}(k - 1 \mid k - 1)F^T(k) + G(k)Q(k)G^T(k). \quad \tag{187}$$

Squaring and taking expectations of Equation 185 yields an expression for the estimate track-to-track cross-correlation

$$P_{ij}(k \mid k) = E\{\hat{x}_i(k \mid k)\hat{x}_j^T(k \mid k)\}$$

$$= E\{(1 - W_i(k)H_i(k))\hat{x}_i(k \mid k - 1) - W_i(k)w_i(k)\}$$

$$\times \left[(1 - W_j(k)H_j(k))\hat{x}_j(k \mid k - 1) - W_j(k)w_j(k)\right] \mid Z^k\}$$

$$= [1 - W_i(k)H_i(k)] E\{\hat{x}_i(k \mid k - 1)\hat{x}_j^T(k \mid k - 1)\} [1 - W_j(k)H_j(k)]^T$$

$$+ W_i(k)E\{w_i(k)w_j^T(k)\} \quad W_j^T(k)$$

$$= [1 - W_i(k)H_i(k)] P_{ij}(k \mid k - 1) [1 - W_j(k)H_j(k)]^T \tag{188}$$
where we have used the fact that $E\{w_i(k)w_i^T(k)\} = 0$. Together, Equations 187 and 188 provide a recursive relationship for computing the cross-correlation $P_{ij}(k | k)$ between the two track estimates $\hat{x}_i(k | k)$ and $\hat{x}_j(k | k)$.

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

$$
\hat{x}(k | k) = \hat{x}(k | k - 1) + P_{xz}^{-1} \left[ z(k) - \hat{z}(k | k - 1) \right]
$$

(189)

and

$$
P(k | k) = P(k | k - 1) - P_{xz}^{-1} P_{xz}^T
$$

(190)

so

$$
\hat{x}_T(k | k) = \hat{x}_i(k | k) + P_{ij}(k | k) P_{ij}^{-1}(k | k) \left[ \hat{x}_j(k | k) - \hat{x}_i(k | k) \right]
$$

(191)

and

$$
P_T(k | k) = P_i(k | k) - P_{ij}(k | k) P_{ij}^{-1}(k | k) P_{ij}^T(k | k)
$$

(192)

identify

$$
P_{ij}(k | k) = P_i(k | k) - P_{ij}(k | k)
$$

(193)

and

$$
P_{i+j}(k | k) = P_i(k | k) + P_j(k | k) - P_{ij}(k | k) - P_{ij}^T(k | k)
$$

(194)

Substituting Equations 193 and 194 into Equation 191 gives

$$
\hat{x}_T(k | k) = \hat{x}_i(k | k) + \left[ P_i(k | k) - P_{ij}(k | k) \right]
\times \left[ P_i(k | k) + P_j(k | k) - P_{ij}(k | k) - P_{ij}^T(k | k) \right]^{-1}
\times \left[ \hat{x}_j(k | k) - \hat{x}_i(k | k) \right]
$$

(195)

as the combined track estimate, and

$$
P_T(k | k) = \left[ P_i(k | k) - P_{ij}(k | k) \right]
\times \left[ P_i(k | k) + P_j(k | k) - P_{ij}(k | k) - P_{ij}^T(k | k) \right]^{-1}
\times \left[ P_i(k | k) - P_{ij}(k | k) \right]^T
$$

(196)

Equations 195 and 196 are in the form of predictor-corrector equations. As written, $\hat{x}_i(k | k)$ is the predicted track, and $\hat{x}_j(k | k) - \hat{x}_i(k | k)$ 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 extensions to the basic track-to-track fusion algorithm. Notable is the use of “equivalent” measurements described in [11]. However, these methods are more appropriately described in context of the information filter.
4 Decentralised Data Fusion Systems

The section addresses the development of algorithms for 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 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 its 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 local interpretation of information.

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.

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.

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 8 and 9, or on the log-likelihood opinion pools shown in Figures 12 and 13. Here the focus is on hierarchical estimation and tracking algorithms (See Figures 25 and 26).

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

\[
\mathbf{z}_i(k) = \mathbf{H}_i(k)\mathbf{x}(k) + \mathbf{w}_i(k), \quad i = 1, \ldots, S
\]  

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

\[
\hat{\mathbf{x}}_i(k \mid k) = \hat{\mathbf{x}}_i(k \mid k - 1) + \mathbf{W}_i(k) [\mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}_i(k \mid k - 1)], \quad i = 1, \ldots, S.
\]

Figure 25: Single Level Hierarchical Multiple Sensor Tracking System
Figure 26: Multiple Level Hierarchical Multiple Sensor Tracking System
These local estimates $\hat{x}_i(k \mid k)$ 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

$$\hat{x}(k \mid k) = \sum_{i=1}^{S} \omega_i(k) \hat{x}_i(k \mid k), \quad (199)$$

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

$$x_i(k) = F_i(k)x(k) + B_i(k)u(k) + G_i(k)v(k), \quad i = 1, \cdots, S.$$ 

This means that the predictions made at local sites are correlated and so the updated local estimates in Equation 198 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 199.

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 195 and 196 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 [28] is notable in employing, indirectly, the information form of the Kalman filter to derive a hierarchical estimation algorithm. The earlier paper by Speyer [54] 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 [52]. In addition, the track-to-track fusion techniques described by Bar-Shalom [16, 2] 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.

![Blackboard Architecture in Data Fusion](image)

Notable is the “Blackboard” architecture (see Figure 27), 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 [45]. In data fusion, the Blackboard approach has been most widely used for knowledge-based
data fusion systems in data interpretation and situation assessment (see [26] and [27] 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 28 and 29 and 30 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 \textit{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.

![Diagram of decentralised data fusion system](image)

**Figure 28:** 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 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 \textit{information} measures as the means of quantifying, communicating and assimilating sensory data. In decentralised
Figure 29: 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 30: A decentralised data fusion system implemented with a hybrid, broadcast and point-to-point, communication architecture.
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 $x(i)$, and yield estimates $\hat{x}(i | j)$ together with a corresponding estimate variance $P(i | j)$. The information filter deals instead with the information state vector $\hat{y}(i | j)$ and information matrix $Y(i | j)$ defined as

$$\hat{y}(i | j) = P^{-1}(i | j)\hat{x}(i | j), \quad Y(i | j) = P^{-1}(i | j).$$

(200)

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:

$$\hat{x}(k | k) = (1 - W(k)H(k))\hat{x}(k | k - 1) + W(k)z(k)$$

(201)

$$P(k | k) = (1 - W(k)H(k))P(k | k - 1)(1 - W(k)H(k))^T + W(k)R(k)W^T(k)$$

(202)

Now, from Equations 158 and 159, we have

$$1 - W(k)H(k) = P(k | k)P^{-1}(k | k - 1),$$

(203)

and

$$W(k) = P(k | k)H^T(k)R^{-1}(k).$$

(204)

Substituting Equations 203 and 204 into Equation 201 gives

$$\hat{x}(k | k) = P(k | k)P^{-1}(k | k - 1)\hat{x}(k | k - 1) + P(k | k)H^T(k)R^{-1}(k)z(k).$$

Pre-multiplying through by $P^{-1}(k | k)$ gives the update equation for the information-state vector as

$$P^{-1}(k | k)\hat{x}(k | k) = P^{-1}(k | k - 1)\hat{x}(k | k - 1) + H^T(k)R^{-1}(k)z(k).$$

(205)

Defining

$$i(k) \triangleq H^T(k)R^{-1}(k)z(k)$$

(206)
as the information-state contribution from an observation \( z(k) \), and with the definitions in Equation 200, Equation 205 becomes

\[
\hat{y}(k | k) = \hat{y}(k | k - 1) + i(k). \tag{207}
\]

A similar expression can be obtained for the covariance update of Equation 202. Substituting Equations 203 and 204 into Equation 202 and rearranging gives

\[
P^{-1}(k | k) = P^{-1}(k | k - 1) + H^T(k)R^{-1}(k)H(k). \tag{208}
\]

Defining

\[
I(k) \triangleq H^T(k)R^{-1}(k)H(k) \tag{209}
\]

as the information matrix associated with the observation, and with the definitions in Equation 200, Equation 208 becomes the information matrix update equation

\[
Y(k | k) = Y(k | k - 1) + I(k). \tag{210}
\]

A comparison of the Kalman filter update stage (Equations 201 and 202) with the information filter update stage (Equations 207 and 210) 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 its 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

\[
P(k | k - 1) = F(k)P(k - 1 | k - 1)F^T(k) + G(k)Q(k)G^T(k) \tag{211}
\]

To derive the prediction stage for the information filter, the following version of the matrix inversion lemma is noted [37]

\[
\begin{pmatrix} A + B^T C \end{pmatrix}^{-1} = A^{-1} - A^{-1}B^T \left( 1 + CA^{-1}B^T \right)^{-1} CA^{-1}.
\]

Identifying

\[
A = F(k)P(k - 1 | k - 1)F^T(k), \quad B^T = G(k)Q(k), \quad C = G^T(k),
\]

the inverse of Equation 211 becomes

\[
P^{-1}(k | k - 1) = M(k) - M(k)G(k) \left[ G^T(k)M(k)G(k) + Q^{-1}(k) \right]^{-1} G^T(k)M(k) \tag{212}
\]

where

\[
M(k) = F^{-T}(k)P^{-1}(k - 1 | k - 1)F^{-1}(k) \tag{213}
\]

when \( Q(k) \) is non singular. Noting the definition of the state transition matrix

\[
F(k) \triangleq \Phi(t_k, t_{k-1})
\]
implies $F^{-1}(k)$ always exists and indeed

$$F^{-1}(k) = \Phi(t_{k-1}, t_k)$$

is simply the state transition matrix defined backwards from a time $t_k$ to $t_{k-1}$. Now, defining

$$\Sigma(k) \triangleq [G^T(k)M(k)G + Q^{-1}(k)] , \quad (214)$$

and

$$\Omega(k) \triangleq M(k)G(k)\left[G^T(k)M(k)G + Q^{-1}(k)\right]^{-1} = M(k)G(k)\Sigma^{-1}(k), \quad (215)$$

the information matrix prediction equation becomes

$$Y(k \mid k - 1) = P^{-1}(k \mid k - 1) = M(k) - \Omega(k)\Sigma(k)\Omega^T(k). \quad (216)$$

A number of alternate expressions for the information prediction stage can also be derived:

$$Y(k \mid k - 1) = \left[1 - \Omega(k)G^T(k)\right]M(k) \quad (217)$$

and

$$Y(k \mid k - 1) = \left[1 - \Omega(k)G^T(k)\right]M(k) \left[1 - \Omega(k)G^T\right]^T + \Omega(k)Q^{-1}(k)\Omega^T(k). \quad (218)$$

The information-state prediction equations may also be obtained as

$$\hat{y}(k \mid k - 1) = \left[1 - \Omega(k)G^T(k)\right]F^{-T}(k) \times [\hat{y}(k - 1 \mid k - 1) + Y(k - 1 \mid k - 1)F^{-1}(k)B(k)u(k)]$$

$$= \left[1 - \Omega(k)G^T(k)\right]F^{-T}(k)\hat{y}(k - 1 \mid k - 1) + M(k)B(k)u(k)$$

$$= \left[1 - \Omega(k)G^T(k)\right]F^{-T}(k)\hat{y}(k - 1 \mid k - 1) + Y(k \mid k - 1)B(k)u(k). \quad (219)$$

It should be noted that the complexity of the inversion of $\Sigma(k)$ is only of order the dimension of the driving noise (often scalar). Further $Q(k)$ is almost never singular, and indeed if it were, singularity can be eliminated by appropriate definition of $G(k)$. The special case in which $Q(k) = 0$ yields prediction equations in the form:

$$Y(k \mid k - 1) = M(k), \quad \hat{y}(k \mid k - 1) = F^{-T}(k)\hat{y}(k - 1 \mid k - 1) + M(k)B(k)u(k) \quad (220)$$

The information filter is now summarised

**Prediction:**

$$\hat{y}(k \mid k - 1) = \left[1 - \Omega(k)G^T(k)\right]F^{-T}(k)\hat{y}(k - 1 \mid k - 1) + Y(k \mid k - 1)B(k)u(k) \quad (221)$$

$$Y(k \mid k - 1) = M(k) - \Omega(k)\Sigma(k)\Omega^T(k) \quad (222)$$
where

\[ M(k) = F^{-T}(k)Y(k - 1 | k - 1)F^{-1}(k), \]  
(223)

\[ \Omega(k) = M(k)G(k)\Sigma^{-1}(k), \]  
(224)

and

\[ \Sigma(k) = \left[ G^T(k)M(k)G(k) + Q^{-1}(k) \right]. \]  
(225)

**Estimate:**

\[ \hat{\gamma}(k | k) = \hat{\gamma}(k | k - 1) + i(k) \]  
(226)

\[ Y(k | k) = Y(k | k - 1) + I(k). \]  
(227)

where

\[ i(k) = H^T(k)R^{-1}(k)z(k), \quad I(k) = H^T(k)R^{-1}(k)H(k) \]  
(228)

Given the interpretation of the information matrix \( Y(i | j) \) as the Fisher information, the update Equation 227 is simply seen to add the information contributed by the observation. Conversely, the prediction Equation 222 subtracts information caused by process uncertainties.

**Example 25**

Consider again Example 16 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 \( E\{v^2(t)\} = q \). The state transition matrix and its inverse over a time interval \( \delta t \) are given by

\[ F(\delta t) = \begin{bmatrix} 1 & \delta t \\
0 & 1 \end{bmatrix}, \quad F^{-1}(\delta t) = \begin{bmatrix} 1 & -\delta t \\
0 & 1 \end{bmatrix} \]

and the noise transition matrix by (Equation 69)

\[ G(\delta t) = \int_0^{\delta t} \begin{bmatrix} 1 & \tau \\
0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\
0 & 1 \end{bmatrix} d\tau = \begin{bmatrix} \delta t^2/2 \\
\delta t \end{bmatrix} \]

Setting

\[ Y(k - 1 | k - 1) = \begin{bmatrix} Y_{xx} & Y_{xv} \\
Y_{xv} & Y_{vv} \end{bmatrix}, \]

the propagated information is

\[ M(k) = F^{-T}(k)Y(k - 1 | k - 1)F^{-1}(k) = \begin{bmatrix} Y_{xx} & Y_{xv} - Y_{xx}\delta t \\
Y_{xv} - Y_{xx}\delta t & Y_{xx}\delta t^2 - 2Y_{xv}\delta t + Y_{vv} \end{bmatrix}. \]

Then

\[ \Sigma(k) = \left[ G^T(k)M(k)G(k) + Q^{-1}(k) \right] = \delta t^2 \left[ Y_{xx}\delta t^2/4 - Y_{xv}\delta t + Y_{vv} \right] + q^{-1}. \]
and
\[
\begin{align*}
\Omega(k) &= M(k)G(k)\Sigma^{-1}(k) \\
&= \frac{1}{Y_{xx}\delta t^4/4 - Y_{xv}\delta t^3 + Y_{vv}\delta t^2 + q^{-1}} \left[ Y_{xv}\delta t - Y_{xx}\delta t^2/2 \right. \\
&\left. - Y_{vv}\delta t^3/2 - 3Y_{xv}\delta t^2/2 + Y_{vv} \right].
\end{align*}
\]

Finally, denoting
\[
M(k) = \begin{bmatrix} M_{xx} & M_{xv} \\ M_{xv} & M_{vv} \end{bmatrix},
\]
the propagation gain matrix is found as
\[
1 - \Omega(k)G^T(k) = \frac{1}{\delta t} \left[ \begin{array}{cc}
\delta t/4 M_{xx} - \delta tM_{xv} + M_{vv} + q^{-1}
\
M_{xv}\delta t/2 + M_{vv} + q^{-1}/\delta t^2
\end{array} \right] \left[ \begin{array}{cc}
M_{xx}\delta t/2 + M_{xv}
\
M_{xx}\delta t^2/4 + M_{vv}\delta t/2 + M_{xx}
\end{array} \right].
\]

Note also the term
\[
F^{-T}(k)\hat{y}(k-1 \mid k-1) = \begin{bmatrix} 1 & 0 \\ -\delta t & 1 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} - \delta t y \end{bmatrix}
\]

Assume observations are made of the position of the particle at discrete synchronous time intervals according to
\[
z(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \dot{x}(k) \end{bmatrix} + w(k)
\]
where \(w(t)\) is a zero mean white noise process with \(E\{w^2(t)\} = r\). The information state is simply given by
\[
i(k) = H^T(k)R^{-1}(k)z(k) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} r^{-1} z_x = \begin{bmatrix} z_x/r \\ 0 \end{bmatrix},
\]
and the information matrix by
\[
I(k) = H^T(k)R^{-1}(k)H(k) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} r^{-1} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1/r & 0 \\ 0 & 0 \end{bmatrix}.
\]

A key problem with this information state vector \(\hat{y}(i \mid j)\) 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 [1]. In particular, identifying the correspondences
\[
\Omega(k) \rightarrow \mathbf{W}(k), \quad \Sigma(k) \rightarrow \mathbf{S}(k), \quad \mathbf{G}^T(k) \rightarrow \mathbf{H}(k)
\]
\(\Omega(k)\) is seen to take on the role of an information prediction gain matrix, \(\Sigma(k)\) 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 226 and 227 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 decentralized 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

\[
P(x(k) | Z^k) = \frac{P(z(k) | x(k))P(x(k) | Z^{k-1})}{P(z(k) | Z^{k-1})}.
\]  

(229)

If it is assumed that the prior and likelihood are Gaussian as

\[
P(x(k) | Z^{k-1}) \propto \exp \left\{ -\frac{1}{2} [x(k) - \hat{x}(k | k - 1)]^T \mathbf{P}^{-1}(k | k - 1) [x(k) - \hat{x}(k | k - 1)] \right\}.
\]

Then the posterior will also be Gaussian in the form

\[
P(z(k) | x(k)) \propto \exp \left\{ -\frac{1}{2} [z(k) - \mathbf{H}(k)x(k)]^T \mathbf{R}^{-1}(k) [z(k) - \mathbf{H}(k)x(k)] \right\},
\]

then the posterior will also be Gaussian in the form

\[
P(x(k) | Z^k) \propto \exp \left\{ -\frac{1}{2} [x(k) - \hat{x}(k | k)]^T \mathbf{P}^{-1}(k | k) [x(k) - \hat{x}(k | k)] \right\}
\]

Now, substituting these distributions into Equation 229 and take logs gives

\[
[x(k) - \hat{x}(k | k)]^T \mathbf{P}^{-1}(k | k) [x(k) - \hat{x}(k | k)] =
\]

\[
[z(k) - \mathbf{H}(k)x(k)]^T \mathbf{R}^{-1}(k) [z(k) - \mathbf{H}(k)x(k)]
\]

\[
+ [x(k) - \hat{x}(k | k - 1)]^T \mathbf{P}^{-1}(k | k - 1) [x(k) - \hat{x}(k | k - 1)] + C(z(k))
\]  

(230)
where $C(z(k))$ is independent of $x(k)$. This equation relates the log likelihoods of these Gaussian distributions and is essentially a quadratic in $x(k)$. Now, differentiating this expression once with respect to $x(k)$ and rearranging gives Equation 226. Differentiating a second time and rearranging gives

$$P^{-1}(k | k) = H^T(k)R^{-1}(k)H(k) + P^{-1}(k | k - 1),$$

which is Equation 227. The first derivative of the log-likelihood is known as the score function. The second derivative is, of course, the Fisher information Equation 59.

