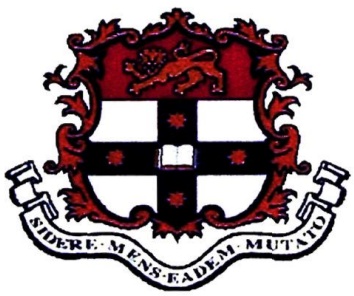
Decentralised Architectures for Tracking and Navigation with Multiple Flight Vehicles

Eric Nettleton

A thesis submitted in fulfillment
of the requirements for the degree of
Doctor of Philosophy



Australian Centre for Field Robotics
Department of Aerospace, Mechanical and Mechatronic Engineering
The University of Sydney

2003

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

Eric Nettleton

February 28, 2003

Abstract

Eric Nettleton
The University of Sydney

Doctor of Philosophy
2003

Decentralised Architectures for Tracking and Navigation with Multiple Flight Vehicles

This thesis is concerned with the development and demonstration of decentralised data fusion (DDF) algorithms in airborne applications. The decentralised architectures described in this thesis require no central fusion centre and no common communications medium. Sensor nodes are joined in a decentralised network without any global knowledge of the network topology or other node capabilities. Each sensor node is able to form a global estimate based on local sensor observations and information communicated to it by adjacent sensor nodes in the network. In a DDF architecture, no sensing, processing or communication component is critical to the operation of the overall system so a failure of any single element results in only an incremental decrease in performance rather than catastrophic system failure. As no node requires knowledge of the global network topology, the system can be scaled simply by connecting new sensing nodes to the system.

The main contribution of this thesis is the development of decentralised algorithms and architectures able to deal with asynchronous and intermittent communications characteristic of real-world sensor nodes communicating over radio networks. The algorithms presented in this thesis are developed using the information form of the Kalman filter running on a variety of structured and unstructured decentralised sensor networks. Although this thesis is primarily concerned with structured communication topologies, that are able to make complete use of global information, sub-optimal algorithms for use in networks with dynamic connectivity changes are also proposed. Exact solutions to the delayed and asequent data problems are developed for the information form of the Kalman filter that enable operation of DDF algorithms in a broad range of real sensor networks.

The DDF architecture is applied to the problem of tracking multiple ground targets using a network of airborne sensing platforms. The objective of this problem is for the sensing nodes on each platform to build a composite global picture of targets in an environment using both local sensor observations and information communicated from other platforms in a decentralised manner.

The second important contribution made by this thesis is in the development of algorithms for multi-vehicle Decentralised Simultaneous Localisation and Mapping (D-SLAM). Closed form solutions to the SLAM covariance and information matrix for simplified single and multiple vehicle D-SLAM problems are presented. These show that; i) in general platform

to platform cross information is zero, and ii) the global map information is simply the sum of the map information on each platform in the system. Using these properties a D-SLAM algorithm is developed which enables multiple platforms to build a common global map in a fully decentralised manner. A constant time communications algorithm is also presented which ensures that the decentralised algorithm scales as the map size grows large.

This thesis also describes the implementation of these algorithms in a demanding environment using multiple uninhabited airborne vehicles (UAVs). The demonstrations of decentralised sensing described in this thesis are believed to be the first ever of multiple cooperative UAVs.

Acknowledgements

I would like to begin by thanking Professor Hugh Durrant-Whyte for his support, guidance and endless enthusiasm throughout the past four years. Hugh was always available to give help and direction whenever it was needed. I would also like to thank my supervisor Dr Peter Gibbens for his help during this period.

I must give special thanks to all the members of the ANSER project, without whom the demonstrations in this thesis could never have happened: Salah, for his superhuman management efforts, Jeremy, for building and flying the aircraft, Ali, with whom I spent many hours and late nights programming the simulator, Matt for his efforts with the vision sensor, Kim for the GPS/IMU loop, Stuart for the flight controller and Gurce, Alan, Chris, Graham and Ikramy for the countless hardware (and software) components they designed and built. Thanks also to the staff at BAE Systems for their help. There was never any shortage of deadlines - only time. When it finally came together, the demonstrations were fantastic to see and it was great to be part of the team.

To all the other members of the ACFR group, I owe a special thank-you for making my time here enjoyable. To Professor Eduardo Nebot, Professor Gamini Dissanayake and Dr David Rye, thanks for the help along the way. To Steve, Ross, Richard and Trevor - the 'Worms' stress relief was fantastic. To Ben, who once took me on a bus ride to a war zone with the line "How bad can it be?" and Jeff, the outings to the Rose and Observer were always entertaining. I would particularly like to thank Mike and Stef for the help they gave, and Anna for keeping everything running so smoothly. To Alex, Fred, Tim, Monica, Jose, Juan, Ralph, Mark, Andrew, Tomo, Shahram and all the others who have come and gone from ACFR, you have all helped to make my time here rewarding.

I would like to give my greatest thanks to Maria, whose love and patience has supported me throughout this thesis (although she would probably use the word 'ordeal'). My parents, Bill and Pat, and sister Jacqui were also always there giving support, as were my "new" family, Gordon, Marena and Mila. I thank you all.

I would also like to thank Dave Nicholson, Simon Julier, Sebastian Thrun and Julia Sutcliffe for the various discussions over the past four years, and BAE Systems for the financial support.

For Maria

MMN2B→MMNUR

Contents

Declaration i

Abstract ii

Acknowledgements iv

Contents vi

List of Figures xii

List of Tables xvi

1 Introduction 1

1.1 Motivation 2

1.2 Decentralised Data Fusion 3

1.2.1 The Benefits of a Decentralised Architecture 4

1.2.2 The Development of the Decentralised Architecture 6

1.3 Objectives of this Thesis 10

1.4 Contributions of this Thesis 11

1.5 Thesis Structure 12

2 Decentralised Estimation Algorithms 14

2.1 Introduction 14

2.2 Bayesian Estimation 15

2.2.1 Bayes Theorem 15

2.2.2 Log-Likelihoods 16

2.2.3 Distributed Bayesian Estimation 18
2.3 The Kalman Filter 21
2.3.1 Motion and Observation Models 22
2.3.2 The Kalman Filter Algorithm 22
2.3.3 The Extended Kalman Filter 26
2.4 The Information Filter 28
2.4.1 State Update in Information Form 28
2.4.2 Prediction of Information 30
2.4.3 The Information Filter and Bayes Theorem 36
2.4.4 The Information Filter in Multi-Sensor Systems 38
2.4.5 The Distributed Information Filter 41
2.4.6 The Decentralised Information Filter 43
2.4.7 The Extended Information Filter 47
2.4.8 Information Metrics 48
2.5 The Covariance Intersect Algorithm 49
2.6 Summary 52

3 Algorithms for Decentralised Systems **54**
3.1 Introduction 54
3.2 Communication in Decentralised Sensing Systems 55
3.2.1 Communication Topologies 56
3.2.2 Bayesian Communication in Sensor Networks 57
3.2.3 Identification of Redundant Information in Sensor Networks 59
3.2.4 Communication Channel Filters 62
3.2.5 Fully Connected and Broadcast Sensor Networks 67
3.2.6 General Network Structures 68
3.3 Delays and Timing: Temporal Propagation of Information 71
3.3.1 System Definitions 71
3.3.2 Forward Propagation of State 72
3.3.3 Backward Propagation of State 73
3.3.4 Notation for Time Invariant Systems 74

3.3.5 Definition of Estimates and Information States 75
3.3.6 Forward Time Propagation 75
3.3.7 Backward Time Propagation 76
3.3.8 Delayed Data 77
3.3.9 Asequent Data 82
3.4 Channel Algorithms 85
3.4.1 Management of Communication 87
3.4.2 Structure of the Communication Algorithm 88
3.4.3 Information Flow 90
3.4.4 Computing the Common Information 90
3.5 A Practical Decentralised Node Algorithm 91
3.5.1 Sensor Preprocessing 92
3.5.2 Local Filter 93
3.5.3 Channel Filter 94
3.5.4 Channel Manager 95
3.5.5 Timing 98
3.5.6 Data Association 99
3.6 Unstructured (Dynamic) Network Topologies 101
3.6.1 Limitations of Structured Topologies 101
3.6.2 Broadcast With CI Update 101
3.6.3 Broadcast With Hybrid CI/IF Update 104
3.6.4 Dynamic Tree Structure 106
3.7 Summary 111

4 ANSER Project: Implementation of a Decentralised System 112
4.1 Introduction 112
4.2 The ANSER Project 113
4.2.1 The Flight Vehicle 113
4.3 Sensing Equipment and Hardware 116
4.3.1 GPS 116
4.3.2 Inertial System 116

4.3.3 Camera 117
4.3.4 SIS: Vision/Laser System 118
4.3.5 Radar 119
4.3.6 Processing and Communications 120
4.4 Implementation of the Decentralised Architecture 122
4.4.1 Communications 122
4.4.2 The Decentralised Filter 123
4.5 ANSER Simulator 125
4.5.1 SimCompiler 126
4.5.2 Flight Simulator 126
4.5.3 Mission Planner 128
4.5.4 Sensor Server 129
4.5.5 DDF 130
4.5.6 Picture Compilation 131
4.5.7 Time Server 131
4.6 Summary 131

5 Decentralised Tracking **133**

5.1 Introduction 133
5.2 Problem Definition 134
5.3 The Vision Sensor 136
5.3.1 Sensor Model 136
5.3.2 Feature Extraction 138
5.4 Vision/Laser Sensor: SIS 140
5.5 Track Formulation 141
5.5.1 Process Model 141
5.5.2 Sensor Preprocessing 142
5.5.3 Artificial Targets 145
5.5.4 Track Maintenance 147
5.5.5 Data Association 148
5.5.6 Communication Strategy 149

5.5.7 Data Fusion 150
5.6 Implementation 151
5.6.1 Offline Implementation 151
5.6.2 Offline Implementation Results 152
5.7 Real-Time Implementation 163
5.7.1 Results of First Real-Time Flight Test - Multiple Aircraft DDF . . . 164
5.7.2 Results of Second Real-Time Flight Test - Single Vehicle DDF . . . 175
5.7.3 Results of Third Real-Time Flight Test - Multiple Vehicle DDF . . . 176
5.8 Summary 179

6 Simultaneous Localisation and Map Building 183

6.1 Introduction 183
6.2 Problem Definition 185
6.2.1 The Augmented State 188
6.2.2 Process Model 189
6.2.3 Observation Model 191
6.2.4 Feature Initialisation in State Space 193
6.2.5 Feature Initialisation in Information Space 195
6.3 Closed Form Solutions 196
6.3.1 Platform Models 197
6.3.2 Solutions to the Riccati Equation 199
6.3.3 State Space Solution 200
6.3.4 Information Space Solution for a Single Platform 203
6.3.5 Information Space Solution for Multiple Platforms 204
6.4 Map Information 207
6.4.1 The Information Map 207
6.5 Algorithms for Decentralised SLAM 209
6.5.1 Decentralised Information Space Formulation 209
6.5.2 Decentralised Hybrid Space Formulation 212
6.5.3 Information Space SLAM Vs Hybrid Space SLAM 212
6.6 Scalable Communication with Large Maps 213

6.6.1 Extracting the Submap to Communicate 214
6.6.2 Channel Update 215
6.6.3 Fusing Information from Other Nodes at the Local Filter 215
6.7 Simulation 216
6.7.1 Results 218
6.8 Summary 227

7 Conclusions and Future Work **229**
7.1 Introduction 229
7.2 Summary of Contributions 229
7.2.1 Decentralised Architectures 229
7.2.2 Delayed and Asequent Data 230
7.2.3 Decentralised Tracking 230
7.2.4 Decentralised SLAM 231
7.3 Further Research 232
7.3.1 System Complexity 232
7.3.2 Real Time Decentralised SLAM 232
7.3.3 Decentralised Control 233
7.3.4 Dynamic Networks 233
7.4 Summary 233

A Normalised Innovation: The Information Gate **234**

Bibliography **237**

List of Figures

1.1 Two Brumby Mk.III UAVs. 1

1.2 Fully connected decentralised network with peer to peer architecture 4

1.3 Fully connected decentralised network with bus architecture 5

1.4 The OXNAV modular, decentralised mobile robot 8

2.1 Centralised log-likelihood data fusion architecture 17

2.2 Distributed log-likelihood data fusion using the independent likelihood architecture 18

2.3 Distributed log-likelihood data fusion using the independent opinion pool architecture 19

2.4 Distributed log-likelihood architecture with global information at the sensor 20

2.5 Decentralised log-likelihood architecture 21

2.6 A distributed information filter form of the independent likelihood architecture 42

2.7 A distributed information filter architecture with an estimate of local information at the sensor 43

2.8 The independent opinion pool architecture in its information filter form . . 44

2.9 A decentralised information filter 45

2.10 Covariance Intersect updates 51

3.1 Communicated information sets in the three classes of topology. 63

3.2 Fully connected (broadcast) network topology 68

3.3 Linear peer to peer tree topology 69

3.4 General peer to peer tree topology 69

3.5 Ideal observation timing 78

3.6 Delayed observation timing 78

3.7 Delayed data information example 81
3.8 Asequent observation timing 82
3.9 Asequent data example 84
3.10 Structure of a general communication channel algorithm 88
3.11 Structure of a decentralised node. 92
3.12 Data flow from sensor observation. 93
3.13 Flow of information for incoming channel data in a structured network. . . 96
3.14 Flow of information for outgoing channel data. 97
3.15 Asynchronous channel filter timing 99
3.16 Target ordering at different nodes 100
3.17 A single node in a dynamic tree configuration. 107
3.18 Structure of a dynamic tree network. 108
3.19 Joining two trees together. 110

4.1 The four Brumby Mk.III airframes. 114
4.2 The original Brumby Mk.I airframe. 114
4.3 Brumby Mk.III airframe 115
4.4 A GPS antenna embedded in the Brumby Mk.III wing. 116
4.5 Inertial Science ISIS IMU 117
4.6 A Sony CCS-SONY-HR vision sensor 118
4.7 Typical images from the camera payload 119
4.8 Millimetre wave radar and scanner fitted to the Brumby Mk.III nose. 120
4.9 Functional diagram of the systems on the Brumby platform. 121
4.10 Communications topology used in ANSER project 123
4.11 Virtual channels 124
4.12 Structure of decentralised filter software 125
4.13 Structure of the ANSER simulator 127
4.14 A typical screenshot from the flight simulator GUI 128
4.15 The ANSER simulator sensor server 129
4.16 The decentralised tracking simulator 130
4.17 The ANSER simulator graphical display in playback mode 132

5.1 A single platform tracking multiple ground targets. 134
5.2 Multiple platforms performing decentralised tracking 136
5.3 Principal point, image and sensor axes for the vision payload. 137
5.4 Image, sensor and body axes for the vision payload. 138
5.5 Image showing extracted feature. 138
5.6 Height estimate and error bounds. 140
5.7 Preprocessed vision observations 146
5.8 Peer-to-peer communications topology used for offline implementation. . . . 152
5.9 Map generated by aircraft 1 and 2 when acting independently 153
5.10 Map generated by aircraft 3 and 4 when acting independently 154
5.11 Map generated by aircraft 1 and 2 when acting in DDF network 155
5.12 Map generated by aircraft 3 and 4 when acting in DDF network 156
5.13 Close up of DDF network map on aircraft 1 157
5.14 x position error of a typical target for 4 aircraft acting individually vs DDF network 158
5.15 y position error of a typical target for 4 aircraft acting individually vs DDF network 158
5.16 z position error of a typical target for 4 aircraft acting individually vs DDF network 159
5.17 x position variance of a typical target for 4 aircraft acting individually vs DDF network 159
5.18 y position variance of a typical target for 4 aircraft acting individually vs DDF network 160
5.19 z position variance of a typical target for 4 aircraft acting individually vs DDF network 160
5.20 Maximum, mean and minimum target errors in x 161
5.21 Maximum, mean and minimum target errors in y 161
5.22 Maximum, mean and minimum target errors in z 162
5.23 Peer-to-peer communications architecture for real-time tracking 165
5.24 Map generated by aircraft 1 in real-time tracking 166
5.25 Map generated by aircraft 2 in real-time tracking 167
5.26 Map generated by ground nodes 1 and 2 in during real-time tracking 168
5.27 x axis position error for a typical target during real-time tracking 169

5.28 y axis position error for a typical target during real-time tracking 170
5.29 z axis position error for a typical target during real-time tracking 170
5.30 x axis position variance for a typical target during real-time tracking 171
5.31 y axis position variance for a typical target during real-time tracking 171
5.32 z axis position variance for a typical target during real-time tracking 172
5.33 Maximum, mean and minimum target errors in x during real-time tracking 172
5.34 Maximum, mean and minimum target errors in y during real-time tracking 173
5.35 Maximum, mean and minimum target errors in z during real-time tracking 173
5.36 1 aircraft, 2 node DDF network. 175
5.37 Real-time maps generated at aircraft and ground node 177
5.38 2 aircraft, 5 node DDF network. 178
5.39 Real-time results: Map at ground node connected to aircraft 3. 179
5.40 Real-time maps generated at the vision and SIS nodes on aircraft 3 180
5.41 Real-time maps generated at the vision and ground nodes of aircraft 4 . . . 181

6.1 The SLAM problem 186
6.2 Continuous time SLAM covariance matrix 202
6.3 Multiple platforms using SLAM have a common map 208
6.4 The D-SLAM simulation world 217
6.5 Error and 2σ bounds for vehicle states under the N^2 communication strategy 219
6.6 Error and 2σ bounds for vehicle states under the constant time communication strategy 220
6.7 Error and 2σ bounds for vehicle states under the no communication strategy 221
6.8 Error and 2σ bounds for a typical feature under the N^2 communication strategy 222
6.9 Error and 2σ bounds for a typical feature under the constant time communication strategy 223
6.10 Error and 2σ bounds for a typical feature under the no communication strategy 224
6.11 Covariance ellipse for a feature under the different communication strategies 225
6.12 The mean variance in x, y and ϕ from all vehicles over the period of the simulation for different maximum map sizes 226

List of Tables

4.1 Brumby performance characteristics. 115

4.2 ISIS IMU specifications. 117

5.1 Vision sensor noise statistics. 139

5.2 SIS noise statistics. 140

5.3 Node configuration for post-processed data. 152

5.4 Configuration for the 4 node real-time DDF flight. 164

5.5 Configuration for the 2 node real-time DDF flight. 176

5.6 Configuration for the 5 node real-time DDF flight. 178

6.1 Number of elements to communicate in D-SLAM 218

Chapter 1

Introduction

The aim of this thesis is to develop and demonstrate algorithms for decentralised data fusion (DDF). The algorithms are applied to tracking and feature based navigation problems and are demonstrated in real-time on multiple uninhabited flight vehicles of the type shown in Figure 1.1. This implementation environment is arguably the most demanding arena in which to demonstrate this research due to the dynamics and complexity of flight vehicles.

The DDF algorithms are developed in an information theoretic framework, and presented initially in a general form applicable to any distributed sensing network. These algorithms are then applied to the specific problem of fusing data from a network of airborne sensors. This requires algorithms to deal with rapid data rates, high degrees of uncertainty and intermittent communication between sensor platforms. Algorithms are developed for two central problems; picture compilation in which the network of sensors is required to track

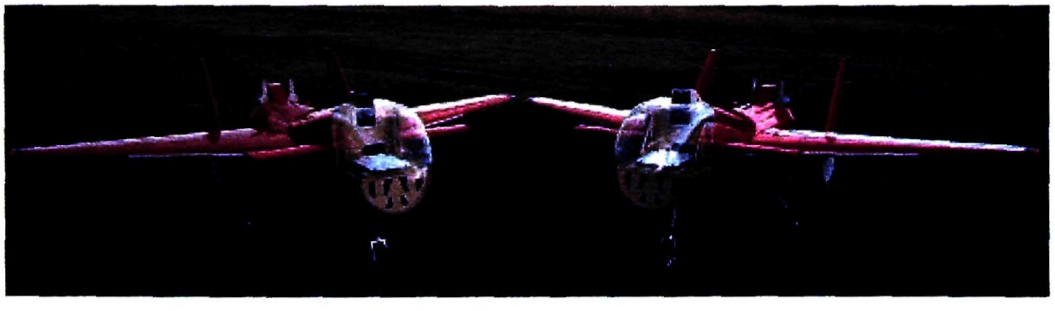


Figure 1.1: Two Brumby Mk.III uninhabited air vehicles (UAVs) capable of carrying a combination of vision and either radar or laser payloads for terrain sensing. These UAVs are used for demonstrating the decentralised data fusion algorithms described in this thesis.

a number of moving ground targets, and Simultaneous Localisation and Mapping (SLAM) in which the network of sensors is required to build a map of terrain landmarks whilst simultaneously using these to determine self location. Both of these problems are addressed in a fully decentralised form. Both simulated and real-time results are of the implementation of DDF algorithms are described. The real-time results in particular describe the first ever use of multiple cooperative uninhabited air vehicles (UAVs).

1.1 Motivation

The goal of the work described in this thesis is to realise practical decentralised data fusion algorithms in real-world multi-sensor data fusion problems. Large, arbitrary networks of heterogeneous intelligent sensor nodes are envisaged. Sensors nodes should exchange information with each other, fusing communicated information with locally observed information to obtain a global picture of the environment of interest. It should be possible to add or remove sensors from the network on-line. It should be possible to scale the network from a few sensors to many thousands. The network should be organic and not require any centralised fusion site, centralised communication medium or any other vulnerable feature. The sensors should be modular and interoperable; there should be no need for other sensors in the network to know how or where a remote sensor gets its information, and a node should be able to change the source of that information without affecting the underlying fusion algorithm of other nodes in the network.

The DDF algorithms described in this thesis go a long way to achieving these goals and vision of network sensing capabilities. A key feature of these algorithms is that they have a sound mathematical basis in ideas of information filtering and information fusion. The resulting algorithms and sensor architectures are broadly applicable to a large number of tasks ranging from map building to tracking and from environment monitoring to security systems, with application in a variety of civilian and military contexts.

There has been a growing realisation over the past decade in the advantages of using sensor network architectures over equivalent centralised or hierarchical structures [10, 30, 31]. Sensor networks offer a degree of flexibility and robustness not possible in more rigid organisations. Sensor networks also offer the possibility of building scalable, modular, and survivable "systems of systems".

The essential algorithms described in this thesis originated over a decade ago. Since this time, the algorithms have been refined, evaluated in simulation and demonstrated in a number of reasonably benign applications including the tracking of people in office environments [85] and in mobile robot navigation [68, 93]. A key objective of this thesis is to take these essential algorithms and develop them for the far more challenging application of decentralised airborne sensor networks. This is possibly the most demanding domain in which to develop these methods. It requires solutions to some important problems in data registration, intermittent and limited communication between sensor sites, and in substantive real-time algorithms for very high data rate sensing devices.

Interest in the use of uninhabited air vehicles (UAVs) is growing at a phenomenal pace. A large number of military systems have been or are being developed. Civilian applications in areas such as geological surveying, search and rescue, bush fire fighting and agricultural monitoring are also being discussed. In military circles, there is a great deal of talk about "network centric warfare". For these reasons the deployment of DDF methods on air vehicles is both timely and appropriate. The system developed in this thesis is one of the first ever developed explicitly for use by multiple UAVs. The DDF methods show how problems of tracking and map building can be deployed and scaled on one and then many UAVs in a complete and scalable network. The results of the application of the DDF algorithms developed in this thesis are the first ever on a true network of UAV systems and demonstrate the first ever cooperative use of multiple UAVs.

1.2 Decentralised Data Fusion

A decentralised data fusion system comprises a network of sensor nodes each incorporating a sensor together with local processing and communications capabilities. Each node runs its own local data fusion algorithm and communicates information to other nodes in its neighbourhood. Nodes fuse incoming information with local estimates to produce a global picture of the state of the world. This strategy results in a network of sensor nodes in which the fusion process is decentralised.

A general decentralised data fusion system can be characterised by three basic constraints:

1. There is no single central fusion centre and no node should be central to the operation of the network.

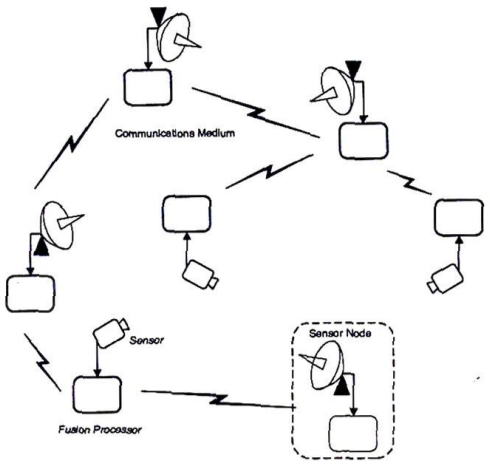


Figure 1.2: A decentralised data fusion structure. Each sensor node incorporates a sensor, local processor and communications capabilities. In the structure communication is on a node-to-node basis and the network topology is arbitrary.

2. There is no common communications facility - communications must be kept on a strictly node-to-node basis.

3. Each node has knowledge only of its immediate neighbours - there is no global knowledge of the network topology.

These constraints combine to ensure that there is no single element that is critical to the operation of the network. If any node or communication link should fail, the result is a gradual degradation in network performance rather than catastrophic failure. Figures 1.2 and 1.3 illustrate two possible architectures for a decentralised system.

1.2.1 The Benefits of a Decentralised Architecture

The conventional structure for multi-sensor systems requires that all sensors send observation information back to a single central point for fusion. This has some advantages in that all information is known at a single point and so, in principle, fusion is based on the best possible information. However centralised structures also have a number of undesirable characteristics: Using a centralised resource for fusion creates both computational and communication bottlenecks as the system increases in size. This limits the ability of a system to scale to large numbers of sensors. A second key problem is that a central fusion site

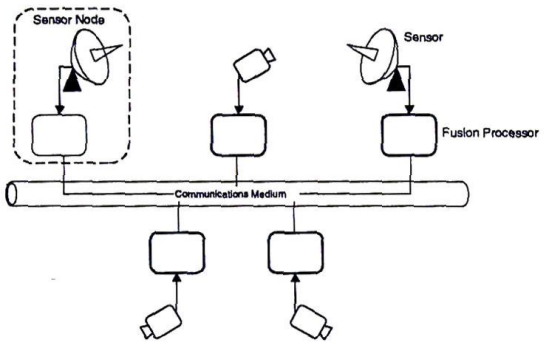


Figure 1.3: A second realisation of a decentralised data fusion architecture. In this structure communication is takes place on a common communications medium (a radio network or a bus). A sensor node may, in principle, talk to any other node in the system. However, the node structure is still such that fusion occurs in a decentralised manner at sensor sites and not at any common fusion centre.

makes the whole system vulnerable to catastrophic failure of the fusion processor. The lack of robustness of centralised architectures is increasingly important as systems become more dependent on the availability of robust and reliable information. A final problem is the rigidity of centralised structures to changes in sensor configurations. Centralised structures do not easily allow sensors to be changed or used in synergistic combination as requirements and tasks vary.

The decentralised system architecture described in this thesis offers a number of advantages over conventional centralised or hierarchical architectures:

- **Scalability**: Eliminating the central fusion centre and any common communication facility ensures that the system is scalable as there are no limits imposed by centralised computational bottlenecks or lack of communication bandwidth.

- **Survivability:** 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.

- **Modularity**: As all fusion processes must take place locally at each sensor site and no global knowledge of the network is required *a priori*, nodes can be constructed and programmed in a modular fashion.

Together these characteristics give decentralised architectures substantial advantages over more conventional data fusion structures.

Interest in the use of networks of modular, locally intelligent, sensors has grown substantially in the last five years. It is recognised that such ad hoc opportunistic structures are better able to deliver the robust and flexible capability required of advanced data fusion systems. This theme has especially been pursued by the defence establishment through the concept of Network-Centric Warfare [1, 66].

A major hurdle in developing network-centric systems is to develop appropriate data fusion algorithms that allow information acquired at a sensor node to be communicated and fused in an efficient and consistent manner. This is the central contribution of this thesis. The algorithms described here are based on the use of information-theoretic concepts of data fusion. The algorithms are based on the development of information-filtering and decentralised estimation undertaken by a number of researchers over the past decade. In this thesis, these are further developed to address issues of communication and computation in real systems with real-time constraints. The algorithms are also demonstrated and evaluated on a network of airborne sensors; one of the most demanding of all network-centric sensor applications.

1.2.2 The Development of the Decentralised Architecture

The potential benefits of decentralised or network-centric systems have been recognised for some time. Research in distributed data fusion algorithms has generally concentrated on hierarchical [51] or parallel systems [22]. These methods generally allow the estimation process to be distributed on a number of local sensor processes often at different physical locations. Algorithms have generally focused on distributing Kalman-filter type estimation algorithms [29, 47, 50]. Of note are track-to-track fusion methods espoused by BarShalom and co-workers [4, 27]. These are of special interest as state tracks from different sensors are necessarily correlated by virtue of the common process being tracked. More recent work has extended these ideas to use full posterior track models [26]. However, all these methods and algorithms continue to require a single central fusion point where sensor data or tracks are fused to create a common representation. Use of a central point for the fusion algorithm immediately reduces the robustness, scalability and flexibility of the system. More general

work in sensor networks is compiled in recent conferences such as IPSN [46]. Although these are "sensor network" conferences, work has focused on communication and data hopping rather than data fusion. The few data fusion algorithms presented generally still require a central fusion site. An interesting example of this is the work being undertaken in the DARPA SensIT programme in which the fusion algorithm is centralised but the fusion site itself hops between sensor nodes [64].

A different approach was developed by the robotics group at Oxford University over a number of years from the late 1980's. Here, the decentralised constraints enumerated in Section 1.2.1, especially the lack of a central fusion site, served as the *starting point* for the development of network data fusion algorithms. The algorithms employed were based on the use of the information form the Kalman filter. The information form has a number of valuable properties in multi-sensor estimation problems that make it particularly suited to decentralisation. These are elucidated in Chapter 2 of this thesis. The initial DDF work undertaken at Oxford demonstrated a practical decentralised tracking system which used multiple cameras to track people and small robots moving through an environment [85–88]. This implementation demonstrated the basic decentralised information fusion algorithm and importantly its equivalence to results produced by a centralised (Kalman-filter based) data fusion system. Initial algorithms required the network to be fully connected; clearly impractical in large networks. The work of Grime [42, 43] made explicit the information form of the Kalman filter and used this to solve the problem of communicating in tree-topology networks. This work employed a structure called the Channel Filter to maintain an estimate of common information between connected nodes. This was a significant conceptual step in identifying the computation of common information as a key problem in network communication. This result was used to demonstrate the operation of decentralised data fusion systems on a model process control plant comprising over 150 distributed sensors.

The decentralised data fusion architecture has been explored and extended in a number of directions [36, 83, 91]. Building on the work of Grime, Utete showed that there is no finite or general solution for optimally identifying common information in sensor networks of arbitrary topology [99–101]. Ways of overcoming this were explored including the use of the distributed Bellman-Ford algorithm to generate spanning trees through which information flow can be managed. Recently however, other researchers in the area have eschewed optimality and developed communication algorithms which use a conservative estimate of

common information to maintain consistency [81, 82]. Such algorithms are especially useful in networks which are subject to highly dynamic changes in communication topology.

The problem of organisation in decentralised systems was addressed by Ho [49]. This work considered how to best configure a decentralised network using nodes with a fixed number of communication links. The performance of all nodes was tracked to determine the effect of different topologies on information flow. For example, in a star connected topology the centre node typically has better results at any given time than the points as it receives information in a more timely manner.

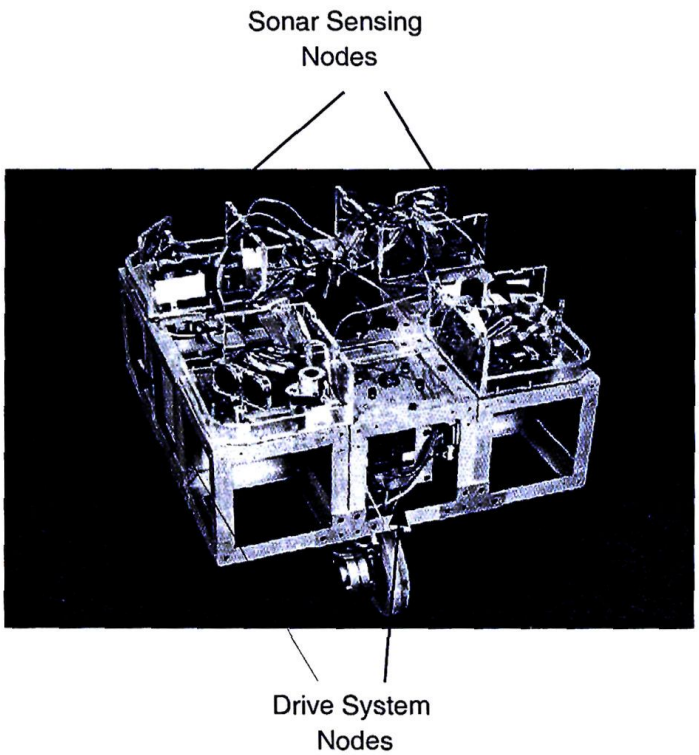


Figure 1.4: The OXNAV indoor mobile robot was a fully modular, decentralised system. The vehicle is constructed of a number of modular cages (nodes), each containing a sensing or actuation system plus power, processing and communication facilities.

The DDF research at Oxford culminated in the development of the OxNav indoor robot shown in Figure 1.4. The vehicle was constructed of modular cages, where each cage contained a specific part of the vehicle sensing or actuation system plus sufficient power, processing and communication facilities to perform local decentralised algorithms. There was

no central processing point where control or navigation algorithms were implemented. The key developments of this project are detailed below.

- The development of modular hardware system was investigated by Burke [20, 21]. The system comprised different hardware modules, each containing an actuator or sensor, with integral processor and decentralised software, connected to other nodes via a high-speed serial line. Using these modules, different vehicles could be built simply by grouping modules for motion, steering, and sensing in different configurations. The communication of information between modules resulted in each having a local estimate of global information, which in turn allowed the steering and drive actuation modules to implicitly coordinate platform motion.

- Research into model distribution in decentralised systems was performed by Berg [11–14]. This concept allows the decentralised network to distribute system models throughout the nodes in the network.

- Manyika used the information-theoretic formulation of the decentralised system for estimation, classification and sensor management [68–70]. The sensing modules on the OxNav vehicle used the DDF information not only to estimate feature/target locations, but to determine which feature/target each sensor should look at in order to maximise information gain.

- The decentralised control problem was investigated by Mutambara [74–76]. This work enabled the OxNav vehicle to calculate trajectories and then follow them using the network of independent actuation modules in a DDF network. Although there was no central processor or global network knowledge, the DDF information allowed the modules to coordinate the actions of the entire vehicle.

The decentralised architecture developed for the OxNav was also used by Deaves to investigate communication management strategies in bandwidth limited problems [33]. A key result of this research was that an information-theoretic approach to communications management out-performs an ad hoc or round-robin approach. The use of information utilities to manage communication in multi-sensor systems was also investigated by Chu [30].

1.3 Objectives of this Thesis

The work in this thesis is part of the Autonomous Navigation and Experimental Research (ANSER) project between the University of Sydney and BAE Systems which aims to demonstrate DDF algorithms on multiple uninhabited aircraft. The specific objectives of this work are:

1. The development of a robust asynchronous decentralised algorithm that is applicable to a variety of different network topologies.
2. The application of the DDF algorithm to the problem of tracking targets from multiple sensing platforms.
3. The application of the DDF algorithm to multiple platform Simultaneous Localisation and Map Building (SLAM).

Previous work on decentralised architectures, discussed in Section 1.2.2, has been limited to more easily managed ground-based applications. These problems were further simplified by using synchronous hardware such as transputers for communication. One of the key objectives of this thesis is to take the step of developing the decentralised algorithms into a form which is readily applicable to complex asynchronous systems and real world problems.

A fundamental problem in distributed or decentralised sensor networks involving intermittent and asynchronous communication is how to temporaly align data prior to fusion. This thesis provides a comprehensive analysis of this problem, and shows how the DDF algorithm can deal with problems of delayed and asequent data transmission. In turn this provides practical solutions to problems of data latency, intermittent and burst communication in sensor networks.

Understanding the structure of communication turns out to be the driver in designing a general purpose architecture for a decentralised sensor node. Within the scope of the ANSER project, this is accomplished in this thesis. The architecture allows for sensor data pre-processing, local prediction and estimation, anonymous communication in a network, time alignment and local assimilation of communicated information. The general node structure is applicable to any decentralised network topology although only restricted subsets of these are employed in the implementations described in this thesis.

The decentralised algorithms and architectures developed in this thesis are applied to two main problems. The first is the classic "tactical picture compilation" problem in which a group of flight vehicles are required to track a number of potentially mobile ground targets within the operating environment. In this case, the location of the flight vehicles is independently obtained from an on-board GPS/INS system. The sensors observe the bearing or range and bearing to targets. A set of filters are established, one for each target, which are used to estimate the position and velocity of the targets. A fully decentralised, multi-sensor multi-air vehicle solution to this task is developed and demonstrated in this thesis. The second application is to the increasingly important problem of map-building and navigation in the absence of GPS or an *a priori* navigation map. This process is known as Simultaneous Localisation and Mapping (SLAM) [35]. It has the potential to provide navigation solutions for platforms operating in unknown GPS-denied environments. The key element in the SLAM problem is the fact that the landmark location estimates generated by the SLAM algorithm are all highly correlated by virtue of the fact that they are estimated from a common vehicle trajectory. The application of decentralised data fusion methods to this problem has a number of features. First it shows the efficacy of the information filter in managing the complexity of exchanging highly correlated target sets between platforms. Second it demonstrates the pay-off in navigation performance that can be obtained by effective sharing of map data between platforms. This thesis presents one of the first effective solutions to the multiple-platform SLAM problem. A combination of theoretical and practical results are used to explore this problem and to demonstrate effective solutions.

1.4 Contributions of this Thesis

The contributions of this thesis are:

- The development of, and exact solution to, both the delayed and asequent data problems for the decentralised information filter.
- The development of a modular sensor architecture permitting scalable asynchronous communication in general networks. A new algorithm for fusion in dynamic tree topologies is also proposed.

- The development and real-time demonstration a decentralised algorithm for tracking ground targets from multiple UAVs.
- Closed form solutions to simple, single degree of freedom, SLAM problems are presented, in covariance and information form, and for single and multiple vehicles. These elucidate the structure of the SLAM problem for multiple platforms.
- A decentralised solution to the multiple vehicle SLAM is presented. This includes methods for managing communication in the light of increasing map dimension.

1.5 Thesis Structure

Chapter 2 presents the mathematical background for the estimation algorithms that are used in this thesis. The fundamental concept of information and its application to multi-sensor estimation is developed.

Chapter 3 considers the structure of decentralised systems and details the algorithms developed for such architectures. A general architecture for a decentralised node is presented and each of its component modules is described. The structure and implications of different communication topologies is also discussed. Communication policies for different network structure are described.

Chapter 4 describes the ANSER system including flight vehicles, sensors, on-board proccssors and ground station. The mapping of the algorithms and architecture presented in Chapter 3 to the ANSER system is also described.

Chapter 5 first describes the decentralised tracking or picture compilation problem. Track models are introduced and presented in full decentralised form. Results of running the decentralised picture compilation system on real data are presented. Results of demonstrating the algorithms in real-time in-flight on multiple flight vehicles are described.

Chapter 6 describes the decentralised multi-vehicle SLAM problem. Closed form solutions to simple variations of this problem are presented and used to develop an understanding of the role of information in building and communicating navigation maps. An algorithm is developed for decentralised SLAM. This algorithm is demonstrated in simulation.

Chapter 7 draws some key conclusions and proposes a number of future directions for research in decentralised systems. The research undertaken in this thesis has gone a long way to realising practical network-centric data fusion systems. In particular the general structure of sensor nodes and communication policies has been elucidated. However, the results are also limited to fairly restrictive estimation algorithms. Key challenges in the future will be to extend these ideas to incorporating more diverse and hetrogeneous information.

Chapter 2

Decentralised Estimation Algorithms

2.1 Introduction

This Chapter describes the algorithms used in decentralised data fusion and discusses their properties. The Chapter begins with an overview of Bayesian estimation in Section 2.2. Bayes Theorem is presented in Section 2.2.1 then converted to its log-likelihood form in Section 2.2.2. This log-likelihood form is then used to develop distributed and decentralised estimation algorithms in Section 2.2.3.

The Kalman filter is briefly introduced in Section 2.3, then presented in its information or inverse covariance form in Section 2.4. The information form of the Kalman filter forms the basis of the decentralised estimation algorithms developed in this research. Section 2.4.3 illustrates the relationship between the information filter and Bayes Theorem by deriving the former directly from the latter. The multi-sensor information filter is then introduced in Section 2.4.4, and applied to distributed and decentralised architectures in Sections 2.4.5 and 2.4.6 respectively. The Extended Information Filter (EIF), used in non-linear estimation problems, is then presented in Section 2.4.7. As the concept of information is a key element in the decentralised architectures of this work, Section 2.4.8 introduces various information metrics which are used to quantify the 'knowledge' contained in an estimate.

The Chapter concludes with a discussion of the Covariance Intersect algorithm in Sec-

tion 2.5. This is included due to its importance in decentralised data fusion methods.

2.2 Bayesian Estimation

This section describes the process of probabilistic data fusion using Bayes theorem. Equivalent log-likelihood forms of these data fusion methods are derived and used to develop distributed and decentralised data fusion architectures.

2.2.1 Bayes Theorem

Define a joint probability distribution $P(\mathbf{x}, \mathbf{z})$ on two random variables $\mathbf{x}$ and $\mathbf{z}$ [84]. The chain rule of conditional probabilities can be used to write

$$P(\mathbf{x}, \mathbf{z}) = P(\mathbf{x} \mid \mathbf{z})P(\mathbf{z}) \tag{2.1}$$

$$= P(\mathbf{z} \mid \mathbf{x})P(\mathbf{x}). \tag{2.2}$$

Rearranging these equations gives Bayes Theorem, [71, 84].

$$P(\mathbf{x} \mid \mathbf{z}) = \frac{P(\mathbf{z} \mid \mathbf{x})P(\mathbf{x})}{P(\mathbf{z})}. \tag{2.3}$$

This simple formula is extremely important in probabilistic estimation theory. Consider the problem of estimating the likelihood of some state of interest $\mathbf{x} \in \mathcal{X}$. Prior belief about the likelihood of the state $\mathbf{x}$ is captured in the distribution $P(\mathbf{x})$. When an observation $\mathbf{z} \in \mathcal{Z}$ of the state is made, it is modeled using the distribution $P(\mathbf{z} \mid \mathbf{x})$ which describes the likelihood of each $\mathbf{z} \in \mathcal{Z}$ for each fixed state $\mathbf{x} \in \mathcal{X}$. Bayes Theorem provides a way to calculate the posterior distribution $P(\mathbf{x} \mid \mathbf{z})$, describing the likelihood of the state $\mathbf{x}$ given the observation $\mathbf{z}$. The distribution $P(\mathbf{z})$ is used to normalise the posterior.

Bayes Theorem can be applied to the problem of recursively computing the likelihood of a state given a sequence of observations $\mathbf{Z}^k = z(1), z(2), \cdots, z(k)$ as

$$P(\mathbf{x} \mid \mathbf{Z}^k) = \frac{P(\mathbf{z}(k) \mid \mathbf{x})P(\mathbf{x} \mid \mathbf{Z}^{k-1})}{P(\mathbf{z}(k) \mid \mathbf{Z}^{k-1})}. \tag{2.4}$$

$P(\mathbf{x} \mid \mathbf{Z}^k)$ can now be interpreted as the likelihood of the state $\mathbf{x}$ conditioned on $\mathbf{Z}^k$, the set of all k observations. $P(\mathbf{z}(k) \mid \mathbf{x})$ is the likelihood function for the k^{th} observation, and $P(\mathbf{x} \mid \mathbf{Z}^{k-1})$ is the prior distribution of the likelihood of the state $\mathbf{x}$. The prior is simply the likelihood the state $\mathbf{x}$ conditioned on $\mathbf{Z}^{k-1}$, the set of observations up to $k-1$. The current observation can now be fused with the prior distribution in a simple manner. This process can be applied recursively at future times by using the posterior $P(\mathbf{x} \mid \mathbf{Z}^k)$ as the prior for the next observation. In most circumstances, the prediction of state requires application of a prediction equation which in turn invokes a convolution of posterior with a probabilistic motion model.

It is straight-forward to extend the use of Bayes Theorem to the fusion of observations from different sensors at a common time. Consider, for example, two observations $\mathbf{z}_i$ and $\mathbf{z}_j$ from two sensors i and j respectively. It is assumed that these two observations are conditionally independent given the true state being observed

$$P(\mathbf{z}_i, \mathbf{z}_j \mid \mathbf{x}) = P(\mathbf{z}_i \mid \mathbf{x})P(\mathbf{z}_j \mid \mathbf{x}).$$

In Bayes theorem, this information is fused as

$$P(\mathbf{x} \mid \mathbf{z}_i, \mathbf{z}_j) = C.P(\mathbf{x})P(\mathbf{z}_i \mid \mathbf{x})P(\mathbf{z}_j \mid \mathbf{x}). \tag{2.5}$$

Practically, individual observation models as functions of both $\mathbf{z}$ and $\mathbf{x}$ are stored in the form $P(\mathbf{z}_i \mid \mathbf{x})$. When a specific observation is made, these models are instantiated with a measured value of z and so generate a function, a likelihood, on $\mathbf{x}$ alone. Equation 2.5 then describes the simple product of three functions, defined on $\mathbf{x}$, appropriately normalised to yield the posterior $P(\mathbf{x} \mid \mathbf{z}_i = z_i, \mathbf{z}_j = z_j)$.

2.2.2 Log-Likelihoods

The natural logarithm of a probability density is termed the log-likelihood. In particular

$$l(\mathbf{x}) \triangleq \log P(\mathbf{x}), \qquad l(\mathbf{x} \mid \mathbf{z}) \triangleq \log P(\mathbf{x} \mid \mathbf{z}), \tag{2.6}$$

are the log-likelihood and conditional log-likelihood respectively. It is often more convenient to work with log-likelihoods rather than the probability distributions as:

1. Multiplication and division operations become simple addition and subtraction. This can reduce the computation required in a fusion algorithm considerably.

2. The log of a distribution is closely related to the amount of 'information' that a distribution contains. Information metrics will be discussed in detail in Section 2.4.8.

The log-likelihood form of Bayes Theorem is simply

$$\begin{aligned} \log P(\mathbf{x} \mid \mathbf{z}) &= \log \frac{P(\mathbf{z} \mid \mathbf{x})P(\mathbf{x})}{P(\mathbf{z})} \\ l(\mathbf{x} \mid \mathbf{z}) &= l(\mathbf{z} \mid \mathbf{x}) + l(\mathbf{x}) - l(\mathbf{z}). \end{aligned} \tag{2.7}$$

As the number of sensors in a data fusion system increases, the log-likelihood formulation of the Bayesian update becomes significantly more attractive as the additive update is computationally simple. Figure 2.1 illustrates a simple centralised log-likelihood data fusion architecture where multiple sensors send their observations to a centralised processor. The central data fusion unit stores the likelihood functions $P(\mathbf{z} \mid \mathbf{x})$ for each sensor in the form of an array. Once an observation is made, these likelihood functions are instantiated with an observation $P(\mathbf{z}_i(k) \mid \mathbf{x})$ resulting in a likelihood on the underlying state $\mathbf{x}$.

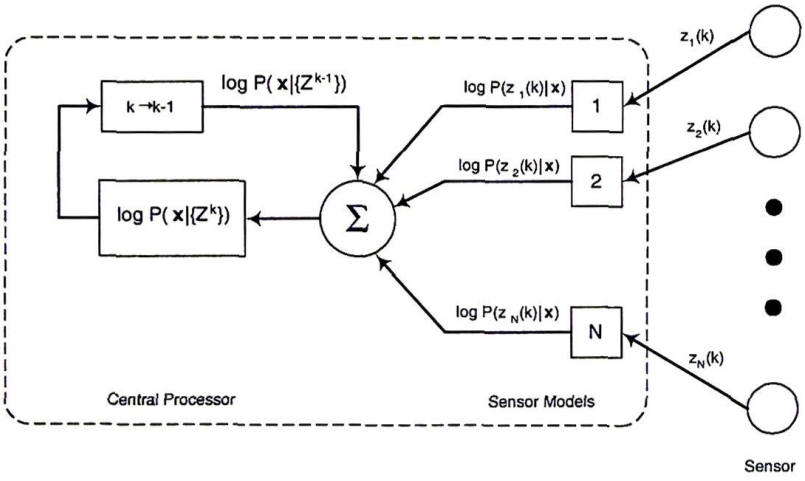


Figure 2.1: Centralised log-likelihood data fusion architecture. Raw observations are transmitted directly to the central fusion algorithm.

2.2.3 Distributed Bayesian Estimation

It is straightforward to construct a distributed Bayesian data fusion architecture exploiting the additive log-likelihood update. The simplist of these architectures, known as the independent likelihood model, is illustrated in Figure 2.2. In this architecture, the likelihood models for each sensor are stored locally at the sensor site. When an observation is made, these likelihoods are instantiated producing a likelihood function on the underlying state $\mathbf{x}$. The log of these likelihoods is communicated to the central processor where they are summed and normalised to produce a posterior incorporating information from all sensors. This architecture has the singular advantage that all sensors talk to the central processor only in terms of the common underlying state $\mathbf{x}$. The specifics of the observation and the model of the sensor itself are hidden from the central fusion centre. To a degree, some of the computational effort required to generate the necessary likelihoods is also undertaken away from the central processor. However, there is in general a cost in the substantially increased amount of total communication required. Figure 2.1 requires the communication of an observation, Figure 2.2 requires the communication of a likelihood function.

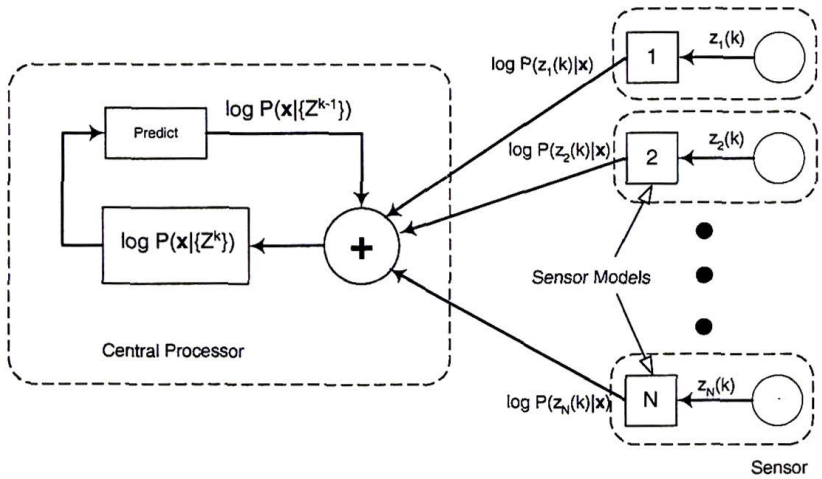


Figure 2.2: Distributed log-likelihood data fusion using the independent likelihood architecture. The sensors compute the log of the sensor likelihood function locally and communicate this to the central fusion algorithm.

Figure 2.3 illustrates another distributed architecture employing Bayes Theorem in its log-likelihood form. This architecture, known as the independent opinion pool, has each sensor maintaining its own models and communicating the log of their local posterior (log-

posterior) to the central fusion point. The log of the global posterior $l(\mathbf{x} \mid \mathbf{Z}^k)$ is therefore available at the central processor, while each sensing node i has a local log-posterior $l(\mathbf{x} \mid \mathbf{Z}_i^k)$ conditioned only on local observations.

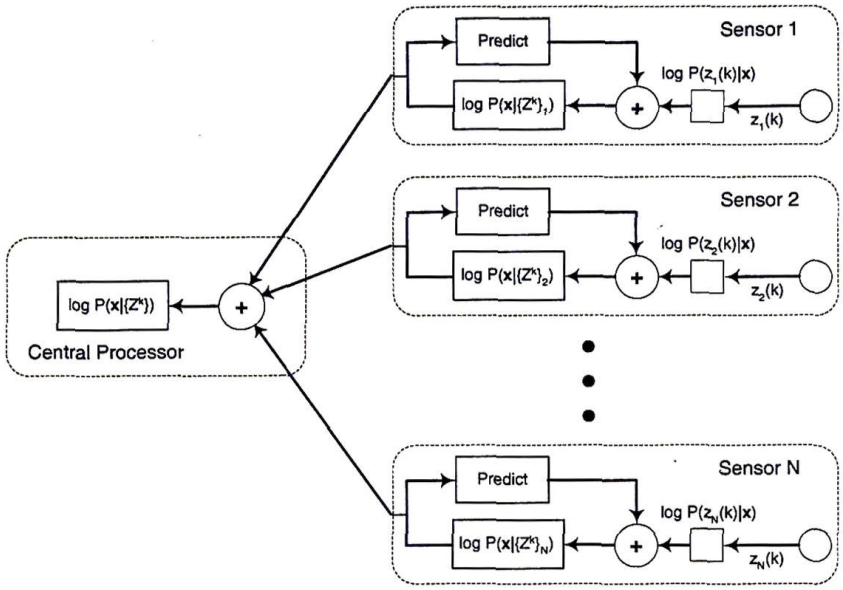


Figure 2.3: Distributed log-likelihood data fusion using the independent opinion pool architecture. The central processor has a global log-posterior conditioned on all observations in the system, while the local log-posterior at each sensing node is conditioned only on local observations.

Maintaining a local log-posterior endows each sensor site with the information necessary to become a little more 'intelligent'. In particular, this information may allow a degree of smart signal processing including frame-to-frame data association, sensor control and power management. The limitations are that each node in this architecture has access only to its own local information. Figure 2.4 shows an architecture in which each sensor site has access to the global posterior log-likelihood. Here the sensors, as before, transmit log-likelihoods to a central fusion point, where the summation yields the global log-posterior. However, in this case, the resulting global log-posterior is then transmitted back to the sensors, which can locally determine the contributions from the rest of the system by simply subtracting their local log-posterior from the global log-posterior. The subtraction is important as it avoids double-counting of local information. In this architecture, each sensor site ends up with the same global posterior information and so can make local decisions based on global information. Importantly, all communications between sensors and fusion centre are in the

form of a common state-space.

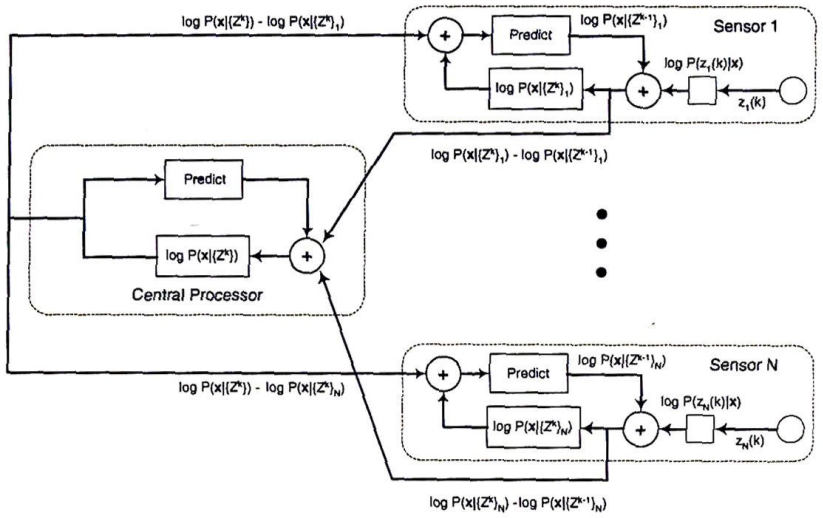


Figure 2.4: A distributed log-likelihood data fusion architecture which has the global log-posterior available to the sensing modules.

A major flaw with all these architectures is that they rely on a single site where information from all sensors is fused. This imposes communication and computational bottle-necks in the system. More fundamentally it also makes the system as a whole vulnerable to failure of the fusion centre leading to catastrophic failure of the system as a whole. Figure 2.5 shows a fully decentralised architecture formed by replicating the function of the central fusion processor locally at each sensor site. In this architecture, each sensor now communicates log-likelihood data to all other nodes rather than to a central processor. Each sensing node sums its own posterior log-likelihood with the contributions communicated by remote nodes. Provided the initial conditions at each node are the same and each node implements the same motion or prediction model, each sensor site will end up with the same global posterior likelihood. Failure of any one sensor site in this architecture will result in a commensurate reduction in system performance, but failure of any component will not lead to catastrophic failure of the system as a whole.

This basic structure underlies the decentralised architecture and decentralised data fusion algorithms developed in this thesis. The practical realisation of such architectures and algorithms requires considerable attention be paid to issues of communication, timing and consistency in the computation of the log-posterior. Nevertheless, the process of com-

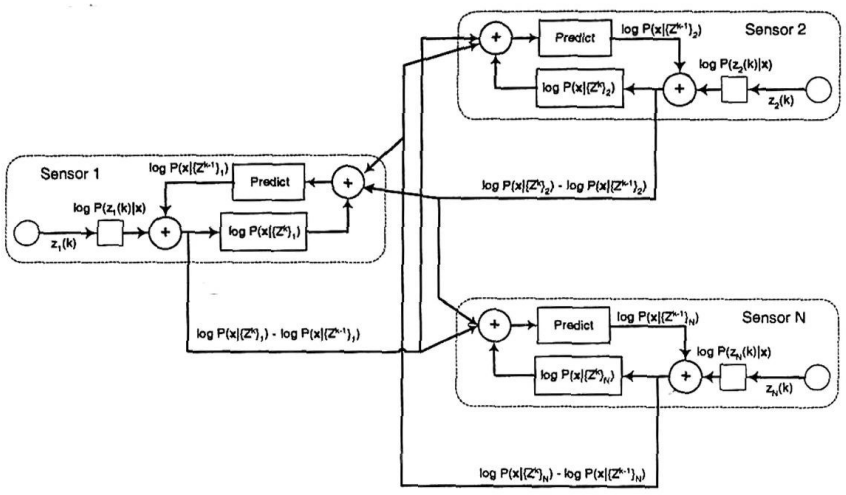


Figure 2.5: A decentralised log-likelihood architecture in which each node has a local log-posterior conditioned on global information. No node is critical to the operation of the network.

municating and summing log-likelihoods drives the essential properties of modularity and network scalability inherent in the methods described in this thesis.

2.3 The Kalman Filter

The Kalman filter is a recursive linear estimation algorithm which serves to minimise the mean squared error between the estimated and true value of a continuous valued state $\mathbf{x}$ [19, 38, 56, 57]. The Kalman filter is one of the most widely used estimation algorithms, not least because it is easy to implement on a digital computer in a simple recursive form. The statistical foundations of the Kalman filter enable the results of the algorithm to be compared and quantified. An explicit description of the process and observations in the algorithm also enables it to be used with a variety of different models.

The Kalman filter is Bayesian and may be derived from Bayes Theorom on the assumption that all distributions are Gaussian and all models linear. In this case, the Kalman filter is essentially a recursive means of computing the mean and variance of the posterior distribution. The Kalman filter is derived here only to define and explain the notation used subsequently in this thesis.

2.3.1 Motion and Observation Models

Consider a system described by the discrete time model

$$\mathbf{x}(k) = \mathbf{F}_k\mathbf{x}(k-1) + \mathbf{B}_k\mathbf{u}(k) + \mathbf{G}_k\mathbf{w}(k), \tag{2.8}$$

where $\mathbf{x}(k)$ is the state of interest, $\mathbf{u}(k)$ is the control input vector and $\mathbf{w}(k)$ a zero mean, white noise sequence with variance $\mathbf{Q}_k$ over the period t_{k-1} to t_k. The matricies $\mathbf{F}_k$, $\mathbf{B}_k$ and $\mathbf{G}_k$ define the propagation of the state, control and noise inputs respectively over the period t_{k-1} to t_k.

Observations of the state are made asynchronously according to a model in the form

$$\mathbf{z}(k) = \mathbf{H}_k\mathbf{x}(k) + \mathbf{v}(k), \tag{2.9}$$

where $\mathbf{v}(k)$ is taken to be a zero mean white noise sequence with covariance $\mathbf{R}_k$ at time k. Define the set of all observations up to k as

$$\mathbf{Z}^k \triangleq \{\mathbf{z}(1), \mathbf{z}(2), \cdots, \mathbf{z}(k)\} . \tag{2.10}$$

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

$$\hat{\mathbf{x}}(i \mid j) \triangleq \mathrm{E}\{\mathbf{x}(t_i) \mid \mathbf{Z}^j\} , \qquad \mathbf{P}(i \mid j) \triangleq \mathrm{E}\{(\mathbf{x}(t_i) - \hat{\mathbf{x}}(i \mid j))(\mathbf{x}(t_i) - \hat{\mathbf{x}}(i \mid j))^T \mid \mathbf{Z}^j\} , \tag{2.11}$$

where $\hat{\mathbf{x}}(i \mid j)$ is the conditional mean, the estimate that minimises mean squared error.

2.3.2 The Kalman Filter Algorithm

The Kalman filter algorithm has two stages. The algorithm starts with an estimate $\hat{\mathbf{x}}(k-1 \mid k-1)$ of the state at time $k-1$ given observations up to time $k-1$ together a corresponding covariance $\mathbf{P}(k-1 \mid k-1)$ in this estimate. The first stage of the Kalman filter algorithm uses the motion model of Equation 2.8 to generates a *prediction* $\hat{\mathbf{x}}(k \mid k-1)$ of the state at time k given observations only up to time $k-1$ together with a corresponding

covariance $\mathbf{P}(k \mid k-1)$ in this prediction according to [71]

$$\hat{\mathbf{x}}(k \mid k-1) = \mathbf{F}_k \hat{\mathbf{x}}(k-1 \mid k-1) + \mathbf{B}_k \mathbf{u}(k) \tag{2.12}$$

$$\mathbf{P}(k \mid k-1) = \mathbf{F}_k \mathbf{P}(k-1 \mid k-1) \mathbf{F}_k^T + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^T. \tag{2.13}$$

At time k an observation $\mathbf{z}(k)$ of the true state is made according to the sensor model described in Equation 2.9. The second stage of the algorithm then updates the state prediction to create a new estimate $\hat{\mathbf{x}}(k \mid k)$ of the state at time k given all information up to and including the observation at time k together with a corresponding covariance in this estimate. The update is computed as a weighted sum according to

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}_k \left[\mathbf{z}(k) - \mathbf{H}_k \hat{\mathbf{x}}(k \mid k-1)\right] \tag{2.14}$$

$$\mathbf{P}(k \mid k) = \mathbf{P}(k \mid k-1) - \mathbf{W}_k \mathbf{S}_k \mathbf{W}_k^T. \tag{2.15}$$

The term $[\mathbf{z}(k) - \mathbf{H}_k \hat{\mathbf{x}}(k \mid k-1)]$ is the error between predicted and actual observation and is referred to as the innovation. Equation 2.14 can therefore be seen to be in a "predictor-corrector" form. The weighting or gain matrix $\mathbf{W}_k$ is selected to minimise the mean squared error in the estimate;

$$\mathbf{W}_k = \mathbf{P}(k \mid k-1) \mathbf{H}_k^T \mathbf{S}_k^{-1}, \tag{2.16}$$

where

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}(k \mid k-1) \mathbf{H}_k^T + \mathbf{R}_k. \tag{2.17}$$

is the innovation covariance.

The algorithm is summarised for future reference :

Kalman Filter Equations

The propagation of the state and covariance is given by

$$\begin{aligned}\hat{\mathbf{x}}(k \mid k-1) &= \mathbf{F}_k \hat{\mathbf{x}}(k-1 \mid k-1) + \mathbf{B}_k \mathbf{u}(k) \\ \mathbf{P}(k \mid k-1) &= \mathbf{F}_k \mathbf{P}(k-1 \mid k-1)\mathbf{F}_k^T + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^T\end{aligned}$$

The update of the state and covariance is given by

$$\begin{aligned}\hat{\mathbf{x}}(k \mid k) &= \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}_k \left[\mathbf{z}(k) - \mathbf{H}_k \hat{\mathbf{x}}(k \mid k-1)\right] \\ \mathbf{P}(k \mid k) &= \mathbf{P}(k \mid k-1) - \mathbf{W}_k \mathbf{S}_k \mathbf{W}_k^T\end{aligned}$$

where

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

and

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

The following example introduces a simple problem that will be used to explore later derivations of the information filter and decentralised data fusion algorithms:

Example 1 *Consider a simple example of the motion of a particle in one dimension. The particle is assumed to move with 'constant' velocity and so it's state can be defined only by its position and velocity.*

$$\hat{\mathbf{x}}(k \mid k) = \begin{bmatrix} \hat{x}(k \mid k) \\ \hat{\dot{x}}(k \mid k) \end{bmatrix}$$

The particle is subject to random forces giving rise to acceleration which are zero mean and white with variance q. The appropriate discrete-time motion models for the particle are thus [17, 19]:

$$\mathbf{F}_k = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \qquad \mathbf{G}_k = \begin{bmatrix} \frac{\Delta T^2}{2} \\ \Delta T \end{bmatrix} \qquad \mathbf{Q}_k = \begin{bmatrix} q \end{bmatrix}$$

The (constant) integration time is taken to be one second $\Delta T = 1$, and the acceleration variance is chosen to be $q = 0.01 m^2/s^4$

If the state estimate and covariance at time $k-1$ are given by

$$\hat{\mathbf{x}}(k-1 \mid k-1) = \begin{bmatrix} 10 \\ 1 \end{bmatrix}, \qquad \mathbf{P}(k-1 \mid k-1) = \begin{bmatrix} 2 & 0.2 \\ 0.2 & 1 \end{bmatrix},$$

the propagation from $k-1$ to k can be calculated by applying Equations 2.12 and 2.13.

$$\begin{aligned}
\hat{\mathbf{x}}(k \mid k-1) &= \mathbf{F}_k \hat{\mathbf{x}}(k-1 \mid k-1) \\
&= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} 11 \\ 1 \end{bmatrix} \\
\mathbf{P}(k \mid k-1) &= \mathbf{F}_k \mathbf{P}(k-1 \mid k-1) \mathbf{F}_k^T + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k \\
&= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0.2 \\ 0.2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix} \begin{bmatrix} 0.01 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 1 \end{bmatrix} \\
&= \begin{bmatrix} 3.4025 & 1.205 \\ 1.205 & 1.01 \end{bmatrix}
\end{aligned}$$

Consider a sensor which observes the position of the target of interest. The observation model for this sensor can be written as

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \mathbf{R}_k = \begin{bmatrix} r_x \end{bmatrix}.$$

Let $r_x = 0.5$ and assume an observation $\mathbf{z}(k)$=11.5 is made. The innovation covariance and Kalman gain can now be calculated using Equations 2.17 and 2.16 respectively.

$$\begin{aligned}
\mathbf{S}_k &= \mathbf{H}_k \mathbf{P}(k \mid k-1) \mathbf{H}_k^T + \mathbf{R}_k \\
&= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 3.4025 & 1.205 \\ 1.205 & 1.01 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.5 \end{bmatrix} \\
&= \begin{bmatrix} 3.9025 \end{bmatrix} \\
\mathbf{W}_k &= \mathbf{P}(k \mid k-1) \mathbf{H}_k^T \mathbf{S}_k^{-1} \\
&= \begin{bmatrix} 3.4025 & 1.205 \\ 1.205 & 1.01 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 3.9025 \end{bmatrix}^{-1}
\end{aligned}$$

$$= \begin{bmatrix} 0.8719 \\ 0.3088 \end{bmatrix}$$

The compete state update can now be performed using Equations 2.14 and 2.15.

$$\begin{aligned}
\hat{\mathbf{x}}(k \mid k) &= \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}_k \left[\mathbf{z}(k) - \mathbf{H}_k \hat{\mathbf{x}}(k \mid k-1)\right] \\
&= \begin{bmatrix} 11 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.8719 \\ 0.3088 \end{bmatrix} \left(\begin{bmatrix} 11.5 \end{bmatrix} - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 11 \\ 1 \end{bmatrix} \right) \\
&= \begin{bmatrix} 11.4359 \\ 1.1544 \end{bmatrix} \\
\mathbf{P}(k \mid k) &= \mathbf{P}(k \mid k-1) - \mathbf{W}_k \mathbf{S}_k \mathbf{W}_k^T \\
&= \begin{bmatrix} 3.4025 & 1.205 \\ 1.205 & 1.01 \end{bmatrix} - \begin{bmatrix} 0.8719 \\ 0.3088 \end{bmatrix} \begin{bmatrix} 3.9025 \end{bmatrix} \begin{bmatrix} 0.8719 \\ 0.3088 \end{bmatrix} \\
&= \begin{bmatrix} 0.4359 & 0.1544 \\ 0.1544 & 0.6379 \end{bmatrix}
\end{aligned}$$

2.3.3 The Extended Kalman Filter

The discrete-time non-linear process or motion model is written in the form

$$\mathbf{x}(k) = \mathbf{f}(\mathbf{x}(k-1), \mathbf{u}(k)) + \mathbf{G}_k \mathbf{w}(k). \tag{2.18}$$

The discrete time non-linear observation model is written in the form

$$\mathbf{z}(k) = \mathbf{h}(\mathbf{x}(k)) + \mathbf{v}(k). \tag{2.19}$$

In both models it is assumed the noise is added in a linear manner. The EKF proceeds by linearising the process and observation models about the predicted state, giving rise to a new set of models which are essentially linear in the state error. These, now linear, models can be used in the conventional linear Kalman filter equations.