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 [46]). 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 are 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 [1, 37]. 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 201 and 202 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 $S(k)$ can never be diagonal, and so cannot be simply partitioned and inverted to yield a gain matrix for each individual observation. Thus, for a set of sensors $i = 1, \ldots, N$, it is not possible to compute the simple sum of innovations as

$$\hat{x}(k | k) \neq \hat{x}(k | k - 1) + \sum_{i=1}^{N} W_i(k) [z_i(k) - H_i(k)\hat{x}(k | k - 1)]$$

in which the individual gains are given by

$$W_i(k) = P(k | k - 1)H_i^T(k)S_i^{-1}(k).$$
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

\[
\hat{y}(k \mid k) = \hat{y}(k \mid k - 1) + \sum_{j=1}^{N} i_j(k),
\]

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

\[
P(z_1(k) \cdots, z_N(k) \mid x(k)) = \prod_{i=1}^{N} P(z_i(k) \mid x(k)),
\]

then Bayes Theorem gives (Equation 19)

\[
P(x(k) \mid Z^n(k)) = P(x(k) \mid Z^n(k - 1)) \prod_{i=1}^{N} P(z_i(k) \mid x(k)).[P(Z^n(k) \mid Z^n(k - 1))]^{-1}.
\]

(232)

Taking logs of Equation 232 then gives

\[
\ln P(x(k) \mid Z^n(k)) = \ln P(x(k) \mid Z^n(k - 1)) + \sum_{i=1}^{n} \ln P(z_i(k) \mid x(k)) - \ln P(Z^n(k) \mid Z^n(k - 1)).
\]

(233)

Given Equation 230, an identification can be made as

\[
\sum_{i=1}^{n} \ln P(z_i(k) \mid x(k)) \doteq \{\sum_{i=1}^{n} i(k), \sum_{i=1}^{n} I(k)\}.
\]

(234)

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

$$z_i(k) = H_i(k)x(k) + v_i(k)$$

(235)

with

$$E\{v_p(i)v_q^T(j)\} = \delta_{ij}\delta_{pq}R_p(i).$$

(236)

The observations can be stacked into a composite observation

$$z(k) = [z_1^T(k), \ldots, z_N^T(k)]^T$$

(237)

The observation model can also be stacked into a composite model

$$H(k) = [H_1^T(k), \ldots, H_N^T(k)]^T,$$  

(238)

and

$$v(k) = [v_1^T(k), \ldots, v_N^T(k)]^T$$

(239)

to give a composite observation equation in familiar form

$$z(k) = H(k)x(k) + v(k).$$

Noting that

$$E\{v(k)v^T(k)\} = R(k) = \text{blockdiag}\{R_1^T(k), \ldots, R_N^T(k)\},$$

(240)

a multiple sensor form of Equation 228 can be obtained as

$$i(k) = \sum_{i=1}^{N} i_i(k) = \sum_{i=1}^{N} H_i^T(k)R_i^{-1}(k)z_i(k)$$

(241)

and

$$I(k) = \sum_{i=1}^{N} I_i(k) = \sum_{i=1}^{N} H_i^T(k)R_i^{-1}(k)H_i(k)$$

(242)

where

$$i_i(k) \triangleq H_i^T(k)R_i^{-1}(k)z_i(k)$$

(243)

is the information-state contribution from observation $z_i(k)$ and

$$I_i(k) \triangleq H_i^T(k)R_i^{-1}(k)H_i(k)$$

(244)

is its associated information matrix.

Equations 241 and 242 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 226 and 227 describing the single sensor information update are easily extended to multiple sensor updates in a straight-forward manner as

\[
\hat{y}(k | k) = \hat{y}(k | k - 1) + \sum_{i=1}^{N} i_i(k) \tag{245}
\]

\[
Y(k | k) = Y(k | k - 1) + \sum_{i=1}^{N} I_i(k). \tag{246}
\]

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 \( i(\cdot) \) and \( I(\cdot) \) 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 [28]. 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 245 and 246 can also be distributed in a simple manner.

![Diagram](image)

Figure 31: 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.

One hierarchical architecture that employs this additive property is shown in Figure 31 (this is the information filter form of Figure 12). Each sensor incorporates a full
state model and takes observations according to Equation 235. They all calculate an information-state contribution from their observations in terms of $i_i(k)$ and $I_i(k)$. These are then communicated to the fusion centre and are incorporated into the global estimate through Equations 245 and 246. The information-state prediction is generated centrally using Equations 221 and 222 and the state estimate itself may be found at any stage from $\hat{x}(i \mid j) = Y^{-1}(i \mid j)\hat{y}(i \mid j)$. To avoid communicating predictions to nodes, any validation or data association should take place at the fusion center.

![Diagram](image)

Figure 32: 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 its local observations. The central processor fuses these tracks.

A second hierarchical system which allows local tracks to be maintained at local sensor sites, is shown in Figure 32 (this is the information filter form Figure 13). 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 $\hat{y}_i(\cdot \mid \cdot)$ be the information-state estimate arrived at by each sensor site based only on its own observations and local information-state prediction $\hat{y}_i(k \mid k - 1)$. This local estimate can be found from a local form of Equations 226 and 227 as

$$\hat{y}_i(k \mid k) = \hat{y}_i(k \mid k - 1) + i_i(k)$$

(247)

and

$$\hat{Y}_i(k \mid k) = Y_i(k \mid k - 1) + I_i(k).$$

(248)

These partial information-state estimates are communicated to a fusion center where they can be assimilated according to

$$\hat{y}(k \mid k) = \hat{y}(k \mid k - 1) + \sum_{i=1}^{N} [\hat{y}_i(k \mid k) - \hat{y}_i(k \mid k - 1)]$$

(249)
and
\[ \mathbf{Y}(k | k) = \mathbf{Y}(k | k-1) + \sum_{i=1}^{N} \left[ \mathbf{Y}_i(k | k) - \mathbf{Y}(k | k-1) \right]. \] (250)

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 245 and 246.

Figure 33: 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 33. This architecture is similar to that shown in Figure 32 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

It is a relatively simple step to decentralize the assimilation Equations 249 and 250 in systems where there is a fully connected network of sensing nodes as shown in Figure 34. 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
Figure 34: 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 assumed that each local sensor site or node maintains a state-space model \((F(k), G(k), Q(k))\) identical to an equivalent centralized model so that \(\hat{y}_i(\cdot | \cdot) \equiv \hat{y}(\cdot | \cdot)\) for all \(i = 1, \ldots, N\). Each node begins by computing a local estimate \(\hat{y}_i(k | k - 1)\) and the observed local information \(i_i(k)\) according to (Equations 221 and 222)

**Prediction:**

\[
\hat{y}_i(k | k - 1) = \left[ I - \Omega_i(k)G^T(k) \right] \mathbf{F}^{-T}(k)\hat{y}_i(k - 1 | k - 1) + Y_i(k | k - 1)\mathbf{B}(k)u(k) \\
Y_i(k | k - 1) = M_i(k) - \Omega_i(k)\Sigma_i(k)\Omega_i^T(k) \tag{251}
\]

where

\[
M_i(k) = \mathbf{F}^{-T}(k)Y_i(k - 1 | k - 1)\mathbf{F}^{-1}(k), \tag{253}
\]

\[
\Omega_i(k) = M_i(k)G(k)\Sigma_i^{-1}(k), \tag{254}
\]

and

\[
\Sigma_i(k) = \left[ G^T(k)M_i(k)G(k) + Q^{-1}(k) \right], \tag{255}
\]

**Estimate:**

\[
\hat{y}_i(k | k) = \hat{y}_i(k | k - 1) + i_i(k) \tag{256}
\]

\[
\hat{Y}_i(k | k) = Y_i(k | k - 1) + I_i(k). \tag{257}
\]

These partial information-State estimates are then communicated to neighboring nodes where they are assimilated according to...
Assimilate:

\[
\hat{y}_i(k \mid k) = \hat{y}_i(k \mid k - 1) + \sum_{j=1}^{N} [\tilde{y}_j(k \mid k) - \hat{y}_j(k \mid k - 1)]
\]  

(258)

\[
Y_i(k \mid k) = Y_i(k \mid k - 1) + \sum_{j=1}^{N} [\tilde{Y}_j(k \mid k) - Y_j(k \mid k - 1)]
\]  

(259)

If each node begins with a common initial information-state estimate \(\hat{y}_j(0 \mid 0) = 0\), \(Y_j(0 \mid 0) = 0\) and the network is fully connected, then the estimates obtained by each node will be identical.

The quantities communicated between sensor nodes; \((\tilde{y}_j(k \mid k) - \hat{y}_j(k \mid k - 1))\) and \((\tilde{Y}_j(k \mid k) - Y_j(k \mid k - 1))\), 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 \(i_j(k)\) and \(I_j(k)\); 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 258 and a summation of matrices in Equation 259. 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 251–259 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 [11] 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.

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 $i(k)$ and $I(k)$. 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 $\hat{y}(k \mid k-1)$ and $Y(k \mid k-1)$. 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 $I(k)$ is required. Generally, the dimension of the observation vector is less than that of the state vector, so the information matrix $I(k) = H^T(k)R^{-1}(k)H(k)$ 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 $I_\dagger(k)$. The generalised-inverse is defined such that

$$I(k)I_\dagger(k) = E,$$

where $E$ is an idempotent matrix which acts as the identity matrix for the information matrix and its generalised-inverse [31, 51]

$$I(k)E = I(k), \quad I_\dagger(k)E = I_\dagger(k),$$

so that

$$I(k)I_\dagger(k)I(k) = I(k), \quad I_\dagger(k)I(k)I_\dagger(k) = I_\dagger(k).$$

Amongst all possible generalised inverses that satisfy Equation 260, 261 and 262, the most appropriate definition is that which projects $I(k)$ into the observation space in the following form;

$$I_\dagger(k) \triangleq H^T(k) \left[ H(k)I(k)H^T(k) \right]^{-1} H(k).$$

This generalised inverse exploits the role of $H(k)$ as a projection operation, taking state space to observation space, and $H^T(k)$ as a back-projection, taking observation space to
state space [56]. Both projections do not change the content of the projected matrix so they can be applied without modifying the information contribution. Multiplying Equation 263 by $H(k)I(k)$ demonstrates this

$$H(k)I(k)I^\dagger(k) = H(k)I(k)H^T(k) \left[ H(k)I(k)H^T(k) \right]^{-1} H(k) = H(k)$$  \hspace{1cm} (264)

The innovation measure is essential for data association and fault detection. The innovation is based on the difference between a predicted and observed measurement

$$\nu(k) = z(k) - H(k)\hat{x}(k \mid k - 1).$$  \hspace{1cm} (265)

In information form, the measurement information is provided by the information vector $i(k) = H^T(k)R^{-1}(k)z(k)$. The information residual vector is therefore defined analogously

$$\nu(k) \triangleq H^T(k)R^{-1}(k)\nu(k),$$  \hspace{1cm} (266)

which is simply the innovation $\nu(k)$ projected into information space. Substituting Equation 265 into Equation 266 gives

$$\nu(k) = H^T(k)R^{-1}(k)z(k) - H^T(k)R^{-1}(k)H(k)\hat{x}(k \mid k - 1)$$  \hspace{1cm} (267)

or

$$\nu(k) = i(k) - I(k)Y^{-1}(k \mid k - 1)\hat{y}(k \mid k - 1)$$  \hspace{1cm} (268)

The information residual variance is computed from

$$Y(k) = E\{\nu(k)\nu^T(k) \mid Z^{k-1}(k)\}.$$  \hspace{1cm} (269)

Substituting in Equation 266 gives

$$Y(k) = H^T(k)R^{-1}(k)E\{\nu(k)\nu^T(k) \mid Z^{k-1}(k)\} R^{-1}(k)H(k)$$

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

$$= I(k) + I(k)Y^{-1}(k \mid k - 1)I(k)$$

$$= I(k) \left[ I(k) + Y^{-1}(k \mid k - 1) \right]^{-1} I(k)$$  \hspace{1cm} (270)

The normalised information residual can now be computed from

$$\Gamma(k) = \nu^T(k)Y^\dagger(k)\nu(k)$$  \hspace{1cm} (271)

noting that the pseudo-inverse for $Y(k)$ is

$$Y^\dagger(k) = H^T(k) \left[ H(k)Y(k)H^T(k) \right]^{-1} H(k)$$

$$= H^T(k) \left[ H(k)H^T(k)R^{-1}(k)S(k)R^{-1}(k)H(k)H^T(k) \right]^{-1} H(k)$$

$$= H^T(k) \left[ H(k)H^T(k) \right]^{-1} R(k)S^{-1}(k)R(k) \left[ H(k)H^T(k) \right]^{-1} H(k)$$  \hspace{1cm} (272)
and substituting Equations 266 and 272 into Equation 271 gives

\[ \nu^T(k)Y^\dagger(k)v(k) = \nu^T(k)R^{-1}(k)H(k)H^T(k) \left[ H(k)H^T(k) \right]^{-1} R(k) \]
\[ \times S^{-1}(k)R(k) \left[ H(k)H^T(k) \right]^{-1} H(k)H^T(k)R^{-1}(k)v(k) \]
\[ = \nu^T(k)S^{-1}(k)v(k). \]  

(273)

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 268 and 270. 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 standard data association methods.

### 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

\[ P(x \mid Z^k) = \frac{P(z(k) \mid x)P(x \mid Z^{k-1})}{P(z(k) \mid Z^{k-1})}, \]

(274)

which may be written in terms of log-likelihoods as

\[ \ln P(x \mid Z^k) = \ln P(x \mid Z^{k-1}) + \ln \frac{P(z(k) \mid x)}{P(z(k) \mid Z^{k-1})}, \]

(275)

where \( x \) is the state to be estimated and \( Z^k \) 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 275, the term \( \ln P(x \mid Z^{k-1}) \) corresponds to information accumulated about the state up to time \( k - 1 \). The term

\[ \ln \frac{P(z(k) \mid x)}{P(z(k) \mid Z^{k-1})} \]

corresponds to the new information generated at time \( k \). Equation 275 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
Figure 35: A Decentralized, fully connected, Bayesian data fusion system appropriate for decentralised identification tasks.

likelihoods of all its neighbours which are then fused to form a global estimate. Rewriting Equation 275 in terms of an individual node $i$, this global estimate is computed as

$$\ln P(x_i \mid Z^k) = \ln P(x_i \mid Z^{k-1}) + \sum_j \ln \frac{P(z_j(k) \mid x_i)}{P(z_j(k) \mid Z^{k-1})},$$

where the summation represents the communicated terms. This is illustrated for a node $i$ in Figure 35.
5 Communication in Decentralised Sensing Systems

The issue of communication is a significant issue in distributed and decentralised data fusion systems. This is because of limited communication bandwidth, time delays and communication failures that can occur between sensing and fusion processes. In decentralised systems, a significant issue is also the potential for time-varying communication topology.

This section considers the communication problem in depth. The fundamental issue of determining what needs to be communicated is first developed in terms of Bayes Theorem. This is then translated into both the information filter and discrete estimation form. Using these basic algorithms, the issues of delayed and asequent data are then considered. Algorithms are then developed for dealing with data that comes in out of time or out of order. Finally, practical issues, including burst and intermittent communication, are addressed in the design of communication channel algorithms.

5.1 Decentralised Communication Theory

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 shown that, in general sensor network topologies, the channel filter will not find all common information. This is because the common information between two nodes can not be established uniquely from what is communicated down a channel. Three (sub-optimal) methods of overcoming this problem are described.

5.1.1 Communication Topologies

There are three basic communication topologies that are considered in this section;
• **Fully-connected** or broadcast topologies in which every node has direct access to every other node in the network (Figure 36(a)). This type of topology occurs when a bus or broadcast medium is used for communication. In fully connected arrangements, individual nodes may have intermittent connection to the communication medium. However, it is generally assumed that once connected, they enjoy equal access to other nodes information.

• **Tree-connected** or singly connected topologies in which only one path exists between any two nodes (Figures 36(b) and (c)). This type of topology rarely occurs in physical practice. However, it can readily be imposed on a more general network through the use of routing or tagging algorithms. In singly-connected topologies, nodes can again communicate intermittently. However, if a link is broken, then the network will become disconnected if a different route can not be established. This makes singly-connected networks generally unreliable.

• **Arbitrary-connected** or multiply-connected topologies allow all nodes to connect to any other node either through a single or, more generally multiple paths (Figure 36(d)). In multiply-connected topologies intermittent communication on any one link will not necessarily halt the flow of information through other routes.

The different topologies determine different algorithms for communication. Fully connected and broadcast networks follow most simply from the algorithms previously developed. However, channel filters (to be described) are often still required to deal with intermittent, delayed and burst communications.

In more general networks, the key data fusion problem turns out to be how to determine the common information between two nodes. This information must be removed before two nodes can freely communicate with each other. In fully connected networks, the common information is simply the information the two nodes had in common last time they communicated. In tree or singly-connected networks, there is also a simple means of determining common information by simply remembering (practically, summing) the information that has previously been communicated down the channel connecting two nodes. This accommodates information that has been previously shared while allowing new information, from further down the network, to be propagated.

In general networks however, there is no general optimal algorithm for both propagating information and maintaining optimality within the locality constraints of a decentralised sensing network (the proof of this result can be found in [58]). However, communication in arbitrary networks can be accommodated either by relaxing the constraints imposed on decentralised systems, or by using sub-optimal fusion algorithms. In the former case, methods such as routing or tagging can be used to reduce the problem to one of a singly-connected network where channel-filter algorithms can be used to determine common information. This approach is most appropriate for networks that already have some stable structure; communication hierarchies, or sensors on a single platform. However, in large-scale networks in which the topology changes dynamically on a regular
basis, sub-optimal methods of determining the common information are most appropriate. This is typical of very large networks operating in tactical situations.

5.1.2 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.

Consider a sensor node \( i \). The information set, consisting of all locally available observations, is denoted \( Z_i \). Let \( x \) define the state to be estimated, then \( P(x \mid Z_i) \) defines the posterior probability on the state given the locally available information. Now the set \( Z_i \) contains information from both observations made locally at node \( i \) as well as information communicated to it from other nodes. When it interacts and exchanges information with another node \( j \), it is essential to understand what information results.

In general, if there are two nodes \( i \) and \( j \), with information sets \( Z_i \) and \( Z_j \) respectively, who engage in communication, it is desirable to construct the common information set \( Z_i \cup Z_j \) constructed from the union of the two local information sets. This may then be used to construct a posterior distribution \( P(x \mid Z_i \cup Z_j) \) for the state based on both sources of information. If the two information sets have no information in common \( Z_i \cap Z_j = \emptyset \), then it is simple to construct the union of the sets from a sum. However if the two sets do have information in common \( Z_i \cap Z_j \neq \emptyset \) then the problem of constructing the union \( Z_i \cup Z_j \) resolves to finding the common information \( Z_i \cap Z_j \). Determining this common information turns out to be key to the decentralised communication problem.