Practically, the prediction step of the EKF is implemented in the form [71]

$$\hat{\mathbf{x}}(k \mid k-1) = \mathbf{f}\left(\hat{\mathbf{x}}(k-1 \mid k-1), \mathbf{u}(k)\right) \tag{2.20}$$

$$\mathbf{P}(k \mid k-1) = \nabla_{\mathbf{x}}\mathbf{f}(k)\mathbf{P}(k-1 \mid k-1)\nabla_{\mathbf{x}}^{\mathbf{T}}\mathbf{f}(k) + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^T, \tag{2.21}$$

where $\nabla_{\mathbf{x}}\mathbf{f}(k)$ is the Jacobian of the function $\mathbf{f}(\mathbf{x}(k-1), \mathbf{u})$ with respect to $\mathbf{x}(k-1)$ evaluated at $\mathbf{x}(k-1) = \hat{\mathbf{x}}(k-1 \mid k-1)$ and with assumed known control input $\mathbf{u}(k)$.

The update step of the EKF is implemented as:

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}_k\left[\mathbf{z}(k) - \mathbf{h}\left(\hat{\mathbf{x}}(k \mid k-1)\right)\right] \tag{2.22}$$

$$\mathbf{P}(k \mid k) = \mathbf{P}(k \mid k-1) - \mathbf{W}_k\mathbf{S}_k\mathbf{W}_k^T, \tag{2.23}$$

where the weighting is given by

$$\mathbf{W}_k = \mathbf{P}(k \mid k-1)\nabla_{\mathbf{x}}^{\mathbf{T}}\mathbf{h}(k)\mathbf{S}_k^{-1}. \tag{2.24}$$

The innovation covariance is

$$\mathbf{S}_k = \nabla_{\mathbf{x}}\mathbf{h}(k)\mathbf{P}(k \mid k-1)\nabla_{\mathbf{x}}^{\mathbf{T}}\mathbf{h}(k) + \mathbf{R}_k, \tag{2.25}$$

where $\nabla_{\mathbf{x}}\mathbf{h}(k)$ is the Jacobian of the function $\mathbf{h}(\mathbf{x}(k))$ with respect to $\mathbf{x}(k)$ evaluated at $\mathbf{x}(k) = \hat{\mathbf{x}}(k \mid k-1)$.

There are many caveats that should be enumerated when stating the extended Kalman filter especially concerned with the pit-falls of linearisation and the consequent stability and consistency of the algorithm. These will not concern us in this thesis and so will not be described here (the reader is referred to [19, 38, 71]).

Extended Kalman Filter Equations

The propagation of the state and covariance is given by

$$\begin{aligned}\hat{\mathbf{x}}(k \mid k-1) &= \mathbf{f}\left(\hat{\mathbf{x}}(k-1 \mid k-1), \mathbf{u}(k)\right) \\ \mathbf{P}(k \mid k-1) &= \nabla_{\mathbf{x}}\mathbf{f}(k)\mathbf{P}(k-1 \mid k-1)\nabla_{\mathbf{x}}^{\mathbf{T}}\mathbf{f}(k) + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^T\end{aligned}$$

The update of the state and covariance is given by

$$\begin{aligned}\hat{\mathbf{x}}(k \mid k) &= \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}_k\left[\mathbf{z}(k) - \mathbf{h}\left(\hat{\mathbf{x}}(k \mid k-1)\right)\right] \\ \mathbf{P}(k \mid k) &= \mathbf{P}(k \mid k-1) - \mathbf{W}_k\mathbf{S}_k\mathbf{W}_k^T\end{aligned}$$

where

$$\mathbf{W}_k = \mathbf{P}(k \mid k-1)\nabla_{\mathbf{x}}^{\mathbf{T}}\mathbf{h}(k)\mathbf{S}_k^{-1},$$

and

$$\mathbf{S}_k = \nabla_{\mathbf{x}}\mathbf{h}(k)\mathbf{P}(k \mid k-1)\nabla_{\mathbf{x}}^{\mathbf{T}}\mathbf{h}(k) + \mathbf{R}_k.$$

2.4 The Information Filter

The inverse covariance or information filter is a numerically identical algorithm to the Kalman filter. However, instead of working directly with the states, it recasts the estimates in terms of information quantities.

2.4.1 State Update in Information Form

The update equations for the information filter can be derived by writing the update stage of the Kalman filter in the form,

$$\hat{\mathbf{x}}(k \mid k) = \left[\mathbf{1} - \mathbf{W}_k\mathbf{H}_k\right]\hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}_k\mathbf{z}(k) \tag{2.26}$$

$$\mathbf{P}(k \mid k) = \left[\mathbf{1} - \mathbf{W}_k\mathbf{H}_k\right]\mathbf{P}(k \mid k-1)\left[\mathbf{1} - \mathbf{W}_k\mathbf{H}_k\right]^T + \mathbf{W}_k\mathbf{R}_k\mathbf{W}_k^T, \tag{2.27}$$

and noting that [71]

$$\mathbf{1} - \mathbf{W}_k\mathbf{H}_k = \mathbf{P}(k \mid k)\mathbf{P}^{-1}(k \mid k-1), \tag{2.28}$$

and

$$\mathbf{W}_k = \mathbf{P}(k \mid k)\mathbf{H}_k\mathbf{R}_k^{-1}. \tag{2.29}$$

Substituting Equations 2.28 and 2.29 into Equation 2.26 gives

$$\hat{\mathbf{x}}(k \mid k) = \mathbf{P}(k \mid k)\mathbf{P}^{-1}(k \mid k-1)\hat{\mathbf{x}}(k \mid k-1) + \mathbf{P}(k \mid k)\mathbf{H}_k\mathbf{R}_k^{-1}\mathbf{z}(k). \tag{2.30}$$

Pre-multiplying Equation 2.30 by $\mathbf{P}^{-1}(k \mid k)$ gives the information state update as

$$\mathbf{P}^{-1}(k \mid k)\hat{\mathbf{x}}(k \mid k) = \mathbf{P}^{-1}(k \mid k-1)\hat{\mathbf{x}}(k \mid k-1) + \mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{z}(k). \tag{2.31}$$

To obtain the covariance update, substitute Equations 2.28 and 2.29 into Equation 2.27 and rearrange terms to give

$$\mathbf{P}^{-1}(k \mid k) = \mathbf{P}^{-1}(k \mid k-1) + \mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{H}_k. \tag{2.32}$$

The state estimate and covariance matrix can now be rewritten in terms of two new quantities, the information vector $\hat{\mathbf{y}}(\cdot \mid \cdot)$ and information matrix $\mathbf{Y}(\cdot \mid \cdot)$. These are defined as

$$\hat{\mathbf{y}}(i \mid j) \triangleq \mathbf{P}^{-1}(i \mid j)\hat{\mathbf{x}}(i \mid j), \qquad \mathbf{Y}(i \mid j) \triangleq \mathbf{P}^{-1}(i \mid j). \tag{2.33}$$

Defining the information associated with an observation taken at k to be

$$\mathbf{i}(k) \triangleq \mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{z}(k), \qquad \mathbf{I}(k) \triangleq \mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{H}_k, \tag{2.34}$$

the update Equations 2.31 and 2.32 can be written in the strikingly simple form

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k) \tag{2.35}$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \mathbf{I}(k). \tag{2.36}$$

It should be re-iterated that the information update Equations 2.35 and 2.36 are numerically identical to Equations 2.14–2.17 describing the update for states. The cost for a far simpler update stage is a commensurate increase in the complexity of the prediction stage for information states.

2.4.2 Prediction of Information

The prediction stage of the information filter can also be derived in a straightforward manner from the Kalman filter equations. The matrix inversion lemma in the form

$$[\mathbf{A}+\mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left[\mathbf{DA}^{-1}\mathbf{B}+\mathbf{C}^{-1}\right]^{-1}\mathbf{DA}^{-1}, \tag{2.37}$$

is employed. Making the following substitutions from the Kalman filter covariance prediction given in Equation 2.13

$$\mathbf{A} = \mathbf{F}_k\mathbf{P}(k-1 \mid k-1)\mathbf{F}_k^T \qquad \mathbf{B} = \mathbf{G}_k \qquad \mathbf{C} = \mathbf{Q}_k \qquad \mathbf{D} = \mathbf{G}_k^T, \tag{2.38}$$

the inverse $\mathbf{P}^{-1}(k \mid k-1)$ can written as

$$\mathbf{P}^{-1}(k \mid k-1) = \mathbf{M}_k - \mathbf{M}_k\mathbf{G}_k\left[\mathbf{G}_k^T\mathbf{M}_k\mathbf{G}_k + \mathbf{Q}_k^{-1}\right]^{-1}\mathbf{G}_k^T\mathbf{M}_k, \tag{2.39}$$

where

$$\mathbf{M}_k \triangleq \mathbf{F}_k^{-T}\mathbf{Y}(k-1 \mid k-1)\mathbf{F}_k^{-1}, \tag{2.40}$$

is a noise-less propagation of the information matrix.

To simplify further, let

$$\mathbf{\Sigma}_k \triangleq \mathbf{G}_k^T\mathbf{M}_k\mathbf{G}_k + \mathbf{Q}_k^{-1}, \tag{2.41}$$

and substitute into Equation 2.39 to obtain the information matrix prediction

$$\mathbf{Y}(k \mid k-1) = \mathbf{M}_k - \mathbf{M}_k\mathbf{G}_k\mathbf{\Sigma}_k^{-1}\mathbf{G}_k^T\mathbf{M}_k. \tag{2.42}$$

The prediction of the information vector is now derived by first writing the Kalman filter state prediction (Equation 2.12) in terms of the information vector and information matrix. This gives

$$\begin{aligned}\hat{\mathbf{y}}(k \mid k-1) &= \mathbf{Y}(k \mid k-1)\left[\mathbf{F}_k\mathbf{Y}^{-1}(k-1 \mid k-1)\hat{\mathbf{y}}(k-1 \mid k-1) + \mathbf{B}_k\mathbf{u}(k)\right] \\ &= \mathbf{Y}(k \mid k-1)\mathbf{F}_k\mathbf{Y}^{-1}(k-1 \mid k-1)\hat{\mathbf{y}}(k-1 \mid k-1) \\ &\quad + \mathbf{Y}(k \mid k-1)\mathbf{B}_k\mathbf{u}(k).\end{aligned} \tag{2.43}$$

When written in this form, the prediction of the information vector requires an inversion $\mathbf{Y}(k-1 \mid k-1)$. In order to obtain a prediction equation which does not require this inversion, substitute Equation 2.42 into Equation 2.43. This yields

$$\begin{aligned}\hat{\mathbf{y}}(k \mid k-1) &= \left[\mathbf{M}_k - \mathbf{M}_k\mathbf{G}_k\boldsymbol{\Sigma}_k^{-1}\mathbf{G}_k^T\mathbf{M}_k\right]\mathbf{F}_k\mathbf{Y}^{-1}(k-1 \mid k-1)\hat{\mathbf{y}}(k-1 \mid k-1) \\ &\quad +\mathbf{Y}(k \mid k-1)\mathbf{B}_k\mathbf{u}(k) \\ &= \left[\mathbf{1} - \mathbf{M}_k\mathbf{G}_k\boldsymbol{\Sigma}_k^{-1}\mathbf{G}_k^T\right]\mathbf{M}_k\mathbf{F}_k\mathbf{Y}^{-1}(k-1 \mid k-1)\hat{\mathbf{y}}(k-1 \mid k-1) \\ &\quad +\mathbf{Y}(k \mid k-1)\mathbf{B}_k\mathbf{u}(k). \qquad (2.44)\end{aligned}$$

Using the definition of $\mathbf{M}_k$ given in Equation 2.40, it is possible to write Equation 2.44 as

$$\begin{aligned}\hat{\mathbf{y}}(k \mid k-1) &= \left[\mathbf{1} - \mathbf{M}_k\mathbf{G}_k\boldsymbol{\Sigma}_k^{-1}\mathbf{G}_k^T\right]\mathbf{F}_k^{-T}\mathbf{Y}(k-1 \mid k-1)\mathbf{F}_k^{-1}\mathbf{F}_k\mathbf{Y}^{-1}(k-1 \mid k-1) \\ &\quad \times\hat{\mathbf{y}}(k-1 \mid k-1) + \mathbf{Y}(k \mid k-1)\mathbf{B}_k\mathbf{u}(k) \\ &= \left[\mathbf{1} - \mathbf{M}_k\mathbf{G}_k\boldsymbol{\Sigma}_k^{-1}\mathbf{G}_k^T\right]\mathbf{F}_k^{-T}\hat{\mathbf{y}}(k-1 \mid k-1) + \mathbf{Y}(k \mid k-1)\mathbf{B}_k\mathbf{u}(k). \qquad (2.45)\end{aligned}$$

The propagation of the information state estimate is now defined by Equations 2.42 and 2.45.

Information Filter Equations

Prediction

$$\begin{aligned}\mathbf{Y}(k \mid k-1) &= \mathbf{M}_k - \mathbf{M}_k\mathbf{G}_k\boldsymbol{\Sigma}_k^{-1}\mathbf{G}_k^T\mathbf{M}_k \\ \hat{\mathbf{y}}(k \mid k-1) &= \left[\mathbf{1} - \mathbf{M}_k\mathbf{G}_k\boldsymbol{\Sigma}_k^{-1}\mathbf{G}_k^T\right]\mathbf{F}_k^{-T}\hat{\mathbf{y}}(k-1 \mid k-1) + \mathbf{Y}(k \mid k-1)\mathbf{B}_k\mathbf{u}(k)\end{aligned}$$

where

$$\boldsymbol{\Sigma}_k = \mathbf{G}_k^T\mathbf{M}_k\mathbf{G}_k + \mathbf{Q}_k^{-1}$$

$$\mathbf{M}_k = \mathbf{F}_k^{-T}\mathbf{Y}(k-1 \mid k-1)\mathbf{F}_k^{-1}$$

Observation information

$$\begin{aligned}\mathbf{i}(k) &= \mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{z}(k) \\ \mathbf{I}(k) &= \mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{H}_k\end{aligned}$$

Update

$$\begin{aligned}\hat{\mathbf{y}}(k \mid k) &= \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k) \\ \mathbf{Y}(k \mid k) &= \mathbf{Y}(k \mid k-1) + \mathbf{I}(k)\end{aligned}$$

As the update stage of the Information filter is additive, the algorithm can be considered as an 'information accumulator'. The update Equations 2.35 and 2.36 demonstrate this as they clearly add new information from observations to the information estimate. In contrast, the prediction cycle removes information from the system. Adding or increasing information in information space is therefore equivalent to reducing covariance in state space. This follows logically as the two algorithms are inversely related.

Algebraically, the information filter has a simple update stage, but a complex prediction phase, in exact contrast to the Kalman filter. Indeed, the information filter and Kalman filter algorithms are known to be dual [3]. Inspection of the prediction phase of the information filter allows us to identify "information innovations" ($\boldsymbol{\Sigma}(k)$ and other terms which have an exact counterpart in the Kalman filter update stage).

It would seem therefore that the information filter and Kalman filter have comparable

computational complexity. While this is outwardly true, the two algorithms scale differently with increasing numbers of observations, that is in their application to multi-sensor systems. Crucially, the simple additive nature of the information filter update makes it extremely simple to fuse information from many sensors with very little computational overhead. Conversely, fusing many observations in the Kalman filter update stage is very complex and scales as the square of the number of observations (fundamentally because the innovations are correlated through a common prediction [9]). On the other hand, the complexity of the prediction stage is invariant to the number of observations made. So while information prediction is more complex than state prediction, it does not get any harder as the number of sensors grows.

The information filter is well known and is described (at least in inverse covariance form) in a number of texts (notably [3, 71]). However it has not found wide-spread use. There are a number of possible reasons for this. First, is the fact that the state space is usually of higher dimension than observation space. While this is true in single sensor estimation problems, it is clearly not true in multi-sensor problems where the combined sensor state space is usually much larger. A second problem is the spread of a false assumption that "it is always cheaper to communicate innovations" in multi-sensor problems. While the sensor dimension is usually smaller than the state dimension, the problem is that innovations between different sensors are correlated and so it is not sufficient to communicate the innovation alone. While information quantities are always of state dimension, they are all that are required to compute a consistent estimate. A third reason is probably that engineers fear using an information quantity rather than a state. States have physical interpretations; position velocity, etc. Information states are not so easy to imagine as they are scaled by the information matrix. The information state-space is not metric; the difference between two information states is not the distance between them in any physical sense. However, the information filter is an algorithm and the information states are quantities from which true states can be deduced (if required). If one can live with this, there is much to be gained by the exploitation of the algorithm.

The information filter turns out to have a number of other advantageous properties of value in multi-sensor estimation problems. In particular, it turns out to be quite closely related to log-likelihoods and related information measures. It has strong relations with inverse-time filters (smoothers) which helps in time-alignment and delayed data problems. It makes

initialisation simple by allowing assignment of zero information (infinite covariance). These properties will be described and used at appropriate points in this thesis.

Example 2 *Consider the information form of Example 1. The initial state at time $k-1$ is written*

$$\begin{aligned}
\mathbf{Y}(k-1 \mid k-1) &= \mathbf{P}^{-1}(k-1 \mid k-1) \\
&= \begin{bmatrix} 0.5102 & -0.1020 \\ -0.1020 & 1.0204 \end{bmatrix} \\
\hat{\mathbf{y}}(k-1 \mid k-1) &= \mathbf{Y}(k-1 \mid k-1)\hat{\mathbf{x}}(k-1 \mid k-1) \\
&= \begin{bmatrix} 5 \\ 0 \end{bmatrix}.
\end{aligned}$$

The propagation from time $k-1$ to k is calculated using Equations 2.40–2.45.

$$\begin{aligned}
\mathbf{M}_k &= \mathbf{F}_k^{-T}\mathbf{Y}(k-1 \mid k-1)\mathbf{F}_k^{-1} \\
&= \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0.5102 & -0.1020 \\ -0.1020 & 1.0204 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0.5102 & -0.6122 \\ -0.6122 & 1.7347 \end{bmatrix} \\
\mathbf{\Sigma}_k &= \mathbf{G}_k^T\mathbf{M}_k\mathbf{G}_k + \mathbf{Q}_k^{-1} \\
&= \begin{bmatrix} \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 0.5102 & -0.6122 \\ -0.6122 & 1.7347 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix} + \begin{bmatrix} 100 \end{bmatrix} \\
&= \begin{bmatrix} 101.25 \end{bmatrix}
\end{aligned}$$

Substituting these into the propagation equations gives

$$\begin{aligned}
\mathbf{Y}(k \mid k-1) &= \mathbf{M}_k - \mathbf{M}_k\mathbf{G}_k\mathbf{\Sigma}_k^{-1}\mathbf{G}_k^T\mathbf{M}_k \\
&= \begin{bmatrix} 0.5102 & -0.6122 \\ -0.6122 & 1.7347 \end{bmatrix} \\
&\quad - \begin{bmatrix} 0.5102 & -0.6122 \\ -0.6122 & 1.7347 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix} \begin{bmatrix} 101.25 \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 0.5102 & -0.6122 \\ -0.6122 & 1.7347 \end{bmatrix}
\end{aligned}$$

$$
\begin{aligned}
&= \begin{bmatrix} 0.5089 & -0.6072 \\ -0.6072 & 1.7145 \end{bmatrix} \\
\hat{\mathbf{y}}(k \mid k-1) &= \left[\mathbf{1} - \mathbf{M}_k \mathbf{G}_k \boldsymbol{\Sigma}_k^{-1} \mathbf{G}_k^T\right] \mathbf{F}_k^{-T} \hat{\mathbf{y}}(k-1 \mid k-1) + \mathbf{Y}(k \mid k-1)\mathbf{B}_k \mathbf{u}(k) \\
&= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.5102 & -0.6122 \\ -0.6122 & 1.7347 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix} \begin{bmatrix} 101.25 \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{2} & 1 \end{bmatrix} \right) \\
&\quad \times \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 4.9912 \\ -4.9647 \end{bmatrix}.
\end{aligned}
$$

Consider now the update stage of Example 1 when formulated in information space. The observation information can be calculated using Equation 2.34.

$$
\begin{aligned}
\mathbf{I}(k) &= \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \\
&= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0.5 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \\
\mathbf{i}(k) &= \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{z}(k) \\
&= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0.5 \end{bmatrix} \begin{bmatrix} 11.5 \end{bmatrix} \\
&= \begin{bmatrix} 23 \\ 0 \end{bmatrix}
\end{aligned}
$$

The update is performed using Equations 2.35 and 2.36.

$$
\begin{aligned}
\mathbf{Y}(k \mid k) &= \mathbf{Y}(k \mid k-1) + \mathbf{I}(k) \\
&= \begin{bmatrix} 2.5089 & -0.6072 \\ -0.6072 & 1.7145 \end{bmatrix} \\
\hat{\mathbf{y}}(k \mid k) &= \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k) \\
&= \begin{bmatrix} 27.9912 \\ -4.9647 \end{bmatrix}
\end{aligned}
$$

Conversion of the information back to state space yields

$$\hat{\mathbf{x}}(k \mid k) = \begin{bmatrix} 11.4359 \\ 1.1544 \end{bmatrix}, \qquad \mathbf{P}(k \mid k) = \begin{bmatrix} 0.4359 & 0.1544 \\ 0.1544 & 0.6379 \end{bmatrix},$$

which is the same result as that obtained using the Kalman filter in Example 1.

2.4.3 The Information Filter and Bayes Theorem

The relationship between the information filter and the underlying probability transformations is fundamental. This is evident as the information filter can be derived in a straightforward manner directly from Bayes Theorem. Recall

$$P(\mathbf{x}(k) \mid \mathbf{Z}^k) = \frac{P(\mathbf{z}(k) \mid \mathbf{x}(k))P(\mathbf{x}(k) \mid \mathbf{Z}^{k-1})}{P(\mathbf{z}(k) \mid \mathbf{Z}^{k-1})}. \tag{2.46}$$

Using Equation 2.46 and the definition of a multi-dimensional Gaussian distribution

$$f(\mathbf{x}) = \frac{1}{\sigma(2\pi)^{\frac{n}{2}}} e^{\left(-\frac{1}{2}[\mathbf{x}-\hat{\mathbf{x}}]^T \mathbf{P}^{-1}[\mathbf{x}-\hat{\mathbf{x}}]\right)}, \tag{2.47}$$

where $\mathbf{x}$ is the true state, $\hat{\mathbf{x}}$ the mean, $\mathbf{P}$ the covariance matrix, n is the dimension of $\mathbf{x}$ and σ is $\mid \mathbf{P} \mid$, define the following distributions:

$$P(\mathbf{x}(k) \mid Z^k) = \frac{1}{\sigma_a(2\pi)^{\frac{n}{2}}} e^{\left(-\frac{1}{2}\left[\mathbf{x}(k)-\hat{\mathbf{x}}(k \mid k)\right]^T \mathbf{P}^{-1}(k \mid k)\left[\mathbf{x}(k)-\hat{\mathbf{x}}(k \mid k)\right]\right)} \tag{2.48}$$

$$P(\mathbf{z}(k) \mid \mathbf{x}(k)) = \frac{1}{\sigma_b(2\pi)^{\frac{m}{2}}} e^{\left(-\frac{1}{2}\left[\mathbf{z}(k)-\mathbf{H}\mathbf{x}(k)\right]^T \mathbf{R}^{-1}\left[\mathbf{z}(k)-\mathbf{H}\mathbf{x}(k)\right]\right)} \tag{2.49}$$

$$P(\mathbf{x}(k) \mid Z^{k-1}) = \frac{1}{\sigma_c(2\pi)^{\frac{n}{2}}} e^{\left(-\frac{1}{2}\left[\mathbf{x}(k)-\hat{\mathbf{x}}(k \mid k-1)\right]^T \mathbf{P}^{-1}(k \mid k-1)\left[\mathbf{x}(k)-\hat{\mathbf{x}}(k \mid k-1)\right]\right)} \tag{2.50}$$

Note that $g(z) = P(\mathbf{z}(k) \mid Z^{k-1})$, σ_a, σ_b and σ_c are all independent of $\mathbf{x}(k)$. Substituting these distributions directly into Bayes Theorem (Equation 2.46) and taking natural logarithms of both sides gives

$$[\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k)]^T \mathbf{P}^{-1}(k \mid k)\,[\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k)] = [\mathbf{z}(k) - \mathbf{H}\mathbf{x}(k)]^T R^{-1}\,[\mathbf{z}(k) - H\mathbf{x}(k)]$$

$$+ [\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k-1)]^T \mathbf{P}^{-1}(k \mid k-1)$$
$$\times [\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k-1)] + C, \qquad (2.51)$$

where C is a constant and is independent of $\mathbf{x}(k)$.

Differentiating Equation 2.51 with respect to $\mathbf{x}$ yields

$$\begin{aligned} \mathbf{P}^{-1}(k \mid k)\,[\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k)] &= -\mathbf{P}^{-1}(k \mid k-1)\,[\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k-1)] \\ &\quad -\mathbf{H}^T\mathbf{R}^{-1}\,[\mathbf{z}(k) - \mathbf{H}\mathbf{x}(k)]\,. \end{aligned} \qquad (2.52)$$

Expanding and rearranging terms,

$$\begin{aligned} \mathbf{P}^{-1}(k \mid k)\,[\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k)] &= -\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}(k) + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{x}(k) \\ &\quad \mathbf{P}^{-1}(k \mid k-1)\mathbf{x}(k) - \mathbf{P}^{-1}(k \mid k-1)\hat{\mathbf{x}}(k \mid k-1) \\ \mathbf{P}^{-1}(k \mid k)\hat{\mathbf{x}}(k \mid k) &= \mathbf{P}^{-1}(k \mid k-1)\hat{\mathbf{x}}(k \mid k-1) + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}(k) + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{x}(k) \\ &\quad +\mathbf{P}^{-1}(k \mid k-1)\mathbf{x}(k) - \mathbf{P}^{-1}(k \mid k)\mathbf{x}(k) \\ \mathbf{P}^{-1}(k \mid k)\hat{\mathbf{x}}(k \mid k) &= \mathbf{P}^{-1}(k \mid k-1)\hat{\mathbf{x}}(k \mid k-1) + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}(k) \\ &\quad \left[\mathbf{P}^{-1}(k \mid k-1) + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} - \mathbf{P}^{-1}(k \mid k)\right]\mathbf{x}(k). \end{aligned} \qquad (2.53)$$

The second derivative of Equation 2.51 with respect to $\mathbf{x}$ gives

$$\mathbf{P}^{-1}(k \mid k) = \mathbf{P}^{-1}(k \mid k-1) + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}, \qquad (2.54)$$

or, after making the substitutions in Equations 2.33 and 2.34,

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \mathbf{I}(k). \qquad (2.55)$$

Substituting Equation 2.54 into Equation 2.53, the last term in the brackets cancels to zero. If the definitions from Equation 2.33 are then applied, the result is

$$\begin{aligned} \mathbf{P}^{-1}(k \mid k)\hat{\mathbf{x}}(k \mid k) &= \mathbf{P}^{-1}(k \mid k-1)\hat{\mathbf{x}}(k \mid k-1) + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}(k) \\ \hat{\mathbf{y}}(k \mid k) &= \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k). \end{aligned} \qquad (2.56)$$

Equations 2.56 and 2.55 can be seen to be the update equations for the information filter.

The derivation of the information filter from Bayes theorem highlights the close relationship between the information states and the underlying probability distributions. Indeed, the second derivative of the log-likelihood is known formally as the Fisher Information. The first derivative of the log-likelihood is known as the Score Function and maybe used in the calculation of relative entropy. The derivation from the log-likelihood form of Bayes theorem also goes to show why the update equation for the information filter is a simple linear sum.

The derivation of the information filter might also be considered in terms of a moment generating function for the simple Gaussian case. The first and second derivatives being the first and second moments respectively. This opens the way for consideration of more general distributions, either continuous or discrete, being treated in a similar information form. Indeed, the information filter is no more than a special case of the log-likelihood filters developed earlier in this Chapter, and as such they inherit many of the architectural properties of log-likelihoods.

2.4.4 The Information Filter in Multi-Sensor Systems

The application of the information filter to solve multi-sensor data fusion problems has, surprisingly, not yet caught on. On the contrary, there is a large academic industry struggling to use state-based methods to do data fusion. Popular examples include the work on track-to-track fusion methods [4], and so-called federated filters [22]. All of these methods grapple with the fact that state estimates generated by different sensor sites are correlated by virtue of a common process model. However, it turns out to be very complex to unravel these correlations and to modify the Kalman filter update (Equations 2.14 and 2.15) to fuse multiple observations at the same time. Specifically, given observations from a number of sensors in the form

$$\mathbf{z}_i(k) = \mathbf{H}_i(k)\mathbf{x}(k) + \mathbf{v}_i(k), \qquad i = 1, \cdots, N, \tag{2.57}$$

the Kalman filter estimate can not be constructed from a simple linear combination of contributions from individual sensors

$$\hat{\mathbf{x}}(k \mid k) \neq \hat{\mathbf{x}}(k \mid k-1) + \sum_{i=1}^{N} \mathbf{W}_i(k) \left[\mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}(k \mid k-1)\right], \tag{2.58}$$

as the innovations $\mathbf{z}_i(k) - \mathbf{H}_i(k)\hat{\mathbf{x}}(k \mid k-1)$ generated from each sensor are correlated through the use of the prediction $\hat{\mathbf{x}}(k \mid k-1)$. However, in information form, estimates can be constructed from linear combinations of observation information as the information terms $\mathbf{i}_i(k)$ from each sensor are uncorrelated.

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \sum_{i=1}^{N} \mathbf{i}_i(k) \tag{2.59}$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \sum_{i=1}^{N} \mathbf{I}_i(k) \tag{2.60}$$

The reason why the multi-sensor information filter can be formulated in the simple form of Equations 2.59 and 2.60 is because the information contributions $\mathbf{i}_i(k)$ and $\mathbf{I}_i(k)$ are directly related to the underlying likelihood functions for the states rather than to the state estimates themselves. To illustrate this, consider the formulation of Bayes Theorem when updating with N observations at the same time step. Assuming that observations from each sensor are conditionally independent given the true state

$$P(\mathbf{z}_1(k) \cdots, \mathbf{z}_N(k) \mid \mathbf{x}(k)) = \prod_{i=1}^{N} P(\mathbf{z}_i(k) \mid \mathbf{x}(k)), \tag{2.61}$$

the multi-sensor update with Bayes Theorem is

$$P(\mathbf{x}(k) \mid \mathbf{Z}^N(k)) = \frac{P(\mathbf{x}(k) \mid \mathbf{Z}^N(k-1)) \prod_{i=1}^{n} P(\mathbf{z}_i(k) \mid \mathbf{x}(k))}{P(\mathbf{Z}^n(k) \mid \mathbf{Z}^N(k-1))}. \tag{2.62}$$

Taking logs of Equation 2.62 gives

$$\begin{aligned} \ln P(\mathbf{x}(k) \mid \mathbf{Z}^N(k)) &= \ln P(\mathbf{x}(k) \mid \mathbf{Z}^N(k-1)) \\ &\quad + \sum_{i=1}^{N} \ln P(\mathbf{z}_i(k) \mid \mathbf{x}(k)) - \ln P(\mathbf{Z}^N(k) \mid \mathbf{Z}^N(k-1)). \end{aligned} \tag{2.63}$$

From the derivation of the information filter in Section 2.4.3, an identification can be made as

$$\sum_{i=1}^{N} \ln P(\mathbf{z}_i(k) \mid \mathbf{x}(k)) \rightleftharpoons \{\textstyle\sum_{i=1}^{N} \mathbf{i}(k), \quad \sum_{i=1}^{N} \mathbf{I}(k)\}. \tag{2.64}$$

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.

Equations 2.59 and 2.60 can also be obtained by algabraic manipulation. Again, consider a system comprising N sensors each taking observations according to

$$\mathbf{z}_i(k) = \mathbf{H}_i(k)\mathbf{x}(k) + \mathbf{v}_i(k), \tag{2.65}$$

where $\mathbf{v}_i(k)$ is assumed to be a zero mean white noise sequence with covariance $\mathbf{R}_i(k)$, and is uncorrelated in both time and between sensors. The observations from each sensor can be stacked into a single composite observation as

$$\mathbf{z}(k) = \left[\mathbf{z}_1^T(k), \cdots, \mathbf{z}_N^T(k)\right]^T. \tag{2.66}$$

The observation models can also be stacked into a composite model

$$\mathbf{H}(k) = \left[\mathbf{H}_1^T(k), \cdots, \mathbf{H}_N^T(k)\right]^T, \tag{2.67}$$

and

$$\mathbf{v}(k) = \left[\mathbf{v}_1^T(k), \cdots, \mathbf{v}_N^T(k)\right]^T, \tag{2.68}$$

to give a composite observation equation in the form of Equation 2.9

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k).$$

Noting that

$$\mathrm{E}\{\mathbf{v}(k)\mathbf{v}^T(k)\} = \mathbf{R}(k) = \mathrm{blockdiag}\{\mathbf{R}_1(k), \cdots, \mathbf{R}_N(k)\}\ , \tag{2.69}$$

the observation information can be calculated using Equation 2.34. This gives

$$\mathbf{i}(k) = \begin{bmatrix} \mathbf{H}_1^T(k) & \cdots & \mathbf{H}_N^T(k) \end{bmatrix} \begin{bmatrix} \mathbf{R}_1(k) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{R}_N(k) \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{z}_1(k) \\ \vdots \\ \mathbf{z}_N(k) \end{bmatrix} \tag{2.70}$$

$$\mathbf{I}(k) = \begin{bmatrix} \mathbf{H}_1^T(k) & \cdots & \mathbf{H}_N^T(k) \end{bmatrix} \begin{bmatrix} \mathbf{R}_1(k) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{R}_N(k) \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{H}_1(k) \\ \vdots \\ \mathbf{H}_N(k) \end{bmatrix}. \tag{2.71}$$

After multiplication, Equations 2.70 and 2.71 reduce to

$$\mathbf{i}(k) = \sum_{i=1}^{N} \mathbf{i}_i(k) = \sum_{i=1}^{N} \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{z}_i(k), \tag{2.72}$$

and

$$\mathbf{I}(k) = \sum_{i=1}^{N} \mathbf{I}_i(k) = \sum_{i=1}^{N} \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{H}_i(k), \tag{2.73}$$

where

$$\mathbf{i}_i(k) \triangleq \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{z}_i(k), \tag{2.74}$$

is the information-state contribution from observation $\mathbf{z}_i(k)$ and

$$\mathbf{I}_i(k) \triangleq \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{H}_i(k), \tag{2.75}$$

is its associated information matrix.

For large multi-sensor systems, the information filter update (Equations 2.59 and 2.60) provides a simple and easily managed solution to the fusion problem. The observation information from multiple sensors can be fused simply by adding the contributions $\mathbf{i}_i(k)$ and $\mathbf{I}_i(k)$ from each sensor i. The equivalent Kalman filter algorithm would either need to sequentially update with each observation, or to implement Equation 2.58 and then compute the cross correlations between each of the observation innovations. Both of these processes become increasingly complex as the number of sensors increases.

2.4.5 The Distributed Information Filter

The multi-sensor information filter can be readily distributed by incorporating modest processing with each sensor capable of transforming the observation to its information form. The sensors then transmit the observation information $\mathbf{i}(k)$ and $\mathbf{I}(k)$ to a central processor where the summation given in Equations 2.59 and 2.60 takes place. This simple architecture relieves the central processor from having to handle any raw sensor data, releasing it to perform the prediction and update steps. Furthermore, all communication is in terms of the information states, which has the benefit that all parts of the system are 'talking in the same language'. This would not be the case in state space as the dimension and structure of the innovations or raw observations which would need to be communicated would be a

function of the particular sensor model. Figure 2.6 illustrates this simple distributed information architecture which is the information filter equivalent of the independent likelihood model described in Section 2.2.3.

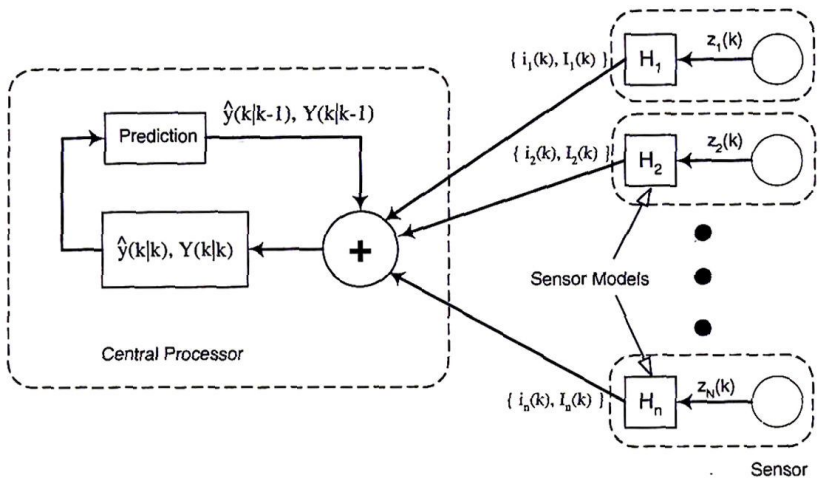


Figure 2.6: A distributed information filter where the observation information is calculated locally at the sensor and communicated to a central processor for fusion. This is the information filter form of the independent likelihood architecture.

Figure 2.7 illustrates another distributed information filter architecture where each sensor now computes the observation information and fuses this with its own local information estimate. The local estimate is maintained using a local implementation of the prediction and update equations and is formed solely from observations from the attached sensor. Each sensor module outputs its own local information estimate to the central processor which fuses the information to form a single global estimate.

However, to fuse these estimates at the central processor, it is necessary to determine the amount of new information that each sensor has contributed. Denoting the local estimate at each sensor i by $\tilde{\mathbf{y}}_i(\cdot \mid \cdot)$ and $\tilde{\mathbf{Y}}_i(\cdot \mid \cdot)$, the update at each local sensor can be written as

$$\tilde{\mathbf{y}}_i(k \mid k) = \tilde{\mathbf{y}}_i(k \mid k-1) + \mathbf{i}_i(k) \tag{2.76}$$

$$\tilde{\mathbf{Y}}_i(k \mid k) = \tilde{\mathbf{Y}}_i(k \mid k-1) + \mathbf{I}_i(k). \tag{2.77}$$

These local updates can then be sent to the central processor where the global update can

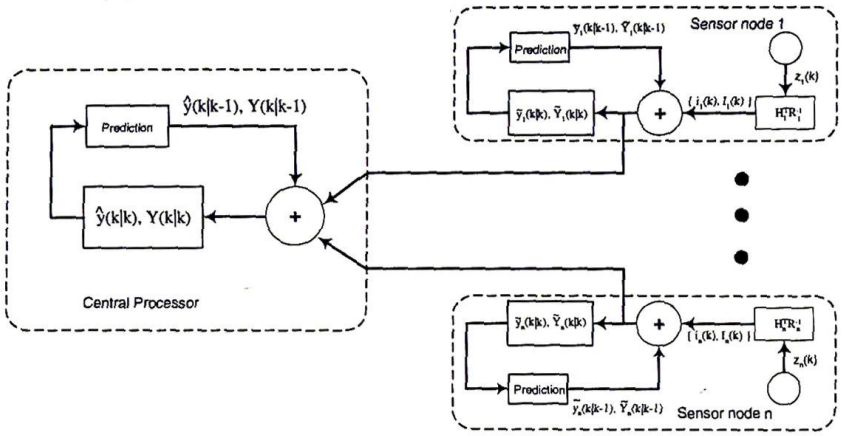


Figure 2.7: A distributed information filter where each sensor has a local estimate based on the local observations. The local estimates are communicated back to a central processor where the global estimate is calculated.

be calculated using

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \sum_{i=1}^{N} [\tilde{\mathbf{y}}_i(k \mid k) - \tilde{\mathbf{y}}_i(k \mid k-1)] \tag{2.78}$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \sum_{i=1}^{N} \left[\tilde{\mathbf{Y}}_i(k \mid k) - \tilde{\mathbf{Y}}_i(k \mid k-1)\right]. \tag{2.79}$$

Extending this architecture one step further to that shown in Figure 2.8 gives the information filter form of the independent opinion pool presented in Section 2.2.3. In this architecture the global estimate is communicated back to the sensor nodes once it has been calculated. As each sensor now has global information it is able to perform further tasks, such as data association, locally with the same accuracy as a central processor. While this further decreases the computational burden of the central processor, the system is still entirely reliant on it in order to function correctly as it is still a critical component. To remove this reliance on a single component, the system must move from being distributed to being decentralised.

2.4.6 The Decentralised Information Filter

It is a relatively simple step to decentralise the assimilation Equations 2.78 and 2.79 in systems where there is a fully connected network of sensing nodes as shown in Figure 2.9.

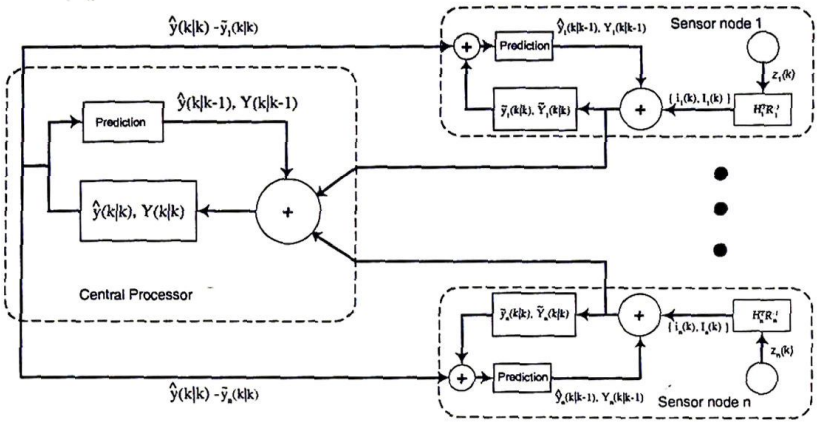


Figure 2.8: The independent opinion pool architecture in its information filter form communicates the global estimate back to the individual sensors each iteration of the filter.

In this type of system, each node generates a prediction, takes an observation and computes a local estimate which is communicated to all neighbouring nodes. Each node receives all local estimates and implements a local form of the assimilation equations to produce a global estimate of information state, equivalent to that obtained by a central fusion center. In effect, this is the same as *replicating* the central assimilation equations of Figure 2.8 at every local sensor site, and then simplifying the result.

It is assumed that each local sensor site or node maintains a state-space model $(\mathbf{F}_k, \mathbf{G}_k, \mathbf{Q}_k)$ identical to an equivalent centralized model so that $\hat{\mathbf{y}}_i(\cdot \mid \cdot) \equiv \hat{\mathbf{y}}(\cdot \mid \cdot)$ for all $i = 1, \cdots, N$. Each node begins by computing a local estimate $\tilde{\mathbf{y}}_i(k \mid k)$ based on a local prediction $\hat{\mathbf{y}}_i(k \mid k-1)$ and the observed local information $\mathbf{i}_i(k)$ according to Equations 2.42 and 2.45.

Prediction:

$$\hat{\mathbf{y}}_i(k \mid k-1) = \left[\mathbf{1} - {}^i\mathbf{M}_k \mathbf{G}_k {}^i\mathbf{\Sigma}_k^{-1} \mathbf{G}_k^T\right] \mathbf{F}_k^{-T} \hat{\mathbf{y}}_i(k-1 \mid k-1) + \mathbf{Y}_i(k \mid k-1)\mathbf{B}_k\mathbf{u}(k) \quad (2.80)$$

$$\mathbf{Y}_i(k \mid k-1) = {}^i\mathbf{M}_k - {}^i\mathbf{M}_k \mathbf{G}_k {}^i\mathbf{\Sigma}_k^{-1} \mathbf{G}_k^T {}^i\mathbf{M}_k \quad (2.81)$$

where

$${}^i\mathbf{M}_k = \mathbf{F}_k^{-T} \mathbf{Y}_i(k-1 \mid k-1) \mathbf{F}_k^{-1}, \quad (2.82)$$

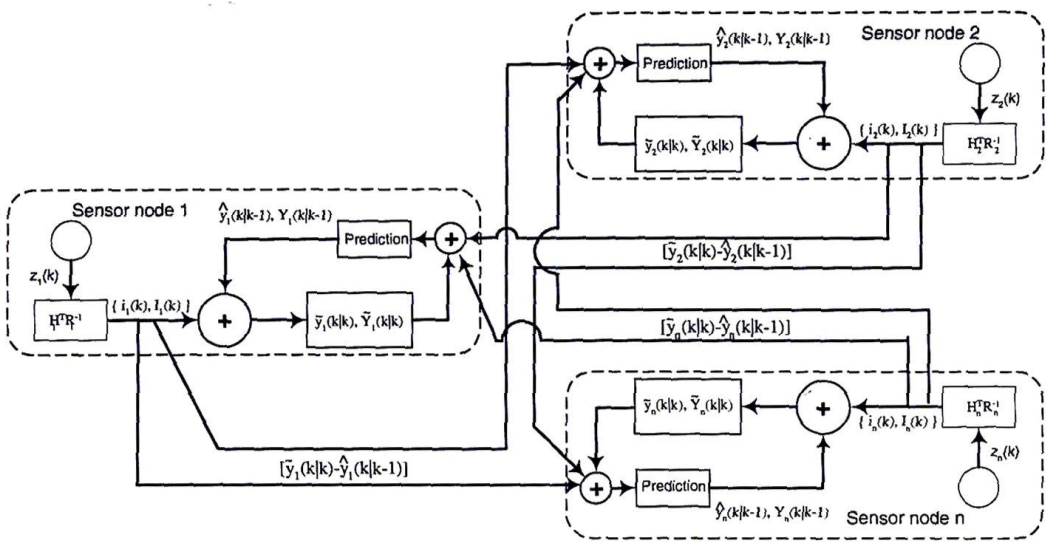


Figure 2.9: The decentralised information filter has global information available at each node, without any central fusion point.

and

$$ {}^{i}\mathbf{\Sigma}_k = \left[\mathbf{G}_k^{T\,i}\mathbf{M}_k\mathbf{G}_k + \mathbf{Q}_k^{-1}\right]. \tag{2.83} $$

Estimate:

$$ \tilde{\mathbf{y}}_i(k \mid k) = \hat{\mathbf{y}}_i(k \mid k-1) + \mathbf{i}_i(k) \tag{2.84} $$

$$ \tilde{\mathbf{Y}}_i(k \mid k) = \mathbf{Y}_i(k \mid k-1) + \mathbf{I}_i(k). \tag{2.85} $$

These partial information state estimates are then communicated to neighbouring nodes where they are assimilated according to

Assimilate:

$$ \hat{\mathbf{y}}_i(k \mid k) = \hat{\mathbf{y}}_i(k \mid k-1) + \sum_{j=1}^{N} \left[\tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_j(k \mid k-1)\right] \tag{2.86} $$

$$ \mathbf{Y}_i(k \mid k) = \mathbf{Y}_i(k \mid k-1) + \sum_{j=1}^{N} \left[\tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_j(k \mid k-1)\right]. \tag{2.87} $$

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

The quantities communicated between sensor nodes; $(\tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_j(k \mid k-1))$ and $(\tilde{\mathbf{Y}}_j(k \mid k) -$

$\mathbf{Y}_j(k \mid k-1)$), consist of the difference between the local information at a time k and the prediction based only on information up to time $k-1$. This can be interpreted as the *new* information obtained by that node at the current time step. Indeed, the communicated terms are algebraically equivalent to $\mathbf{i}_j(k)$ and $\mathbf{I}_j(k)$; logically the new information available at a time k is just the information obtained through observation at that time. Thus, the operation of the sensor network can be envisioned as a group of local estimators which communicate new, independent, information between each other and which assimilate both local and communicated information to individually obtain a globally optimal local estimate.

There are three interesting points that can be made about these decentralised equations:

- The additional computation required of each node to assimilate information from adjacent nodes is small; a summation of vectors in Equation 2.86 and a summation of matrices in Equation 2.87. 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 centalised architecture. 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 in a halving of required communication bandwidth, which may be further improved if model distribution is incorporated.

- The algorithm defined by Equations 2.80–2.87 is appropriate for both fully connected sensor networks or for sensors connected to a broadcast communication facility (such as a bus).

The decentralised architecture has no central fusion site. Each sensor node forms an estimate using information from its local sensor and new information communicated by neighbouring nodes. In a decentralised architecture there is no single point of failure. The loss of any one node results only in a commensurate loss of information to the system as a whole. This essential architecture forms the basis for the development of the decentralised sensing networks described in detail in later Chapters.

2.4.7 The Extended Information Filter

The extended information filter is the information space equivalent of the EKF. It uses the Jacobian of the non-linear process and observation models given by Equations 2.18 and 2.19, taken with respect to the states, in the same way as the EKF.

For a non-linear process model, the prediction stage simply uses the Jacobian in the standard information filter equations. This gives

$$\mathbf{Y}(k \mid k-1) = \mathbf{M}_k - \mathbf{M}_k \mathbf{G}_k \boldsymbol{\Sigma}_k^{-1} \mathbf{G}_k^T \mathbf{M}_k \tag{2.88}$$

$$\hat{\mathbf{y}}(k \mid k-1) = \mathbf{Y}(k \mid k-1)\mathbf{f}\left(\hat{\mathbf{x}}(k-1 \mid k-1), \mathbf{u}(k)\right), \tag{2.89}$$

where

$$\boldsymbol{\Sigma}_k = \mathbf{G}_k^T \mathbf{M}_k \mathbf{G}_k + \mathbf{Q}_k^{-1},$$

and

$$\mathbf{M}_k = \nabla_{\mathbf{x}}^{-\mathbf{T}} \mathbf{f}(k) \mathbf{Y}(k-1 \mid k-1) \nabla_{\mathbf{x}}^{-\mathbf{1}} \mathbf{f}(k).$$

The observation information is calculated using

$$\mathbf{i}(k) = \nabla_{\mathbf{x}}^{\mathbf{T}} \mathbf{h}(k) \mathbf{R}_k^{-1} \left[\mathbf{z}(k) - \left(\mathbf{h}(\hat{\mathbf{x}}(k \mid k-1)) - \nabla_{\mathbf{x}}^{\mathbf{T}} \mathbf{h}(k) \hat{\mathbf{x}}(k \mid k-1)\right)\right] \tag{2.90}$$

$$\mathbf{I}(k) = \nabla_{\mathbf{x}}^{\mathbf{T}} \mathbf{h}(k) \mathbf{R}_k^{-1} \nabla_{\mathbf{x}} \mathbf{h}(k), \tag{2.91}$$

and updated using the standard addition equations

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k) \tag{2.92}$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \mathbf{I}(k). \tag{2.93}$$

Extended Information Filter Equations

Prediction

$$\begin{aligned}\mathbf{Y}(k \mid k-1) &= \mathbf{M}_k - \mathbf{M}_k \mathbf{G}_k \boldsymbol{\Sigma}_k^{-1} \mathbf{G}_k^T \mathbf{M}_k \\ \hat{\mathbf{y}}(k \mid k-1) &= \mathbf{Y}(k \mid k-1)\mathbf{f}\left(\hat{\mathbf{x}}(k-1 \mid k-1), \mathbf{u}(k)\right)\end{aligned}$$

where

$$\boldsymbol{\Sigma}_k = \mathbf{G}_k^T \mathbf{M}_k \mathbf{G}_k + \mathbf{Q}_k^{-1}$$

$$\mathbf{M}_k = \nabla_{\mathbf{x}}^{-\mathbf{T}} \mathbf{f}(k) \mathbf{Y}(k-1 \mid k-1) \nabla_{\mathbf{x}}^{-\mathbf{1}} \mathbf{f}(k)$$

Observation information

$$\begin{aligned}\mathbf{i}(k) &= \nabla_{\mathbf{x}}^{\mathbf{T}} \mathbf{h}(k) \mathbf{R}_k^{-1} \left[\mathbf{z}(k) - \left(\mathbf{h}(\hat{\mathbf{x}}(k \mid k-1)) - \nabla_{\mathbf{x}}^{\mathbf{T}} \mathbf{h}(k) \hat{\mathbf{x}}(k \mid k-1)\right)\right] \\ \mathbf{I}(k) &= \nabla_{\mathbf{x}}^{\mathbf{T}} \mathbf{h}(k) \mathbf{R}_k^{-1} \nabla_{\mathbf{x}} \mathbf{h}(k)\end{aligned}$$

Update

$$\begin{aligned}\hat{\mathbf{y}}(k \mid k) &= \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k) \\ \mathbf{Y}(k \mid k) &= \mathbf{Y}(k \mid k-1) + \mathbf{I}(k)\end{aligned}$$

2.4.8 Information Metrics

Ideas of information underlie all the algorithms developed for decentralised data fusion networks. It is therefore appropriate to define the two key common measures of information employed in estimation and to understand their relation to each other and to the information estimation algorithms so far described.

Consider a probability distribution $P(\mathbf{x})$ defined on a random variable $\mathbf{x}$. The Shannon information (or entropy) represented by this distribution is the expected value of the negative log-likelihood. This is calculated as [24, 71]

$$H_P(\mathbf{x}) = -E\{\log P(\mathbf{x})\} = -\int_{-\infty}^{\infty} P(\mathbf{x}) \log P(\mathbf{x}) d\mathbf{x}, \tag{2.94}$$

for continuous valued random variables and

$$H_P(\mathbf{x}) = -E\{\log P(\mathbf{x})\} = -\sum P(\mathbf{x}) \log P(\mathbf{x}) d\mathbf{x}, \tag{2.95}$$

for discrete random variables.

Importantly, despite the convention of having $\mathbf{x}$ as an argument for H_P, the Shannon information is function of the shape of the distribution and is not a function of the underlying state $\mathbf{x}$. Indeed, the Shannon information measures the volumetric compaction of the distribution on in the state space. Entropy is low (information is high) when the probability distribution is confined to a few (ideally a single) state. Conversely, the entropy is a maximum (information is a minimum) when the distribution is uniformly distributed over the state space.

The Fisher information is defined as the second derivative of the log-likelihood of a distribution [71]

$$J(\mathbf{x}) = \frac{d^2}{d\mathbf{x}^2} \log P(\mathbf{x}). \tag{2.96}$$

When the state $\mathbf{x}$ is a vector, the Fisher Information will be a matrix (known as the Information Matrix). This matrix measures the axes of the bounding surface of the probability mass. The Fisher information measures the compactness of a distribution by its surface area. Unlike Shannon information, Fisher information is directional.

Both Shannon and Fisher information measure compactness of a distribution; through a volume and through a surface measure respectively. Unsurprisingly, the two measures can be shown to be related to each other. In the case where $P(\mathbf{x})$ is an n-dimensional Gaussian with covariance $\mathbf{\Sigma}$, the Fisher information is simply $\mathbf{\Sigma}^{-1}$, and the Shannon information is $-\frac{1}{2}\log(2\pi e)^n|\ \mathbf{\Sigma}\ |$.

2.5 The Covariance Intersect Algorithm

The covariance intersect (CI) algorithm, developed by Uhlmann and Julier [55, 98], allows the fusion of two random vectors whose degree of correlation is unknown. The fusion process uses a geometric interpretation of the covariance matrix in order to consistently fuse the information.

Consider the problem of fusing two estimates at some arbitrary time k. Let the two estimates and their associated covariances be

$$\begin{aligned} &\hat{\mathbf{x}}_A(k \mid k), \qquad \mathbf{P}_A(k \mid k) \\ &\hat{\mathbf{x}}_B(k \mid k), \qquad \mathbf{P}_B(k \mid k). \end{aligned} \tag{2.97}$$

The CI update is written

$$\mathbf{P}_C(k \mid k) = \left[\omega \mathbf{P}_A^{-1}(k \mid k) + (1-\omega)\,\mathbf{P}_B^{-1}(k \mid k)\right]^{-1} \tag{2.98}$$

$$\hat{\mathbf{x}}_C(k \mid k) = \mathbf{P}_C(k \mid k)\left[\omega \mathbf{P}_A^{-1}(k \mid k)\hat{\mathbf{x}}_A(k \mid k) + (1-\omega)\,\mathbf{P}_B^{-1}(k \mid k)\hat{\mathbf{x}}_B(k \mid k)\right], \tag{2.99}$$

where ω is selected to minimise some heuristic such as the determinate or trace of the resulting covariance matrix.

When the same problem is formulated in information space, the estimates can be rewritten as

$$\begin{aligned} &\hat{\mathbf{y}}_A(k \mid k) = \mathbf{P}_A^{-1}(k \mid k)\hat{\mathbf{x}}_A(k \mid k), \qquad \mathbf{Y}_A(k \mid k) = \mathbf{P}_A^{-1}(k \mid k) \\ &\hat{\mathbf{y}}_B(k \mid k) = \mathbf{P}_B^{-1}(k \mid k)\hat{\mathbf{x}}_B(k \mid k), \qquad \mathbf{Y}_B(k \mid k) = \mathbf{P}_B^{-1}(k \mid k). \end{aligned} \tag{2.100}$$

In this form, the CI update can be written more naturally as

$$\mathbf{Y}_C(k \mid k) = \omega \mathbf{Y}_A(k \mid k) + (1-\omega)\,\mathbf{Y}_B(k \mid k) \tag{2.101}$$

$$\hat{\mathbf{y}}_C(k \mid k) = \omega \hat{\mathbf{y}}_A(k \mid k) + (1-\omega)\,\hat{\mathbf{y}}_B(k \mid k), \tag{2.102}$$

where ω is again selected based on some heuristic.

Consider now the geometric representation of these equations with the two estimates illustrated in Figure 2.10. If the two estimates are not independent then the Kalman filter update shown in Figure 2.10 (a) will be overly confident and hence inconsistent. Using the CI update with varying values for ω from 0 to 1 gives the range of ellipses shown in Figure 2.10 (b). The geometric constraint of the algorithm is visible in that all of the CI updates for the entire range of ω enclose the area bounded by the union of the two original estimates. This ensures that the fused estimate is never overly confident. The update

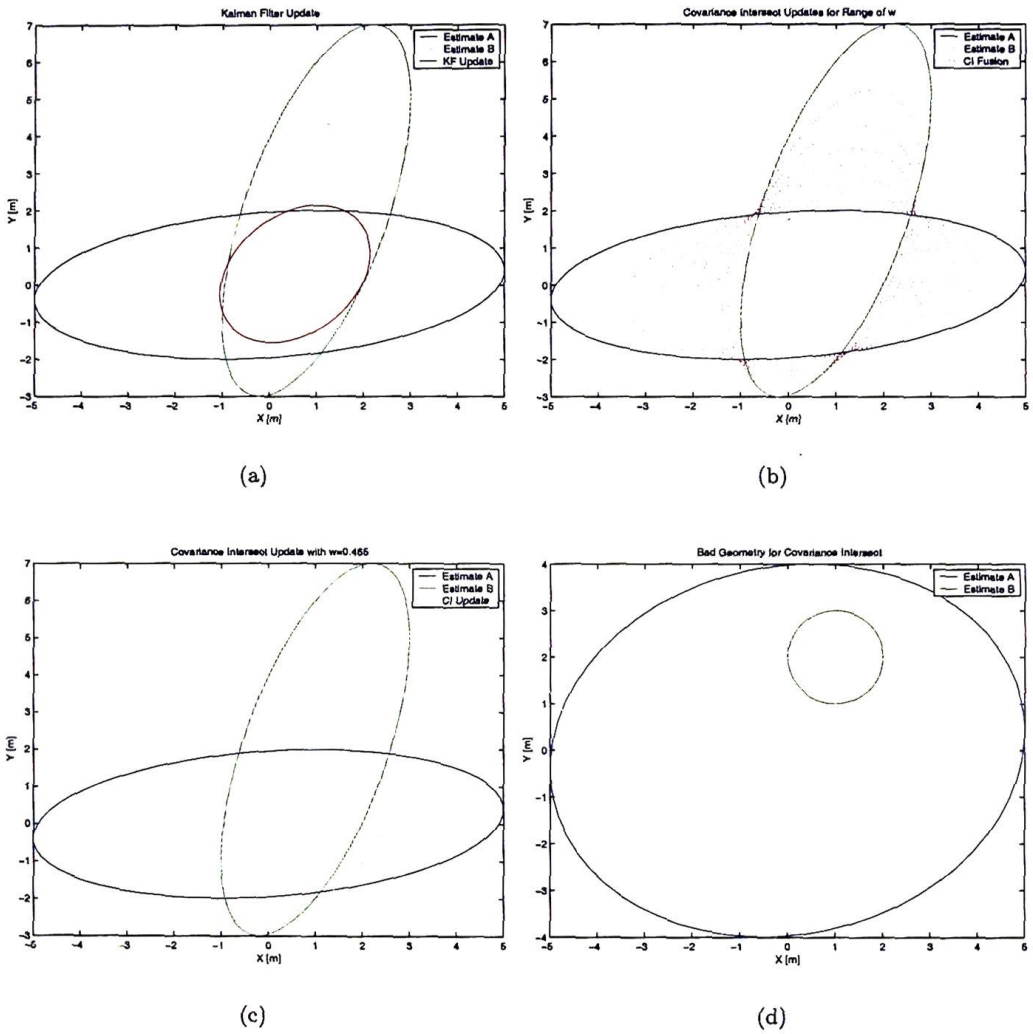


Figure 2.10: Covariance Intersect updates for two estimates. (a) shows the result of fusing the two estimates using the Kalman filter. (b) illustrates the CI update of the estimates over a range of values of ω. (c) shows the CI update with $\omega = 0.465$ minimising the determinant of the resulting covariance. (d) shows a situation where the CI update will be dominated by the more accurate estimate and ignore the less accurate one completely.

using the value of ω which minimises the determinant of the resulting covariance matrix is shown in Figure 2.10 (c). Note that this result is clearly more conservative than that of the Kalman filter in Figure 2.10 (a).

In the extreme, if ω is set to either 0 or 1 the algorithm ignores one of the estimates completely and uses the other as the result. In practice this can happen if one estimate is significantly better than another as illustrated in Figure 2.10 (d). This highlights the fact that CI discards information which can result in very conservative estimates. Although the conservative nature of the algorithm is the very reason it is able to work consistently in situations where the degree of correlation between estimates is unknown, it is also a major drawback and a reason to use CI only where necessary. In a decentralised network where information from different sources with an unknown degree of correlation is abundant, it is a useful tool when used correctly.

Covariance Intersect Equations

$$\begin{aligned} \mathbf{Y}_C(k \mid k) &= \omega \mathbf{Y}_A(k \mid k) + (1-\omega)\,\mathbf{Y}_B(k \mid k) \\ \hat{\mathbf{y}}_C(k \mid k) &= \omega \hat{\mathbf{y}}_A(k \mid k) + (1-\omega)\,\hat{\mathbf{y}}_B(k \mid k) \end{aligned}$$

where ω is selected according to some heuristic such as minimising the determinate of the resulting covariance matrix.

2.6 Summary

This Chapter developed the mathematical foundations for decentralised data fusion methods. The log-likelihood formulation of Bayes theorem was applied to explore centralised, distributed and decentralised data fusion architectures. The information filter was then derived from both the the Kalman filter and directly from Bayes theorem. The information filter was also applied to explore centralised, distributed and decentralised estimation methods. The relationship between these methods and the underlying log-likelihood formulation of the data fusion problem were established. It was argued extensively that the information filter has many advantages over the conventional Kalman filter in data fusion problems and indeed is crucial in the development of distributed and decentralised data fusion systems.

The link between the information filter and the Shannon and Fisher information metrics was also established. The Covariance Intersect (CI) algorithm was also described. This provides a method of fusing quantities which have an unknown degree of correlation. These algorithms are now employed in subsequent Chapters to develop and demonstrate real-world DDF algorithms and architecture.

Chapter 3

Algorithms for Decentralised Systems

3.1 Introduction

This Chapter describes the algorithmic implementation of decentralised data fusion systems. A decentralised system is described by the local data processing algorithms at each node and by the algorithms that decide what and when to communicate to other nodes. These local processing and local communication algorithms define the operation of the overall decentralised system.

This Chapter begins in Section 3.2 by discussing the problem of communication and network structure in decentralised systems. In particular, the key issue of identifying common information between two nodes in a network is described. This leads to the notion of a channel filter; a filter which tracks the information common to two adjacent nodes. The channel filter is fundamental to the efficient realisation of a decentralised network. It is shown that in a structured network, the channel filter can optimally track this common information. In unstructured networks, sub optimal methods of tracking common information are described.

In real networks, with estimates being transmitted from both locally attached sensors and from remote nodes, account must be made of timing. Section 3.3 discusses the temporal propagation of information in the network and presents solutions to the problem of delayed

and asequent data communication.