The most general solution to the common information problem is in the form of Bayes Theorem. Consider the interaction of pairs of nodes. For each communicating pair \( (i, j) \), the required probability is \( P(x \mid Z_i \cup Z_j) \). Let the union of the individual observation information sets be partitioned into disjoint sets as

\[
Z_i \cup Z_j = Z_{i\setminus j} \cup Z_{j\setminus i} \cup Z_{ij}
\]

where

\[
Z_{i\setminus j} = Z_i \setminus Z_{ij}, \quad Z_{j\setminus i} = Z_j \setminus Z_{ij}, \quad Z_{ij} = Z_i \cap Z_j,
\]

and where the notation \( p \setminus r \) (the restriction operation) means elements of the set \( p \) excluding those elements that are also in set \( r \). Note also that

\[
Z_{i\setminus j} \cup Z_{ij} = Z_i, \quad Z_{j\setminus i} \cup Z_{ij} = Z_j.
\]

Then,

\[
P(Z_i \cup Z_j \mid x) = P(Z_{i\setminus j} \cup Z_{j\setminus i} \cup Z_{ij} \mid x)
\]

\[
= P(Z_{i\setminus j} \mid Z_{j\setminus i} \cup Z_{ij}, x)P(Z_{j\setminus i} \cup Z_{ij} \mid x) = P(Z_{i\setminus j} \mid Z_{ij}, x)P(Z_j \mid x)
\]

\[
= \frac{P(Z_{i\setminus j} \cup Z_{ij} \mid x)}{P(Z_j \mid x)} = \frac{P(Z_i \mid x)}{P(Z_{ij} \mid x)} P(Z_j \mid x)
\]

\[
= \frac{P(Z_i \mid x)P(Z_j \mid x)}{P(Z_i \cap Z_j \mid x)}.
\]
Figure 36: Different communication topologies for a 20 node network: (a) Fully connected; (b) and (c) tree connected; (d) general network topology.
Substituting Equation 278 into Bayes theorem gives

\[
P(x \mid Z_i \cup Z_j) = \frac{P(Z_i \cup Z_j \mid x)P(x)}{P(Z_i \cup Z_j)}
\]

\[
= \frac{P(Z_i \mid x)P(Z_j \mid x)P(x)}{P(Z_i \cup Z_j)}
\]

\[
= \frac{P(x \mid Z_i)P(x \mid Z_j)P(z_i)P(z_j)}{P(x \mid Z_i \cap Z_j)P(Z_i \cap Z_j)}
\]

\[
= c \cdot \frac{P(x \mid Z_i)P(x \mid Z_j)}{P(x \mid Z_i \cap Z_j)}. \tag{279}
\]

This shows that the relation between the posterior probability in the unknown state given information from both nodes, \(P(x \mid Z_i \cup Z_j)\), as a function of the posteriors based only on locally available information, \(P(x \mid Z_i)\) and \(P(x \mid Z_j)\), and the information the two nodes have in common \(P(x \mid Z_i \cap Z_j)\).

Taking logs of Equation 279, gives the intuitive result

\[
\ln P(x \mid Z_i \cup Z_j) = \ln P(x \mid Z_i) + \ln P(x \mid Z_j) - \ln P(x \mid Z_i \cap Z_j). \tag{280}
\]

Equation 280 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 \(\ln P(x \mid Z_i \cap Z_j)\) describes the common information between two nodes which must be removed before fusion.

### 5.1.3 Identification of Redundant Information in Sensor Networks

The key step in deriving fusion equations for decentralised sensing networks is to identify the common information \(Z_i \cap Z_j\) between estimates so that it is not used redundantly. The problem of accounting for redundant information in decentralised communication structures is encountered in many applications\(^1\). 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 \(Z_i^k\) at time \(k\) based on a local sensor observation \(z_i(k)\) 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 \([N_i] : \bigcup_{j \in [N_i]} Z_j^k\). 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

\[
Z_i^k \cup Z_j^k = Z_i^k + Z_j^k - Z_i^k \cap Z_j^k, \tag{281}
\]

\(^1\)In human communication decentralised and cyclic communication structures give rise to “rumour propagation” or the “chicken licken” problem.
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 \( Z_{k}^{i} \cap j \) 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 38). 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

\[
\bigcup_{j \in [N_i]} Z_{k}^{i} = \left[ \sum_{j \in [N_i]} Z_{k}^{j} - \bigcup_{j \in [N_i]} Z_{k-1}^{j} \right].
\] (282)

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 282 is an information-set equivalent of Equations 299 and 300 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.

- **Tree Connection:** In a tree connected topology, there is only one path between each pair of nodes (see Figures 39 and 40). 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 \((i, j)\),

\[
Z_{i}^{k} \cup Z_{j}^{k} = Z_{i}^{k} + Z_{j}^{k} - Z_{i}^{k} \cap j,
\]

where, crucially, the intersection can be found from information sets at the previous timestep,

\[
Z_{i}^{k-1} = Z_{i}^{k-1} \cup Z_{j}^{k-1}.
\] (283)
This generalises to a communication strategy over the neighbourhood as

\[ \bigcup_{j \in [N_i]} Z^k_j = Z^k_i + \left[ \sum_{j \in N_i} Z^k_j - \bigcup_{j \in N_i} Z^{k-1}_j \right], \]

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

\[ Z^k_i \cup Z^k_j = Z^k_i + Z^k_j - Z^k_{i \cap j}. \]

In the arbitrary network case, the intersection term \(Z^k_{i \cap j}\) cannot 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 37 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 \(Z_j\) upon arrival at \(j\). This information must then be acquired by \(i\) through some operation of union or intersect of information sets. To maintain 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 282 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 37, 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.

### 5.1.4 Communication Channel Filters

Equation 280 serves as the basis for developing information communication policies for non-fully connected sensor networks.
Figure 37: Communicated information sets in the three classes of topology.
The probability density functions in Equation 280 can represent four different information or Kalman filter estimates:

- The local estimate at node $i$:
  \[ \hat{x}_i(k | k) \triangleq \mathbb{E}\{x(k) | Z_i^k\} \]
  with covariance $P_i(k | k)$.

- The local estimate at node $j$:
  \[ \hat{x}_j(k | k) \triangleq \mathbb{E}\{x(k) | Z_j^k\} \]
  with covariance $P_j(k | k)$.

- The estimate based on the union of all information possessed by nodes $i$ and $j$ (in effect the global estimate):
  \[ \hat{x}_{i \cup j}(k | k) \triangleq \mathbb{E}\{x(k) | Z_i^k \cup Z_j^k\} \]
  with covariance $P_{i \cup j}(k | k)$.

- The estimate based on the common information between nodes $i$ and $j$:
  \[ \hat{x}_{i \cap j}(k | k) \triangleq \mathbb{E}\{x(k) | Z_i^k \cap Z_j^k\} \]
  with covariance $P_{i \cap j}(k | k)$.

Following Equation 230; Substituting Gaussian distributions for the probability density functions in Equation 280 and taking natural logarithms immediately gives the information filter equivalent of Equation 280 as

\[
\hat{y}_{i \cup j}(k | k) = \hat{y}_i(k | k) + \hat{y}_j(k | k) - \hat{y}_{i \cap j}(k | k),
\]

\[
Y_{i \cup j}(k | k) = Y_i(k | k) + Y_j(k | k) - Y_{i \cap j}(k | k),
\]

where

\[
\hat{y}_i(k | k) = \hat{y}_i(k | k - 1) + i_i(k), \quad \hat{y}_j(k | k) = \hat{y}_j(k | k - 1) + i_j(k)
\]

\[
\hat{Y}_i(k | k) = Y_i(k | k - 1) + I_i(k), \quad \hat{Y}_j(k | k) = Y_j(k | k - 1) + I_j(k)
\]

It remains to evaluate the common information terms

\[
\hat{y}_{ij}(k | k) \triangleq \hat{y}_{i \cap j}(k | k), \quad Y_{ij}(k | k) \triangleq Y_{i \cap j}(k | k).
\]

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

$$
\hat{y}_{ij}(k \mid k) = \hat{y}_i(k \mid k - 1) = \hat{y}_j(k \mid k - 1)
$$

$$
Y_{ij}(k \mid k) = Y_i(k \mid k - 1) = Y_j(k \mid k - 1).
$$

Substitution of Equations 289, 287 and 288 into Equations 285 and 286, yields the previously derived decentralised data fusion Equations 299 and 300.

• **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 258 and 259 provide a recursive expression for the total information communicated between the two nodes as

$$
\hat{y}_{ij}(k \mid k) = \hat{y}_{ij}(k \mid k - 1)
$$

$$
+ [\hat{\gamma}_i(k \mid k) - \hat{\gamma}_{ij}(k \mid k - 1)]
$$

$$
+ [\hat{\gamma}_j(k \mid k) - \hat{\gamma}_{ij}(k \mid k - 1)]
$$

$$
= \hat{\gamma}_i(k \mid k) + \hat{\gamma}_j(k \mid k) - \hat{\gamma}_{ij}(k \mid k - 1)
$$

(290)

and

$$
Y_{ij}(k \mid k) = Y_{ij}(k \mid k - 1)
$$

$$
+ [\hat{\gamma}_i(k \mid k) - Y_{ij}(k \mid k - 1)]
$$

$$
+ [\hat{\gamma}_j(k \mid k) - Y_{ij}(k \mid k - 1)]
$$

$$
= \hat{\gamma}_i(k \mid k) + \hat{\gamma}_j(k \mid k) - Y_{ij}(k \mid k - 1)
$$

(291)

This estimate of common information replaces the prior information terms in Equations 258 and 259. The local updates at each node remain unchanged

$$
\hat{\gamma}_i(k \mid k) = \hat{\gamma}_i(k \mid k - 1) + \delta_i(k)
$$

(292)

$$
\hat{\gamma}_i(k \mid k) = Y_i(k \mid k - 1) + \delta_i(k)
$$

(293)

and the assimilation stage becomes

$$
\hat{\gamma}_i(k \mid k) = \hat{\gamma}_i(k \mid k) + \sum_{j \in N_i} [\hat{\gamma}_j(k \mid k) - \hat{\gamma}_{ji}(k \mid k - 1)]
$$

(294)

$$
Y_i(k \mid k) = \hat{\gamma}_i(k \mid k) + \sum_{j \in N_i} [\hat{\gamma}_j(k \mid k) - Y_{ji}(k \mid k - 1)]
$$

(295)

This filter is clearly symmetric, $\hat{y}_{ij}(\cdot \mid \cdot) = \hat{y}_{ji}(\cdot \mid \cdot)$, 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 \( \hat{y}_{ij}(k | k) \) and \( Y_{ij}(k | k) \) 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 5.1.6.

Equations 290 and 291 define an information filter which estimates the common information between nodes. The filter is completed by addition of a corresponding prediction stage as

\[
\hat{y}_{ij}(k | k - 1) = \left[ 1 - \Omega_{ij}(k)G^T(k) \right] F^{-T}(k)\hat{y}_{ij}(k - 1 | k - 1) + Y_{ij}(k | k - 1)B(k)u(k)
\]

\[
Y_{ij}(k | k - 1) = M_{ij}(k) - \Omega_{ij}(k)\Sigma_{ij}(k)\Omega_{ij}^T(k)
\]

where

\[
M_{ij}(k) = F^{-T}(k)Y_{ij}(k - 1 | k - 1)F^{-1}(k), \quad \Omega_{ij}(k) = M_{ij}(k)G(k)\Sigma_{ij}^{-1}(k),
\]

and

\[
\Sigma_{ij}(k) = \left[ G^T(k)M_{ij}(k)G(k) + Q^{-1}(k) \right].
\]

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.

5.1.5 **Fully Connected and Broadcast Sensor Networks**

It is appropriate to make some comments about the fully connected or broadcast communication Equations.

Recall the assimilation equations

\[
\hat{y}_i(k | k) = \hat{y}_i(k | k - 1) + \sum_j [\hat{y}_j(k | k) - \hat{y}_j(k | k - 1)]
\]

\[
= \hat{y}_i(k | k - 1) + \sum_{j \in N_i} i_j(k)
\]

(299)
and

\[
Y_i(k | k) = Y_i(k | k - 1) + \sum_j \left[ \hat{Y}_j(k | k) - Y_j(k | k - 1) \right]
\]

\[
= Y_i(k | k - 1) + \sum_{j \in N_i} I_j(k).
\] (300)

Equations 299 and 300 make it clear that the estimate arrived at locally by a node \(i\) is based on the observation information \(i_j(k)\) and \(I_j(k)\) 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 \(\hat{y}_j(k | k)\) is obtained at a node \(j\), the prior information at this node \(\hat{y}_j(k | k - 1)\) is subtracted to provide the ‘new’ information to be communicated to a node \(i\). The assumption here is that the prior information \(\hat{y}_j(k | k - 1)\) 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 \(i_j(k)\) 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 \(i_j(k)\) 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 38 shows a fully connected network in which local estimates will be equal to global estimates using only Equations 299 and 300 for assimilation. Figure 39 shows a linear connected sensor network and Figure 40 shows a tree connected sensor network. Here, the estimates arrived at by each node, using Equations 299 and 300 for assimilation, will be based only on observations made by sensors in the immediate neighbourhood.

5.1.6 General Network Structures

Tree networks are 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 \(\hat{y}_{ij}(\cdot | \cdot)\), 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
Figure 38: 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 299 and 300.

Figure 39: A linear sensor network consisting of four nodes, A, B, C, and D. Nodes in this network, using only Equations 299 and 300, can only generate estimates based on information available in their immediate neighbourhood.
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 $\hat{y}_{ij}(\cdot | \cdot)$ 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 it 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. In effect, this allows 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 $\hat{y}_{ij}(\cdot | \cdot) = \tilde{y}_{ij}(\cdot | \cdot)$, 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 described in Section 3.1.6. 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 very conservative. The determination of the common information between two nodes in an arbitrary dynamic network is the subject of current research [44].

5.2 Delays and Timing: Temporal Propagation of Information

This section defines the general state model and the notation and procedure for propagating state both forward and backward in time. This is necessary to enable information to be correctly time-aligned between different sensors in communication and also to provide algorithms to deal with delayed or asequent information. Together these algorithms then allow a capability of dealing with intermittent and burst communications.

5.2.1 System Definitions

For the sake of completeness, some standard definitions are repeated here.

Consider the standard linear continuous-time state space model

$$\dot{x}(t) = F(t)x(t) + B(t)u(t) + G(t)v(t),$$  \hspace{1cm} (301)

where $x(t)$ is the state of interest, $u(t)$ the control input, $v(t)$ the driving noise, and $F(t), B(t), G(t)$ matrices modeling effects of state, control and noise inputs respectively. Equation 301 admits a well known closed-form solution

$$x(t) = \Phi(t,t_0)x(t_0) + \int_{t_0}^{t} \Phi(t,\tau)B(\tau)u(\tau)d\tau + \int_{t_0}^{t} \Phi(t,\tau)G(\tau)v(\tau)d\tau$$ \hspace{1cm} (302)

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

$$\dot{\Phi}(t,t_0) = F(t)\Phi(t,t_0), \quad \Phi(t_0,t_0) = I.$$ \hspace{1cm} (303)

The state transition matrix has three important properties:
1. It is uniquely defined for all $t, t_0$ in $[0, \infty]$.

2. (The semi-group property) $\Phi(t_3, t_1) = \Phi(t_3, t_2)\Phi(t_2, t_1)$.

3. $\Phi(t, t_0)$ is non singular and $\Phi^{-1}(t, t_0) = \Phi(t_0, t)$.

Specifically when $F(t) = F$ is a constant matrix, the state transition matrix is dependent only on the time interval and is given by

$$\Phi(t, t_0) = \Phi(t - t_0) = \exp(F(t - t_0)), \quad (304)$$

5.2.2 Forward Propagation of State

Define an asynchronous discrete time set $t = \{t_0, t_1, \cdots, t_k, \cdots\}$, and a state transition matrix taking the state from a time slot $t_k$ to $t_{k+1}$

$$F_{(k+1,k)} \triangleq \Phi(t_{k+1}, t_k). \quad (305)$$

It is assumed that the control and noise inputs over the time period $t_k$ to $t_{k+1}$ can be approximated as constant and equal to $u(k)$ and $v(k)$ so that with the definitions

$$B_{(k+1,k)} \triangleq \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau)B(\tau)d\tau$$

$$G_{(k+1,k)} \triangleq \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau)G(\tau)d\tau; \quad (306)$$

Equation 302 becomes

$$x(t_{k+1}) = F_{(k+1,k)}x(t_k) + B_{(k+1,k)}u(k) + G_{(k+1,k)}v(k), \quad (307)$$

where the noise sequence $v(k)$ is assumed zero mean and white with variance $Q_k$. Equation 307 describes the evolution of the state forward from a time $t_k$ to $t_{k+1}$. Equation 307 can be generalised to describe propagation forward by $n$ time slots as

$$x(t_{k+n}) = F_{(k+n,k)}x(t_k) + \sum_{i=1}^{n} B_{(k+i,k)}u(k+i-1) + \sum_{i=1}^{n} G_{(k+i,k)}v(k+i-1). \quad (308)$$

5.2.3 Backward Propagation of State

Starting with Equation 307, decrementing the time index and rearranging gives

$$x(t_{k-1}) = F_{(k,k-1)}^{-1}x(t_k) - F_{(k,k-1)}^{-1}B_{(k,k-1)}u(k-1) - F_{(k,k-1)}^{-1}G_{(k,k-1)}v(k-1).$$

Noting that

$$F_{(k,k-1)}^{-1} = F_{(k-1,k)} = \Phi(t_{k-1}, t_k)$$
so consequently

$$-F_{(k,k-1)}^{-1} B_{(k,k-1)} = -\Phi(t_{k-1}, t_k) \int_{t_{k-1}}^{t_k} \Phi(t_k, \tau) B(\tau) d\tau$$

$$= -\int_{t_{k-1}}^{t_k} \Phi(t_{k-1}, t_k) \Phi(t_k, \tau) B(\tau) d\tau$$

$$= -\int_{t_{k-1}}^{t_k} \Phi(t_{k-1}, \tau) B(\tau) d\tau$$

$$= \int_{t_{k-1}}^{t_k} \Phi(t_k, \tau) B(\tau) d\tau$$

$$= B_{(k-1,k)},$$

and similarly

$$-F_{(k-1,k)}^{-1} G_{(k,k-1)} = G_{(k-1,k)},$$

gives

$$x(t_{k-1}) = F_{(k-1,k)} x(t_k) + B_{(k-1,k)} u(k - 1) + G_{(k-1,k)} v(k - 1), \quad (309)$$

describing the evolution of the state backward in time from $t_k$ to $t_{k-1}$. Equation 309 can
be generalised to describe propagation backward by $m$ time slots as

$$x(t_{k-m}) = F_{(k-m,k)} x(t_k) + \sum_{i=1}^{m} B_{(k-i,k)} u(k - i) + \sum_{i=1}^{m} G_{(k-i,k)} v(k - i). \quad (310)$$

5.2.4 Notation for Time Invariant Systems

If the system $F(t)$, $B(t)$, $G(t)$ is time invariant, then the state, control and noise transition matrices depend only on the integration time $t_n - t_m$. It is a significant aid to understanding to define a simplified notation for the case of time-invariant systems. At a time $t_k$, define

$$F_k = F_{k+} = F_{(k+1,k)}, \quad B_k = B_{k+} = B_{(k+1,k)}, \quad G_k = G_{k+} = G_{(k+1,k)}$$
as the ‘one-step forward’ transition matrices,

$$F_{k+n} = F_{(k+n,k)}, \quad B_{k+n} = B_{(k+n,k)}, \quad G_{k+n} = G_{(k+n,k)}$$
as the ‘n-step forward’ transition matrices,

$$F_{k-} = F_{(k-1,k)}, \quad B_{k-} = B_{(k-1,k)}, \quad G_{k-} = G_{(k-1,k)}$$
as the ‘one-step backward’ transition matrices, and

$$F_{k-m} = F_{(k-m,k)}, \quad B_{k-m} = B_{(k-m,k)}, \quad G_{k-m} = G_{(k-m,k)}$$
as the ‘m-step backward’ transition matrices. Note, because of the asynchronous one-step time difference

$$F_k = F_{(k+1,k)} \neq F_{k-}^{-1} = F_{(k-1,k)}^{-1} = F_{(k,k-1)}$$
in general.