The channel communication algorithm is presented in Section 3.4. The communications management problem is described and a general method of handling communications in channels is presented. The structure of this channel algorithm is discussed. The section concludes by looking at the flow of information through the channel and how common information is maintained.

With the communications algorithm defined, Section 3.5 describes the overall algorithmic structure of a node in a realistic decentralised system. Sensor pre-processing, local filters, channel filters and the communications manager are described. The section then addresses a number of practical considerations for the implementation of decentralised sensing networks. In particular, issues associated with timing and data association are discussed.

Section 3.6 uses the nodal architecture described in Section 3.5 to describe algorithms for communicating in unstructured network topologies. The most complex of these, given in Section 3.6.4, allows nodes to dynamically form into a tree topology which makes use of complete information.

The algorithms and architecture developed in this Chapter sets the stage for the implementations described in Chapters 5 and 6.

3.2 Communication in Decentralised Sensing Systems

The issue of communication is significant 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 in to both the information filter and discrete estimation form. 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.

3.2.1 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. 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. 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 until a different route can be established.

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

Throughout this thesis, the first two of the aforementioned communication topologies are collectively known as structured networks as they both place restrictions on the overall architecture in order to give it structure. Arbitrary connected networks will be known as unstructured networks as they do not impose any restrictions on the network structure.

The different topologies determine different algorithms for communication. Fully connected and broadcast networks follow most simply from the algorithms introduced in Chapter 2. 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 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 [100]). 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. Algorithms for communication in this type of network are discussed in detail in Section 3.6.

3.2.2 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 $\mathbf{Z}_i$. Let $\mathbf{x}$ define the state to be estimated, then $P(\mathbf{x} \mid \mathbf{Z}_i)$ defines the posterior probability on the state given the locally available information. Now the set $\mathbf{Z}_i$ 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 $\mathbf{Z}_i$ and $\mathbf{Z}_j$ respectively, who

engage in communication, it is desirable to construct the common information set $\mathbf{Z}_i \cup \mathbf{Z}_j$ constructed from the union of the two local information sets. This may then be used to construct a posterior distribution $P(\mathbf{x} \mid \mathbf{Z}_i \cup \mathbf{Z}_j)$ for the state based on both sources of information. If the two information sets have no information in common $\mathbf{Z}_i \cap \mathbf{Z}_j = \emptyset$, then it is simple to construct the union of the sets from a sum. However if the two sets do have information in common $\mathbf{Z}_i \cap \mathbf{Z}_j \neq \emptyset$ then the problem of constructing the union $\mathbf{Z}_i \cup \mathbf{Z}_j$ resolves to finding the common information $\mathbf{Z}_i \cap \mathbf{Z}_j$. 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 $P(\mathbf{x} \mid \mathbf{Z}_i \cup \mathbf{Z}_j)$. Let the union of the individual observation information sets be partitioned into disjoint sets as

$$\mathbf{Z}_i \cup \mathbf{Z}_j = \mathbf{Z}_{i\backslash j} \cup \mathbf{Z}_{j\backslash i} \cup \mathbf{Z}_{ij} \tag{3.1}$$

where

$$\mathbf{Z}_{i\backslash j} = \mathbf{Z}_i \backslash \mathbf{Z}_{ij}, \quad \mathbf{Z}_{j\backslash i} = \mathbf{Z}_j \backslash \mathbf{Z}_{ij}, \quad \mathbf{Z}_{ij} = \mathbf{Z}_i \cap \mathbf{Z}_j,$$

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

$$\mathbf{Z}_{i\backslash j} \cup \mathbf{Z}_{ij} = \mathbf{Z}_i, \qquad \mathbf{Z}_{j\backslash i} \cup \mathbf{Z}_{ij} = \mathbf{Z}_j.$$

Then,

$$\begin{aligned} P(\mathbf{Z}_i \cup \mathbf{Z}_j \mid \mathbf{x}) &= P(\mathbf{Z}_{i\backslash j} \cup \mathbf{Z}_{j\backslash i} \cup \mathbf{Z}_{ij} \mid \mathbf{x}) \\ &= P(\mathbf{Z}_{i\backslash j} \mid \mathbf{Z}_{j\backslash i} \cup \mathbf{Z}_{ij}, \mathbf{x}) P(\mathbf{Z}_{j\backslash i} \cup \mathbf{Z}_{ij} \mid \mathbf{x}) = P(\mathbf{Z}_{i\backslash j} \mid \mathbf{Z}_j, \mathbf{x}) P(\mathbf{Z}_j \mid \mathbf{x}) \\ &= \frac{P(\mathbf{Z}_{i\backslash j} \cup \mathbf{Z}_{ij} \mid \mathbf{x})}{P(\mathbf{Z}_{ij} \mid \mathbf{x})} P(\mathbf{Z}_j \mid \mathbf{x}) = \frac{P(\mathbf{Z}_i \mid \mathbf{x})}{P(\mathbf{Z}_{ij} \mid \mathbf{x})} P(\mathbf{Z}_j \mid \mathbf{x}) \\ &= \frac{P(\mathbf{Z}_i \mid \mathbf{x}) P(\mathbf{Z}_j \mid \mathbf{x})}{P(\mathbf{Z}_i \cap \mathbf{Z}_j \mid \mathbf{x})}. \end{aligned} \tag{3.2}$$

Substituting Equation 3.2 into Bayes theorem gives

$$P(\mathbf{x} \mid \mathbf{Z}_i \cup \mathbf{Z}_j) = \frac{P(\mathbf{Z}_i \cup \mathbf{Z}_j \mid \mathbf{x}) P(\mathbf{x})}{P(\mathbf{Z}_i \cup \mathbf{Z}_j)}$$

$$
\begin{aligned}
&= \frac{P(\mathbf{Z}_i \mid \mathbf{x})P(\mathbf{Z}_j \mid \mathbf{x})}{P(\mathbf{Z}_i \cap \mathbf{Z}_j \mid \mathbf{x})} \frac{P(\mathbf{x})}{P(\mathbf{Z}_i \cup \mathbf{Z}_j)} \\
&= \frac{P(\mathbf{x} \mid \mathbf{Z}_i)P(\mathbf{x} \mid \mathbf{Z}_j)}{P(\mathbf{x} \mid \mathbf{Z}_i \cap \mathbf{Z}_j)} \frac{P(\mathbf{Z}_i)P(\mathbf{Z}_j)}{P(\mathbf{Z}_i \cap \mathbf{Z}_j)} P(\mathbf{Z}_i \cup \mathbf{Z}_j) \\
&= c.\frac{P(\mathbf{x} \mid \mathbf{Z}_i)P(\mathbf{x} \mid \mathbf{Z}_j)}{P(\mathbf{x} \mid \mathbf{Z}_i \cap \mathbf{Z}_j)}. \qquad (3.3)
\end{aligned}
$$

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

Taking logs of Equation 3.3, gives the intuitive result

$$\ln P(\mathbf{x} \mid \mathbf{Z}_i \cup \mathbf{Z}_j) = \ln P(\mathbf{x} \mid \mathbf{Z}_i) + \ln P(\mathbf{x} \mid \mathbf{Z}_j) - \ln P(\mathbf{x} \mid \mathbf{Z}_i \cap \mathbf{Z}_j). \qquad (3.4)$$

Equation 3.4 simply implies that the fused information is constructed from the sum of the information from each of the nodes minus the information they have in common. The term $\ln P(\mathbf{x} \mid \mathbf{Z}_i \cap \mathbf{Z}_j)$ describes the common information between two nodes which must be removed before fusion.

3.2.3 Identification of Redundant Information in Sensor Networks

The key step in deriving fusion equations for decentralised sensing networks is to identify the common information $\mathbf{Z}_i \cap \mathbf{Z}_j$ between estimates so that it is not used redundantly. In decentralised data fusion systems, the incorporation of redundant information may lead to bias, over-confidence and divergence in estimates.

The problem of identifying common information is most generally considered in terms of information sets. A neighbourhood of a node is the set of nodes to which it is linked directly. A complete neighbourhood includes the node itself. A node i forms an information set $\mathbf{Z}_i^k$ at time k based on a local sensor observation $\mathbf{z}_i(k)$ and the information communicated by its neighbours. The objective of each node is to form the *union* of the information sets in the complete neighbourhood $[N_i] : \bigcup_{j \in [N_i]} \mathbf{Z}_j^k$. In particular, consider communications between node i and a neighbour j. The union of information sets, on which estimates are

to be based, may be written as

$$\mathbf{Z}_i^k \cup \mathbf{Z}_j^k = \mathbf{Z}_i^k + \mathbf{Z}_j^k - \mathbf{Z}_{i\cap j}^k, \tag{3.5}$$

that is, the union is equal to the sum of information sets communicated minus the intersection of, or common information between, these information sets. A fully decentralised solution to the estimation problem is only possible where the intersection or common communicated information $\mathbf{Z}_{i\cap j}^k$ can be determined from information which is available locally.

The topology of the sensing network is the most important factor in determining common communicated information. Consider the following three cases:

- **Full Connection:** Consider the case in which each node is connected to every other in a fully connected, or completely connected topology (Figure 3.2). In this case, the sensor nodes may acquire observation information from the entire network through direct communication and the neighbourhood of any node is the full network. In a fully connected network, the problem of locally detecting and eliminating redundant information is considerably simplified as every node has access to the same information. In this situation, the estimates formed by the nodes are identical and new information is immediately communicated after the removal of the estimate from the previous time-step as

$$\bigcup_{j\in[N_i]} \mathbf{Z}_j^k = \left[\sum_{j\in[N_i]} \mathbf{Z}_j^k - \bigcup_{j\in[N_i]} \mathbf{Z}_j^{k-1}\right]. \tag{3.6}$$

  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 3.6 is an information-set equivalent of Equations 2.86 and 2.87 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 3.3 and 3.4). Therefore, a receiving node can be certain that the only redundant information communicated by a neighbour is the information that they have exchanged in the past. Thus, the observation information history that

is required for a tree communication system extends only to the previous timestep. The only common information between node i and a neighbour j at time k is the information which they exchanged in the last time interval $k-1$. This results in a pair-wise communication algorithm. For a node i on link (i,j),

$$\mathbf{Z}_i^k \cup \mathbf{Z}_j^k = \mathbf{Z}_i^k + \mathbf{Z}_j^k - \mathbf{Z}_{i \cap j}^k,$$

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

$$\mathbf{Z}_{i \cap j}^{k-1} = \mathbf{Z}_i^{k-1} \cup \mathbf{Z}_j^{k-1}. \tag{3.7}$$

This generalises to a communication strategy over the neighbourhood as

$$\bigcup_{j \in [N_i]} \mathbf{Z}_j^k = \mathbf{Z}_i^k + \left[\sum_{j \in N_i} \mathbf{Z}_j^k - \bigcup_{j \in N_i} \mathbf{Z}_j^{k-1} \right], \tag{3.8}$$

where $[N_i]$ is the complete neighbourhood, and $i \notin N_i$.

- **Arbitrary Networks:** In an arbitrary network, the connectedness of nodes is unknown and may be partially connected, or non fully connected non-trees. Removal of common information in arbitrary network topologies is complex because the pattern of communication varies from node to node, yet each node is required to implement the same algorithm. Consider again two communicating nodes

$$\mathbf{Z}_i^k \cup \mathbf{Z}_j^k = \mathbf{Z}_i^k + \mathbf{Z}_j^k - \mathbf{Z}_{i \cap j}^k.$$

In the arbitrary network case, the intersection term $\mathbf{Z}_{i \cap j}^k$ can not be simply determined from past communication on a single link. In the tree case, the common information term depends only on the observation information of i and j. In a non-tree network, the information common to i and j may contain terms from other nodes outside of the neighbourhood. This is because multiple propagation paths are possible. Figure 3.1 illustrates the information that is communicated to nodes in three cases. The communicated information terms at i and j are denoted T_i and T_j respectively. The information T_j is integrated into the observation information sets $\mathbf{Z}_j$ upon arrival at j. This information must then be acquired by i through some operation of union or

intersect of information sets. To 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 3.6 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 3.1, 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.

3.2.4 Communication Channel Filters

Equation 3.4 serves as the basis for developing information communication policies for non-fully connected sensor networks. The probability density functions in Equation 3.4 can represent four different information or Kalman filter estimates:

- The local estimate at node i:

$$\hat{\mathbf{x}}_i(k \mid k) \triangleq \mathrm{E}\{\mathbf{x}(k) \mid \mathbf{Z}_i^k\} ,$$

  with covariance $\mathbf{P}_i(k \mid k)$.

- The local estimate at node j:

$$\hat{\mathbf{x}}_j(k \mid k) \triangleq \mathrm{E}\{\mathbf{x}(k) \mid \mathbf{Z}_j^k\} ,$$

  with covariance $\mathbf{P}_j(k \mid k)$.

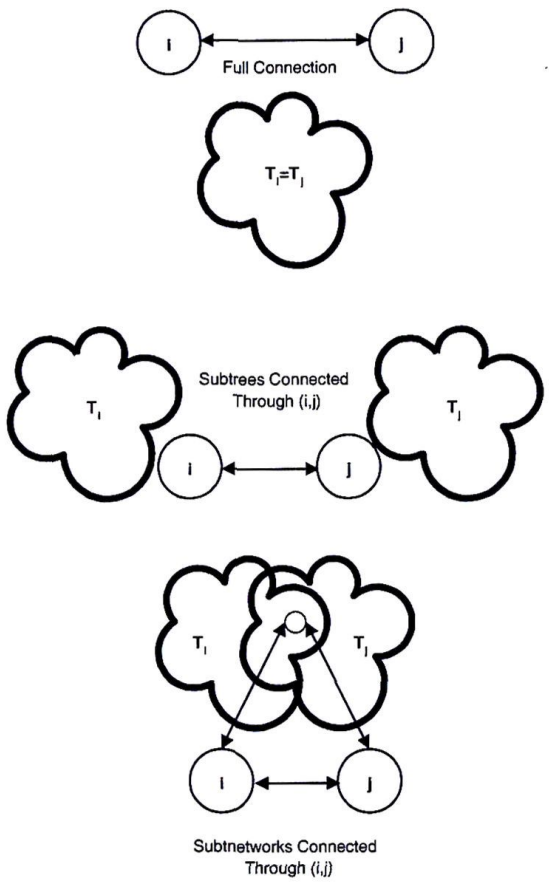


Figure 3.1: Communicated information sets in the three classes of topology.

- The estimate based on the union of all information possessed by nodes i and j (in effect the global estimate):

$$\hat{\mathbf{x}}_{i\cup j}(k \mid k) \triangleq \mathrm{E}\{\mathbf{x}(k) \mid \mathbf{Z}_i^k \cup \mathbf{Z}_j^k\} ,$$

  with covariance $\mathbf{P}_{i\cup j}(k \mid k)$.

- The estimate based on the common information between nodes i and j:

$$\hat{\mathbf{x}}_{i\cap j}(k \mid k) \triangleq \mathrm{E}\{\mathbf{x}(k) \mid \mathbf{Z}_i^k \cap \mathbf{Z}_j^k\} ,$$

  with covariance $\mathbf{P}_{i\cap j}(k \mid k)$.

Following Equation 2.51; Substituting Gaussian distributions for the probability density functions in Equation 3.4 and taking natural logarithms immediately gives the information filter equivalent of Equation 3.4 as

$$\hat{\mathbf{y}}_{i\cup j}(k \mid k) = \tilde{\mathbf{y}}_i(k \mid k) + \tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_{i\cap j}(k \mid k) \tag{3.9}$$

$$\mathbf{Y}_{i\cup j}(k \mid k) = \tilde{\mathbf{Y}}_i(k \mid k) + \tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_{i\cap j}(k \mid k), \tag{3.10}$$

where

$$\tilde{\mathbf{y}}_i(k \mid k) = \hat{\mathbf{y}}_i(k \mid k-1) + \mathbf{i}_i(k), \qquad \tilde{\mathbf{y}}_j(k \mid k) = \hat{\mathbf{y}}_j(k \mid k-1) + \mathbf{i}_j(k) \tag{3.11}$$

$$\tilde{\mathbf{Y}}_i(k \mid k) = \mathbf{Y}_i(k \mid k-1) + \mathbf{I}_i(k), \qquad \tilde{\mathbf{Y}}_j(k \mid k) = \mathbf{Y}_j(k \mid k-1) + \mathbf{I}_j(k). \tag{3.12}$$

It remains to evaluate the common information terms

$$\hat{\mathbf{y}}_{ij}(k \mid k) \triangleq \hat{\mathbf{y}}_{i\cap j}(k \mid k), \qquad \mathbf{Y}_{ij}(k \mid k) \triangleq \mathbf{Y}_{i\cap j}(k \mid k).$$

There are three cases

- **Fully Connected:** When the network is fully connected, the common information between two nodes at a time k is exactly the information communicated up to time $k-1$. This is simply the common prediction

$$\hat{\mathbf{y}}_{ij}(k \mid k) \;=\; \hat{\mathbf{y}}_i(k \mid k-1) = \hat{\mathbf{y}}_j(k \mid k-1) \tag{3.13}$$

$$\mathbf{Y}_{ij}(k \mid k) = \mathbf{Y}_i(k \mid k-1) = \mathbf{Y}_j(k \mid k-1). \tag{3.14}$$

Substitution of Equations 3.11–3.14 into Equations 3.9 and 3.10, yields the previously derived decentralised data fusion Equations 2.86 and 2.87.

- **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 2.86 and 2.87 provide a recursive expression for the total information communicated between the two nodes as

$$\begin{aligned}\hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_{ij}(k \mid k-1) \\ &\quad +[\tilde{\mathbf{y}}_i(k \mid k) - \hat{\mathbf{y}}_{ij}(k \mid k-1)] \\ &\quad +[\tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_{ij}(k \mid k-1)] \\ &= \tilde{\mathbf{y}}_i(k \mid k) + \tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_{ij}(k \mid k-1),\end{aligned} \tag{3.15}$$

and

$$\begin{aligned}\mathbf{Y}_{ij}(k \mid k) &= \mathbf{Y}_{ij}(k \mid k-1) \\ &\quad +[\tilde{\mathbf{Y}}_i(k \mid k) - \mathbf{Y}_{ij}(k \mid k-1)] \\ &\quad +[\tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_{ij}(k \mid k-1)] \\ &= \tilde{\mathbf{Y}}_i(k \mid k) + \tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_{ij}(k \mid k-1).\end{aligned} \tag{3.16}$$

This estimate of common information replaces the prior information terms in Equations 2.86 and 2.87. The local updates at each node remain unchanged

$$\tilde{\mathbf{y}}_i(k \mid k) = \hat{\mathbf{y}}_i(k \mid k-1) + \mathbf{i}_i(k) \tag{3.17}$$

$$\tilde{\mathbf{Y}}_i(k \mid k) = \mathbf{Y}_i(k \mid k-1) + \mathbf{I}_i(k), \tag{3.18}$$

and the assimilation stage becomes

$$\hat{\mathbf{y}}_i(k \mid k) = \tilde{\mathbf{y}}_i(k \mid k) + \sum_{j \in N_i} [\tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_{ji}(k \mid k-1)] \tag{3.19}$$

$$\mathbf{Y}_i(k \mid k) = \bar{\mathbf{Y}}_i(k \mid k) + \sum_{j \in N_i} \left[\bar{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_{ji}(k \mid k-1) \right]. \tag{3.20}$$

This filter is clearly symmetric, $\hat{\mathbf{y}}_{ij}(\cdot \mid \cdot) = \hat{\mathbf{y}}_{ji}(\cdot \mid \cdot)$, as two nodes have the same common information, so it need only be computed once for each channel.

- **General Networks:** In networks which admit multiple cycles, it is possible for information from a node j to be communicated to a node i both directly through the link (i, j) and indirectly through other nodes connected to both i and j. In such cases, simply adding up information transmitted through the direct connection (i, j) is *not* the same as determining the common information $\hat{\mathbf{y}}_{ij}(k \mid k)$ and $\mathbf{Y}_{ij}(k \mid k)$ between these two nodes. Indeed, in general it is *not* possible to determine this common information on the basis of local information only. A number of alternative approaches to fusion in general networks are briefly considered and compared in Section 3.2.6, and expanded in detail in Section 3.6.

Equations 3.15 and 3.16 define an information filter which estimates the common information between nodes. The filter is completed by addition of a corresponding prediction stage as

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k-1) &= \left[\mathbf{1} - {}^{ij}\mathbf{M}_k \mathbf{G}_k {}^{ij}\boldsymbol{\Sigma}_k^{-1} \mathbf{G}_k^T\right] \mathbf{F}_k^{-T} \hat{\mathbf{y}}_{ij}(k-1 \mid k-1) \\ &\quad + \mathbf{Y}_{ij}(k \mid k-1) \mathbf{B}_k \mathbf{u}(k) \qquad (3.21) \\ \mathbf{Y}_{ij}(k \mid k-1) &= {}^{ij}\mathbf{M}_k - {}^{ij}\mathbf{M}_k \mathbf{G}_k {}^{ij}\boldsymbol{\Sigma}_k^{-1} \mathbf{G}_k^T {}^{ij}\mathbf{M}_k \qquad (3.22) \end{aligned}$$

where

$${}^{ij}\mathbf{M}_k = \mathbf{F}_k^{-T} \mathbf{Y}_{ij}(k-1 \mid k-1) \mathbf{F}_k^{-1}, \tag{3.23}$$

and

$${}^{ij}\boldsymbol{\Sigma}_k = \left[\mathbf{G}_k^{T\,ij}\mathbf{M}_k \mathbf{G}_k + \mathbf{Q}_k^{-1}\right]. \tag{3.24}$$

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 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.

3.2.5 Fully Connected and Broadcast Sensor Networks

Recall the assimilation equations given in Section 2.4.6

$$\begin{aligned}\hat{\mathbf{y}}_i(k \mid k) &= \hat{\mathbf{y}}_i(k \mid k-1) + \sum_j [\tilde{\mathbf{y}}_j(k \mid k) - \hat{\mathbf{y}}_j(k \mid k-1)] \\ &= \hat{\mathbf{y}}_i(k \mid k-1) + \sum_{j \in N_i} \mathbf{i}_j(k) \end{aligned} \tag{3.25}$$

and

$$\begin{aligned}\mathbf{Y}_i(k \mid k) &= \mathbf{Y}_i(k \mid k-1) + \sum_j \left[\tilde{\mathbf{Y}}_j(k \mid k) - \mathbf{Y}_j(k \mid k-1)\right] \\ &= \mathbf{Y}_i(k \mid k-1) + \sum_{j \in N_i} \mathbf{I}_j(k). \end{aligned} \tag{3.26}$$

Equations 3.25 and 3.26 make it clear that the estimate arrived at locally by a node i is based on the observation information $\mathbf{i}_j(k)$ and $\mathbf{I}_j(k)$ communicated to it by nodes j. If node i only communicates with a local neighbourhood N_i, a subset of the complete network, then the estimate arrived at locally will be based only on this information and will not be equivalent to a centralised estimate.

In effect, each time a new local estimate $\tilde{\mathbf{y}}_j(k \mid k)$ is obtained at a node j, the prior information at this node $\hat{\mathbf{y}}_j(k \mid k-1)$ is subtracted to provide the 'new' information to be communicated to a node i. The assumption here is that the prior information $\hat{\mathbf{y}}_j(k \mid k-1)$ at node j is the information that nodes i and j have in common up to time $k-1$. Subtracting this from the local estimate should then give the new information to be communicated from node j to node i. In the fully-connected case, the prior information at node j is indeed the common information between nodes i and j and so only the information $\mathbf{i}_j(k)$ is communicated. In the non-fully connected case, the prior information at node j includes not only information common to node i but also information from other branches in the network. Subtracting this from the local estimate however, again results in only the information $\mathbf{i}_j(k)$ being communicated to node i. This is because it assumes that node i already has the information communicated to node j through other branches of the network. The net result

of this is that nodes only exchange information in their immediate neighbourhoods and do not propagate information from distant branches.

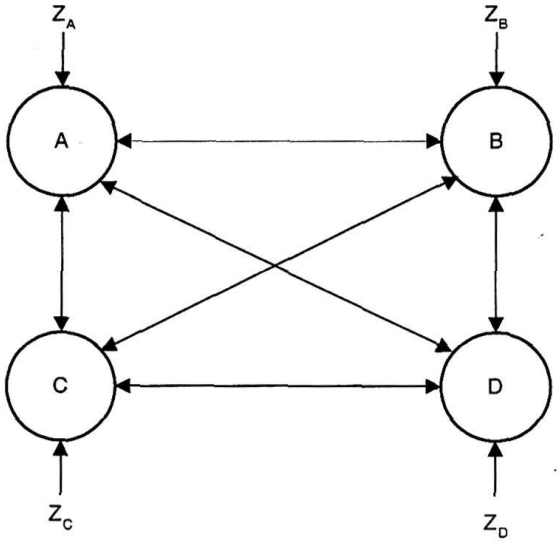


Figure 3.2: A fully connected sensor network consisting of four nodes, A, B, C, and D. All nodes in this network can generate estimates, equal to a centralised estimate, using only Equations 3.25 and 3.26.

Figure 3.2 shows a fully connected network in which local estimates will be equal to global estimates using only Equations 3.25 and 3.26 for assimilation. Figure 3.3 shows a linear connected sensor network and Figure 3.4 shows a tree connected sensor network. Here, the estimates arrived at by each node, using Equations 3.25 and 3.26 for assimilation, will be based only on observations made by sensors in the immediate neighbourhood.

3.2.6 General Network Structures

Tree networks are restricted to having only a single path between any two nodes. Therefore, a failure of any one channel or node will divide the network into two non-communicating halves. However, a decentralised network would ideally be composed of multiple, redundant, paths between sensor nodes.

Unfortunately, the algorithms described in the previous section will not produce consistent information-state estimates for networks with multiple paths. The reason for this is that the channel filter $\hat{\mathbf{y}}_{ij}(\cdot \mid \cdot)$, which is intended to determine information common to nodes i and

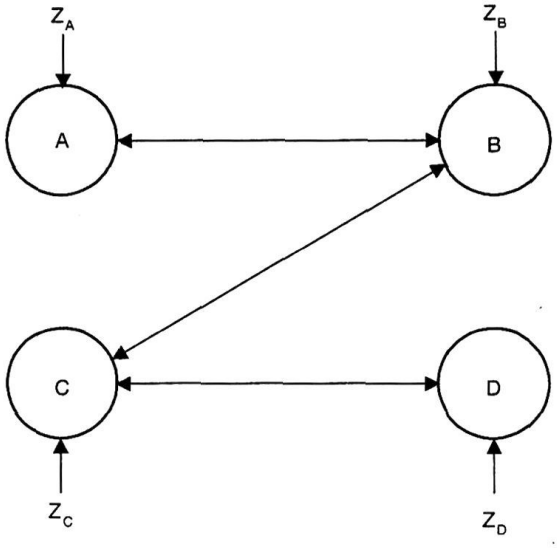


Figure 3.3: A linear sensor network consisting of four nodes, A, B, C, and D. Nodes in this network, using only Equations 3.25 and 3.26, can only generate estimates based on information available in their immediate neighbourhood.

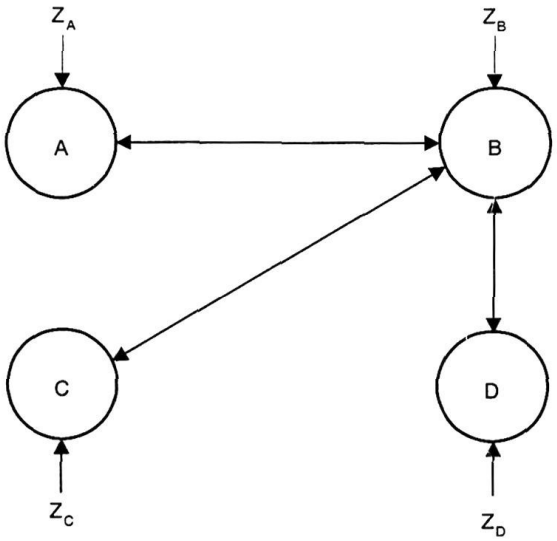


Figure 3.4: A tree connected sensor network consisting of four nodes, A, B, C, and D. Nodes in this network, using only Equations 3.25 and 3.26, generate estimates based on information available in their immediate neighbourhood.

j, does so only by summing up information communicated through the channel linking these two nodes. The assumption is that there is only one path between two neighbouring nodes so the information they have in common must have passed through the communication link connecting them. If information is communicated to both nodes i and j directly by some third node then this information will clearly be common information held by both i and j, but will not appear in the channel filter $\hat{\mathbf{y}}_{ij}(\cdot \mid \cdot)$ because it has not passed through the channel between them.

There are three possible solutions to this problem

- **Data Tagging:** An obvious solution is to tag information as 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 $\hat{\mathbf{y}}_{ij}(\cdot \mid \cdot) = \tilde{\mathbf{y}}_j(\cdot \mid \cdot)$, ensuring no information is communicated across the cut. The distributed Belman-Ford algorithm is one algorithm which allows such a tree to be constructed in a fully distributed or decentralised manner (it is commonly used for internet routing). Overall system performance depends on reducing data delays by choosing a topology with short paths between nodes. The best topology can then be implemented by artificially breaking the necessary links and using the channel filter algorithm. The network remains scalable although reconfiguration may be necessary if nodes are to be added or removed. This may, at first, seem to be a disadvantage but, should the network sustain damage to links or nodes such a strategy will enable the surviving nodes to reconfigure to form a new topology, bringing formerly disused links into service to bypass damaged nodes or links. Such a system will be more robust than a purely tree connected system which will be divided into two non-communicating parts by the loss of a node or link.

- **Dynamic Determination of Cross-Information:** A realistic and scalable method in highly dynamic large scale networks is to use a local algorithm which does not

require knowledge of the common information between nodes. One such method is the covariance intersect filter described in Section 2.5. 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. A number of realistic algorithms using this approach are presented in detail in Section 3.6.

3.3 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.

3.3.1 System Definitions

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

Consider the standard linear continuous-time state space model

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{G}(t)\mathbf{w}(t), \tag{3.27}$$

where $\mathbf{x}(t)$ is the state of interest, $\mathbf{u}(t)$ the control input, $\mathbf{w}(t)$ the driving noise, and $\mathbf{F}(t)$, $\mathbf{B}(t)$, $\mathbf{G}(t)$ matrices modeling effects of state, control and noise inputs respectively. Equation 3.27 admits a well known closed-form solution

$$\mathbf{x}(t) = \mathbf{\Phi}(t, t_0)\mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{\Phi}(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)\mathrm{d}\tau + \int_{t_0}^{t} \mathbf{\Phi}(t, \tau)\mathbf{G}(\tau)\mathbf{w}(\tau)\mathrm{d}\tau, \tag{3.28}$$

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

$$\dot{\mathbf{\Phi}}(t,t_0) = \mathbf{F}(t)\mathbf{\Phi}(t,t_0), \qquad \mathbf{\Phi}(t_0,t_0) = \mathbf{1}. \tag{3.29}$$

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) $\mathbf{\Phi}(t_3,t_1) = \mathbf{\Phi}(t_3,t_2)\mathbf{\Phi}(t_2,t_1)$.
3. $\mathbf{\Phi}(t,t_0)$ is non singular and $\mathbf{\Phi}^{-1}(t,t_0)$=$\mathbf{\Phi}(t_0,t)$.