Equation 307 is now written as

$$x(t_{k+1}) = F_k x(t_k) + B_k u(k) + G_k v(k),$$

and Equation 309 as

$$x(t_{k-1}) = F_{k-1} x(t_k) + B_{k-1} u(k - 1) + G_{k-1} v(k - 1).$$

The remaining sections define notation for state estimates, covariances, information states, information matrices and the propagation of these both forward and backward in time.

### 5.2.5 Definition of Estimates and Information States

Observations of the state are assumed to be made (asynchronously) according to a standard linear model in the form

$$z(k) = H(k) x(t_k) + w(k)$$

where $w(k)$ is taken to be a zero mean white noise sequence with covariance $R_k$. Define the set of all observations up to time $t_k$ as

$$Z^k = \{z(1), z(2), \cdots, z(k)\}.$$ 

The state estimate $\hat{x}(i \mid j)$ and its variance $P(i \mid j)$ at time $t_i$ given observations up to time $t_j$ are defined as

$$\hat{x}(i \mid j) \triangleq E\{x(t_i) \mid Z^k\}, \quad P(i \mid j) \triangleq E\{(x(t_i) - \hat{x}(i \mid j))(x(t_i) - \hat{x}(i \mid j))^T \mid Z^j\}.$$ 

The information vector $\hat{y}(i \mid j)$ and information matrix $Y(i \mid j)$ at time $t_i$ given observations up to time $t_j$ are defined as

$$\hat{y}(i \mid j) \triangleq P^{-1}(i \mid j) \hat{x}(i \mid j), \quad Y(i \mid j) \triangleq P^{-1}(i \mid j).$$

### 5.2.6 Forward Time Propagation

The propagation of information from a time $t_k$ to a time $t_{k+1}$ is considered. For the Kalman filter, state estimates and covariances are propagated according to

$$\begin{align*}
\hat{x}(k+1 \mid k) &= F_k \hat{x}(k \mid k) + B_k u(k) \\
P(k+1 \mid k) &= F_k P(k \mid k) F_k^T + G_k Q_k G_k^T
\end{align*}$$

For the information filter propagation, define

$$\Sigma(k + 1 \mid k) = G^T_k M_{(k+1,k)} G_k + Q_k^{-1},$$

(311)
where

\[
M_{(k+1,k)} = F_{(k+1,k)}^- Y(k \mid k) F_{(k+1,k)}^{-1} = F_{(k,k+1)}^T Y(k \mid k) F_{(k,k+1)}
\]

\[
\triangle = M_k
\]

is a noise-less propagation of information. The ‘one-step forward’ propagation of the information matrix is then

\[
Y(k + 1 \mid k) = M_k - M_k G_k \Sigma^{-1}(k + 1 \mid k) G_k^T M_k
\] (312)

and the ‘one-step forward’ propagation of the information state

\[
\hat{y}(k + 1 \mid k) = \left[1 - M_k G_k \Sigma^{-1}(k + 1 \mid k) G_k^T\right] F_k^{-T} \hat{y}(k \mid k) + M_k B_k u(k)
\]

\[
= \left[1 - M_k G_k \Sigma^{-1}(k + 1 \mid k) G_k^T\right] F_k^{-T} \hat{y}(k \mid k) + Y(k + 1 \mid k) B_k u(k).
\] (313)

The ‘n-step forward’ propagation are best obtained through direct evaluation of $F_k$, $G_k$ and $B_k$ over a combined time interval.

### 5.2.7 Backward Propagation

For convenience, the propagation of information from a time $t_{k+1}$ to a time $t_k$ is considered. The propagation of state estimates and covariances follows according to

\[
\hat{x}(k - 1 \mid k) = F_{k-1} \hat{x}(k \mid k) + B_k u(k - 1)
\]

\[
\hat{p}(k - 1 \mid k) = F_{k-1} \hat{p}(k \mid k) F_{k-1}^T - G_{k-1} Q_{k-1} G_{k-1}^T
\]

For the information filter propagation, define

\[
\Sigma(k - 1 \mid k) = Q_{k-1}^{-1} - G_{k-1}^T M_{(k-1,k)} G_{k-1},
\] (314)

where

\[
M_{(k-1,k)} = F_{(k-1,k)}^- Y(k \mid k) F_{(k-1,k)}^{-1} = F_{(k,k-1)}^T Y(k \mid k) F_{(k,k-1)}
\]

\[
\triangle = M_{k-1}
\]

is a noise-less back-propagation of information. The ‘one-step backward' propagation of the information matrix is

\[
Y(k - 1 \mid k) = M_{k-1} + M_{k-1} G_{k-1} \Sigma^{-1}(k - 1 \mid k) G_{k-1}^T M_{k-1}
\] (315)

\[\text{Note, the indices } (k \mid k) \text{ and } (k - 1 \mid k) \text{ do not imply an update at time } k; \text{ exactly the same propagation equations would apply for any indices of the form } (k \mid k - m) \text{ and } (k - 1 \mid k - m).\]
and the ‘one-step backward’ propagation of the information state

\[
\dot{y}(k - 1 | k) = \left[ 1 + M_{k-}G_{k-}\Sigma^{-1}(k - 1 | k)G_{k-}^T \right] F_{k-}^{-T}\dot{y}(k | k) + M_{k-}B_{k-}\pi(k - 1)
\]

\[
= \left[ 1 + M_{k-}G_{k-}\Sigma^{-1}(k - 1 | k)G_{k-}^T \right] F_{k-}^{-T}\dot{y}(k | k) + Y(k - 1 | k)B_{k-}\pi(k - 1).
\]

(316)

The ‘m-step backward’ propagation is also best obtained through direct evaluation of \( F_{k-}, G_{k-} \) and \( B_{k-} \) over a combined time interval.

### 5.2.8 Delayed Data

The state can be optimally updated at time \( k \) using delayed data from time \( k - 1 \) using the following

\[
Y^+(k + 1 | k) = Y^-(k + 1 | k) + I^+(k)
\]

\[
\dot{y}^+(k + 1 | k) = \dot{y}^-(k + 1 | k) + i^+(k)
\]

(317)

\[
I^+(k) = M_k^T + M_k^T G_k \left[ G_k^T M_k G_k + Q_k^{-1} \right]^{-1} G_k^T M_k^Y
\]

\[
= \left[ M_k^T G_k + M_k^T G_k \right] \left[ G_k^T M_k^T G_k + G_k^T M_k G_k + Q_k^{-1} \right]^{-1} \left[ M_k Y_k G_k + M_k Y_k G_k \right]^T
\]

\[
i^+(k) = Y^-(k + 1 | k) \left[ F_k (Y(k | k) + I(k))^{-1} (i(k) - I(k)Y^-(k | k)\dot{y}(k | k)) \right]
\]

\[
+ I^+(k) \left[ F_k (Y(k | k) + I(k))^{-1} (\dot{y}(k | k) + i(k)) + B_k \pi(k) \right]
\]

(318)

where

\[
M_k^Y = F_k^T Y(k | k) F_{k-}
\]

\[
M_k^I = F_k^T I(k) F_{k-}
\]

(319)

are the predictions of the information matrix and the observation information respectively from time \( k - 1 \) to time \( k \), \( Y^-(k + 1 | k) \) represents the state at time \( k \) without the delayed data and \( Y^+(k + 1 | k) \) represents the updated state at time \( k \) which includes the delayed data.

### 5.2.9 Information Observation and Update

Finally, the information associated with an observation taken at time \( t_k \) is written in the form

\[
i(k) \triangleq H^T(k)R^{-1}(k)z(k) \quad I(k) \triangleq H^T(k)R^{-1}(k)H(k)
\]

(320)

The update stage of the information filter is associative and is given by

\[
\dot{y}(k | k) = \dot{y}(k | k - 1) + i(k)
\]

\[
Y(k | k) = Y(k | k - 1) + I(k)
\]

(321)
5.3 Delays in Channels and Burst Communication

The ability to propagate information forward in time (Equations 312 and 313), the ability to propagate information backward in time (Equations 315 and 316), and the ability to fuse two pieces of information from two different times in a consistent manner (Equation 317) provides all the requisite tools for dealing with delayed and also burst communication.

5.3.1 Node Algorithms

The essential equations can easily be extended to cases of multi-step delays. Consider the information matrix $Y(k | k)$ obtained locally at a node at time $k$. Following Equation 312, this information matrix can be propagated forward in time $n$ steps according to

$$Y(k + n | k) = M_n(k) - M_n(k)G_n(k)\Sigma_n^{-1}(k)G_n^T(k)M_n(k)$$  \hspace{1cm} (322)

where

$$M_n(k) = [F^{-T}(k)]^n Y(k | k) [F^{-1}(k)]^n,$$

$$\Sigma_n(k) = G_n^T(k)M_n(k)G_n(k) + Q_n^{-1}(k)$$

and where the subscript $n$ denotes that the associated transition matrices are evaluated from the integrals in Equation 302 over the interval $n\Delta T$. Likewise, from Equation 313 the information-state vector can be propagated forward according to

$$\hat{y}(k + n | k) = [1 - M_n(k)G_n(k)\Sigma_n^{-1}(k)G_n^T(k)] [F^{-T}(k)]^n \hat{y}(k | k)$$

$$+ Y(k + n | k)B_n(k)u_n(k).$$  \hspace{1cm} (323)

An equivalent expression can be obtained for propagating information backwards in time as

$$Y(k - n | k) = M_{-n}(k) + M_{-n}(k)G_{-n}(k)\Sigma_{-n}^{-1}(k)G_{-n}^T(k)M_{-n}(k)$$  \hspace{1cm} (324)

where

$$M_{-n}(k) = [F^T(k)]^n Y(k | k) [F(k)]^n,$$

$$\Sigma_{-n}(k) = Q_n^{-1}(k) - G_{-n}^T(k)M_{-n}(k)G_{-n}(k)$$

and where the subscript $-n$ denotes that the associated transition matrices are evaluated from the integrals in Equation 302 over the interval $-n\Delta T$. Similarly, the information-state vector can be propagated backward according to

$$\hat{y}(k - n | k) = [1 - M_{-n}(k)G_{-n}(k)\Sigma_{-n}^{-1}(k)G_{-n}^T(k)] [F^T(k)]^n \hat{y}(k | k)$$

$$+ Y(k - n | k)B_{-n}(k)u_n(k).$$  \hspace{1cm} (325)

The delayed data problem can now be solved using the following algorithm:

1. Back-propagate the estimate $Y(k | k)$ to the time, $k - n$, at which the information $I(k - n)$ was obtained, using Equations 324 and 325.
2. Add the delayed data $I(k-n)$ to the estimate $Y(k-n | k)$ in the normal manner.

3. Propagate the new fused estimate back to time $k$ using Equations 322 and 323.

It can be seen (Equation 317) that the net effect of this algorithm is to produce an estimate in the form

$$Y(k | k) = Y_Y(k | k) + Y_I(k | k) + Y_{IY}(k | k)$$

where $Y_Y(k | k)$ is the estimate obtained without the delayed information, $Y_I(k | k)$ is the estimate obtained using only the delayed information, and $Y_{IY}(k | k)$ is a cross-information term (uniquely defined $Y_Y(k | k)$ and $Y_I(k | k)$) describing the cross-correlation between the information states caused by propagation through a common process model. It should be noted that $Y_{IY}(k | k)$ is additive, so the obvious approximation

$$Y(k | k) = Y_Y(k | k) + Y_I(k | k)$$

is conservative.

5.3.2 Channel Algorithms

The channel filter is a conventional 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 $\hat{y}_{ij}(k | k)$ and the common information matrix $Y_{ij}(k | k)$.

The prediction equations (for both forward and backward propagation) used in the channel filter are the same as Equations 322–325 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 $j$

$$m_j(k) = \hat{y}_{j}(k | k-n) - \hat{y}_{ij}(k | k-m)$$

$$M_j(k) = Y_j(k | k-n) - Y_{ij}(k | k-m)$$

(326)

Note now that the time indices $Y_j(k | k-n)$ and $Y_{ij}(k | k-m)$ are different as the different nodes are asynchronous and the two information sets can be predicted over different time horizons. This information gain $m(k)$ and $M(k)$ is analogous to the observation information vector $i(k)$ and $I(k)$, 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

$$\hat{y}_{ij}(k | k) = \hat{y}_{ij}(k | k-m) + m_j(k)$$

$$Y_{ij}(k | k) = Y_{ij}(k | k-m) + M_j(k)$$

(327)
This can be further simplified by substituting Equation 326

\[
\hat{y}_{ij}(k \mid k) = \hat{y}_{ij}(k \mid k - m) + m_j(k)
\]
\[
\hat{y}_{ij}(k \mid k) = \hat{y}_{ij}(k \mid k - m) + \hat{y}_j(k \mid k - n) - \hat{y}_{ij}(k \mid k - m)
\]
\[
\hat{y}_{ij}(k \mid k) = \hat{y}_j(k \mid k - n)
\] (328)

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

\[
\hat{y}_{ij}(k \mid k) = \hat{y}_j(k \mid k - n)
\]
\[
Y_{ij}(k \mid k) = Y_j(k \mid k - n)
\] (329)

Alternatively, if the new information at the channel filter of node \(i\) is from the local filter at node \(i\), the update becomes

\[
\hat{y}_{ij}(k \mid k) = \hat{y}_i(k \mid k)
\]
\[
Y_{ij}(k \mid k) = Y_i(k \mid k)
\] (330)

While the normal information filter update of Equation 327 is perfectly valid, implementation in the form of Equation 329 and 330 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. 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.

### 5.4 The Communication Channel Algorithm

This section provides a practical algorithm for the general implementation of a communication channel.

#### 5.4.1 Management of Communication

All communication algorithms should be ‘node’ rather than ‘network’ centric. This is because, locally, the structure of the network can change dynamically whereas the operation of a node can be considered fixed and known. Thus a node is considered as an ‘isolated’ entity which engages in communication intermittently and opportunistically rather than on a fixed basis. This view also firmly fixes a modular node-centric model of system structure. This is shown diagramatically in Figure 41. Broadly, the node generates information measures \(\hat{y}_i(k \mid k)\) 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 \(\hat{y}_i(k \mid k - 1)\) given all local and communicated data up to time \(k - 1\).
At this time, local observations produce local information measures $i(k)$ on the basis of local observations. The prediction and local information measures are combined, by simple addition, into a total local information measure $\hat{y}_i(k | k)$ 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 $\hat{y}_{ji}(k | k)$ 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 $\hat{y}_i(k | k)$.

The communication channels exploit the associativity property of information measures. The channels take the total local information $\hat{y}_i(k | k)$ and subtract out all information that has previously been communicated down the channel, $\hat{y}_{ij}(k | k)$, 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.

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.

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,

Figure 41: Algorithmic structure of a decentralised sensing node.
dynamic network changes and signal jamming).

The remainder of this section details a practical algorithm for the implementation of a channel.

### 5.4.2 Structure of the Communication Algorithm

![Diagram of communication algorithm](image)

**Figure 42:** Structure of the general communication channel algorithm: (a) Initial state of the node $p$ prior to establishing communication with node $q$; (b) establishing common information between nodes $p$ and $q$ at a time $k$ in a channel $pq$; (c) computation of local, partial estimates at the two nodes; (d) calculation of information gains at channel inputs and subsequent updating of local estimates.

Consider a node $p$ and its neighbourhood $N(p)$ of nodes shown in Figure 42. Node $p$ has a local information estimate $Y_p(k-1 \mid k-1)$ at time $k - 1$. The node has a number of communication channels open with its neighbourhood $j \in N(p)$. Node $p$ now decides to also communicate with node $q$ through a new channel $pq$. It is not generally known a priori if $p$ and $q$ have previously communicated or indeed have been in indirect communication with each other through another branch of the network. The first step in the general communication algorithm is establish what information node $p$ and $q$ have in common. This is done in three stages:

1. A separate point-to-point channel, labeled $pq$, is established between the two nodes. The current estimates at the two nodes $Y_p(k - 1 \mid k - 1)$ and $Y_q(k - 1 \mid k - 1)$ are placed in the channel.
2. The two estimates are then time-aligned using the information state prediction or retrodiction Equations 312-316 described in Section 5.2. The time alignment is normally taken to some time $k$ at which the next available observation information is to occur. This yields process yields two predictions $Y_p(k | k - 1)$ and $Y_q(k | k - 1)$.

3. The common information $Y_{p\cap q}(k | k - 1)$ between nodes $p$ and $q$ is then determined. This information may be computed by any of the methods described in Section 5.4.4. The result of this process is a channel between the two nodes with a measure of the common information $Y_{p\cap q}(k | k - 1)$ between them at a time $k$. Similar measures $Y_{p\cap j}(k | k - 1)$ are computed for every connected node $j \in N(p)$ in the neighbourhood of $p$. This is shown diagramatically in Figure 42(b).

The next step is to communicate new information or information gain between node $p$ and its neighbours. As a first step, it is assumed that node $p$ acquires some local observation information $I_p(k)$ at time $k$ and fuses this with its local prediction $Y_p(k | k - 1)$ to produce a local, partial, estimate $\tilde{Y}_p(k | k)$ at time $k$ as

$$\tilde{Y}_p(k | k) = Y_p(k | k - 1) + I_p(k). \quad (331)$$

Other nodes in the neighbourhood may also produce a similar estimate using observation data acquired since the last communication or otherwise based only on the prediction (so $\tilde{Y}_p(k | k) = Y_p(k | k - 1)$). This is shown diagramatically in Figure 42(c). It is important here to reiterate the node-centric form of the algorithm: The node time-aligns common information in a single point-to-point communication channel. The channels themselves may well act asynchronously with each other. The different nodes in the network will certainly act asynchronously.

The final step is to compute the information gain from the channel following communication. The most robust method of doing this is to communicate the total information $Y_q(k | k)$ from a node $q$ to node $p$ and then to subtract the common information $Y_{p\cap q}(k | k - 1)$ in the input channel of node $p$. This allows for the possibility that communication may be halted or changed part way through a message: By performing the information gain calculation at the input stage, there is no need to guarantee a communication until the message actually arrives. The resulting computation performed in the incoming channel is simply

$$\Delta Y_{pq}(k | k) = Y_q(k | k) - Y_{p\cap q}(k | k - 1). \quad (332)$$

Finally, the local estimate is updated by simply adding information gains coming from each channel as

$$Y_p(k | k) = \tilde{Y}_p(k | k) + \sum_{j \in N(p)} \Delta Y_j(k | k). \quad (333)$$

This calculation is may be performed asynchronously at each node and with respect to each channel at a node. This update is shown diagramatically in Figure 42(d).
5.4.3 Information Flow

Qualitatively, the effect of the algorithm is to pass all information between two nodes, subtracting out only the information considered to be common. Substituting Equation 331 into Equation 332 to obtain

\[ \Delta Y_{pq}(k \mid k) = I_q(k) + Y_q(k \mid k-1) - Y_{p\cap q}(k \mid k-1) \]  

(334)

shows that local measurement information is always communicated in its entirety. Further, the term \( Y_q(k \mid k-1) - Y_{p\cap q}(k \mid k-1) \) captures the additional information at node \( q \) not already obtained at (or in common with) node \( p \). This may include observations previously obtained by at node \( q \) but not yet communicated, or information communicated to \( q \) by other nodes in the network neighbourhood \( N(q) \). In the latter case, the effect of this term is to propagate information through the network, from node to node, at each stage being incorporated into a single information gain.

5.4.4 Computing the Common Information

The simplest method of computing common information is to assume that the two nodes have all prior information in common so \( Y_{p\cap q}(k \mid k-1) = Y_p(k \mid k-1) = Y_q(k \mid k-1) \). With this substituted in Equation 334, the information gain is simply the locally observed information \( \Delta Y_{pq}(k \mid k) = I_q(k) \). This algorithm will be called the “all-common” communication method and would be used in a fully connected or broadcast structure.

If the channel \( pq \) is subject to delay or intermittent availability, the all-common algorithm can be easily extended to communicate all information observed as a single ‘burst’ communication. In this case, the common prior \( Y_{p\cap q}(k \mid k-n) \) is set to the common estimate \( Y_p(k-n \mid k-n) \) at the last communication time \( k-n \), appropriately predicted forward to the new communication time \( k \). The local estimates \( Y_q(k \mid k) \) computed in Equation 331 now include all observation information \( I_q(k-j), j = n, \ldots, 0 \) taken over the interval \( k-n \) to \( k \) (and again, appropriately predicted through to time \( k \)). The computed information gain \( \Delta Y_{pq}(k \mid k) \) obtained from Equation 332 now consists of all this observation information with the last common prior \( Y_p(k-n \mid k-n) \) properly subtracted. It should be clear that no matter what the time delay \( n \) or the number of observations made, only a single information gain \( \Delta Y_{pq}(k \mid k) \) is computed and communicated. Fundamentally, this is what a burst communication consists of. However, it is also worth noting that the time propagation of observation information terms \( I_q(k-n) \) over a time \( n \) generally results in a net loss of information through the addition of the process uncertainty accumulated over the delay time.
6 Management in Decentralised Systems

Managing sensor systems involves making decisions and taking actions under uncertainty. In decentralised data fusion systems, three types of decisions must be made:

1. Sensor Management: When and what to sense;
2. Communications Management: When and what to communicate;
3. Organisation: How should the network be structured and controlled.

These decisions are associated with the evaluation, acquisition and exchange of information. Consequently, they all derive from very similar principles. Further, they follow quite naturally from the ideas of information already developed in decentralised data fusion systems.

In this course, Bayesian decision theory is used as the underlying basis for these decision making problems. Bayesian decision theory provides a comprehensive foundation for making decisions under uncertainty ([10] is a definitive if challenging introduction to Bayesian decision theory). Section 2 and subsequent discussions has also shown that Bayesian methods also provide a powerful means of describing information. Thus Bayesian decision theory is a most appropriate methodology for studying management problems in decentralised sensor networks. Section 6.1 provides a brief and necessary introduction to Bayesian Decision theory and particularly its application to data fusion problems. Section 6.3 describes the extension of single-decision maker ideas to multiple decision makers. The extension of single-person decision making to multiple-person decision making is not trivial, particularly if the principles of fully decentralised decision making are to be adhered to.

Together Sections 6.1 and 6.3 cover the underlying theory necessary for the study of sensor management problems. Section 6.4 then describes the application of the decision-theoretic approach to the problem of sensor management. Sensor management focuses on issues such as sensor cuing, sensor-to-target assignment, hand-off and emission management. The focus is on single sensor abilities to obtain information and to share this information with other sensors without other constraint. The sensor management problem admits some simple and direct solutions in decentralised form and has consequently seen the most comprehensive development.

Section 6.5 considers the communication management problem. This focuses on problems imposed by limited communication bandwidth and limited achievable communication topologies in decentralised sensor networks. In particular, the problems are concerned with who to communicate with, when to communicate, and what to communicate within some established constraints. Although less work has been done in this area than for the sensor management problem, the ideas of information embodied in decentralised data fusion algorithms naturally lead to a well-formed group of “information maximisation” communication algorithms.

Section 6.6 considers the problem of organisation in decentralised networks. This focuses on specific issues in establishing communication topologies and also in broader
issues of sensor-to-platform assignment and platform trajectory optimisation. This is the least developed area of the decentralised management problem and is an area of current and future research. It involves much broader ideas of optimal control in distributed systems. However, it also admits useful “information maximisation” solutions which, while not optimal, give useful and quantifiable organisation algorithms.

6.1 Bayesian Decision Theory

This section briefly summarises the main elements of normative Bayesian decision theory as it applies to sensor and sensor network management.

6.1.1 Structure of the Problem

Decision making begins with an unknown state \( \mathbf{x} \in \mathcal{X} \). The state may be either continuous or discrete and may indeed constitute a trajectory in time. Knowledge about the state is assumed to be encoded in a probability density function in the form of a prior \( P(\mathbf{x}) \) or as a posterior \( P(\mathbf{x} | Z^k) \).

Given knowledge of the state \( \mathbf{x} \), some action \( \mathbf{a} \in \mathcal{A} \) is to be taken. The action may be to determine the true value of \( \mathbf{x} \) (an estimation problem), to take some more general control action, or indeed to decide to take a further observation. The action may be single, or it may consist of some time sequence of actions \( \mathbf{a} = \{a_1, a_2, \cdots, a_n\} \).

The action \( \mathbf{a} \) is selected with respect to a utility function \( U(\mathbf{a}, \mathbf{x}) \) or equivalently\(^{12}\) a loss function \( L(\mathbf{a}, \mathbf{x}) \). Utility or loss functions are real-valued functions which assign a unique value (respectively a loss) incurred if the action \( \mathbf{a} \) is taken when the true value of the state is \( \mathbf{x} \).

A utility or loss function has the effect of placing a preference ordering on possible actions. Simply an optimal action is one that is most preferred. The choice of optimal action is encoded in a decision rule \( \delta \). A decision rule is a function taking knowledge of the state and computing an action as \( \mathbf{a} = \delta_p(\mathbf{x}) \). Practically, a decision rule is a function of the probabilistic knowledge on the state, \( P(\mathbf{x}) \), rather than the actual state \( \mathbf{x} \) (which is not normally known with precision). However, it is traditional to blur this distinction in notation and talk about a decision \( \delta_p(\mathbf{x}) \).

Knowledge of the state \( P(\mathbf{x}) \), the set of actions \( \mathbf{a} \), the utility (or loss) function \( U(\mathbf{a}, \mathbf{x}) \), and specification of the decision rule \( \delta \) constitute all the required elements of the decision making and management problem.

\(^{12}\)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.
6.1.2 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;

$$L : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}, \quad U : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}.$$ 

The interpretation of a loss function $L(x, a)$ is that $L$ is the loss incurred in taking the action $a$ when the true state of nature is $x$. Similarly, the interpretation of a utility function $U(x, a)$ is that $U$ is the gain obtained in taking the action $a$ when the true state of nature is $x$. In principle, it should be the case that for a fixed state of nature, both utility and loss will induce a preference ordering on $\mathcal{A}$. To ensure this, loss and utility functions must obey three 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 $x \in \mathcal{X}$:

1. Given any $a_1, a_2 \in \mathcal{A}$, either $U(x, a_1) < U(x, a_2), U(x, a_1) = U(x, a_2)$, or $U(x, a_1) > U(x, a_2)$. That is, given any two actions we can assign real numbers which indicate our preferred alternative.

2. If $U(x, a_1) < U(x, a_2)$ and $U(x, a_2) < U(x, a_3)$ then $U(x, a_1) < U(x, a_3)$ and $U(x, a_3) < U(x, a_1)$. That is, if we prefer action $a_2$ to $a_1$ and action $a_3$ to $a_2$, then we must prefer action $a_3$ to action $a_1$; the preference ordering is transitive.

3. If $U(x, a_1) < U(x, a_2)$, then $\alpha U(x, a_1) + (1 - \alpha)U(x, a_3) < \alpha U(x, a_2) + (1 - \alpha)U(x, a_3)$, for any $0 < \alpha < 1$. That is, if $a_2$ is preferred to $a_1$ then, in a choice between two random situations which are identical except that both $a_1$ and $a_2$ occur with probability $\alpha$, the situation involving $a_2$ will be preferred.

A proof that these axioms imply the existence of a utility function can be found in [10].

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.

6.1.3 Expected Utility or Loss

Except in case of perfect knowledge, a utility function in the form of $U(x, a)$ is not very useful because the true state of nature $x \in \mathcal{X}$ will not be known with precision and so the true utility gained in taking an action will not be known. Rather, there will be a probability distribution $P(x)$ summarizing all the (probabilistic) information we have about the state at the time of decision making. With this information, one natural
method of defining utility is as an expected utility (or Bayes expected utility) which for continuous-valued state variables is simply

\[
\beta(a) \triangleq \mathbb{E}\{U(x, a)\} = \int_{-\infty}^{\infty} U(x, a) P(x) dx,
\]

and for discrete-valued state variables is given by

\[
\beta(a) \triangleq \mathbb{E}\{U(x, a)\} = \sum_{x \in \mathcal{X}} U(x, a) P(x).
\]

Clearly, Bayes expected utility simply weights the utility gained by the probability of occurrence (an average utility). More pessimistic approaches could also be used. For example, we could ask what possible value of \( x \) could would incur the “smallest utility gain”. This approach results in a “minmax” or game-theoretic decision algorithm.

### 6.1.4 Bayes Actions

It is most likely that probabilistic information concerning \( x \) is obtained after taking a number of observations, in the form of a posterior distribution \( P(x \mid Z^n) \). An *expected utility* (or loss) \( \beta \) following an action \( a \) may then be defined with respect to a specified posterior distribution in the true state, as

\[
\beta_P(x \mid Z^n)(a) \triangleq \mathbb{E}_P(x \mid Z^n)\{U(x, a)\} = \int_x U(x, a) P(x \mid Z^n) dx.
\]

The Bayes action \( a^* \) is defined as the strategy which maximises the posterior expected utility

\[
a^* = \arg \max_a \beta_P(x \mid Z^n)(a).
\]

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 (\( A = \mathcal{X} \)). In particular, the MMSE estimate defined by

\[
\hat{x} = \arg \min_{a \in \mathcal{X}} \int_x (x - a)^T(x - a) P(x \mid Z^n) dx
\]

is clearly a Bayes action with respect to a squared error loss function defined by \( L(x, \hat{x}) = (x - \hat{x})^T(x - \hat{x}) \).

### 6.2 The Utility of Information

For the data fusion problems under consideration in this course, the goal is almost always the maximisation of information available to the system. As was demonstrated in Section
2.3, the Shannon and Fisher measures provide a natural concept of information in terms of the compactness of probability density functions. Further, these information measures are made available directly in the decentralised data fusion algorithms developed in this course. It is therefore logical to establish a relationship between the utility of sensor decision making and these information quantities.

In the data fusion arena, the action set \( A = \{a_1, a_2, \ldots, a_n\} \) corresponds to the different sensing configurations or possible sensing strategies available to the system. In general, the effect of each possible action \( a_i \) is to induce a posterior probability distribution \( P_i(x) \) on the state \( x \). A probabilistic outcome \( \rho_i \) of an action \( a_i \) is defined simply as the posterior pdf for each possible sensing action

\[
\rho_i = P_i(x \mid a_i), \quad i = 1, \ldots, n
\]  

(340)

Utility theory can now be used to encode the preferential structure of a sensor faced with several choices of sensing action in the form of an ordered set

\[
\langle \rho_1, \rho_2, \ldots, \rho_m \rangle, \quad \text{such that } \rho_1 \succeq \rho_2 \succeq \cdots \succeq \rho_m,
\]  

(341)

where \( \rho_1 \) is the most preferred outcome. Decisions are now made through a maximization of the expected utilities for all possible actions. Following Equation 338, the optimal action \( a^* \) is defined through

\[
a^* = \arg \max_a \beta(\rho) = \arg \max_a E\{U(x, a)\}. 
\]

(342)

It remains to identify appropriate utility functions for the sensing process.

---

Figure 43: Structure of the sensor management function.
6.2.1 Information as Expected Utility

An appropriate definition of utility turns out to be the log likelihood

\[ U(x, a) = \log P(x | a) \] (343)

or for each action

\[ U(x, a_i) = \log P(x | a_i) = \log \rho_i, \quad i = 1, \ldots, n. \] (344)

The log-likelihood clearly satisfies the utility axioms in providing a transitive, convex, preference structure as

\[ \log P(\alpha x_1 + (1 - \alpha) x_2) \geq \alpha \log P(x_1) + (1 - \alpha) \log P(x_2) \]
\[ U(\alpha x_1 + (1 - \alpha) x_2) \geq \alpha U(x_1) + (1 - \alpha) U(x_2), \] (345)

for \(0 \leq \alpha \leq 1\). So,

\[ U(E^\rho \{x\}, a) \geq E^\rho \{U(x, a)\}. \] (346)

A utility function satisfying this convex inequality is called risk-averse. Risk-averse behaviour takes into account the probabilities associated with acquiring information, whereas risk-prone behaviour only considers the value of the information which could be gained irrespective of the probability of acquiring it.

Taking expected values of Equation 343 gives

\[ E^\rho \{U(x, a)\} = E^\rho \{\log P(x | a)\} = -H_\rho(x | a) \] (347)

or for each action

\[ E^\rho_i \{U(x, a_i)\} = E^\rho_i \{\log P(x | a_i)\} = -H_{\rho_i}(x | a_i), \quad i = 1, \ldots, n \] (348)

where the dependence on the specific action \(a_i\) is made explicit. The efficacy of the choice of utility as the log-likelihood is now clear: The expected utility is simply the negative entropy or Shannon Information.

The Bayes action defined in Equation 338 now has a natural interpretation as the action which maximises information. This then allows all the information-theoretic tools developed in Section 2.3 to be brought to bear on the sensor and communication management problems.
6.2.2 Information Filter Metrics

In the information filter, Information is quantified through the information matrix $Y(i \mid j)$ or inverse covariance $P^{-1}(i \mid j)$. It follows from Equation 51, that the posterior information following the $i^{th}$ sensing action at time $k$ is given by

$$I(k,i) \triangleq \frac{1}{2} \log \left[ (2\pi e)^{n} | Y_{i}(k \mid k) | \right]$$

$$= \frac{1}{2} \left[ (2\pi e)^{n} | Y(k \mid k - 1) + I_{i}(k) | \right].$$  \hspace{1cm} (349)

The possible information matrices, $I_{i}(k)$, result directly from the geometry of the sensing process as is described in Example 22.

A second related measure is the mutual information gain from a sensing action. For the information filter (from the definition in Equation 53), this is given by

$$I(k,i) = \frac{1}{2} \log \left[ \frac{| Y(k \mid k - 1) + I_{i}(k) |}{| Y(k \mid k - 1) |} \right]$$  \hspace{1cm} (350)

An example of the use of this information measure, in the context of sensor management, will be provided in Example 28.

These two information measures can be generalised and used in all three areas of sensor decision making. In sensor management, Equations 349 and 350 can be used to decide on the assignment of sensors to targets and in sensor hand-off and cuing. In communications, the same equations can be used to determine which information it is most informative to communicate.

6.2.3 Discrete State Metrics

For discrete states, the Shannon information is used directly. The posterior information about a set of states $x = \{x_1, x_2, \cdots \}$ is simply given by

$$I(k) \triangleq \mathbb{E}\{\log P(x \mid Z^k)\} = \sum_{i} P(x_i \mid Z^k) \log \left[ P(x_i \mid Z^k) \right],$$  \hspace{1cm} (351)

The mutual information (from Equation 53) is

$$I(k) = \mathbb{E}\left\{ \log \left[ \frac{P(z(k) \mid x)}{P(z(k))} \right] \right\} = \sum_{i} \frac{P(z(k) \mid x)}{P(z(k))} \log \left[ \frac{P(z(k) \mid x)}{P(z(k))} \right]$$

$$= \sum_{i} \alpha \Lambda_{z}(x_i) \log \left[ \alpha \Lambda_{z}(x_i) \right].$$ \hspace{1cm} (352)

As with the information filter metrics, the discrete metrics in Equations 351 and 352 can be used for the purpose of both sensor management and communication optimisation.
6.2.4 Composite Metrics

In multi-objective optimisation problems, simultaneous localisation and identification of targets for example, composite information utility metrics can be employed. This is normally done in a simple linear combination, for example;

\[ I_{\text{composite}}(x) = C_1 I_{\text{identity}}(x) + C_2 I_{\text{location}}(x), \]

where \( C_1 \) and \( C_2 \) are arbitrary weights.

6.3 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. This is generally the case for the decentralised data fusion problems studied here in which utility is defined globally in terms of information.

6.3.1 The Super Bayesian

The simplest solution to the multiple-person decision making problem is to assume that everyone can send information to a single decision maker or "Super Bayesian". The task of this decision maker is, in the first instance, to combine probabilistic information from all sources and then to make decisions based on the global posterior \( P(x \mid Z^k) \) in the form

\[ a^* = \arg \max_a \beta(P(x \mid Z^k), a) \]

\[ = \arg \max_a \mathbb{E}\{U(x, a) \mid Z^k\}, \quad (353) \]

where \( U(x, a) \) is a single group utility function. The solution in this case is well defined in terms of classical Bayesian analysis. However, this “Super Bayesian” approach is generally not applicable or appropriate in decentralised data fusion systems where no single decision maker is generally permitted.

6.3.2 Multiple Bayesians

Consider a system consisting of local Bayesians each able to obtain probabilistic information which they can shares with all other Bayesians. Locally, each Bayesian can formulate
a posterior distribution \( P(x_i \mid Z^k) \) and from this select a Bayes action with respect to a local utility \( U_i(x, a) \). Acting locally, a Bayesian would choose the action which maximises local expected utility:

\[
\begin{align*}
a^*_i &= \arg \max_a \beta(P(x_i \mid Z^k), a) \\
&= \arg \max_a E\{U_i(x, a) \mid i^k_Z\}. \quad (354)
\end{align*}
\]

The essential problem here is that each decision maker may end up choosing a different action. There is no guarantee that the resulting set of actions will be optimal in a global sense or indeed that the actions do not interfere with each other.

A general solution to these problems presents a considerable challenge. The trivial case exists where the optimal action \( a^*_i \) at each Bayesian \( i \) happens to be the same for all the Bayesians and thus becomes the group action. However, in general these local decisions will not the same.

In the general case, there are two generally recognised solutions to this problem. The first is to optimize a weighted sum of local utilities in the form

\[
a^* = \arg \max_a \sum_j w_j [E\{U_j(x, a)\} - c(j)] \quad (355)
\]

where \( 0 \leq w_j \leq 1 \) and \( \sum_j w_j = 1 \). The value \( c(j) \) is known as decision maker \( j \)’s “security level”. Equation 355 is known as a linear decision pool. A second solution is to optimize a product of local utilities in the form

\[
a^* = \arg \max_a \prod_j [E\{U_j(x, a)\} - c(j)], \quad (356)
\]

This is known as the Nash product [43]. In an information context, the linear decision pool is most appropriate as the global information will be the sum of locally available information.

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 smaller 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 which are not described further here.

### 6.3.3 Practical Bargaining Solutions

To apply Equation 356 or Equation 355 to compute an optimal action requires information about every sensors utilities for each possible group action. In a decentralised system, this would require each sensor node to acquire all other sensor nodes expected utilities. In general this would require each node to communicate all it’s expected utilities to all other nodes. This is clearly not practical.

However an approximate, and iteratively optimal solution, to this problem can also be obtained Suppose each sensor, by considering each \( a_i \in A \), computes the following
ordered set i.e.

\[ \langle \beta_i(a^{i1}), \beta_i(a^{i2}), \ldots, \beta_i(a^{ir}) \rangle, \]

where \( \beta_i(a^{i1}) \geq \beta_i(a^{i2}) \geq \cdots \geq \beta_i(a^{ir}) \),

such that \( a^{i1} \) is sensor \( i \)'s most preferred action and \( a^{ir} \) is \( i \)'s least preferred action. The communication required can be reduced if each sensor \( i \) communicates only a subset of the expected utility set corresponding to the “more” preferred actions. For example, such a subset can be chosen by considering only those actions which result in expected utilities greater than or equal to \( i \)'s current expected utility i.e.

\[ \{ \beta_i(a_i) : \beta_i(a_i) \geq c(i) \}, \]

where \( c(i) \) is sensor \( i \)'s current utility at time of decision-making. Another possibility is to consider only expected utilities above a given threshold value. The effect of this is to reduce the size of the expected utility set considered and hence the number of possible actions to be considered.