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

$$\mathbf{\Phi}(t,t_0) = \mathbf{\Phi}(t-t_0) = \exp \mathbf{F}(t-t_0). \tag{3.30}$$

3.3.2 Forward Propagation of State

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

$$\mathbf{F}_{(k+1,k)} \triangleq \Phi(t_{k+1}, t_k). \tag{3.31}$$

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 $\overline{\mathbf{u}}(k)$ and $\overline{\mathbf{w}}(k)$ so that with the definitions

$$\begin{aligned} \mathbf{B}_{(k+1,k)} &\triangleq \int_{t_k}^{t_{k+1}} \Phi(t_{k+1},\tau)\mathbf{B}(\tau)d\tau \\ \mathbf{G}_{(k+1,k)} &\triangleq \int_{t_k}^{t_{k+1}} \Phi(t_{k+1},\tau)\mathbf{G}(\tau)d\tau. \end{aligned} \tag{3.32}$$

Equation 3.28 becomes

$$\mathbf{x}(t_{k+1}) = \mathbf{F}_{(k+1,k)}\mathbf{x}(t_k) + \mathbf{B}_{(k+1,k)}\overline{\mathbf{u}}(k) + \mathbf{G}_{(k+1,k)}\overline{\mathbf{w}}(k), \tag{3.33}$$

where the noise sequence $\overline{\mathbf{w}}(k)$ is assumed zero mean and white with variance $\mathbf{Q}_k$. Equation 3.33 describes the evolution of the state forward from a time t_k to t_{k+1}. Equation 3.33

can be generalised to describe propagation forward by n time slots as

$$\mathbf{x}(t_{k+n}) = \mathbf{F}_{(k+n,k)}\mathbf{x}(t_k) + \sum_{i=1}^{n} \mathbf{B}_{(k+i,k)}\overline{\mathbf{u}}(k+i-1) + \sum_{i=1}^{n} \mathbf{G}_{(k+i,k)}\overline{\mathbf{w}}(k+i-1). \quad (3.34)$$

3.3.3 Backward Propagation of State

Starting with Equation 3.33, decrementing the time index and rearranging gives

$$\mathbf{x}(t_{k-1}) = \mathbf{F}^{-1}_{(k,k-1)}\mathbf{x}(t_k) - \mathbf{F}^{-1}_{(k,k-1)}\mathbf{B}_{(k,k-1)}\overline{\mathbf{u}}(k-1) - \mathbf{F}^{-1}_{(k,k-1)}\mathbf{G}_{(k,k-1)}\overline{\mathbf{w}}(k-1).$$

Noting that

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

so consequently

$$\begin{aligned}
-\mathbf{F}^{-1}_{(k,k-1)}\mathbf{B}_{(k,k-1)} &= -\Phi(t_{k-1}, t_k)\int_{t_{k-1}}^{t_k} \Phi(t_k, \tau)\mathbf{B}(\tau)d\tau \\
&= -\int_{t_{k-1}}^{t_k} \Phi(t_{k-1}, t_k)\Phi(t_k, \tau)\mathbf{B}(\tau)d\tau \\
&= -\int_{t_{k-1}}^{t_k} \Phi(t_{k-1}, \tau)\mathbf{B}(\tau)d\tau \\
&= \int_{t_k}^{t_{k-1}} \Phi(t_{k-1}, \tau)\mathbf{B}(\tau)d\tau \\
&= \mathbf{B}_{(k-1,k)},
\end{aligned}$$

and similarly

$$-\mathbf{F}^{-1}_{(k,k-1)}\mathbf{G}_{(k,k-1)} = \mathbf{G}_{(k-1,k)},$$

gives

$$\mathbf{x}(t_{k-1}) = \mathbf{F}_{(k-1,k)}\mathbf{x}(t_k) + \mathbf{B}_{(k-1,k)}\overline{\mathbf{u}}(k-1) + \mathbf{G}_{(k-1,k)}\overline{\mathbf{w}}(k-1), \quad (3.35)$$

describing the evolution of the state backward in time from t_k to t_{k-1}. Equation 3.35 can be generalised to describe propagation backward by m time slots as

$$\mathbf{x}(t_{k-m}) = \mathbf{F}_{(k-m,k)}\mathbf{x}(t_k) + \sum_{i=1}^{m} \mathbf{B}_{(k-i,k)}\overline{\mathbf{u}}(k-i) + \sum_{i=1}^{m} \mathbf{G}_{(k-i,k)}\overline{\mathbf{w}}(k-i). \quad (3.36)$$

3.3.4 Notation for Time Invariant Systems

If the system $\mathbf{F}(t)$, $\mathbf{B}(t)$, $\mathbf{G}(t)$ 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

$$\mathbf{F}_k = \mathbf{F}_{k+} = \mathbf{F}_{(k+1,k)}, \quad \mathbf{B}_k = \mathbf{B}_{k+} = \mathbf{B}_{(k+1,k)}, \quad \mathbf{G}_k = \mathbf{G}_{k+} = \mathbf{G}_{(k+1,k)},$$

as the 'one-step forward' transition matrices,

$$\mathbf{F}_{k+n} = \mathbf{F}_{(k+n,k)}, \quad \mathbf{B}_{k+n} = \mathbf{B}_{(k+n,k)}, \quad \mathbf{G}_{k+n} = \mathbf{G}_{(k+n,k)},$$

as the 'n-step forward' transition matrices,

$$\mathbf{F}_{k-} = \mathbf{F}_{(k-1,k)}, \quad \mathbf{B}_{k-} = \mathbf{B}_{(k-1,k)}, \quad \mathbf{G}_{k-} = \mathbf{G}_{(k-1,k)},$$

as the 'one-step backward' transition matrices, and

$$\mathbf{F}_{k-m} = \mathbf{F}_{(k-m,k)}, \quad \mathbf{B}_{k-m} = \mathbf{B}_{(k-m,k)}, \quad \mathbf{G}_{k-m} = \mathbf{G}_{(k-m,k)},$$

as the 'm-step backward' transition matrices. Note, because of the asynchronous one-step time difference

$$\mathbf{F}_k = \mathbf{F}_{(k+1,k)} \neq \mathbf{F}_{k-}^{-1} = \mathbf{F}_{(k-1,k)}^{-1} = \mathbf{F}_{(k,k-1)},$$

in general.

Equation 3.33 is now written as

$$\mathbf{x}(t_{k+1}) = \mathbf{F}_k \mathbf{x}(t_k) + \mathbf{B}_k \overline{\mathbf{u}}(k) + \mathbf{G}_k \overline{\mathbf{w}}(k),$$

and Equation 3.35 as

$$\mathbf{x}(t_{k-1}) = \mathbf{F}_{k-} \mathbf{x}(t_k) + \mathbf{B}_{k-} \overline{\mathbf{u}}(k-1) + \mathbf{G}_{k-} \overline{\mathbf{w}}(k-1).$$

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

3.3.5 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

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(t_k) + \mathbf{v}(k),$$

where $\mathbf{v}(k)$ is taken to be a zero mean white noise sequence with covariance $\mathbf{R}_k$. Define the set of all observations up to t_k as

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

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

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

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

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

3.3.6 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

$$\begin{aligned} \hat{\mathbf{x}}(k+1 \mid k) &= \mathbf{F}_k\hat{\mathbf{x}}(k \mid k) + \mathbf{B}_k\mathbf{u}(k) \\ \mathbf{P}(k+1 \mid k) &= \mathbf{F}_k\mathbf{P}(k \mid k)\mathbf{F}_k^T + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^T. \end{aligned}$$

For the information filter propagation, define

$$\mathbf{\Sigma}_{(k+1,k)} = \mathbf{G}_k^T\mathbf{M}_{(k+1,k)}\mathbf{G}_k + \mathbf{Q}_k^{-1}, \tag{3.37}$$

where

$$\mathbf{M}_{(k+1,k)} = \mathbf{F}_{(k+1,k)}^{-T}\mathbf{Y}(k \mid k)\mathbf{F}_{(k+1,k)}^{-1}$$

$$\begin{aligned} &= \mathbf{F}^T_{(k,k+1)}\mathbf{Y}(k \mid k)\mathbf{F}_{(k,k+1)} \\ &\triangleq \mathbf{M}_k, \end{aligned}$$

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

$$\mathbf{Y}(k+1 \mid k) = \mathbf{M}_k - \mathbf{M}_k\mathbf{G}_k\boldsymbol{\Sigma}^{-1}_{(k+1,k)}\mathbf{G}^T_k\mathbf{M}_k, \tag{3.38}$$

and the 'one-step forward' propagation of the information state is

$$\begin{aligned} \hat{\mathbf{y}}(k+1 \mid k) &= \left[1 - \mathbf{M}_k\mathbf{G}_k\boldsymbol{\Sigma}^{-1}_{(k+1,k)}\mathbf{G}^T_k\right]\left[\mathbf{F}^{-T}_k\hat{\mathbf{y}}(k \mid k) + \mathbf{M}_k\mathbf{B}_k\mathbf{u}(k)\right] \\ &= \left[1 - \mathbf{M}_k\mathbf{G}_k\boldsymbol{\Sigma}^{-1}_{(k+1,k)}\mathbf{G}^T_k\right]\mathbf{F}^{-T}_k\hat{\mathbf{y}}(k \mid k) + \mathbf{Y}(k+1 \mid k)\mathbf{B}_k\mathbf{u}(k). \end{aligned} \tag{3.39}$$

The 'n-step forward' propagation are best obtained through direct evaluation of $\mathbf{F}_k$, $\mathbf{G}_k$ and $\mathbf{B}_k$ over a combined time interval.

3.3.7 Backward Time Propagation

For convenience, the propagation of information from a time t_k to a time t_{k-1} is considered. The propagation of state estimates and covariances follows according to[1]

$$\begin{aligned} \hat{\mathbf{x}}(k-1 \mid k) &= \mathbf{F}_{k-}\hat{\mathbf{x}}(k \mid k) + \mathbf{B}_{k-}\mathbf{u}(k-1) \\ \mathbf{P}(k-1 \mid k) &= \mathbf{F}_{k-}\mathbf{P}(k \mid k)\mathbf{F}^T_{k-} - \mathbf{G}_{k-}\mathbf{Q}_{k-1}\mathbf{G}^T_{k-}. \end{aligned}$$

For the information filter propagation, define

$$\boldsymbol{\Sigma}_{(k-1,k)} = \mathbf{Q}^{-1}_{k-1} - \mathbf{G}^T_{k-}\mathbf{M}_{(k-1,k)}\mathbf{G}_{k-}, \tag{3.40}$$

where

$$\mathbf{M}_{(k-1,k)} = \mathbf{F}^{-T}_{(k-1,k)}\mathbf{Y}(k \mid k)\mathbf{F}^{-1}_{(k-1,k)}$$

[1]Note, the indices $(k \mid k)$ and $(k-1 \mid k)$ do not imply an update at time k; exactly the same propagation equations would apply for any indices of the form $(k \mid k-m)$ and $(k-1 \mid k-m)$.

$$\begin{aligned} &= \mathbf{F}_{(k,k-1)}^{T}\mathbf{Y}(k \mid k)\mathbf{F}_{(k,k-1)} \\ &\triangleq \mathbf{M}_{k-}, \end{aligned}$$

is a noise-less back-propagation of information. The 'one-step backward' propagation of the information matrix is

$$\mathbf{Y}(k-1 \mid k) = \mathbf{M}_{k-} + \mathbf{M}_{k-}\mathbf{G}_{k-}\boldsymbol{\Sigma}_{(k-1,k)}^{-1}\mathbf{G}_{k-}^{T}\mathbf{M}_{k-}, \tag{3.41}$$

and the 'one-step backward' propagation of the information state is

$$\begin{aligned} \hat{\mathbf{y}}(k-1 \mid k) &= \left[1 + \mathbf{M}_{k-}\mathbf{G}_{k-}\boldsymbol{\Sigma}_{(k-1,k)}^{-1}\mathbf{G}_{k-}^{T}\right]\left[\mathbf{F}_{k-}^{-T}\hat{\mathbf{y}}(k \mid k) + \mathbf{M}_{k-}\mathbf{B}_{k-}\mathbf{u}(k-1)\right] \\ &= \left[1 + \mathbf{M}_{k-}\mathbf{G}_{k-}\boldsymbol{\Sigma}_{(k-1,k)}^{-1}\mathbf{G}_{k-}^{T}\right]\mathbf{F}_{k-}^{-T}\hat{\mathbf{y}}(k \mid k) + \mathbf{Y}(k-1 \mid k)\mathbf{B}_{k-}\mathbf{u}(k-1). \end{aligned} \tag{3.42}$$

The 'm-step backward' propagation is also best obtained through direct evaluation of $\mathbf{F}_{k-}$, $\mathbf{G}_{k-}$ and $\mathbf{B}_{k-}$ over a combined time interval.

3.3.8 Delayed Data

The delayed data problem, where information or observations arrive late at the fusion process, can affect the real time implementation of any data fusion system. One of the simplest and most common causes is as a result of a communication latency between different parts of the system. Other systems may use sensors that take some form of scan for an observation which must undergo some time consuming batch processing to transform it to a usable format. For these and many other reasons, it is not uncommon for data to arrive late at the fusion process.

There has been a large amount of research on delayed data in the Kalman filter, including both conservative [9, 25] and exact [7, 50, 67] solutions to the problem. However, these solutions are all developed in state-space form. An exact solution to the problem for the information form of the Kalman filter was therefore derived [77].

The problem that arises is how to fuse an observation taken at $k-1$ that arrives sufficiently late that the filter has already been propagated forward to k. The most common solution to this problem has previously been to propagate the delayed observation information through

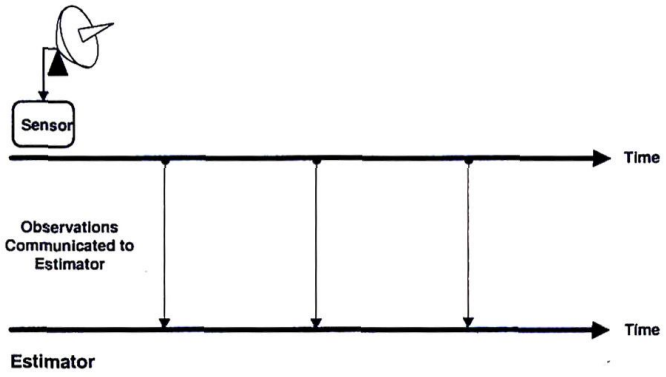


Figure 3.5: Ideal timing where observations are communicated to the estimator and arrive immediately.

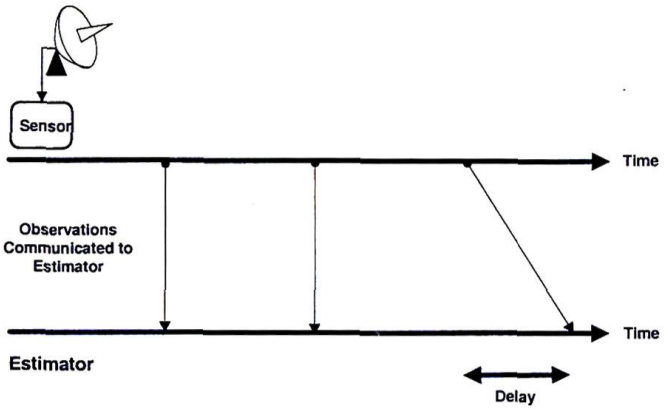


Figure 3.6: Delayed data timing occurs when observations are communicated to the estimator but arrive with some delay.

the standard information prediction cycle. This is done by using $\mathbf{I}(.)$ and $\mathbf{i}(.)$ instead of $\mathbf{Y}(. \mid .)$ and $\hat{\mathbf{y}}(. \mid .)$ in Equations 3.38 and 3.39. For this cycle, $\mathbf{M}_k^Y$ is simply replaced by $\mathbf{M}_k^I$, given below.

$$\mathbf{M}_k^I = \mathbf{F}_k^{-T}\mathbf{I}(k)\mathbf{F}_k^{-1}$$

The predicted observation information is then fused at the current time k using the standard update Equations 2.36 and 2.35. While this method is simple and consistent, it is conservative.

An exact solution to the delayed data update problem can be obtained in the following manner: Define $\mathbf{Y}^+(k \mid k-1)$ to be the information at time k once the delayed data has

been fused, and $\mathbf{Y}^-(k \mid k-1)$ the same information but without the delayed data having been fused. The difference between these two quantities is the delayed information gain $\mathbf{I}^+(k)$ defined through

$$\mathbf{Y}^+(k \mid k-1) = \mathbf{Y}^-(k \mid k-1) + \mathbf{I}^+(k) \tag{3.43}$$

$$\hat{\mathbf{y}}^+(k \mid k-1) = \hat{\mathbf{y}}^-(k \mid k-1) + \mathbf{i}^+(k). \tag{3.44}$$

An expression for $\mathbf{Y}^+(k \mid k-1)$ can be obtained in terms of contributions from both the previous prediction and the delayed information as

$$\begin{aligned}\mathbf{Y}^+(k \mid k-1) &= \left(\mathbf{M}_k^Y + \mathbf{M}_k^I\right) \\ &\quad - \left(\mathbf{M}_k^Y + \mathbf{M}_k^I\right)\mathbf{G}_k\left(\boldsymbol{\Sigma}_k + \mathbf{G}_k^T\mathbf{M}_k^I\mathbf{G}_k\right)^{-1}\mathbf{G}_k^T\left(\mathbf{M}_k^Y + \mathbf{M}_k^I\right) \qquad (3.45)\\ \hat{\mathbf{y}}^+(k \mid k-1) &= \left[\mathbf{1} - \left(\mathbf{M}_k^Y + \mathbf{M}_k^I\right)\mathbf{G}_k\left(\boldsymbol{\Sigma}_k + \mathbf{G}_k^T\mathbf{M}_k^I\mathbf{G}_k\right)^{-1}\mathbf{G}_k^T\right]\mathbf{F}_k^{-T} \\ &\quad \times \left(\hat{\mathbf{y}}(k-1 \mid k-1) + \mathbf{i}(k-1)\right) + \mathbf{Y}^+(k \mid k-1)\mathbf{B}_k\mathbf{u}(k), \qquad (3.46)\end{aligned}$$

where $\mathbf{M}_k^Y$ and $\mathbf{M}_k^I$ are the noiseless propagation of the information matrix and observation information respectively and $\boldsymbol{\Sigma}_k$ is defined by

$$\boldsymbol{\Sigma}_k = \left(\mathbf{G}_k^T\mathbf{M}_k^Y\mathbf{G}_k + \mathbf{Q}_k^{-1}\right).$$

Equations 3.45 and 3.46 are simply the prediction from $k-1$ to k for the ideal case where the observation is *not* delayed.

As the prediction of the information estimate without the delayed observation has already been calculated as

$$\mathbf{Y}^-(k \mid k-1) = \mathbf{M}_k^Y - \mathbf{M}_k^Y\mathbf{G}_k\boldsymbol{\Sigma}_k^{-1}\mathbf{G}_k^T\mathbf{M}_k^Y \tag{3.47}$$

$$\begin{aligned}\hat{\mathbf{y}}^-(k \mid k-1) &= \left[\mathbf{1} - \mathbf{M}_k^Y\mathbf{G}_k\boldsymbol{\Sigma}_k^{-1}\mathbf{G}_k^T\right]\mathbf{F}_k^{-T}\hat{\mathbf{y}}(k-1 \mid k-1) \\ &\quad + \mathbf{Y}^-(k \mid k-1)\mathbf{B}_k\mathbf{u}(k), \qquad (3.48)\end{aligned}$$

it is necessary to remove this from Equations 3.45 and 3.46 in order to obtain $\mathbf{I}^+(k)$ and $\mathbf{i}^+(k)$. Therefore, subtracting Equation 3.47 and 3.48 from Equations 3.45 and 3.46 respec-

tively gives

$$\begin{aligned}
\mathbf{I}^{+}(k) &= \mathbf{M}_k^I + \mathbf{M}_k^Y \mathbf{G}_k {\boldsymbol{\Sigma}_k}^{-1} \mathbf{G}_k^T \mathbf{M}_k^Y \\
&\quad - \left[\mathbf{M}_k^Y + \mathbf{M}_k^I\right] \mathbf{G}_k \left[\boldsymbol{\Sigma}_k + \mathbf{G}_k^T \mathbf{M}_k^I \mathbf{G}_k\right]^{-1} \mathbf{G}_k^T \left[\mathbf{M}_k^Y + \mathbf{M}_k^I\right]^T \qquad (3.49) \\
\mathbf{i}^{+}(k) &= \mathbf{F}_k^{-T} \mathbf{i}(k-1) + \left[\mathbf{M}_k^Y \mathbf{G}_k \boldsymbol{\Sigma}_k^{-1} \mathbf{G}_k^T\right] \mathbf{F}_k^{-T} \hat{\mathbf{y}}(k \mid k) \\
&\quad - \left[\left(\mathbf{M}_k^Y + \mathbf{M}_k^I\right) \mathbf{G}_k \left(\boldsymbol{\Sigma}_k + \mathbf{G}_k^T \mathbf{M}_k^I \mathbf{G}_k\right)^{-1} \mathbf{G}_k^T\right] \mathbf{F}_k^{-T} \\
&\quad \times \left(\hat{\mathbf{y}}(k-1 \mid k-1) + \mathbf{i}(k-1)\right) + \mathbf{I}^{+}(k) \mathbf{B}_k \mathbf{u}(k), \qquad (3.50)
\end{aligned}$$

which corresponds to the increment of information that the observation at $k-1$ contributes at the current time k. This increment can now be fused with the current information state estimate without the delayed observation (given by $\mathbf{Y}^{-}(k \mid k-1)$ and $\hat{\mathbf{y}}^{-}(k \mid k-1)$) using Equations 3.43 and 3.44 to yield the exact result as if the observation had not been delayed.

It can be seen from Equation 3.43 that the net effect of this algorithm is to produce an estimate in the form

$$\mathbf{Y}(k \mid k) = \mathbf{Y}_Y(k \mid k-1) + \mathbf{Y}_I(k \mid k-1) + \mathbf{Y}_{YI}(k \mid k-1),$$

where $\mathbf{Y}_Y(k \mid k-1)$ is the estimate obtained without the delayed information, $\mathbf{Y}_I(k \mid k-1)$ is the estimate obtained using only the delayed information, and $\mathbf{Y}_{YI}(k \mid k-1)$ is a cross-information term (uniquely defined $\mathbf{Y}_Y(k \mid k-1)$ and $\mathbf{Y}_I(k \mid k-1)$) describing the cross-information between the information states caused by propagation through a common process model.

It should also be noted that although the exact delayed data solutions requires both $\mathbf{M}_k^Y$ and $\hat{\mathbf{y}}(k-1 \mid k-1)$, these values can be determined by backward propagation (Equations 3.41 and 3.42) as long as no other information has arrived since the delayed observation time. If another observation has arrived in this period, then some filter history must be remembered in order to obtain an exact solution. The following section covers this in detail.

Example 3 *Consider again the simple tracking system using the models given in Example 1.*

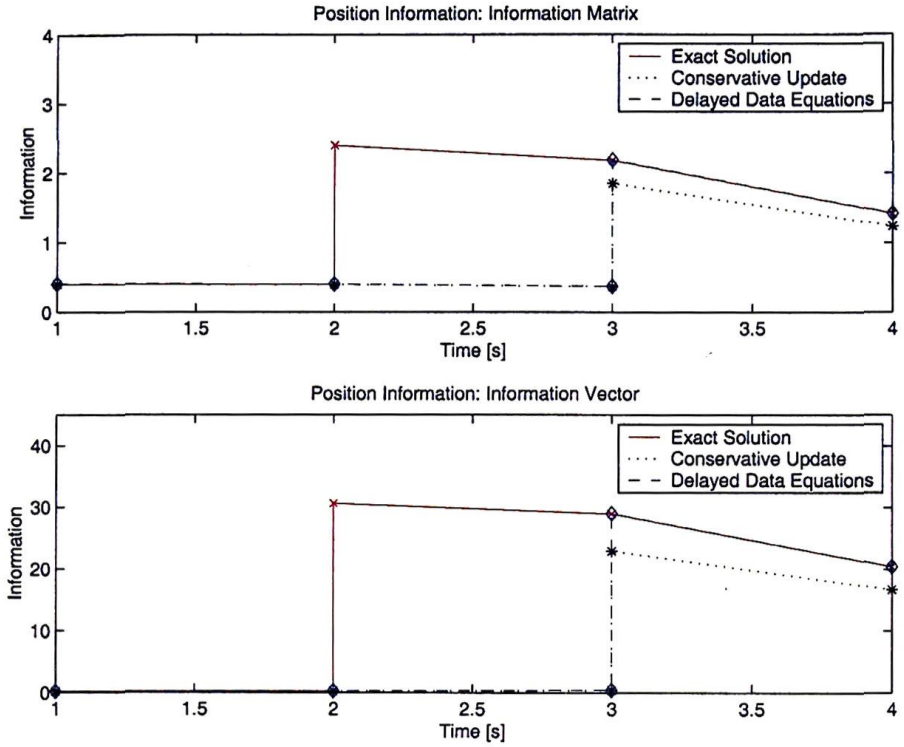


Figure 3.7: Time history of the information filter with delayed data. The red plot illustrates the exact solution where the data is not delayed. The black plot demonstrates the conservative result obtained by predicting the delayed observation information forward to the next time step using the standard filter motion model. The blue plot shows the results of using the exact delayed update equations to predict the observation forward to the next time step, accounting for the correlation between the observation and the estimate over that period.

Let the information state at some time $t = 1$ *be given by*

$$\hat{\mathbf{y}}(1 \mid 1) = \begin{bmatrix} 2 & 3.3333 \end{bmatrix} \qquad \mathbf{Y}(1 \mid 1) = \begin{bmatrix} 0.4 & 0 \\ 0 & 0.3333 \end{bmatrix}$$

The information estimate is propagated forward from time $t = 1$ *to time* $t = 2$ *using the information filter prediction equations. The estimate is again propagated forward to time* $t = 3$. *An observation of the target location, taken at time* $t = 2$ *arrives late at time* $t = 3$. *The increment of new information at time* $t = 3$ *from the delayed observation at* $t = 2$ *is calculated using Equations 3.49 and 3.50. This is then fused with the existing information*

state estimate at $t = 3$ using Equations 3.43 and 3.44 to give the same result as if the data had not been delayed.

Figure 3.7 shows the 3 step time history of this simple example for the exact solution (where the data is not delayed), the delayed data solution handled using Equations 3.50 and 3.49, and for the case where the observation information is predicted forward in a conservative manner. Note that the delayed data equations yield exactly the same result as if the data had arrived on time, whereas the method of predicting the observation forward to the current time is more conservative.

3.3.9 Asequent Data

Data that arrives both late and out of sequence is termed asequent. For example, data from a particular sensor may arrive sufficiently late that the estimate has already been updated by one or more observations from some other sensor.

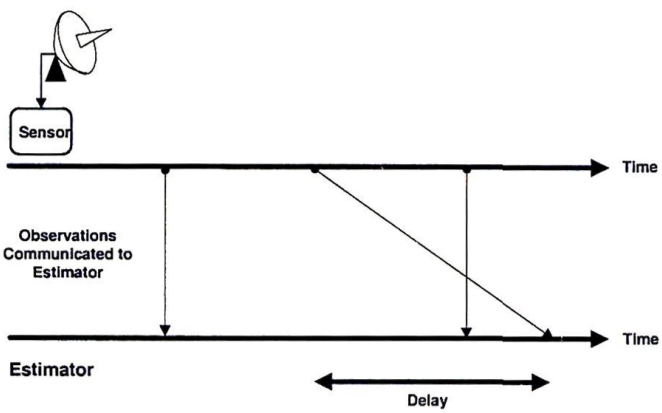


Figure 3.8: Asequent data timing occurs when observations are communicated to the estimator but arrive both late and out of order.

The delayed data Equations 3.49 and 3.50 can be used to yield an exact solution to this problem. However, the solution requires the system to store state history in memory. At each time step, the system should store the current values of

$$\mathbf{M}_k = \mathbf{M}_k^Y + \mathbf{M}_k^I, \qquad \hat{\mathbf{y}}(k \mid k).$$

When the asequent data from some time $k - n$ arrives at the current time k, the update is

performed using the following steps.

1. The values of $\mathbf{M}_{k-n}$ and $\hat{\mathbf{y}}(k-n \mid k-n)$ are recalled from memory to be used as $\mathbf{M}^Y_{k-n}$ and $\hat{\mathbf{y}}(k-n \mid k-n)$ respectively in Equations 3.49 and 3.50. The new observation at $k-n$ is used to calculate $\mathbf{M}^I_{k-n}$ and $\mathbf{i}(k-n)$.

$$\begin{aligned} \mathbf{M}^I_{k-n} &= \mathbf{F}^{-T}_{k-n}\mathbf{I}(k-n)\mathbf{F}^{-1}_{k-n} \\ \mathbf{i}(k-n) &= \mathbf{i}(k-n) \end{aligned}$$

2. Equations 3.43 and 3.44 are applied to yield $\mathbf{I}^+(k-n+1)$ and $\mathbf{i}^+(k-n+1)$.

3. The increments $\mathbf{M}^I_{k-n}$ and $\mathbf{i}(k-n)$ are added to the stored values at time $k-n$ and the resulting new value saved, overwriting the previous value.

$$\begin{aligned} \mathbf{M}_{k-n} &\leftarrow \mathbf{M}_{k-n} + \mathbf{M}^I_{k-n} \\ \hat{\mathbf{y}}(k-n \mid k-n) &\leftarrow \hat{\mathbf{y}}(k-n \mid k-n) + \mathbf{i}(k-n) \end{aligned}$$

This completes the one step forward propagation cycle. The information from the observation at $k-n$ has now been propagated forward to $k-n+1$. The algorithm then repeats to propagate the data from $k-n+1$ to $k-n+2$:

1. The increments $\mathbf{I}^+(k-n+1)$ and $\mathbf{i}^+(k-n+1)$ are propagated one step forward again. This is done by setting $\mathbf{M}^Y_{k-n+1}$ and $\hat{\mathbf{y}}(k-n+1 \mid k-n+1)$ to the stored values at $k-n+1$, and calculating $\mathbf{M}^I_{k-n+1}$ and $\mathbf{i}(k-n+1)$ using $\mathbf{I}^+(k-n+1)$ and $\mathbf{i}^+(k-n+1)$ as

$$\begin{aligned} \mathbf{M}^I_{k-n+1} &= \mathbf{F}^{-T}_{k-n+1}\mathbf{I}^+(k-n+1)\mathbf{F}^{-1}_{k-n+1} \\ \mathbf{i}(k-n+1) &= \mathbf{i}^+(k-n+1). \end{aligned}$$

2. These values are again used in the delayed data Equations 3.49 and 3.50, which yield the one step forward propagation of the delayed observation information to $\mathbf{I}^+(k-n+2)$ and $\mathbf{i}^+(k-n+2)$.

3. The information added to this step by $\mathbf{I}^+(k-n+1)$ and $\mathbf{i}^+(k-n+1)$ is added to stored values at time $k-n+1$, with the resulting new value replacing the previous

one.

$$\begin{aligned}
\mathbf{M}_{k-n+1} &\leftarrow \mathbf{M}_{k-n+1} + \mathbf{F}_{k-n+1}^{-T}\mathbf{I}^{+}(k-n+1)\mathbf{F}_{k-n+1}^{-1} \\
&\leftarrow \mathbf{M}_{k-n+1} + \mathbf{M}_{k-n+1}^{I} \\
\hat{\mathbf{y}}(k-n+1 \mid k-n+1) &\leftarrow \hat{\mathbf{y}}(k-n+1 \mid k-n+1) + \mathbf{i}^{+}(k-n+1)
\end{aligned}$$

The algorithm then continues, using these same steps, until the data has been propagated the n steps forward to the current time where it is added to the information state using the update Equations 3.43 and 3.44.

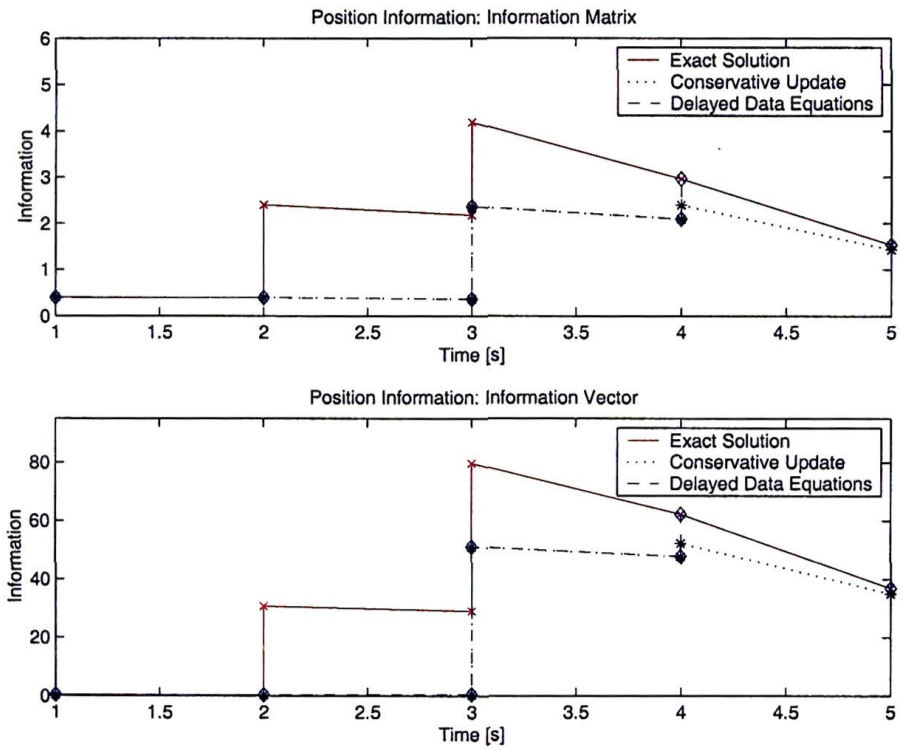


Figure 3.9: Time history of the information matrix of a filter with asequent data. The red plot illustrates the exact solution where the data arrived in time and in order. The black plot demonstrates the conservative result obtained by the tradition method of predicting the delayed observation information from $t = 2$ forward to the time step $t = 4$ using the standard filter motion model. The blue plot shows the results of using the exact asequent update equations to predict the observation forward through each step it was delayed while accounting for the correlation between the observation and the estimate over that period.

Example 4 *Consider the same system used in Example 3. Again, let the initial information*

state at some time $t = 1$ be given by

$$\hat{\mathbf{y}}(1 \mid 1) = \begin{bmatrix} 2 & 3.3333 \end{bmatrix} \qquad \mathbf{Y}(1 \mid 1) = \begin{bmatrix} 0.4 & 0 \\ 0 & 0.3333 \end{bmatrix}.$$

The sensor makes 2 observations at times $t = 2$ and $t = 3$ which are given by

$$\mathbf{z}(2) = [15.2] \qquad \mathbf{z}(3) = [25.3].$$

Due to a communications delay, the observation from $t = 2$ does not arrive until $t = 4$. The system therefore predicts to time 2 then again to time 3 where the filter updates with $\mathbf{z}(3)$. The system is then predicted to $t = 4$ when observation $\mathbf{z}(2)$ arrives.

Figure 3.9 illustrates the results of this simple example and shows that the results of using the asequent data algorithm are exactly the same as the case where the data is not delayed and arrives in order. At time $t = 4$ when the delayed observation arrives, the update returns the exact value. The results of predicting the late observation forward are also shown to be conservative.

3.4 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 $\hat{\mathbf{y}}_{ij}(k \mid k)$ and the common information matrix $\mathbf{Y}_{ij}(k \mid k)$.