An iterative bargaining algorithm can be stated as follows:

1. **Order preferences.** Each sensor \( i \) computes its ordered expected utility set and, by implication, the corresponding ordered preferred action set i.e.

\[ \langle \beta_i(a^{i1}), \ldots, \beta_i(a^{ir}) \rangle, \Rightarrow \langle a^{i1}, \ldots, a^{ir} \rangle. \]

2. **Communicate first preferences.** Each sensor \( i \) communicates its most preferred action \( a^{i1} \) and the corresponding expected utility \( \beta_i(a^{i1}) \) to other sensors. In the event that there are two actions with an identical expected utility then both the actions are communicated.

3. **Compare first preferences.** Each sensor \( i \) then compares its own first preference \( a^{i1} \) with other received first preferences \( a^{j1}, \forall j \in \mathcal{N} \), if all refer to the same action then \( \hat{a} = a^{i1} \). In practice this will be unlikely except for very simple management problems. If the communicated preferences do not refer to the same action, then each sensor \( i \) must communicate its expected utility corresponding to each received preferred action \( a^{j1}, \forall j \in \mathcal{N} \).

4. **Maximize on first preferences.** Each sensor is then able to compute the group expected utility

\[ B_c(a^{j1}) = \beta_1(a^{j1}) + \beta_2(a^{j1}) + \cdots + \beta_N(a^{j1}), \]

corresponding to the most preferred action \( a^{j1} \) for each sensor \( j \): \( \forall j \in \mathcal{N} \). From this, each sensor can find the most optimal solution based on all the sensors’ first preferences by maximizing over the group expected utilities corresponding to each sensors’ first preference actions i.e.

\[ \arg \max_{a^{j1}} B_c(a^{j1}). \]

This completes the first iteration of the process.

5. **Repeat and maximize on subsequent preferences.** i.e. next communicate and compute \( B_c(a^{j2}), \forall j \in \mathcal{N} \) and so on for subsequent preferences.

The number of iterations required in the bargaining depends on the “degree of optimality” required in the actions taken.
6.4 Decentralised Sensor Management

Sensor management is generally considered to encompass three main decisions:

- **Sensor-Target Assignment.** It is important to manage and coordinate the assignment of sensors to targets in order to use sensing resources effectively. This also ensures that all the targets that need to be observed are covered in a manner which is consistent with system goals. To be most effective, the management of sensor-target assignments must take into account the dynamic nature of the problem, such as in the case of moving targets or moving sensor platforms, by continually reviewing current assignments.

- **Sensor Cueing and Hand-off.** When using sensors with limited fields of view it is important to ensure that targets which may pass out of view are not lost. Hence, it may become necessary to cue sensors into whose field of view a target may be entering. Cueing may be done in a cooperative manner e.g. when sensors, capable of obtaining different information, cooperate to resolve ambiguity concerning a particular target. Hand-off refers to the transfer of the observation of a target by one sensor to another. As expected, sensor cueing and hand-off should be consistent with system goals.

- **Mode management.** Often sensors are capable of operating in several modes, therefore, it becomes necessary to make decisions concerning the most appropriate mode for a given situation. This is similar to the situation where there are several physically diverse sensors available. In such cases it becomes necessary to manage such diversity i.e. make decisions regarding the appropriate sensor or sensor mode for a particular observation.

6.4.1 Sensor to Target Assignment

The application of the methods described in Section 6.3 to the sensor to target assignment problem is now described through a number of examples.

**Example 26**

Consider the sensing scenario depicted in Figure 44. Each of three (decentralised) sensors are able to observe three possible targets \( T = \{t_1, t_2, t_3\} \). Thus, all possible actions are constrained by a 1-1 sensor-target assignment. In this case, the action set is defined by the various possible sensor-target assignments. These can, for example, be

\[
\mathcal{A} = \begin{cases} 
    a_1 = (t_1 \rightarrow j; \ t_2 \rightarrow i; \ t_3 \rightarrow k), \\
    a_2 = (t_1 \rightarrow i; \ t_2 \rightarrow k; \ t_3 \rightarrow j), \\
    a_3 = (t_1 \rightarrow k; \ t_2 \rightarrow i; \ t_3 \rightarrow j), \\
    a_4 = (t_1 \rightarrow j; \ t_2 \rightarrow k; \ t_3 \rightarrow i), \\
    \vdots
\end{cases}.
\]  

Some of these sensor-target assignments are illustrated in Figure 44.
Figure 44: Some alternate sensor to target assignments for the problem described in Example 26.
Using the action-outcome association suggested by Equation 361, values are assigned to each of the utilities for each of the sensors \(i\), \(j\) and \(k\). For ease of illustration, scaled values of typical expected utilities are used:

\[
\begin{align*}
&\mathcal{B}_c \\
a_1 : & \quad t_1 \overset{20}{\rightarrow} i; \quad t_2 \overset{16}{\rightarrow} j; \quad t_3 \overset{18}{\rightarrow} k; \quad 54 \\
a_2 : & \quad t_1 \overset{20}{\rightarrow} i; \quad t_2 \overset{14}{\rightarrow} k; \quad t_3 \overset{15}{\rightarrow} j; \quad 49 \\
a_3 : & \quad t_1 \overset{10}{\rightarrow} k; \quad t_2 \overset{16}{\rightarrow} j; \quad t_3 \overset{33}{\rightarrow} i; \quad 59 \\
a_4 : & \quad t_1 \overset{10}{\rightarrow} k; \quad t_2 \overset{22}{\rightarrow} i; \quad t_3 \overset{15}{\rightarrow} j; \quad 47 \\
a_5 : & \quad t_1 \overset{10}{\rightarrow} j; \quad t_2 \overset{22}{\rightarrow} i; \quad t_3 \overset{18}{\rightarrow} k; \quad 50 \\
a_6 : & \quad t_1 \overset{10}{\rightarrow} j; \quad t_2 \overset{14}{\rightarrow} k; \quad t_3 \overset{33}{\rightarrow} i; \quad 57
\end{align*}
\]  

(362)

where \(\mathcal{B}_c\) is the group expected utilities obtained from Equation 355 with a security level of zero \(c(j) = 0\).

In Equation 362, sensor node \(i\) has its highest local utility when it is assigned to target \(t_3\). This will occur if either the group action \(a_3\) or \(a_6\) is taken. In contrast, sensor node \(j\) has its highest local utility when it is assigned to target \(t_2\), which will occur in the group actions \(a_1\) or \(a_3\) is taken. However, there is a degree of conflict in local decision making as sensor node \(k\) also has its highest local utility when it is assigned to target \(t_3\); thus simply selecting locally maximising actions will not be optimal as target \(t_1\) would go unobserved.

In total, the the ordered action preferences for each sensor node are

\[
\begin{align*}
i & : \quad \langle (a_3, a_6), (a_4, a_5), (a_1, a_2) \rangle \\
j & : \quad \langle (a_3, a_1), (a_4, a_2), (a_5, a_6) \rangle \\
k & : \quad \langle (a_5, a_1), (a_6, a_2), (a_3, a_4) \rangle
\end{align*}
\]

where each tuple represents a pair for which the preferences, that is, the expected utilities are the same. It can be seen that by communicating only the expected utilities corresponding to their first preference actions, the solution obtained from maximizing on these, that is \(a_3\), is the same as the one obtained by maximizing over all possible actions. This is simply because the global optimum is indeed one of the local optimums; a fortuitous result for this particular example.

The iterative algorithm is now applied to this example. The first iteration yields the following; considering \(i\)’s most preferred actions \(a^{i1}\)

\[
\begin{align*}
\mathcal{B}_c(a^{i1}) & = \mathcal{B}_c(a_3) = \beta_i(a_3) + \beta_j(a_3) + \beta_k(a_3) = 59 \\
\mathcal{B}_c(a^{i1}) & = \mathcal{B}_c(a_6) = \beta_i(a_6) + \beta_j(a_6) + \beta_k(a_6) = 57,
\end{align*}
\]

considering \(j\)’s most preferred actions \(a^{j1}\)

\[
\begin{align*}
\mathcal{B}_c(a^{j1}) & = \mathcal{B}_c(a_3) \quad \text{(already computed)} \\
\mathcal{B}_c(a^{j1}) & = \mathcal{B}_c(a_1) = \beta_i(a_1) + \beta_j(a_1) + \beta_k(a_1) = 54,
\end{align*}
\]
considering \( k \)'s most preferred actions \( a^{k1} \)

\[
\begin{align*}
B_c(a^{k1}) &= B_c(a_1) \quad \text{(already computed)} \\
B_c(a^{k1}) &= B_c(a_5) = \beta_i(a_5) + \beta_j(a_5) + \beta_k(a_5) = 50.
\end{align*}
\]

A second bargaining iteration yields

\[
\begin{align*}
B_c(a^{i2}) &= B_c(a_5) \quad \text{(already computed)} \\
B_c(a^{i2}) &= B_c(a_4) = \beta_i(a_4) + \beta_j(a_4) + \beta_k(a_4) = 47,
\end{align*}
\]

considering \( j \)'s 2nd most preferred actions \( a^{j2} \)

\[
\begin{align*}
B_c(a^{j2}) &= B_c(a_2) \quad \text{(already computed)} \\
B_c(a^{j2}) &= B_c(a_4) = \beta_i(a_4) + \beta_j(a_4) + \beta_k(a_4) = 49,
\end{align*}
\]

considering \( k \)'s 2nd most preferred actions \( a^{k2} \)

\[
\begin{align*}
B_c(a^{k2}) &= B_c(a_2) \quad \text{(already computed)} \\
B_c(a^{k2}) &= B_c(a_4) \quad \text{(already computed)}.
\end{align*}
\]

Thus, a maximization based only on the first iteration in this example yields the solution \( \hat{\alpha}_c = a_3 \) as before. This highlights the fact that for relatively simple management problems, there is rarely a need to have more than a single iteration of the bargaining process.

For management problems where there are a large number of sensors and a large action set, more than one iteration may be required as the following example illustrates:

**Example 27**

Consider a system of 4 sensor nodes making observations of targets such that there are 8 distinct sensing configurations possible, that is, \( \mathcal{A} = \{a_1, \ldots, a_8\} \). The following are the ordered sensor expected utilities for each of the actions:

<table>
<thead>
<tr>
<th>sensor ( i )</th>
<th>sensor ( j )</th>
<th>sensor ( k )</th>
<th>sensor ( l )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 ) - 20</td>
<td>( a_5 ) - 40</td>
<td>( a_3 ) - 20</td>
<td>( a_7 ) - 35</td>
</tr>
<tr>
<td>( a_2 ) - 15</td>
<td>( a_6 ) - 35</td>
<td>( a_4 ) - 18</td>
<td>( a_2 ) - 30</td>
</tr>
<tr>
<td>( a_3 ) - 10</td>
<td>( a_1 ) - 20</td>
<td>( a_5 ) - 16</td>
<td>( a_3 ) - 10</td>
</tr>
<tr>
<td>( a_4 ) - 5</td>
<td>( a_2 ) - 15</td>
<td>( a_6 ) - 14</td>
<td>( a_5 ) - 5</td>
</tr>
<tr>
<td>( a_5 ) - 4</td>
<td>( a_8 ) - 10</td>
<td>( a_7 ) - 12</td>
<td>( a_6 ) - 4</td>
</tr>
<tr>
<td>( a_6 ) - 3</td>
<td>( a_4 ) - 5</td>
<td>( a_8 ) - 10</td>
<td>( a_4 ) - 3</td>
</tr>
<tr>
<td>( a_7 ) - 2</td>
<td>( a_3 ) - 2</td>
<td>( a_1 ) - 8</td>
<td>( a_8 ) - 2</td>
</tr>
<tr>
<td>( a_8 ) - 1</td>
<td>( a_7 ) - 1</td>
<td>( a_2 ) - 6</td>
<td>( a_1 ) - 1</td>
</tr>
</tbody>
</table>

The 1st iteration yields

\[
\begin{align*}
B_c(a^{i1}) &= B_c(a_1) = 49 \\
B_c(a^{j1}) &= B_c(a_5) = 65 \leftarrow a^* \\
B_c(a^{k1}) &= B_c(a_3) = 42 \\
B_c(a^{l1}) &= B_c(a_7) = 50.
\end{align*}
\]
Thus according to the first iteration the optimal action is $a_5$. However, proceeding to the 2nd iteration yields

$$B_c(a^2) = B_c(a_2) = 66 \quad \leftarrow a^*$$

$$B_c(a^2) = B_c(a_6) = 56$$

$$B_c(a^2) = B_c(a_4) = 31$$

$$B_c(a^2) = B_c(a_2) \text{ (already computed)},$$

showing that the optimal action is updated to $a_2$. A third iteration yields actions that have already been considered and similarly for the 4th iteration. In the 5th iteration all the actions have been considered except $a_8$ which yields

$$B_c(a^5) = B_c(a_8) = 30.$$

These examples demonstrate the process of decision making or bargaining between decentralised decision makers. The remaining sections focus more on the computation of information utilities.

### 6.4.2 Sensor Hand-off and Cueing

The control of a number of different sensors to track, and possibly identify is a key sensor management problem. The following example demonstrates the problem of cooperative tracking in a decentralised framework using information measures to evaluate target choices.

**Example 28**

Three tracking sensors at fixed locations $x_{s,i} = [x_{s,i}, y_{s,i}]^T$ $i = 1, 2, 3$ make range and bearing observations of three targets located at $x_j(k) = [x_j(k), y_j(k)]^T$ $j = 1, 2, 3$ at time $k$. Each sensor can only track one target. The discrete control action is the target assignment. The control objective is to find the 1 to 1 sensor to target mapping that maximises global knowledge of the target states. Target dynamics are represented by a probabilistic constant velocity model. The observation vector $z_i(k) = [r(k), \theta(k)]^T$, is a function of the state of the target being observed.

$$z_i(k) = h(x(k)) + v_i(k)$$

where $v_i(k)$ is taken to be a zero-mean uncorrelated Gaussian sequence with variance,

$$\mathbb{E}\{v_i(k)v_i^T(k)\} = R = \begin{bmatrix} \sigma_r & 0 \\ 0 & \sigma_\theta \end{bmatrix}$$

The observation model is

$$h(x(k)) = \begin{bmatrix} r(k) \\ \theta(k) \end{bmatrix} = \begin{bmatrix} \sqrt{(x(k) - x_s)^2 + (y(k) - y_s)^2} \\ \arctan \frac{x(k) - x_s}{y(k) - y_s} \end{bmatrix}$$
The Jacobian with respect to target state is
\[
H(x(k)) = \begin{bmatrix}
\frac{x(k) - x_s}{\sqrt{(x(k) - x_s)^2 + (y(k) - y_s)^2}} & \frac{y(k) - y_s}{\sqrt{(x(k) - x_s)^2 + (y(k) - y_s)^2}}
\end{bmatrix}
\]

The expected observation information for this sensor model is given by
\[
I(k) = H^T(k)R^{-1}H(k) = \begin{bmatrix}
\frac{\sin^2(\theta(k))}{r^2\sigma_r^2} + \frac{\cos^2(\theta(k))}{r^2\sigma_\theta^2} - \frac{\sin(\theta(k)) \cos(\theta(k))}{r^2\sigma_r \sigma_\theta} & \frac{\sin(\theta(k)) \cos(\theta(k))}{r^2\sigma_r \sigma_\theta} + \frac{\sin^2(\theta(k)) \cos(\theta(k))}{r^2\sigma_\theta^2}
\end{bmatrix}
\]

Note, the determinant $|I(k)| = \frac{1}{r^2\sigma_r \sigma_\theta}$. Hence, the observation information for this model is range dependent. Each sensor runs an information filter with local knowledge of the global information
\[
\text{\hat{Y}}_i(k | k) = \begin{bmatrix}
\text{\hat{Y}}_{i,1}(k | k) \\
\text{\hat{Y}}_{i,2}(k | k) \\
\text{\hat{Y}}_{i,3}(k | k)
\end{bmatrix}, \quad Y_i(k | k) = \begin{bmatrix}
Y_{i,1}(k | k) & 0 & 0 \\
0 & Y_{i,2}(k | k) & 0 \\
0 & 0 & Y_{i,3}(k | k)
\end{bmatrix}
\]

With entropic information measure
\[
i_i(k) = -\frac{1}{2} \log [(2\pi e)^{12} | Y_i(k | k) |] = -\frac{1}{2} \sum_{j=1}^{3} \log [(2\pi e)^{4} | Y_{i,j}(k | k) |]
\]

With it’s estimate of the target state after each prediction step, sensor $i$ constructs the expected information gain from observing target $j$
\[
I_{i,j}(k) = H^T(\text{\hat{x}}_{i,j}(k | k - 1), x_{s,i})R^{-1}H(\text{\hat{x}}_{i,j}(k | k - 1), x_{s,i})
\]

This is communicated between all sensor nodes. Each node can then form a matrix of the mutual information gain for each sensor target assignment.
\[
I_{i,j}(k) = \frac{1}{2} \log \left[ \frac{|Y_{i,j}(k | k - 1) + I_{i,j}(k) |}{|Y_{i,j}(k | k - 1) |} \right]
\]

The utility for control action $a_t$ is
\[
U_{a_t}(k) = \sum_{n=1}^{3} I_{n,j_n}(k), \quad \text{Where } a_{t,n} = i_n \to j_n, \quad a_t \in \left\{ a_1 = (1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3), a_2 = (1 \rightarrow 1, 2 \rightarrow 3, 3 \rightarrow 2), a_3 = (1 \rightarrow 2, 2 \rightarrow 1, 3 \rightarrow 3), a_4 = (1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 1), a_5 = (1 \rightarrow 3, 2 \rightarrow 1, 3 \rightarrow 2), a_6 = (1 \rightarrow 3, 2 \rightarrow 2, 3 \rightarrow 1) \right\}
\]

The optimal action is selected by
\[
a(k) = \arg \max_{a_t \in A} U_{a_t}(k)
\]
The sensors now make observation $\mathbf{z}_i(k)$ of their allocated target and communicate the observation information $\mathbf{i}_i(k) = H^T (\hat{x}_{i,j}(k | k-1), \mathbf{x}_{s,i}) R^{-1} \mathbf{z}_i(k)$ to all nodes. The nodes then update their local information by

$$\hat{y}_{i,j}(k | k) = \hat{y}_{i,j}(k | k-1) + i_j(k),$$

$$Y_{i,j}(k | k) = Y_{i,j}(k | k-1) + I_{j}(k).$$

Results from a solution to this example problem are shown in Figure 45. Figure 45 (b) shows that the value of observing a target is range dependent. As the targets move the optimal group control action switches. This decision and communication structure has provided a coordinated solution to the global control objective. Local prior and communicated external information allows each node to arrive at a solution it believes is best for the group. This example indicates that without this external information the resulting control action would differ. There are points where “greedy” allocation based on individual sensor to target mutual information gain is not the best group decision. A powerful result of the communicated information is that combined with the observation model, each sensor can determine the utility associated with observing a target. The decision to switch target allocation is made based on this utility comparison without the sensors observing their future target. This action is referred to as sensor hand-off and cueing.

It can be seen that the sensor nodes do not require any knowledge of the other sensors location or characteristics. Everything required to select the optimal action is contained in the local $\hat{y}_i(k | k)$, $Y_i(k | k)$ and the communicated $I_{i,j}(k)$. This is an important property of the decentralized information filter.

Note that in this case the local sensor knowledge is the true global information since all information is communicated between the sensor nodes. If all information is not available at each node this formulation is still valid. However, the local utility value associated with an action will differ between nodes.

Optimal configuration of vehicle and sensor trajectories for feature tracking and localisation is a logical extension of this example. The reason is the sensor model. While highly simplified, the range bearing model used is representative of a real world sensor. The combined observation information from multiple simultaneous observations is a function of the relative range and bearing between the sensors, vehicles and features. The dynamics of the information measures suggest an optimal system configuration.

### 6.5 Communications Management

The communications resource in a decentralised system provides a mechanism for platforms and sensors to exchange information. If the communications resource is constrained, in bandwidth, latency or connectivity, then it must generally be managed.

The information metrics developed in this section can be used to control the flow of information between sensor nodes. The subject of communications management was
Figure 45: Information theoretic approach to a discrete sensor management task. (a) Problem geography, (b) Information based Utility for each individual sensor to target assignment, (c) Entropic target information, (d) optimal group sensor to target assignment
studied in detail by Rob Deaves in his thesis [18] and is also reported in the paper [19]. Here we briefly discuss the key points:

- The value of information to be communicated can be measured using Equations 349–352.

- These information measurements can be implemented as part of the communication channel structure. This is illustrated in Figure 46.

- Information on different tracks and other objects can be evaluated. Communications can then be ranked by information value. Strategies include:
  
  - Only communicate the most informative track information when bandwidth becomes available.
  
  - Save up information in a channel until the total becomes sufficiently informative (burst communication).
  
  - Using a measure of mutual information with respect to previously communicated information, decide which node to connect to to communicate track information.

  A number of other strategies are also possible.

- Generally, each node operates opportunistically, computing the mutual information gains possible on each connected channel to determine when and what to communicate.

- In a limited medium, shared by a number of sensors, some element of the bargaining solutions described in Section 6.3.3 would need to be introduced. This is still a subject of current research.

### 6.6 Organisation and Control of Network Resource

The problem of organisation and control in decentralised networks has not yet received a great deal of attention. The problem of optimal organisation was considered in Peter Ho’s thesis [29]. This considered the problem of determining the connectivity arrangements in large-scale networks. Typical objectives included maximising the mean information levels, maximising the minimum information level, and minimizing information deviation. The network optimisation was performed using a distributed Bellman-Ford algorithm with the information metrics of Equations 349–352 serving as distance measures.

However, the problem of network connectivity and topology is only part of the overall problem. There are also issues coupled to the placement and indeed the trajectory of platform mounted sensors. This is illustrated in Example 29.

Control in decentralised systems is complex because of the potential coupling between decision makers actions (two bearing-only sensors engaged in tracking for example). The general problem is an area of current research and is not further discussed in this course.
Figure 46: Communications management process is used to control the function of channel filters.

Example 29

Continuous Area Coverage Example Two vehicles $i = 1, 2$ are exploring a terrain area $T(x, y)$ defined on the $(x, y)$ plane. The (two-dimensional) trajectories for the $i^{th}$ vehicle is defined by $x_i = [x_i(k), y_i(k)]^T$, $k \in [1, N]$ and shown in Figure 47. Each vehicle makes observations $z_i(k)$ of the terrain according to

$$z_i(k) = T(x, y) + v_i(k)$$

where $v_i(k)$ is taken to be a zero-mean uncorrelated Gaussian sequence with a variance proportional to the range between the vehicle and terrain feature,

$$E\{v_i(k)v_i^T(k)\} = R_{T,i}(k) = R\sqrt{(x_z(k) - x_i(k))^2 + (y_z(k) - y_i(k))^2},$$

where $x_z, y_z$ are the true terrain locations being observed.

The vehicle trajectories are assumed known and the kinematics of the terrain are stationary. It is required to generate estimates for the terrain $T(x, y)$ over the whole surface. In this case, the state transition and observation matrices are simply the identity matrix; $F(k) = 1$ and $H(k) = 1$, so the prediction and update stages of the information filter reduce to:

Prediction:

$$Y_T(k \mid k-1) = Y_T(k-1 \mid k-1)$$

Update:

$$Y_T(k \mid k) = Y_T(k \mid k-1) + \sum_{i=1}^{2} R_{T,i}^{-1}(k)$$

where the subscript $T$ makes it clear that these information quantities of defined at every point $x, y$ on the terrain.
The posterior entropic information on $T(x,y)$ and contained in the estimate is given by

$$i_T(k) = -\frac{1}{2} \log [2\pi e Y_T(k | k)]$$

The mutual information gain expected for an observation $z_i(k)$ of a terrain element $T(x,y)$ is

$$I_T(x_i(k)) = \frac{1}{2} \log \left[ \frac{Y_T(k | k - 1) + R_T^{-1}(k)}{Y_T(k | k - 1)} \right]$$

This information measure is a function over $T(x,y)$. A utility measure for an observation made at $x_i(k)$ is

$$U(x_i(k), k) = \int \int I_T(x_i(k)) dx dy$$

This can now be employed to generate a control metric to determine the trajectory of each vehicle to maximize the total information over the whole area. The trajectory utility;

$$U(x_i) = \sum_{k=1}^{N} U(x_i(k), k)$$

Figure 47 shows snapshots of the information measures over time. The vehicles start on opposite sides of the region. They travel at constant velocity over the indicated trajectories. The last leg of the first vehicles path overlaps the first leg of the vehicles path. Plots (a,d,g,j) show the entropic information over the area. (b,e,h,k) indicate the mutual information gain for the current observation. (c,f,i,l) show the potential information gain utility measure for an observation was made at location $(x,y)$. This example is set up to illustrate the information theoretic metrics. It highlights that the value of making observations at a location is dependent on time and that the value of vehicle trajectories are coupled. These metrics can be combined with other constraints, objectives and costs to form an optimal area coverage control problem and solution.

Vehicle trajectories in this example are deliberately set up to illustrate the information measures. In the proposed decentralised architecture the information obtained by each node is propagated throughout the system. Local control actions are selected with the benefit of communicated or estimated external information. It can be seen that the mutual information measure decreases exponentially as observations are made. Hence, the value of making observations from a position is coupled to time. A consequence of this probabilistic approach is exact knowledge of state can never be obtained. A wide range of possible control objectives, minimum control effort, minimize maximum uncertainty in minimum time uniformity across region, are possible. In this example information is never lost.
Figure 47: Snapshots of information measures for an area coverage example over time. Plots (a, d, g, j) show information (as negative entropy) over the area, (b, e, h, k) indicate the mutual information gain for the current observation. (c, f, i, l) show the potential information gain if an observation was made at location (x, y).
7 Applications and Practical Considerations

This section provides a brief description of a number of key decentralised data fusion projects. The objective is to describe some of the practical issues involved in developing and implementing decentralised data fusion systems.

We begin by reviewing three past projects; SKIDS, ISSS, and OxNav. These served as the basis for the original development of the various decentralised data fusion algorithms. The current ANSER project is then described in some detail. Finally, we include some discussion on large-scale sensor networks characteristic of ground and combined air-ground surveillance systems.

7.1 Historical Projects

7.1.1 SKIDS

In the SKIDS project, a fully decentralised surveillance system was implemented using four cameras and a Transputer based architecture. The network was a fully-connected point-to-point topology. The system was capable of tracking multiple targets (humans and robots). A decentralised data-association algorithm was developed, but this was superseded in later research and so is not described here. A decentralised identification algorithm was also developed but this too was later superseded. The algorithms developed for the SKIDS project are fully described in Bobby Rao’s D.Phil. thesis [49], and in the papers [50, 48, 47]. The SKIDS demonstrator, which continued to be refined and operated for almost 10 years, laid the basis for all subsequent work on decentralised data fusion.

7.1.2 ISSS

An essential limitation with the original decentralized Kalman filter algorithm is that it requires the sensor network to be fully connected so ultimately limiting the size of any realisable decentralized sensing system. The need for fully connectedness is a consequence of the assumption that common information between two neighbouring sites is simply the prior information they share.

This observation led to an analysis of information flow in decentralised sensing networks. By introduction of an additional filter associated with each communication link, it was shown that tree connected network topologies can also be supported by the decentralized Kalman filter algorithm. This is described in detail in Stewart Grime’s D.Phil. thesis [24] and in [25]. Channel filters, as they became known, also address a number of key issues in data asynchronicity, communications management and network reliability. Simukai Utete’s D.Phil. thesis [58] showed that, within the constraints imposed by the definition of a decentralized sensing network, it is not in general possible to construct a set of filters which can provide consistent estimates across an arbitrary network topology. To overcome this, decentralized routing algorithms were developed to enable construction of tree connected networks from networks of arbitrary topology [57, 60, 61, 59, 62]. Such algorithms maintain both consistency across the estimators in the network and satisfy the
constraints of locality and modularity of a decentralized system. Many of these ideas were developed on a large-scale demonstrator as part of the ISSS project. This demonstrator consisted of a model process control system consisting of over 200 sensors linked to over thirty purpose designed decentralized processing sites. The ISSS demonstrator allowed on-line network reconfiguration and software imposed communication bandwidth limitations. The demonstrator was designed to show scalability of decentralised data fusion algorithms to large numbers of sensors.

Further work on communications management has also been undertaken by Rob Deaves of Sowerby research center [18, 19]. This work builds on the idea of a channel filter and other work on information modeling [36] to manage communication between sensing platforms. The work is developed in a military context.

7.2 OxNav

A second limitation of the original decentralized Kalman filter algorithm is that it requires a complete system model to be maintained at every sensor site. As the size and complexity of a system the grows the need for a global model at each site becomes prohibitive. The problem of distributing system models across a sensor network was initially considered by Berg [6, 8, 9, 7]. Starting with a central state model and with the local observation models associated with each sensor node, it was shown that only a locally observable sub-model of the central state model is required at each site to ensure consistent estimates across a network. It was also shown that the transformations of model from central to local sites can be combined to provide a single transformation from one site to another which can be implemented as part of the node-to-node communication mechanism.

An important contribution of this work was in the geometric interpretation of information and the consequent explanation as to why information measures, and not state estimates, are uncorrelated (orthogonal) and thus why information fusion is associative. This result was also exploited in the development of the information gate; an information form of the equivalent data validation or innovation gate commonly used data association algorithms [21, 22, 23]. Data validation is essential in practical data fusion problems in providing a means of associating different internal models to observations made and in rejecting observations that are considered to be outliers or spurious. In decentralized systems validation must often take place remotely from the original observation. Understanding how an observation taken at one site with only a partial local model of the overall system compares to observations taken at another site with a different partial model is essential in being able to associate and validate information across a decentralized sensing network. This work is also continuing in the development of high integrity navigation systems [32].

A second consequence of the ability to perform model distribution is that it also allows a connection to be made between the decentralized data fusion techniques and the field of decentralized control. In decentralized control, components of an overall system model are distributed amongst a number of sites that both take observations of the world and exert control over the environment. The extension of the decentralized and distributed
Kalman filter to problems in decentralized control is described in [41, 42, 40, 39, 38]. It is shown how the model distribution results obtained for the decentralized Kalman filter lead to both the design of a controller and to the design of a control structure which is characteristic of the interconnections involved in the physical system to be controlled.

Figure 48: The OxNav Vehicle; a fully modular fully decentralised navigation and control system

These ideas on control were implemented in the OxNav project aimed at demonstrating fully decentralised and modular mobile robot navigation and control. This was a particularly challenging project combining almost all aspects of the theory in a single demonstration system and requiring the physical realization of a mechanical and electrically modular system, and demonstrates more clearly than any other application the potential advantages to be gained from a decentralized approach to systems design. A modular vehicle consisting of a number of standardized modular cages was designed (Figure 48). Each cage contained a specific part of the overall vehicle function; drive unit,
sensor, power distribution, communication systems [13, 14]. Each cage contained a processor, power and communication facilities, and all local software to implement the required decentralized functions of that unit. There is no central unit or processor where information is combined or where control is coordinated. A wide range of different vehicle systems were constructed from a small number of standardized cages, without the need to change either hardware or software. The decentralized control system for the vehicle demonstrated that the design of local decentralized control algorithms for an individual driven wheel unit allows the control of vehicles with any number of and kinematic configuration of driven and steered wheels. The decentralized navigation system is also described. The system employs a number of modular tracking sonar units. Each unit employs a model of vehicle motion to track environment features to provide independent estimates of vehicle location. The estimates are exchanged between sonar units to provide global navigation information. Vehicle guidance was achieved through exchange of information between vehicle drive units and sonar navigation sensors. The OxNav project won a number of major awards for innovation and industrial collaboration.

7.2.1 Organisation and Management

All early work on decentralized data fusion relied on the simple algebraic manipulation of a centralized estimation algorithm to obtain an equivalent decentralized estimator. The work by Manyika [33, 36, 34, 35] provided a completely different approach to the decentralized data fusion problem based on an information-theoretic model of sensor observation and data assimilation. Information-theoretic models are much more fundamental to the data fusion problem than conventional state estimation techniques. In particular, the work demonstrated that all of the decentralized algorithms derived to this point can in fact be directly derived from information theoretic concepts once the problem has been defined through Bayes theorem in probabilistic form. It was also demonstrated that the quantities computed at sensor nodes in both the discrete and continuous filters may be described in terms of information quantities alone. Furthermore, the information-theoretic development of the problem demonstrates clearly why the decentralized filter is structurally and computationally a more natural formulation of the data fusion problem than is a conventional multi-sensor filter.

The information formulation of the decentralised data fusion problem was first demonstrated on the OxNav project; in particular for navigation and for sensor management.

Once the nature of decentralized sensing has been described in the form of information measures it became possible to quantify the value of a particular observation and the value of a sensor site within a sensing organization. If the value of an observation or site can be measured, then a number of additional questions may be asked of the sensing network. In particular, it becomes possible to address the problem of sensor management and network organization on the basis of information maximizing strategies. Sensor management addresses the problem of how individual sensor sites should act so as to maximize the amount of information available to the network.

The solution to the sensor management problem relies on the local measurement of
mutual information gain resulting from a particular sensing action, and through the global maximization of this quantity by each individual sensor site. Sensor organization addresses the problem of what structure the sensing network should take under specific quantifiable system requirements. Given a probabilistic model for each sensor site and the observations made by each sensor, what network structure will provide, for example, the most robust system or the most informative system [29]. The solution to the sensor organization problem relies on a composite measure of the information generated by each node in the system, accounting for the possible connectivity between sensor sites and the local observability of different states at different nodes in the network.

The sensor management and organization problem demonstrate more clearly than any other aspect of the decentralized data fusion problem the value of using information-theoretic quantities to describe both individual sensors and the overall structure of the sensing network.

7.2.2 OxNav System and Demonstrations

The OxNav system shown in Figure 48 consists of a number of modular system components. The main sensors are tracking sonars as shown in Figure 49. These are able to lock on to and track different types of features in an environment and to produce estimates of relative feature location and identity (corner, line, etc). The main control elements are driven and steered wheels coupled with spacer blocks of similar geometry. Figure 50 shows an arrangement of nine such units.

![Figure 49: A photograph of the tracking sonar used on the OxNav vehicle.](image)

Each module, sensor or drive unit, contains a standard processor and processor motherboard which serve as the computational element of the node. Nodes are connected to each other via a network of point-to-point high speed serial lines. The interconnects
between nodes (which can be seen atop the vehicles in Figures 48 and 50), can be made arbitrarily. There is no central processor on the vehicle.

The arrangement of sonars on the four corners of a standard vehicle is shown in Figure 51. As the vehicle moves through the environment, the lead sonars scan and lock onto new features. These features are tracked with respect to the estimated vehicle location to provide both location and identity elements. As the vehicle moves through the environment, these targets are handed-off to the rear-mounted sensors. These sensors lock on and track these features, providing vehicle location data. The management of the sensors, the cuing and hand-off of targets between sensors, is handled in a fully decentralised manner using a composite information measure of identity and location for the targets. The algorithm employed is based on that described in Section 6.3.3. A typical run through an environment, with targets located and tracked, is shown in Figure 52.

7.3 ANSER

7.3.1 Objectives

The primary objective of the ANSER project is to demonstrate fully decentralised and modular picture compilation and terrain navigation on single and multiple flight platforms (UAVs). In turn, this will be used to demonstrate, in a highly relevant form to BAE Systems operating divisions, how decentralised systems architectures result in:

1. Modular packaging of sensor and data fusion algorithms.

2. Scalability and on-line flexibility to addition of single and multiple sensors and flight systems.
3. Robustness and fault tolerance to failure in sensors, systems and platforms.

4. Providing a quantitative means of analysing systems performance issues such as communication and integrity.

5. Increasing degrees of controllability and autonomy in sensors and flight systems.

These objectives are being met by developing, from the outset, fully modular navigation and picture compilation instrumentation following the theoretical and practical principles developed in previous collaborative research projects. This specifies logical packaging of sensors with processors where possible, and explicit separation of functions such as observation pre-processing, data assimilation and inter-module communication.

7.3.2 Demonstrations

Fundamentally, ANSER is a demonstration project: The objective is to demonstrate functionality of decentralised data fusion theory and algorithms developed over the past 10 years in a form which is of direct relevance to BAE Systems business units.

The ANSER project calls for the simultaneous deployment of up to four UAVs in decentralised configuration (see Figures 53 and 57). Four platforms are the minimum allowing demonstration of non-trivial decentralised communication policies. Each UAV is
Figure 52: A run of the OxNav vehicle along a corridor showing different targets being acquired and tracked.
Figure 53: The Mark I ANSER UAV.

Figure 54: The Mark III UAV showing nose housing for radar.
Figure 55: A further view of the Mark III UAV.

Figure 56: A further view of the Mark III UAV.
The UAVs are flown at the ACFR flight test facilities at Marulin, 175Km south of Sydney.

The UAVs will use the payloads to perform two primary functions:

1. Picture compilation: The detection and tracking of multiple ground targets given known locations (derived from GPS/IMU) for the flight platforms. Each function will be developed and demonstrated in fully decentralised and modular form; across payloads on any one flight platform and across multiple payloads on multiple flight platforms. Figure 58 shows the structure of on-board algorithms.

The scenarios to be flown are aimed at demonstration of the key elements of the decentralised data fusion method. The scenarios can be broken down in to four main groups (see scenario definition documentation for details):

1. Single platform picture compilation: A single platform will be flown with multiple payloads in picture compilation mode. Demonstrations include the modular exchange of payloads (modularity, interoperability), and the in-flight failure and reconfiguration of payloads (survivability and flexibility).
Figure 58: Algorithmic architecture for the UAV. Each payload operates an independent decentralised filter and corresponding channel filters for both picture compilation and SLAM modes. The internal function of each sensor node is hidden and their operation is transparent to the operation and location of other payloads.

2. Single platform SLAM: A single platform will be flown with multiple payloads in SLAM mode. Demonstrations include flight proving the SLAM method (not discussed in this document), generation of terrain maps from decentralised payloads, with the same modularity and payload reconfiguration abilities as are demonstrated for picture compilation functions.

3. Multiple platform picture compilation: Multiple flight platforms will be flown with multiple payloads in picture compilation mode. Demonstrations will include the extension of function from one to four platforms (scalability), the transparent use of sensors on one platform by assimilation processes on other platforms (modularity, interoperability), reconfiguration of sensing due to failure of individual payloads, and task reconfiguration due to failure of complete flight platforms (survivability and flexibility).