The prediction equations (for both forward and backward propagation) used in the channel filter are the standard information filter equations described in Sections 3.3.6 and 3.3.7. 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 synchronous time (the local synchronous time is referred to as a '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

$$\mathbf{m}_j(k) \;=\; \hat{\mathbf{y}}_j(k \mid k-n) - \hat{\mathbf{y}}_{ij}(k \mid k-m)$$

$$\mathbf{M}_j(k) = \mathbf{Y}_j(k \mid k-n) - \mathbf{Y}_{ij}(k \mid k-m). \tag{3.51}$$

Note now that the time indices $\mathbf{Y}_j(k \mid k-n)$ and $\mathbf{Y}_{ij}(k \mid k-m)$ are different as the different nodes are asynchronous and the two information sets can be predicted over different time horizons. This information gain $\mathbf{m}(k)$ and $\mathbf{M}(k)$ is analogous to the observation information vector $\mathbf{i}(k)$ and $\mathbf{I}(k)$, and thus is added in the update stage for the channel. The update for the channel filter between nodes i and j when new information has been received from node j can then be written as

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_{ij}(k \mid k-m) + \mathbf{m}_j(k) \\ \mathbf{Y}_{ij}(k \mid k) &= \mathbf{Y}_{ij}(k \mid k-m) + \mathbf{M}_j(k). \end{aligned} \tag{3.52}$$

This can be further simplified by substituting Equation 3.51

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_{ij}(k \mid k-m) + \mathbf{m}_j(k) \\ \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_{ij}(k \mid k-m) + \hat{\mathbf{y}}_j(k \mid k-n) - \hat{\mathbf{y}}_{ij}(k \mid k-m) \\ \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_j(k \mid k-n). \end{aligned} \tag{3.53}$$

This reveals a trivial update stage where the channel filter merely overwrites the previous state with the newer one. When a new information set arrives at node i from node j, the channel filter is updated by

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_j(k \mid k-n) \\ \mathbf{Y}_{ij}(k \mid k) &= \mathbf{Y}_j(k \mid k-n). \end{aligned} \tag{3.54}$$

Alternatively, if the new information at the channel filter of node i is from the local filter at node i, the update becomes

$$\begin{aligned} \hat{\mathbf{y}}_{ij}(k \mid k) &= \hat{\mathbf{y}}_i(k \mid k) \\ \mathbf{Y}_{ij}(k \mid k) &= \mathbf{Y}_i(k \mid k). \end{aligned} \tag{3.55}$$

While the normal information filter update of Equation 3.52 is perfectly valid, implementation in the form of Equation 3.54 and 3.55 is clearly much simpler.

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. There are a number of important implications of these results which will be discussed in this section.

3.4.1 Management of Communication

The communication channels exploit the associativity property of information measures. The channels take the total local information $\tilde{\mathbf{y}}_i(k \mid k)$ and subtract out all information that has previously been communicated down the channel, $\hat{\mathbf{y}}_{ij}(k \mid k)$, thus transmitting only new information obtained by node i since the last communication. Intuitively, communicated data from node i thus consists only of information not previously transmitted to a node j; because common data has already been removed from the communication, node j can simply assimilate incoming information measures by addition.

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 accumulated data can be employed to substantially reduce communication bandwidth requirements (and indeed be used to manage communications); if a node is disconnected from the communications network, it can be re-introduced and information synchronised in one step (the same applies to new nodes entering the system, dynamic network changes and signal jamming).

The remainder of this section details a practical algorithm for the implementation of a channel.

3.4.2 Structure of the Communication Algorithm

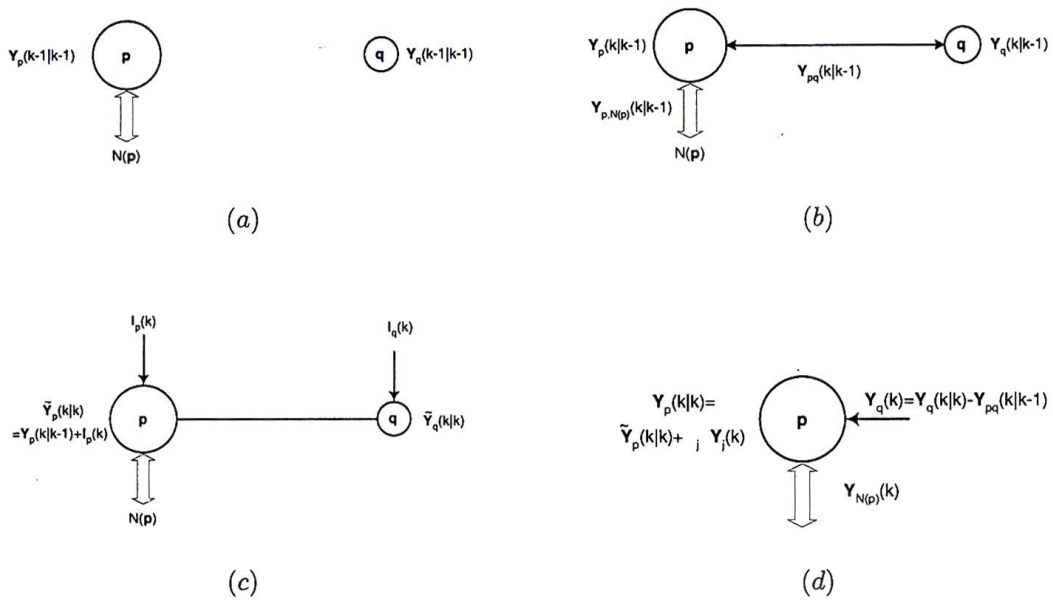


Figure 3.10: Structure of the general communication channel algorithm: (a) Initial state of the node p prior to establishing communication with node q; (b) establishing common information between nodes p and q at a time k in a channel pq; (c) computation of local, partial estimates at the two nodes; (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 3.10. Node p has a local information estimate $\mathbf{Y}_p(k-1 \mid k-1)$ 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 $\mathbf{Y}_p(k-1 \mid k-1)$ and $\mathbf{Y}_q(k-1 \mid k-1)$ are placed in the channel.

2. The two estimates are then time-aligned using the information state prediction or retrodiction Equations 3.38-3.42 described in Section 3.3. 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 $\mathbf{Y}_p(k \mid k-1)$ and $\mathbf{Y}_q(k \mid k-1)$.

3. The common information $\mathbf{Y}_{p \cap q}(k \mid k-1)$ between nodes p and q is then determined. This information may be computed by any of the methods described in Section 3.4.4. The result of this process is a channel between the two nodes with a measure of the common information $\mathbf{Y}_{p \cap q}(k \mid k-1)$ between them at a time k. Similar measures $\mathbf{Y}_{p \cap j}(k \mid k-1)$ are computed for every connected node $j \in N(p)$ in the neighbourhood of p. This is shown diagramatically in Figure 3.10(b).

4. New information (the information gain) between node p and it's neighbours is communicated. As a first step, it is assumed that node p acquires some local observation information $\mathbf{I}_p(k)$ at time k and fuses this with its local prediction $\mathbf{Y}_p(k \mid k-1)$ to produce a local, partial, estimate $\tilde{\mathbf{Y}}_p(k \mid k)$ at time k as

$$\tilde{\mathbf{Y}}_p(k \mid k) = \mathbf{Y}_p(k \mid k-1) + \mathbf{I}_p(k). \tag{3.56}$$

   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 $\tilde{\mathbf{Y}}_p(k \mid k) = \mathbf{Y}_p(k \mid k-1)$). This is shown diagramatically in Figure 3.10(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.

5. 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 $\tilde{\mathbf{Y}}_q(k \mid k)$ from a node q to node p and then to subtract the common information $\mathbf{Y}_{p \cap q}(k \mid k-1)$ 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

$$\Delta \mathbf{Y}_{pq}(k \mid k) = \tilde{\mathbf{Y}}_q(k \mid k) - \mathbf{Y}_{p \cap q}(k \mid k-1). \tag{3.57}$$

Finally, the local estimate is updated by simply adding information gains coming from each channel as

$$\mathbf{Y}_p(k \mid k) = \tilde{\mathbf{Y}}_p(k \mid k) + \sum_{j \in N(p)} \Delta \mathbf{Y}_j(k \mid k). \tag{3.58}$$

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 3.10(d).

3.4.3 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 3.56 into Equation 3.57 to obtain

$$\Delta \mathbf{Y}_{pq}(k \mid k) = \mathbf{I}_q(k) + \mathbf{Y}_q(k \mid k-1) - \mathbf{Y}_{p \cap q}(k \mid k-1), \tag{3.59}$$

shows that local measurement information is always communicated in its entirety. Further, the term $\mathbf{Y}_q(k \mid k-1) - \mathbf{Y}_{p \cap q}(k \mid k-1)$ 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.

3.4.4 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 $\mathbf{Y}_{p \cap q}(k \mid k-1) = \mathbf{Y}_p(k \mid k-1) = \mathbf{Y}_q(k \mid k-1)$.

With this substituted in Equation 3.59, the information gain is simply the locally observed information $\Delta\mathbf{Y}_{pq}(k \mid k) = \mathbf{I}_q(k)$. 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 $\mathbf{Y}_{p\cap q}(k \mid k-n)$ is set to the common estimate $\mathbf{Y}_p(k-n \mid k-n)$ at the last communication time $k-n$, appropriately predicted forward to the new communication time k. The local estimates $\tilde{\mathbf{Y}}_q(k \mid k)$ computed in Equation 3.56 now include all observation information $\mathbf{I}_q(k-j)$, $j = n, \cdots, 0$ taken over the interval $k-n$ to k (and again, appropriately predicted through to time k). The computed information gain $\Delta\mathbf{Y}_{pq}(k \mid k)$ obtained from Equation 3.57 now consists of all this observation information with the last common prior $\mathbf{Y}_p(k-n \mid k-n)$ 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 $\Delta\mathbf{Y}_{pq}(k \mid k)$ 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 $\mathbf{I}_q(k-n)$ over a time n generally results in a net loss of information through the addition of the process uncertainty accumulated over the delay time.

3.5 A Practical Decentralised Node Algorithm

This section provides a practical algorithm for the general implementation of a decentralised node. The communication algorithm is '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 3.11.

The structure of a decentralised data fusion node can generalised to four specific modules. Shown in graphically in Figure 3.11, these modules are the sensor preprocessing, local filter, channel manager and channel filter.

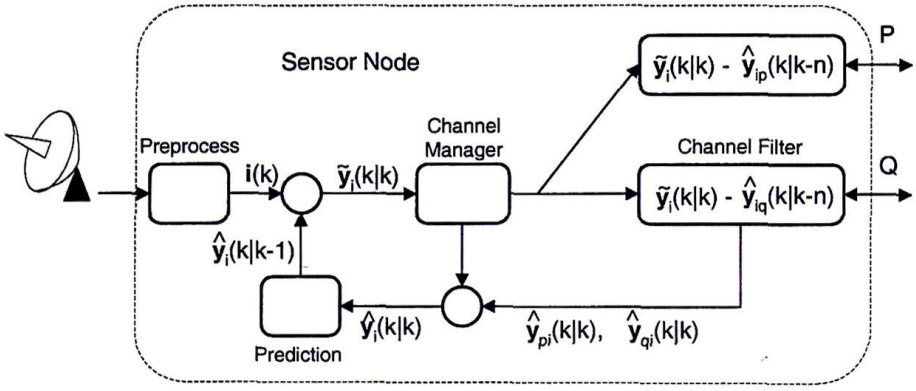


Figure 3.11: Structure of a decentralised node.

Regardless of the network topology, it is possible to use this as a generic structure for a node. The algorithm implemented within each module can be changed, as can the number of channels, in order to construct virtually any decentralised network configuration. For example, when looking at the fully connected class of networks the only difference between a bus architecture and a tree connection is the number of channels on each node (the bus architecture needs only one channel, whereas the tree structure may have many). Therefore, Figure 3.11 can be used to represent a node in a tree connected network when it has multiple channels (shown), or as a node in a broadcast network by only having the one channel.

The extension to more general broadcast networks which make no requirements about topology can also fit this structure at a node level by using only one channel and implementing different update mechanisms such as covariance intersection in the channel filter. These general networks will be discussed in more detail in Section 3.6.

The different modules comprising a sensing node are now presented and their purpose described. The implementation of each module is then discussed in the context of a decentralised tree network which is employed in subsequent implementations and demonstrations.

3.5.1 Sensor Preprocessing

When a sensor makes an observation, a number of preprocessing stages must occur to transform observation data into information form ($\mathbf{i}(k)$ and $\mathbf{I}(k)$) before it is fused in the nodal filter. This includes coordinate transformations and conversion of the observation to

information form. Section 5.5.2 describes the specific equations used for this preprocessing in the ANSER project.

For example, when a range/bearing observation is made at some time k, both the observation and its associated covariance matrix are transformed to cartesian coordinates. The observation information matrix and information vector are then computed.

The final preprocessing stage is a data association step. Certain sensors have extra information they can use for data association that may not be available to the decentralised filtering node. This information can be used and transmitted by the sensor preprocessing module to increase association reliability. The sensor preprocessing then outputs the observation information to the decentralised filtering node.

3.5.2 Local Filter

The local information filter, also known as the nodal 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. It is represented in Figure 3.11 by the prediction and summation (update) loop.

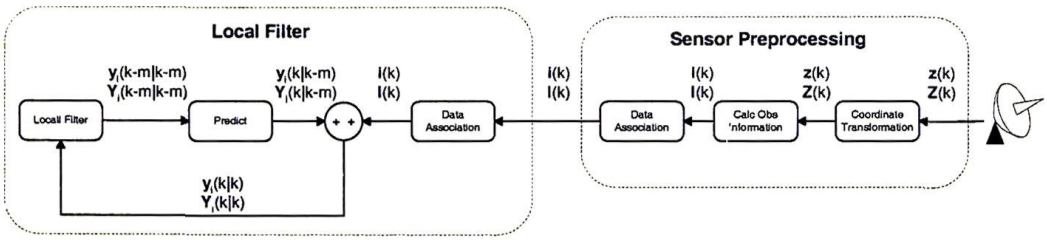


Figure 3.12: Data flow from sensor observation.

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 $\mathbf{I}(k)$. These are then communicated to the local filter. The generation of observations is typically asynchronous. This observation information is communicated to the local node filter where (see Figure 3.12):

1. The local information state is predicted forward to observation time.

2. It passes through a data association stage using a combination of both the information gate of Section 3.5.6 and the results of any data association performed by the sensor preprocessing.

3. With a correct association, the node filter then fuses observation and prediction information through the summation in Equation 2.35.

4. The fused estimate is then propagated forward from observation time to the synchronous node time horizon.

5. Equation 3.38 is generally used to predict the information state forward in time.

If any delayed or asequent data arrives, the exact solution is computed using the algorithms developed in Sections 3.3.8 and Section 3.3.9.

At the given node time horizon, the node filter also receives new information from the channel manager. The state is predicted forward to this time using Equation 3.38 and updated with Equation 2.35. The full information matrix $\mathbf{Y}(k \mid k)$ and information vector $\hat{\mathbf{y}}(k \mid k)$ are then output to the channel manager for transmission to neighbouring nodes.

3.5.3 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 3.13 shows the flow of incoming information through the channels, channel manager and local filter. Note that the channel filter in this figure is implemented for a structured network as a conventional information 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 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 3.38. 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 3.14 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 then cycles to the next available communication time.

3.5.4 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 from the nodal filter and disseminates this to the channel filters for transmission. The channel manager also handles the online channel connectivity and allocation in any network allowing dynamic changes in configuration.

In any implementation of a decentralised network using the architecture presented, it is necessary to keep an estimate of the common information between nodes in a channel filter. However, since this architecture supports multiple nodes operating on multiple processors in different physical locations, it is important to define where the channel filter actually gets implemented. If a single channel filter is used on any given communications link, then there will be two subtly different node types implemented - there will be those nodes with the channel filter and those nodes without. When organising the structure of the network, it is vital to ensure that it was arranged such that non-channel filter nodes were not connected to each other.

In order to remove this requirement for different node types, the architecture presented here uses a channel filter on every node, effectively putting one at each end of a communication link. This dual channel filter system on any given link means that every node is exactly the same and that it is a simple matter to add and remove nodes without concern to possible mismatch between differing node types. It does however, present some new difficulties not present in a single channel filter implementation.

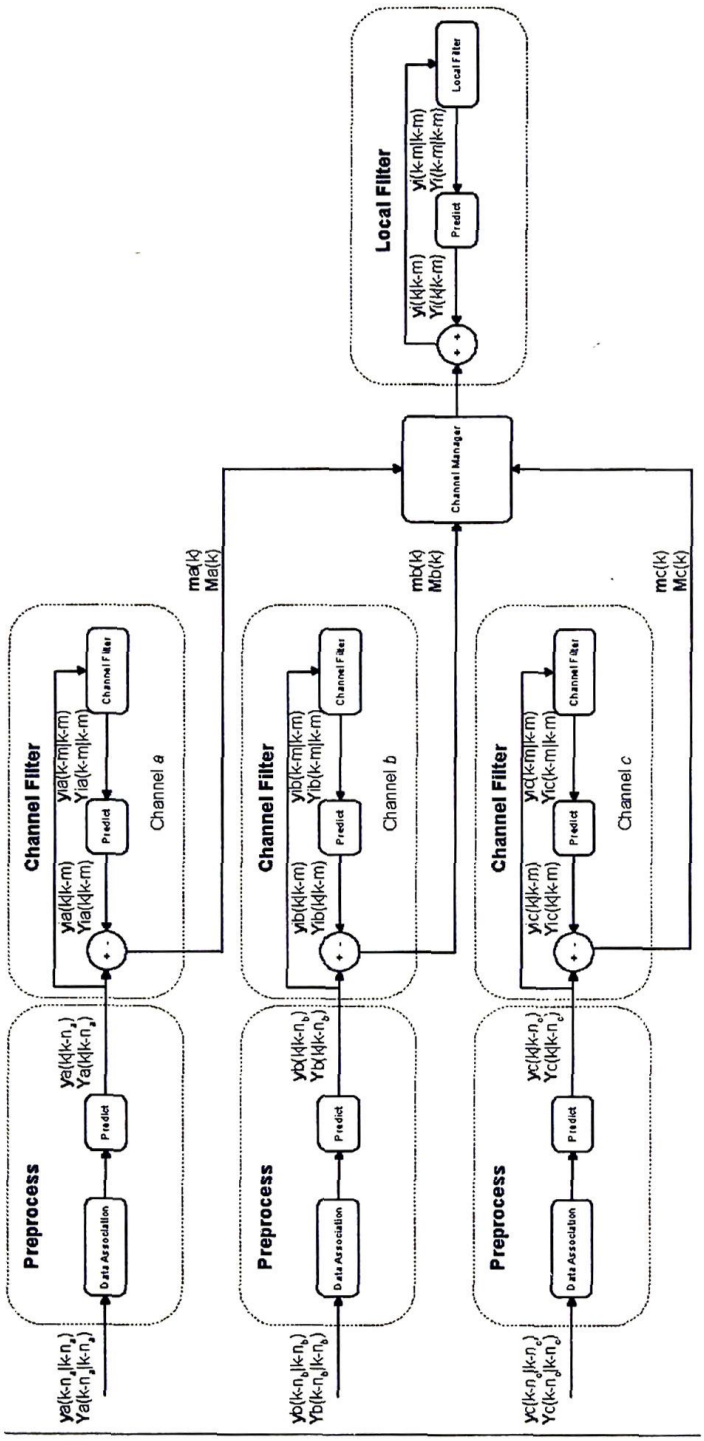


Figure 3.13: Flow of information for incoming channel data in a structured network.

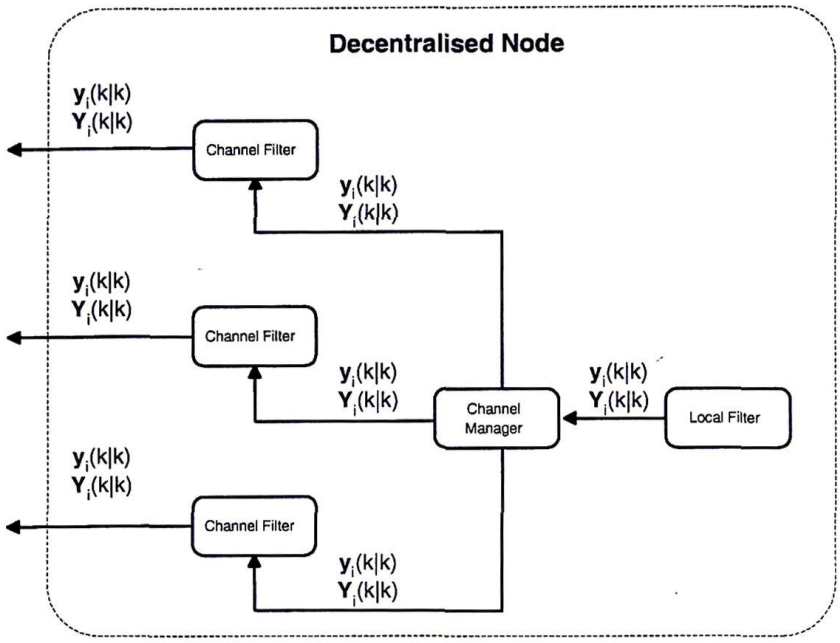


Figure 3.14: Flow of information for outgoing channel data.

It is possible for the dual channel filters to go out of alignment when communication messages are lost between nodes. This occurs as the sender updates its channel estimate when transmitting and the receiver never gets to update as the message is lost. In this situation, the receiver's channel filter will be different from that of the sender. If the receiver then transmits a message of its own and that message is also lost then the problem only gets worse.

Not handling this misalignment between the channel filters does not cause any serious problems such as filter instability, it simply means that it is possible for a node to lose information. If the channel filters are misaligned and the one with less information is the first to successfully get a message through, then the receiving node with greater information will end up with a negative information gain and essentially loses information when aligning itself with the other node.

The simplest way to to handle this problem while ensuring that nodes never lose information is to compare the information content of a received message with the existing channel estimate using some heuristic such as trace or determinate. If the received message has less information than the existing estimate, then discard it. The existing estimate can then be

transmitted back to the other node as it is known to have more information. It is important to note that this method does not necessarily end up using all information, it simply aligns them to the node with the most information.

An exact solution for misaligned channel filters is also possible, however it requires the storage of channel states and the transmission of channel timing information. The channel filter would be required to store the full state $\mathbf{Y}(k \mid k)$ and $\hat{\mathbf{y}}(k \mid k)$ and the time after every update with information received from another node. It would also need to store the time it last transmitted information to another node. The DDF message structure would then be modified to include the time time at which the transmitting node last received information from the node to which it is connected. Using this information, it is possible to keep dual channel filters aligned such that the total amount of information is always used.

3.5.5 Timing

The clock at each node in a decentralised system must be synchronised to a common system time in order to eliminate errors in clock drifts and the subsequent time alignment of data. The method used in this thesis was to synchronise all nodes to GPS time. This is necessary as every node operates asynchronously with respect to the other nodes. The nodes are programmed to communicate their information every time period ΔT (where ΔT 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 (recall that the time horizon is the 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 3.15 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.

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.

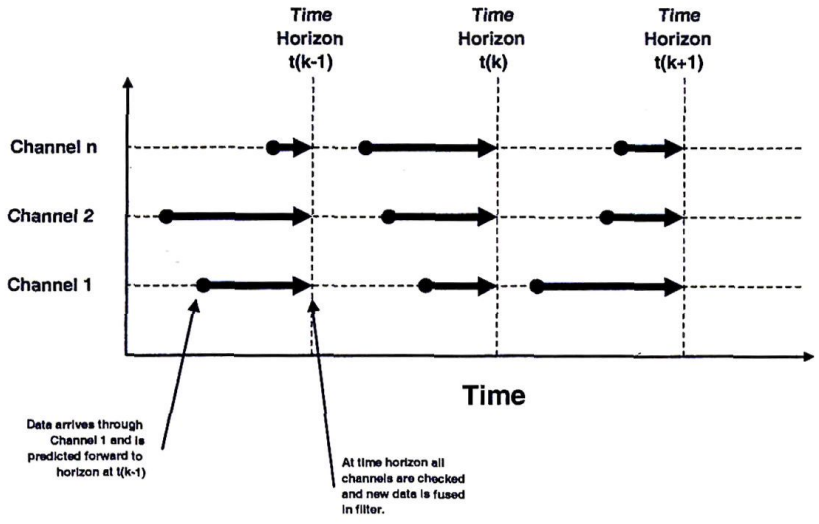


Figure 3.15: Data arrives at the channel filter asynchronously from other nodes. It is predicted forward to a local time horizon and fused then.

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 the 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. In this way, the timing within each node is kept synchronous, but the timing between nodes is asynchronous while ensuring that the filters remain consistent.

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 presented in Section 3.3.8.

3.5.6 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 3.16 illustrates this notion where two nodes are estimating the same target set, but the targets are ordered differently on each node. When node 1 communicates information about its target 1, node 2 must correctly associate it with its own target 3.

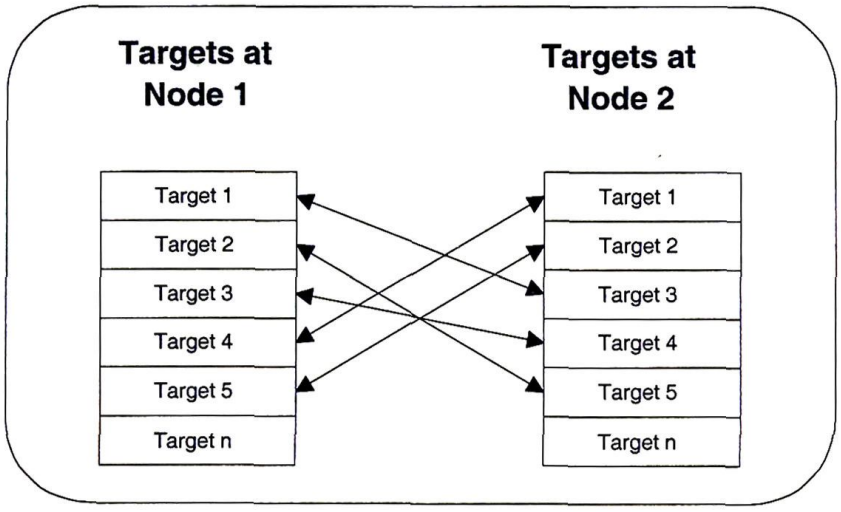


Figure 3.16: Different nodes may have the same physical targets stored in different orders.

The information gate [37] can be used for data association with the information filter. Shown in Equation 3.60 below, it is the information equivalent of the state space innovation gate. The primary advantage of this algorithm is that it is cast in terms of the information states.

$$\mathbf{v}^T(k)\mathbf{B}(k)^+\mathbf{v}(k) < \gamma, \tag{3.60}$$

where γ is selected from a Chi-Squared distribution with the number of degrees of freedom of the observation and

$$\begin{aligned}
\mathbf{v}(k) &= \mathbf{I}(k)\left[\mathbf{I}(k)^+\mathbf{i}(k) - \mathbf{Y}^{-1}(k \mid k-1)\hat{\mathbf{y}}(k \mid k-1)\right] \\
\mathbf{B}(k) &= \mathbf{I}(k)\left[\mathbf{I}(k) + \mathbf{Y}^{-1}(k \mid k-1)\right]\mathbf{I}(k) \\
\mathbf{B}(k)^+ &= \mathbf{H}^T(k)\left[\mathbf{H}(k)\mathbf{B}(k)\mathbf{H}^T(k)\right]\mathbf{H}(k) \\
\mathbf{I}(k)^+ &= \mathbf{H}^T(k)\left[\mathbf{H}(k)\mathbf{I}(k)\mathbf{H}^T(k)\right]\mathbf{H}(k).
\end{aligned} \tag{3.61}$$

A complete derivation of the information gate is given in Appendix A.

The 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 will 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 rather than having to apply a computationally expensive association algorithm at each update iteration. Figure 3.16 illustrates this case, where a lookup table mapping the targets on different nodes is created. Along with the data association index, nodes also include a probability of the data association index being correct.

3.6 Unstructured (Dynamic) Network Topologies

This section presents network configurations which allow for dynamic changes in the network topology. The algorithms presented use the same internal node structure as structured networks, but have modified channel updates to allow for the dynamic aspect of the information.

3.6.1 Limitations of Structured Topologies

The primary advantage of structured network topologies is that common information between nodes can be maintained exactly. However, structured networks also limit online connectivity changes between nodes and are therefore not flexible to change.

Although unstructured networks are flexible they cannot make use of information in an optimal manner. As the source of information is unknown, updates must be performed in a conservative and sub-optimal manner. The unstructured network algorithms presented in this section use the CI algorithm to compute a conservative update.

3.6.2 Broadcast With CI Update

The simplest model for a dynamic network is for nodes to broadcast their full state in information form and for any receiving node to fuse this in the channel using a CI update. Nodes still use the standard information filter update for information received from locally

attached sensors. It is important to note that since CI makes no assumptions about the degree of correlation between nodes, it is technically possible to remove the channel filter completely and simple fuse the communicated estimate directly with the estimate at the nodal filter on reception. However, this will result in the conservative CI update degrading the quality of any information obtained from the locally attached sensor at the receiving node since the last communication. This information is clearly independent as it has not yet been communicated, so it is sensible to preserve its integrity. The use of the channel filter allows this as it only performs the conservative update on information that nodes may have in common.

This method of communication update is completely scalable and is extremely simple to implement. Nodes will never do worse than if they were operating independently, however, it is possible that they will also never do better. This is function of the inherent conservative behavior of the CI update. In practice, what usually occurs is that a few nodes with the best sensors end up dominating the network as they provide the best information. Information from nodes with less accurate sensors is often not used. Figure 2.10(d) illustrates this situation.

Example 5 *Consider the situation where two nodes i and j have each been operating in their own separate sub-network and communicate for the first time. Although they have never communicated with each other before, both nodes have communicated with their sub-networks and will therefore have a non-zero channel estimate. As it is not known whether other nodes have interacted between the two sub-networks, the amount of common information between i and j is completely unknown.*

Using the same constant velocity tracking model developed in Example 1, let the estimates

on the two vehicles be

$$\mathbf{Y}_i(k \mid k) = \begin{bmatrix} 0.4000 & -0.2000 \\ -0.2000 & 0.6000 \end{bmatrix} \quad \hat{\mathbf{y}}_i(k \mid k) = \begin{bmatrix} 3.6000 \\ -0.8000 \end{bmatrix}$$

$$\mathbf{Y}_{i_{Chan}}(k \mid k-1) = \begin{bmatrix} 0.2000 & -0.2000 \\ -0.2000 & 0.6000 \end{bmatrix} \quad \hat{\mathbf{y}}_{i_{Chan}}(k \mid k-1) = \begin{bmatrix} 2.5500 \\ -0.8000 \end{bmatrix}$$

$$\mathbf{Y}_j(k \mid k) = \begin{bmatrix} 0.2667 & -0.1333 \\ -0.1333 & 1.0667 \end{bmatrix} \quad \hat{\mathbf{y}}_j(k \mid k) = \begin{bmatrix} 2.7333 \\ 0.1333 \end{bmatrix}$$

$$\mathbf{Y}_{j_{Chan}}(k \mid k-1) = \begin{bmatrix} 0.0167 & -0.1333 \\ -0.1333 & 1.0667 \end{bmatrix} \quad \hat{\mathbf{y}}_{j_{Chan}}(k \mid k-1) = \begin{bmatrix} 0.1083 \\ 0.1333 \end{bmatrix}$$

When node i now communicates to node j, the channel filter on node i is overwritten with the estimate being sent out, and a CI update is done between the channel estimate at j and the information $\mathbf{Y}_i(k \mid k)$ and $\hat{\mathbf{y}}_i(k \mid k)$. This gives

$$\begin{aligned} \mathbf{Y}_{j_{Chan}}(k \mid k) &= \omega \mathbf{Y}_{j_{Chan}}(k \mid k-1) + (1-\omega)\,\mathbf{Y}_i(k \mid k) \\ \hat{\mathbf{y}}_{j_{Chan}}(k \mid k) &= \omega \hat{\mathbf{y}}_{j_{Chan}}(k \mid k-1) + (1-\omega)\,\hat{\mathbf{y}}_i(k \mid k) \end{aligned}$$

Selecting ω to minimise the determinant of the resulting covariance yields $\omega = 0.0059$. In this case, $\mathbf{Y}_i(k \mid k)$ is clearly dominant as illustrated in Figure 2.10(d). The channel update then gives

$$\mathbf{Y}_{j_{Chan}}(k \mid k) = \begin{bmatrix} 0.3977 & -0.1996 \\ -0.1996 & 0.6028 \end{bmatrix} \quad \hat{\mathbf{y}}_{j_{Chan}}(k \mid k) = \begin{bmatrix} 3.5794 \\ 0.7945 \end{bmatrix}$$

The increment of new information can then be determined by subtracting the old channel estimate at node j from the newly updated one, which gives

$$\mathbf{I}_i(k) = \begin{bmatrix} 0.3811 & -0.0663 \\ -0.0663 & -0.4639 \end{bmatrix} \quad \mathbf{i}_i(k) = \begin{bmatrix} 3.4710 \\ -0.9278 \end{bmatrix}$$

This increment of new information is then added to the estimate at j and the process is

complete. The final result gives

$$\mathbf{Y}_i(k \mid k) = \begin{bmatrix} 0.4000 & -0.2000 \\ -0.2000 & 0.6000 \end{bmatrix} \quad \hat{\mathbf{y}}_i(k \mid k) = \begin{bmatrix} 3.6000 \\ -0.8000 \end{bmatrix}$$

$$\mathbf{Y}_{i_{Chan}}(k \mid k) = \begin{bmatrix} 0.4000 & -0.2000 \\ -0.2000 & 0.6000 \end{bmatrix} \quad \hat{\mathbf{y}}_{i_{Chan}}(k \mid k) = \begin{bmatrix} 3.6000 \\ -0.8000 \end{bmatrix}$$

$$\mathbf{Y}_j(k \mid k) = \begin{bmatrix} 0.6477 & -0.1996 \\ -0.1996 & 0.6028 \end{bmatrix} \quad \hat{\mathbf{y}}_j(k \mid k) = \begin{bmatrix} 6.2044 \\ -0.7945 \end{bmatrix}$$

$$\mathbf{Y}_{j_{Chan}}(k \mid k) = \begin{bmatrix} 0.3977 & -0.1996 \\ -0.1996 & 0.6028 \end{bmatrix} \quad \hat{\mathbf{y}}_{j_{Chan}}(k \mid k) = \begin{bmatrix} 3.5794 \\ 0.7945 \end{bmatrix}$$

The resulting determinate of $\mathbf{P}_j(k \mid k)$ *using this update method is* 2.8524, *compared with its initial determinate of* 3.7500. *Therefore, the covariance after the update is clearly more compact, due to the information communicated from node* i.