4. Multiple platform SLAM: Multiple flight platforms will be flown in SLAM mode. Demonstrations will include the ability to share terrain maps between payloads on different flight platforms, to fuse maps from geographically separated payloads, and to demonstrate scalability and robustness of decentralised SLAM methods.

The scenarios form a logical progression for the implementation of the theory and demonstration of algorithms.
7.3.3 Research

The ANSER project is focused on demonstration of existing theory. However, the complexity of the demonstrator still requires research in three main areas:

1. The extension of theory and methods to airborne scenarios: There is a necessary process of mapping a general theory and set of methods to this specific, and demanding, application. Most previous work undertaken in decentralised methods has been done on ground-based sensors or land vehicles. An air scenario has more degrees of freedom, faster data rates, and larger demands on processing and communication management.

2. The development of sensing, terrain data acquisition, and terrain or target representation methods: The development and packaging of payload sensors for airborne application has been a significant issue; particularly weight, volume and data acquisition speed. The extraction of appropriate terrain features for picture compilation and SLAM, the modeling and communication of these between different payloads has also been an area of significant study.

3. The development of decentralised and information-theoretic SLAM methods: One significant theoretical advance made in this project is the development of a decentralised formulation of the map building and SLAM problem. This has required the re-formulation of map building equations in information-theoretic form and the study of how maps from different platforms can be exchanged and assimilated.

ANSER has also provoked a number of new research areas, beyond the scope of the current project, but of considerable future value in decentralised systems.

7.4 Implementation of DDF Algorithms in ANSER

This section describes the decentralised data fusion algorithms implemented in the ANSER system. The approach is to present generic equations for the decentralised data fusion problem and subsequently (in Sections 7.4.7 and 7.4.8 to instantiate process and observation models specific to the different demonstrations and sensor nodes employed. Figure 59 shows the general structure of point-to-point communications in the decentralised data fusion architecture. This is generally what is implemented within the ANSER system. On a single vehicle, this is practically mapped to a bus or broadcast network as shown in Figure 60. However, this does not change the essential decentralised algorithms for fusion and communication.

All the decentralised data fusion functions to be demonstrated in ANSER are continuous state estimation problems. The decentralised data fusion algorithms therefore all employ the information filter for continuous state estimation. The more general discrete state decentralised estimation problem employs the log-likelihood form of Bayes theorem from which the information filter is derived. State estimates (tracks or landmark estimates) are therefore all described in terms of an information matrix \( Y_p(i \mid j) \) and an information state \( \hat{y}_p(i \mid j) \).
With these definitions, the overall structure of a decentralised sensing node is shown in Figure 61. The node executes four main functions:

1. Information state prediction, generating $\hat{y}(k | k - 1)$ and $Y(k | k - 1)$ at the observation time (generally asynchronous) from past information state values.

2. Observation preprocessing to provide the observation information vector and matrix $i(k)$ and $I(k)$. If necessary these need to be associated with correct predictions, but then fusion is a simple process of summation.

3. Information transmission through the channel filter. This requires the calculation of new information to be communicated to connected nodes.

4. Assimilation of incoming data from channels. This generally requires temporal alignment and compensation of delayed data effects.

State estimation takes place at the local filter, which simply sums the total information from observation, communication and prediction. All communications to the network pass through the channel manager, which handles timing and interfacing with the channel filters. The channel filters keep an estimate of all information that has passed down a particular channel and physically connects with the communications medium.

The main node operations are now described:
7.4.1 Preprocessing and Coordinate Transformations

When a sensor makes an observation, a number of preprocessing stages must occur to transform observation data into information form (i(k) and I(k)) before it is fused in the nodal filter.

The sensors employed are all range/bearing or bearing only. Observation information is converted in to a Cartesian form for processing by the DDF algorithms. The Cosine Matrix relating frame i to frame j (rotating $\psi$ in yaw, $\theta$ in pitch and $\phi$ in roll is

$$
C_j^i = \begin{bmatrix}
\cos(\psi) \cos(\theta) & \cos(\psi) \sin(\theta) \sin(\phi) - \sin(\psi) \cos(\phi) & \cos(\psi) \sin(\theta) \cos(\phi) + \sin(\psi) \sin(\phi) \\
\sin(\psi) \cos(\theta) & \sin(\psi) \sin(\theta) \sin(\phi) + \cos(\psi) \cos(\phi) & \sin(\psi) \sin(\theta) \cos(\phi) - \cos(\psi) \sin(\phi) \\
-\sin(\theta) & \cos(\theta) \sin(\phi) & \cos(\theta) \cos(\phi)
\end{bmatrix}
$$

With this, the observed target is related to the sensor, the sensor to the body and then the body to the earth frame, as,

$$
P^e_b = P^e_b + C^e_b P^b_s + C^e_b C^b_s P^s_b \quad (365)
$$
where

- \( P_b^e = [x_b^e, y_b^e, z_b^e] \) is the position of the body in the earth frame and is provided by the INS.
- \( C_b^e \) is the direction cosine matrix which relates how the body frame is rotated with respect to the earth frame.
- \( P_b^s = [x_b^s, y_b^s, z_b^s] \) is the position of the mission sensor to the body frame. This is determined by calibration.
- \( C_b^s \) is the direction cosine matrix which relates how the sensor frame is rotated to the body frame and is also provided by calibration. If the sensor frame is perfectly aligned so that the frame axes of both the sensor and body frames are parallel then the matrix is simply an Identity matrix.
- \( P_t^s = [x_t^s, y_t^s, z_t^s] \) is the position of the target with respect to the sensor. The vector which points to the target from the sensor frame can be described by two angles and the range measurement. These angles will be taken as \( \theta_t^s \) which is the look down angle of the target from the sensor frame on the \( xy \) plane. This angle will transform the vector over to the \( xy \) plane from which the azimuth of the target \( \psi_t^s \) can be determined. Thus the position of the target becomes

\[
P_t^s = \begin{bmatrix} x_t^s \\ y_t^s \\ z_t^s \end{bmatrix} = \begin{bmatrix} r_t^s \cos \theta_t^s \cos \psi_t^s \\ r_t^s \cos \theta_t^s \sin \psi_t^s \\ r_t^s \sin \theta_t^s \end{bmatrix} \tag{366}
\]

Note that for the radar/laser system the look down angle (grazing angle) is fixed, that is, \( \theta_t^s \) is constant. Also note that positive looks down from the \( xy \) plane. The azimuth angle is positive from the \( x \) axis clockwise when looking from above.

Once the measurements have been converted, it is necessary to convert the observation noise to cartesian space. For a sensor with a standard deviation of \( \sigma_r \) in range, \( \sigma_\psi \) in azimuth and \( \sigma_\theta \) in pitch, the converted observation noise matrix \( R_{Cart}(k) \) is a function of the square of the range of the observation and the rotation matrices.

\[
R_{Cart}(k) = \left[ C_b^e C_b^s \right] \begin{bmatrix} \sigma_r^2 & 0 & 0 \\ 0 & r^2 \sigma_\psi^2 & 0 \\ 0 & 0 & r^2 \sigma_\theta^2 \end{bmatrix} \left[ C_b^e C_b^s \right]^T \tag{367}
\]

As the platform location is used in the observation coordinate transformation, it is necessary to adjust the observation information for the vehicle uncertainty. Failure to do this could result in overly confident target estimates and possible filter divergence. This is done using the following equation.

\[
I_{Norm}(k) = Y_{Platform}(k | k - n) I(k) Y_{Platform}(k | k - n) \tag{368}
\]
In summary, when a range/bearing observation is made at some time $k$, it is firstly transformed to cartesian coordinates using Equation 365. The observation uncertainty matrix is also transformed to the new reference frame by Equation 367. The observation information matrix and information vector are then computed using Equation 209 and the observation information matrix adjusted to account for the platform uncertainty by Equation 368. This is necessary as the platform location information is used in the coordinate transformation.

7.4.2 Data Association

Data association is necessary to correctly match information about the same target or feature from different sources or at different time steps. When an observation is made, it is necessary to determine if the target is same as one that has already been seen. Also, in a decentralised system, it is necessary to associate information from other nodes with that stored locally. Figure 62 illustrates this notion where decentralised system exists with two nodes estimating the same targets, but they are ordered differently. When node one communicates information about its target 1, node 2 must correctly associate it with its own target 3.

![Figure 62: Different nodes may have the same physical targets stored in different orders.](image_url)

The information gate and nearest-neighbour algorithm is used for data association with the information filter. This is very much a “brute force” approach. Nodes also include a data association index with each information communication. The data association index is the location of the target/feature at the transmitting node. When received for the first time, the targets pass through the data association algorithm to determine if they match any targets at the receiving node. Once the receiving node knows the index of that target locally, it can store the relationship between the targets on different nodes. In this way, a look up table is generated once targets are identified. Figure 62 illustrates this.
A lookup table mapping the targets on different nodes is shown. Along with the data association index, nodes also include a probability of the data association index being correct.

7.4.3 Local Filter

The local information filter generates information state estimates on the basis of observed, predicted and communicated information. Other infrastructure such as the channel filter and channel manager exist only to support the correct implementation of the local filter.

![Diagram of data flow from sensor observation.](image)

Figure 63: Data flow from sensor observation.

The local filter takes input from local sensors (if present) and from the channel manager (if connected). Local sensors preprocess observation data to produce an observation information vector \( \mathbf{i}(k) \), observation information matrix \( \mathbf{I}(k) \), a data association index, and an associated probability of correctness to the filter. The generation of observations is typically asynchronous. This observation information is communicated to the local node filter where (see Figure 63:

1. The local information state is predicted forward to observation time.
2. It passes through a data association stage using the a combination of both the information gate and the data association index sent with the information package.
3. With a correct association, the node filter then fuses observation and prediction information through the summation in Equation 321.
4. The fused estimate is then propagated forward from observation time to the synchronous node step.
5. Equation 312 is generally used to predict the information state forward in time. If data is delayed in communication or observation to the node filter, it is fused using delayed data Equation 317 given in Section 5.2.8.

At the given node synchronous rate, the node filter also receives new information from the channel manager. The state is predicted forward to this time using Equation 312 and updated with Equation 321. The full information matrix and information vector are then output to the channel manager for transmission to neighbouring nodes.
7.4.4 Channel Filter

The channel filter is used to manage communication between nodes. It serves two main functions; to keep track of information previously communicated on the channel, and to synchronize incoming and outgoing information with the node filter. Information previously communicated is used to compute new information gain to other sensor nodes in the network. Synchronization serves to accommodate any delays in information or differences in timing between node filters at remote sites.

Figure 64 shows the flow of incoming information through the channels, channel manager and local filter. Information arrives asynchronously at each channel from remote nodes. The information first passes through a preprocessing stage where track-to-track association is performed using both data association index and an information gate, in order to associate the new data with an existing filter track. The information state is then predicted forward to the time horizon using the standard prediction method of Equation 312. As the data enters the channel filter, the new information is determined and is then transmitted to the channel manager. The channel filter is then updated.

Figure 64: Flow of information for incoming channel data.

Figure 65 shows the flow of outgoing information from the nodal filter to the channel manager and to the channels. As information is sent to the channel, the channel filter is updated and the current state is transmitted down the channel. In the event that the channel becomes blocked or disconnected, the channel filter effectively fuses the new data and cycles to the next available communication time.

7.4.5 Channel Manager

The channel manager serves as the interface between the nodal filter and the channel filters (and through these, the other nodes in the network). The channel manager collects incoming data from the channels at the time horizon, assimilates it, and then communicates the result to the nodal filter. It also receives outgoing updated information states.
Figure 65: Flow of information for outgoing channel data.

from the nodal filter and disseminates this to the channel filters for transmission. The channel manager manages online channel connectivity and allocation of, the allocation is handled by the channel manager as well.

7.4.6 Timing

Each clock at each node is synchronised to a common system time, in order to eliminate errors in clock drifts.

Every node in the decentralised system operates asynchronously with respect to the other nodes. The nodes are programmed to communicate their information every time period $\Delta T$ (where $\Delta T$ is the period between time horizons), which need not be the same for all nodes. When a node receives new information through a channel, it is predicted to the time horizon of the local node and temporarily buffered. The time horizon is a time in the future at which the local node will assimilate the new data from all of its channels with the local state estimate. The prediction to this horizon is done using the same model as the local filter. Thus, data will arrive asynchronously at a nodes channel filter, then be predicted and stored until the next time horizon. Figure 66 illustrates this timing. If a situation arises where two pieces of information arrive on the same channel between any two time horizons, only the most recent to arrive should be used and the other(s) discarded. As the channels have a one to one mapping, the most recent of these measurements will always contain all the information of the earlier message in addition to any newer data. This situation can occur if different nodes are transmitting their
information at different frequencies, a scenario which is entirely valid under the system specifications. However, if the channel filter is implemented correctly then this will not cause a problem.

Figure 66: Data arrives at the channel filter asynchronously from other nodes. It is predicted forward to a local time horizon and fused then.

It is important that data not be transmitted from the node if there is still new data buffered in the channels. Prior to transmitting information, the channels should be read and cleared. This is necessary as the channel filter at the transmitting node will already have updated its channel filter with the data waiting in the buffer at the receiving node, and if that receiving node transmits information back that does not include this information then the system becomes inconsistent. In practice, what occurs is at the time horizon the channels and the local filter are updated and the node then outputs its new state then. That is, each node outputs its state at the local time horizon after it has updated with the information in the channels. In this way, the timing within each node is kept synchronous, but the timing between nodes is asynchronous. This ensures the filters remain consistent and do not have any timing ambiguities.

If the system is implemented in this manner, it also removes any delayed or asequent data problem through the channel filter. When the data arrives, it is automatically predicted forward to the time horizon. There still exists the possibility of delayed data from the local sensor, however, this can be handled using the delayed data algorithm of Section 5.2.8.

### 7.4.7 Picture Compilation

For the decentralised picture compilation problem, a separate channel filter/local filter pair is maintained for each target. It is possible to augment all of the targets into a single
vector, however as the targets are uncorrelated the information matrix is sparse.

In picture compilation mode, the location of the sensors and platform is assumed known with precision. This location information is used to convert all relative observation information to a global cartesian ground coordinates in which tracking takes place. This conversion is accomplished in the sensor preprocessing stage for both bearing only and range and bearing sensor types (see Section 7.4.1).

In global ground coordinates, all targets are modeled using a three-dimensional ‘constant velocity’ ballistic model in the form:

$$\mathbf{x}(k) = F_{(k,k-1)} \mathbf{x}(k-1 | k-1) + \mathbf{w}_{(k,k-1)}$$

where the state vector is

$$\mathbf{x}(k) = [x(k), \dot{x}(k), y(k), \dot{y}(k), z(k), \dot{z}(k)]^T$$

The state transition matrix for this system is given by

$$F_{(k,k-1)} = \begin{bmatrix}
1 & \Delta T & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & \Delta T & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \Delta T \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

The process noise is written as $\mathbf{G}_k \mathbf{Q}_{(k,k-1)} \mathbf{G}_k^T$ where

$$\mathbf{Q}_{(k,k-1)} = \begin{bmatrix}
q_x & 0 & 0 \\
0 & q_y & 0 \\
0 & q_z & 0
\end{bmatrix}$$

and

$$\mathbf{G}_k = \begin{bmatrix}
\frac{\Delta T^2}{2} & 0 & 0 \\
\Delta T & 0 & 0 \\
0 & \frac{\Delta T^2}{2} & 0 \\
0 & \Delta T & 0 \\
0 & 0 & \frac{\Delta T^2}{2} \\
0 & 0 & \Delta T
\end{bmatrix}$$

While the real observations are of the range/bearing type, they are converted to cartesian coordinates in order to linearise the problem. After this conversion has been made, the observation matrix is simply given by

$$\mathbf{H}(k) = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}$$

where the observations are in the form $\mathbf{z}(k) = [x, y, z]^T$. 
The picture compilation system uses an information gate for data association. Nodes also use the data association index to map targets at one node to targets at another, as detailed in Section 7.4.2.

The coordinate transformations to convert the range/bearing measurements to cartesian will be done in the sensor preprocessing stage. The equations, given in Section 7.4.1, convert the measurement and the observation covariance taking into account the platform uncertainty. Once the sensor preprocessing has completed these transformations, the observation information vector is calculated according to Equation 209 and transmitted to the decentralised filter node.

### 7.4.8 Terrain Navigation

The decentralised terrain navigation system is a single vehicle problem. Each platform has all of its sensors decentralised and communicating internally, however, there is no inter-platform communication. The platforms start with a prior map and navigate by observing these known features. Each node in the system runs a single local filter (and associated channel manager and channel filters) which estimates the platform position, velocity and attitude. The state vector for this is given by

\[
x(k) = [x(k), y(k), z(k), \dot{x}(k), \dot{y}(k), \dot{z}(k), \psi(k), \theta(k), \phi(k)]^T
\]  

The decentralised system includes some combination of radar, laser and camera as well as an IMU and GPS. The local filter at the inertial node will run a filter estimating the same states as the rest of the system, but need not necessarily be in the same linear form as other nodes. The result of this is that when the inertial node outputs its state through the channel filter, it may in effect be outputting a loss in information. When the node receiving this information determines the information gain, it will be therefore be negative. The prediction stage of the channel filter in the inertial node uses the same linear model as other nodes for the time alignment of data.

The extension to the Simultaneous Localisation and Mapping (SLAM) problem involves an expanded state incorporating the locations of all (stationary) land marks.

### 7.4.9 The Communication Layer

While the decentralised algorithms are to be implemented in a point to point network, the physical communication medium onboard the Brumby platforms is a CAN bus. Therefore, the point to point architecture will be developed in software using virtual channels to connect nodes. This is illustrated in Figure 67 where multiple decentralised nodes can be seen to reside on a single processor.

The virtual channels are supported by the CommLib communication software developed for the project. In addition to the connections between the nodes, separate channels are required for

- DDF links between node pairs
• The platform states from the GPS/IMU process
• Time synchronisation messages

All nodes in the system will be synchronised with the flight control computer, which will be synchronised with GPS time in order to keep the clocks across different platforms from drifting. If the GPS signal is lost for a period of time, all nodes within a platform stay synchronised with the flight control computer, however, the flight control computers on different platforms may drift as they are not being reset.

7.5 Large Scale Sensor Networks

Work is also underway at Sydney to develop a ground-based sensor network to be located at the same test site as the flight platforms. The aim of this work is to demonstrate large-scale scalability of decentralised data fusion methods and to show distribution across both air and land environments.

Currently the project is developing a group of ground-based sensor nodes, based on the same architecture as the ANSER project. The nodes include vision, and multi-spectral cameras, lasers and (in the future) radars. Sensor nodes are being physically constructed in a modular fashion around a PC104 architecture and linked by radio ethernet.

The network will be used for both air-target tracking and hand-off of ground targets from air to land sensors. The practical objectives are to demonstrate the applicability of decentralised methods to a broader range of data fusion problems, and to serve as the primary test-vehicle for system-of-system theories and algorithms.

The large-scale sensor network laboratory associated with this course aims to explore some of the issues in large-scale sensor networks. In particular, the issue of dynamic
communication topologies, and network organisation. Figure 68 shows the typical network configurations to be explored. Figure 69 shows typical tracks. Figure 70 shows tracking results from a typical node.
Figure 68: Different communication topologies for a 20 node network: (a) Fully connected; (b) and (c) tree connected; (d) general network topology.
Figure 69: Typical tracks and observations generated: (a) true $x$–$y$ target tracks and observations; (b) detail of estimated tracks from all nodes.
Figure 70: Estimates from a node: (a) $x$-position; (b) $y$-position; (c) velocity; (d) heading.
References