3.6.3 Broadcast With Hybrid CI/IF Update

One method of improving the conservative nature of the CI update in channels is to use a hybrid CI/full filter implementation. This aims to use the full information filter update on data that is known to be independent while using CI on information for which independence cannot be guaranteed.

When a node communicates, it is required to send two complete estimates in one message. The first estimate, which will be referred to as type 1 data, is the current estimate of common information at that node (ie the channel filter estimate). This estimate has some unknown correlation with the rest of the network as it contains information communicated previously. The second estimate, known as type 2 data, is the complete estimate at the current communication time. This estimate contains all information that was in the type 1 data, plus any new information that may have arrived from the locally attached sensor. This new sensor information is known to be independent from the rest of the network as it has not yet been communicated.

When a node receives a message from the network, it first does a CI update with the type 1 data. This update is exactly the same as that described in Section 3.6.2 and has the same conservative and sub-optimal properties. The second part of the update is to subtract the

type 1 from the type 2 data, and perform a standard information filter update with the result. As this information comes only from the sensor(s) attached locally to the node which transmitted the data, the information is independent and can be fused using the additive information filter update equations. When a node i communicates to node j, the complete channel update at j can be written as

$$\begin{aligned}\mathbf{Y}_{j_{Chan}}(k \mid k) &= [\omega \mathbf{Y}_{j_{Chan}}(k \mid k-1) + (1-\omega)\,\mathbf{Y}_{i_{Chan}}(k \mid k-1)] \\ &\quad + [\mathbf{Y}_i(k \mid k) - \mathbf{Y}_{i_{Chan}}(k \mid k-1)] \quad (3.62) \\ \hat{\mathbf{y}}_{j_{Chan}}(k \mid k) &= [\omega \hat{\mathbf{y}}_{j_{Chan}}(k \mid k-1) + (1-\omega)\,\hat{\mathbf{y}}_{i_{Chan}}(k \mid k-1)] \\ &\quad + [\hat{\mathbf{y}}_i(k \mid k) - \hat{\mathbf{y}}_{i_{Chan}}(k \mid k-1)]\,. \quad (3.63)\end{aligned}$$

This method is as scalable and as easy to implement as the broadcast with CI update of Section 3.6.2, but with the added benefit that it makes use of more information while still maintaining consistency. The fact that less information is thrown away serves to increase the accuracy of the estimates, and hence increase performance. The trade off is that this requires twice the communication bandwidth to communicate the two estimates, however, as this is a linear increase it does not void the scalability of the network.

Example 6 *Consider again the situation in Example 5 where the two nodes i and j need to communicate. Using this algorithm, node i now sends both the full current estimate $\mathbf{Y}_i(k \mid k)$ and $\hat{\mathbf{y}}_i(k \mid k)$ as well as the channel estimate $\mathbf{Y}_{i_{Chan}}(k \mid k-1)$ and $\hat{\mathbf{y}}_{i_{Chan}}(k \mid k-1)$. As this communication contains all information contained at node i, the channel filter at i is updated by*

$$\begin{aligned}\mathbf{Y}_{i_{Chan}}(k \mid k) &\leftarrow \mathbf{Y}_i(k \mid k) \\ \hat{\mathbf{y}}_{i_{Chan}}(k \mid k) &\leftarrow \hat{\mathbf{y}}_i(k \mid k).\end{aligned}$$

At the receiving node, the channel filter at node j now performs a CI update on the channel information from node i and its own existing channel estimate, then adds the difference between $\mathbf{Y}_i(k \mid k)$ and $\mathbf{Y}_{i_{Chan}}(k \mid k-1)$. This is done using Equations 3.62 and 3.63.

Selecting ω to minimise the determinate of the resulting covariance gives $\omega = 0.0061$. The increment of new information can now be determined by subtracting old channel estimate

at j from the new updated one.

$$\mathbf{I}_i(k) = \begin{bmatrix} 0.4816 & -0.0663 \\ -0.0663 & -0.4638 \end{bmatrix} \quad \mathbf{i}_i(k) = \begin{bmatrix} 4.5268 \\ -0.9276 \end{bmatrix}$$

This increment of new information is then added to the estimate at j and the process is complete. The final result gives

$$\begin{aligned} \mathbf{Y}_i(k \mid k) &= \begin{bmatrix} 0.4000 & -0.2000 \\ -0.2000 & 0.6000 \end{bmatrix} & \hat{\mathbf{y}}_i(k \mid k) &= \begin{bmatrix} 3.6000 \\ -0.8000 \end{bmatrix} \\ \mathbf{Y}_{i_{Chan}}(k \mid k) &= \begin{bmatrix} 0.4000 & -0.2000 \\ -0.2000 & 0.6000 \end{bmatrix} & \hat{\mathbf{y}}_{i_{Chan}}(k \mid k) &= \begin{bmatrix} 3.6000 \\ -0.8000 \end{bmatrix} \\ \mathbf{Y}_j(k \mid k) &= \begin{bmatrix} 0.7483 & -0.1996 \\ -0.1996 & 0.6028 \end{bmatrix} & \hat{\mathbf{y}}_j(k \mid k) &= \begin{bmatrix} 7.2601 \\ -0.7943 \end{bmatrix} \\ \mathbf{Y}_{j_{Chan}}(k \mid k) &= \begin{bmatrix} 0.3983 & -0.1996 \\ -0.1996 & 0.6028 \end{bmatrix} & \hat{\mathbf{y}}_{j_{Chan}}(k \mid k) &= \begin{bmatrix} 3.5851 \\ 0.7943 \end{bmatrix} \end{aligned}$$

The resulting determinate of $\mathbf{P}_j(k \mid k)$ using this update method is 2.4316, *compared with the initial determinate of* 3.7500 *and* 2.8524 *given by Example 5.*

3.6.4 Dynamic Tree Structure

While the standard tree structure for network connectivity makes the best use of communicated information, it is also very brittle to loss of communication links. Conversely, the CI broadcast network structure copes well with dynamic connectivity but at the cost of poor use of communicated information.

This section describes a network structure that uses a combination of both architectures. This network structure operates as a tree whenever possible but is able to dynamically reconfigure in the face of link failure. The key elements of this architecture are:

1. A hierarchical parent/child link structure which ensures that it is impossible to create a loop in any tree structure formed.

2. The use of CI to initialise a new channel filter rather than act as the nodal fusion algorithm.

A node is first defined with n links. Link 1 is defined to be a parent with the remaining $n-1$ links being children. The parent link can operate as a peer-to-peer or as a broadcast link, depending on the network configuration. The child links can only be used in peer-to-peer communications with other nodes. Figure 3.17 illustrates this node architecture.

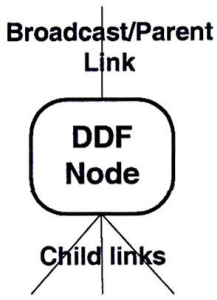


Figure 3.17: A single node in a dynamic tree configuration.

The parent link on a node can operate in either a broadcast mode or can be joined to a tree, but never both. When the nodes in the network form themselves into tree structures, they do so using a hierarchical link structure. The only way that nodes may join is by joining a parent and a child link. As each node can, by definition, only ever have one parent link, this ensures that any tree structure formed can only ever have one single node that sits at the top of the tree and has its parent link in broadcast mode. This node is referred to as the tree master. All other nodes in the tree must, by definition, have been attached to a node using their parent links and therefore do not receive broadcast information. This ensures that there is only one single point at which the tree interacts with other nodes or trees. In turn, this eliminates the possibility of double counting broadcast data.

This concept is illustrated in Figure 3.18 where a number of trees and an independent node are all communicating. Note that it is only the master node in each tree which broadcasts. Once the tree master has information, it can propagate this to all other nodes within the tree in an optimal manner. An important difference in the dynamic tree architecture is that nodes know the identity of both their immediate neighbours *and* the tree master node. No further local knowledge of the global topology is required however.

The parent link on each node maintains a channel filter regardless of whether it is in a broadcast or tree topology. This is necessary as the channel mode may change with changes in network configuration. However, all child links can only ever be used in a peer-to-peer

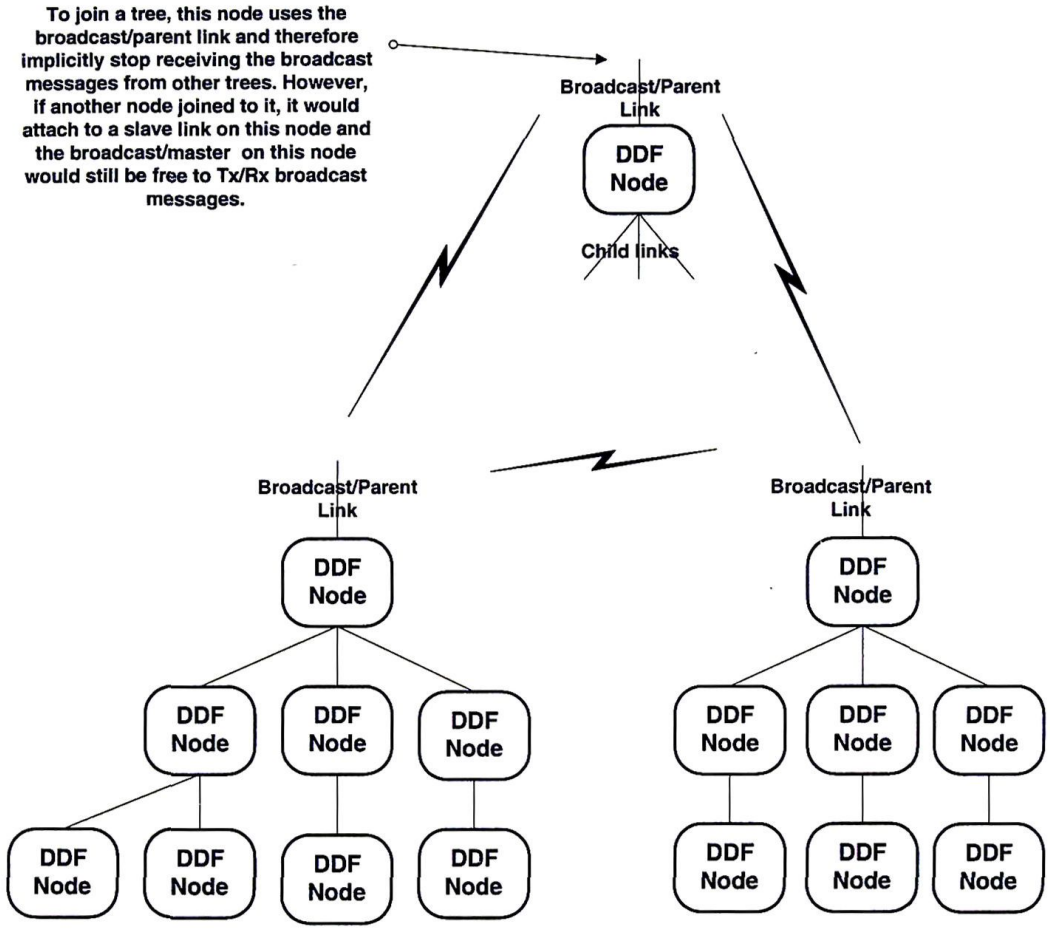


Figure 3.18: Structure of a dynamic tree network.

mode so a channel filter is only required when a connection is made.

When a parent link is not directly connected to the child on another node, it operates in a broadcast mode. The channel filter update in broadcast mode needs to be conservative as there is no knowledge of the source of any information. The channel filter could be implemented using the conservative algorithms described in either of Section 3.6.2 or Section 3.6.3 above, although the latter would be preferable as it makes better use of the information. When configured in a tree structure, channel filters are implemented to make use of all information.

To join a tree structure, a node broadcasts a request to attach to another node. Any node with a spare child link replies and they negotiate joining. When joining a tree it is important

to ensure consistency, no double counting, between estimates in the joining nodes. The node joining the tree must send a broadcast information message to the attaching node. The child link on the attaching node must be updated with this estimate using the broadcast update algorithm. The attaching node then sends this update back down the channel and the new node need only overwrite its previous channel estimates with this new information. This new information is guaranteed to be at least as good as its information prior to connection.

Multiple trees can be joined in the same way as an independent node joins a tree. The node at the top of the tree uses its parent link to negotiate joining a child link on another tree.

As an example of this in operation, consider the simple case of two independent nodes, A and B, wishing to join together:

1. Node A broadcasts a Request To Join (RTJ) another node.
2. Node B receives the RTJ message. It has spare child links so it sends an Available To Join (ATJ) message to node A.
3. Node A receives the ATJ message from node B and from any other node in the vicinity that may also be responding to the original RTJ. Node A selects the best response (node B) and commits to a joining by transmitting a Confirm Selected Link (CSL) message.
4. Node B receives the CSL message and must ensure that the request to join does not corrupt the integrity of the tree. It must pass a message to the tree master with the identity of the node that wishes to join the tree. When ready, node B transmits a Confirm Link (CL) message. Part of the acceptance message is the ID of the trees master node, in this case Node B.
5. Node A receives the CL message and joins the tree. The parent link on Node A is now in peer-to-peer mode and does not listen to any broadcast messages. In order to align the channel filters, Node A sends its usual broadcast message structure through the point to point link to node B. Node B uses this information with its own state and performs a CI update. The result of this CI update is then written to the channel filter and sent back to Node A down the same peer-to-peer link.
6. Node A receives the information from Node B and uses this to update its own channel filter. The channel filter between the nodes is now initialised with a conservative

estimate of the common information and the two nodes can communicate from this point onward in a tree structure.

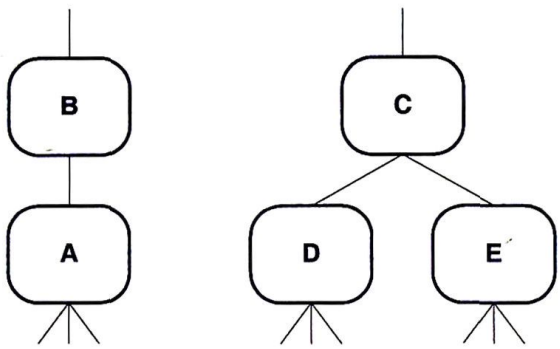


Figure 3.19: Joining two trees together.

This simple example can now be extended to a more complex situation where two trees are joined together. Consider the scenario in Figure 3.19. The two nodes A and B have now formed into a tree of two and would like to merge with another tree. When node B broadcasts its request to join, it will be received by nodes A, C, D and E. Node A will reject joining node B again as node A knows the identity of the tree master. In the example shown, node C has two child links and they are taken by nodes D and E. It cannot accept any new connections so it ignores the RTJ message. The two remaining nodes, D and E, both have available child links and so will respond with ATJ messages.

Node B will receive the ATJ messages from both D and E and would select whichever is best or which arrived first. Assuming the message from node D was the winner, B would reply to D with a confirm selected link (CSL) message.

Node D receives the CSL message from B and must now pass a message to the top of the tree to confirm that making this link will not violate the integrity of the tree. This is necessary in order to prevent a race condition where B is joining to D at the same time that C is negotiating to join to A. If this were to happen without any safeguards, the result would be a tree looped back on itself.

Node D now sends a message up through its parent link (in peer-to-peer mode) to its parent in the tree. The content of this message is simply that it is attempting to join with another node. If the tree master is currently in the process of negotiating to join another tree, it would deny the request to allow another node to join to its own tree. If it not negotiating,

it can immediately accept a new node and pass a message back through the tree to node D. An interesting side issue here is that with a simple counter on this message increments as nodes relay it down the tree, it is possible for nodes to know their position in the tree relative to the top.

If any link in a tree connection is broken, then two trees are formed. However, as soon as the link is broken the node lower in the tree has its parent link freed again and can use this in broadcast mode or negotiate to join to another tree. Importantly, the channel filter on the broadcast link must not be reset. This is because if a a tree splits then manages to rejoin there will clearly be information in common up to the time of the split. The channel filter at the child link can be reset and cleared safely.

While having knowledge of the identity of node at the top of the tree is new, it does not violate any of the criteria for a decentralised system. It is not necessary to know all nodes on the path to the top, merely the top itself. Another point of note is that it is easy and useful to implement a system for counting the number of nodes on the path to the top as this would enable a network manager to limit the size of any single tree, should that be required.

3.7 Summary

This Chapter has developed the algorithmic structure of a decentralised node. It has addressed the issues of common information between nodes and presented algorithms capable of robustly handling problems of timing and delay. Using these, an architecture for a generic decentralised node was described. The architecture allows each node in the system to run using exactly the same software, resulting in true "plug and play" operation.

The modules in this node architecture are general to any decentralised network configuration. They can be mapped to specific network structures by modification of the channel filter update method. The node architecture is able to run completely asynchronously, and is able to handle communication delays, asequent data transmission and burst communication. Algorithms were also presented which allow for operation in dynamic networks.

Chapter 4 describes the practical realisation of this architecture.

Chapter 4

ANSER Project: Implementation of a Decentralised System

4.1 Introduction

This Chapter describes the systems and hardware used in the ANSER project to demonstrate the DDF architectures and algorithms developed in this thesis.

Section 4.2 begins by outlining the ANSER project. This includes an overview of the scope of the project and its aims. Brief details of the Brumby aircraft are then presented in Section 4.2.1. This details the evolution of the airframe through the three different generations and gives a short account of its performance specifications.

Section 4.3 gives an overview of the main sensing equipment used on the ANSER project. While there are many sensors and actuators required to operate the vehicles, only the primary sensors used for the decentralised algorithms are described in this thesis. This includes GPS, inertial, vision and radar sensors in Sections 4.3.1, 4.3.2, 4.3.3 and 4.3.5 respectively. An overview of the processing and communications hardware is also given in Section 4.3.6

Section 4.4 describes how the decentralised architecture presented in Chapter 3 is mapped to the physical hardware resources on the ANSER project. This includes details of where the algorithms are physically implemented and which resources they use. A description of how the communication resources are used is given in Section 4.4.1.

Section 4.5 details the ANSER simulator. The simulator is a complex distributed software tool that was designed to develop and test the decentralised algorithms prior to the flight trials, and to play back and evaluate the flight data for post flight analysis. The software is completely modular and designed to match the real system as closely as possible.

4.2 The ANSER Project

The Autonomous Navigation and Sensing Experimental Research (ANSER) project is a large scale collaborative research project between the University of Sydney and BAE Systems. The goal of the project is to develop decentralised data fusion algorithms and demonstrate these in real time aboard uninhabited air vehicles. Due to the scope of the project, a large number of people across a variety of disciplines are involved. The contribution of this thesis to the project is the development of the decentralised algorithms for tracking and feature based navigation. However, in order to successfully demonstrate these algorithms in an airborne environment it is important to have a team capable of manufacturing and maintaining both the airframes and other specialised equipment.

The system which was used consists of four uninhabited flight vehicles of the type shown in Figure 4.1. Each flight vehicle is equipped with GPS and inertial sensors and carries two terrain payloads; a vision system and either a mm-wave radar or laser sensor. Each payload incorporates it's own modular, fully decentralised processing hardware. On-board, the payloads communicate with each other using a CAN bus. Inter-vehicle communication is via radio ethernet. Each payload processor implements a fully decentralised data fusion algorithm. Payloads communicate with each other directly in terms of terrain information; all data fusion and assimilation occurs at the payload site. There is no separate fusion centre on any flight platform and no fusion centre elsewhere on the ground. The architecture is thus decentralised, fully modular and scalable. Further details of the system are given in the following sections.

4.2.1 The Flight Vehicle

The Brumby airframe was designed and built at the University of Sydney in collaboration with BAE Systems as a low cost, rapid development uninhabited aircraft. The airframe is

Figure 4.1: The four Brumby Mk.III airframes.

constructed from sandwiched nomex honeycomb and fibreglass. It has a tubular fuselage with a large volume in order to accommodate multiple sensing payloads. A pusher (rear mounted) engine design was used to allow a payload of sensing equipment to be installed in the nose without any forward obstructions. In total, three versions of the Brumby airframe have been developed. All are tricycle undercarriage, delta wing airframes capable of flying at approximately 100 knots (180km/hr).

Figure 4.2: The original Brumby Mk.I airframe.

The original Brumby Mk.I airframe, shown in Figure 4.2, used a 74cc engine with a two blade propeller and was capable of flying for 20 minutes with a 5kg payload. This airframe was used for the initial vision data logging flights.

In order to increase the maximum payload capacity, the Brumby Mk.II was designed. This used the same fuselage as the Mk.I, but had a larger wing area and a more powerful 80cc engine which could deliver approximately 6hp. This aircraft was also used with a vision system but was unable to accommodate the extra weight and volume of a terrain scanning sensor such as the radar.

The current aircraft design is the Brumby Mk.III shown in Figure 4.3. This aircraft was

Model	Engine	Max Speed	Payload	Endurance
Mk.I	74cc/5hp	100kts (180km/hr)	5kg	20min
Mk.II	80cc/6hp	100kts (180km/hr)	14kg	20min
Mk.III	150cc/16hp	100kts (180km/hr)	20kg	45min

Table 4.1: Brumby performance characteristics.

designed to fill the demonstration requirements of the ANSER project. The most noticeable differences between this and earlier designs are addition of canards and the large bulbous nose. The large nose was required in order to accommodate the scanner for the radar and laser sensors, while the canards improve the low speed handling characteristics of the aircraft. In addition to these modifications, a much more powerful 150cc engine was used with a four blade propeller which increases the power over the Mk.II by approximately 10hp. In its ANSER configuration, the Brumby Mk.III has an endurance in the order of 45 minutes with a 20kg payload. These parameters can be modified by the addition of extra fuel tanks if required.

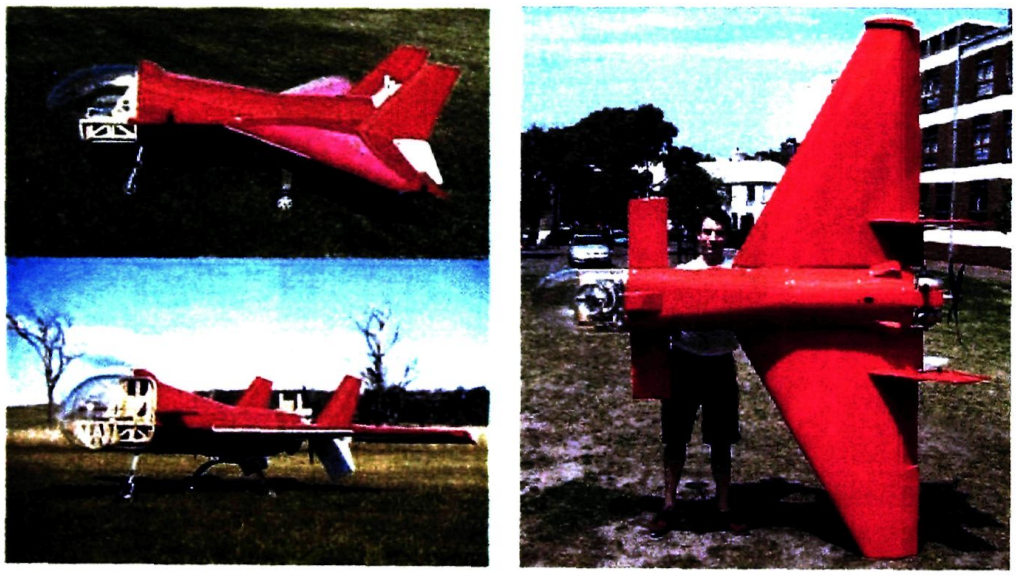


Figure 4.3: (Left) The Brumby Mk.III airframe on the runway at the flight test facility. (Right) Although it has a 2.9m wingspan, the Brumby Mk.III often looks deceptively small. Mr Jeremy Randle, the UAV engineer responsible for the design, construction and maintenance of the aircraft, gives some perspective to its true size.

4.3 Sensing Equipment and Hardware

A variety of sensing hardware is required on each UAV in the ANSER system. These include sensors for measuring quantities such as temperature, RPM and battery voltage, as well as those for measuring the vehicle position and for the detection and tracking of external features. While all are required for the UAV to function correctly, the discussion in this section is restricted to only those sensors used directly in the implementation of the DDF algorithms.

4.3.1 GPS

Each Brumby Mk.III airframe has two CMC Allstar DGPS receivers on board for localisation. The antenna for these receivers are built into the top surface of each wing during construction. Figure 4.4, taken during the assembly of a wing, shows the location of the antenna. The base station for the differential GPS solution is a permanent fixture at the University flight test facility.

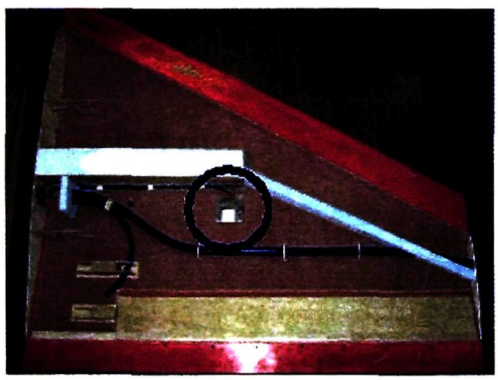


Figure 4.4: A GPS antenna is embedded in the top of each wing under a layer of carbon fibre. The black circle highlights the location of the antenna as the wing is being constructed.

4.3.2 Inertial System

Each aircraft is equipped with an Inertial Science ISIS inertial measurement unit, shown in Figure 4.5. These six-degree of freedom systems are ideally suited for application in uninhabited air vehicles due to their small size, light weight and low power consumption.

The gyro and accelerometer ranges used on these units are given in Table 4.2. A complete listing of the specifications for this IMU are included in the Technical Manual [52].

Axis	Gyro Range	Accelerometer Range
X	$+/-500^o/s$	$+/-2G$
Y	$+/-200^o/s$	$+/-2G$
Z	$+/-200^o/s$	$+/-10G$

Table 4.2: ISIS IMU specifications.

When operating on the aircraft, the ISIS IMU outputs the acceleration and rotation rates at the maximum rate of $400Hz$ using the built in RS422 serial link.

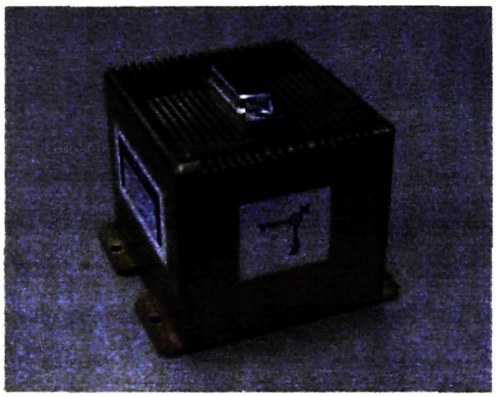


Figure 4.5: An Inertial Science ISIS IMU, measuring approximately $71mm$ x $76mm$ x $59mm$.

4.3.3 Camera

A Sony CCS-SONY-HR camera, shown in Figure 4.6, is used for the vision system as it is lightweight, inexpensive and readily available. This CCD imaging sensor has a resolution of 600 horizontal lines and runs from a 12V power source. It has a composite video output which gives images at up to 50Hz, or 25Hz when the images are interlaced for viewing. This occurs as the odd and even lines are used to form separate images 20ms apart.

The vision sensor is mounted in the second payload bay of the Brumby Mk.III, immediately behind the forward bulkhead. This places the camera slightly aft of the nosewheel on the centre axis of the airframe. The sensor is mounted pointing directly down. Typical airborne images from this sensor are illustrated in Figure 4.7.

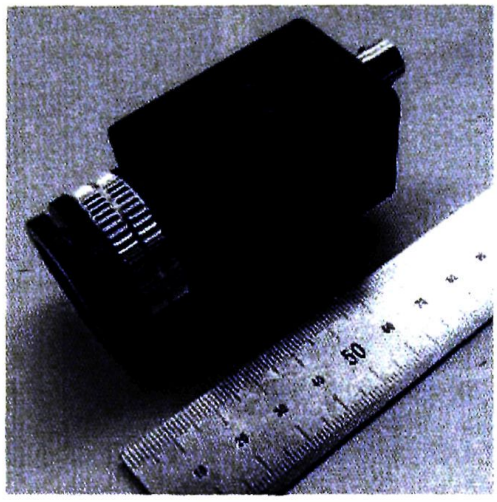


Figure 4.6: A Sony CCS-SONY-HR vision sensor used on the Brumby Mk.III airframe.

The camera is used to detect and track artificial targets on the ground. A simple intensity threshold algorithm is used in real time to extract these targets from the images. The bearing to the target can then be generated. Although the vision sensor does not give range directly, an estimated value is generated based on the size of the targets. A more accurate method based on the height of the vehicle was also demonstrated as the terrain at the test facility is relatively flat.

4.3.4 SIS: Vision/Laser System

The secondary information sensor (SIS) is a virtual range/bearing sensor obtained by packaging a Sony CCS-SONY-HR camera (described in Section 4.3.3) with a laser ranging device. It can be mounted in the Brumby Mk.III nose as a terrain sensing payload option.

The ranging sensor used in the SIS is a Measurement Devises Ltd ILM300HR laser. It uses a GaAs laser diode to provide range measurements over distances from 5m to 300m at a frequency of 1000Hz. The wavelength of the laser is 905nm, and range accuracy is in the order of 30cm [65].

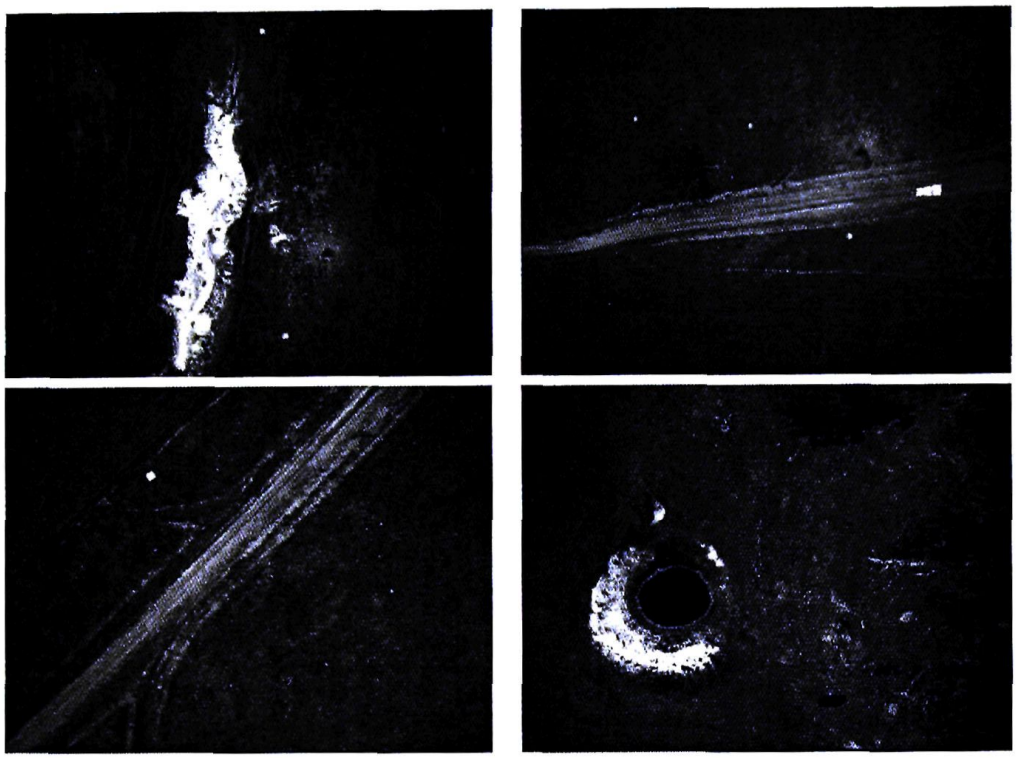


Figure 4.7: Typical images from the camera payload. The white $0.9m$ x $0.9m$ camera targets are visible in the upper and lower left frames. The top right frame shows a white Toyota Landcruiser 4 wheel drive in the image which was used to test performance with moving targets. The bottom right frame is taken over one of the dams on the test facility which could easily be used as a natural target.

4.3.5 Radar

A millimeter wave radar and scanner are housed in the bulbous Brumby Mk.III nose. This configuration is shown in Figure 4.8. The radar was designed at the University in order to meet the stringent size, weight and power consumption limitations of the aircraft. However, at the time of writing this thesis the radar is still undergoing calibration trials it is therefore not used in any of the DDF experiments.

The frequency modulated continuous wave (FMCW) radar front end transmits a linear chirp signal at 77GHz which is deflected to the ground using a gimballed mirror scanner. This scanner rotates at constant speed and directs the radar beam at a mean grazing angle of 18^o from the horizontal which results in a constant width annular swathe being scanned as the aircraft flies. As the aircraft is constantly pitching and rolling, the system is gimballed

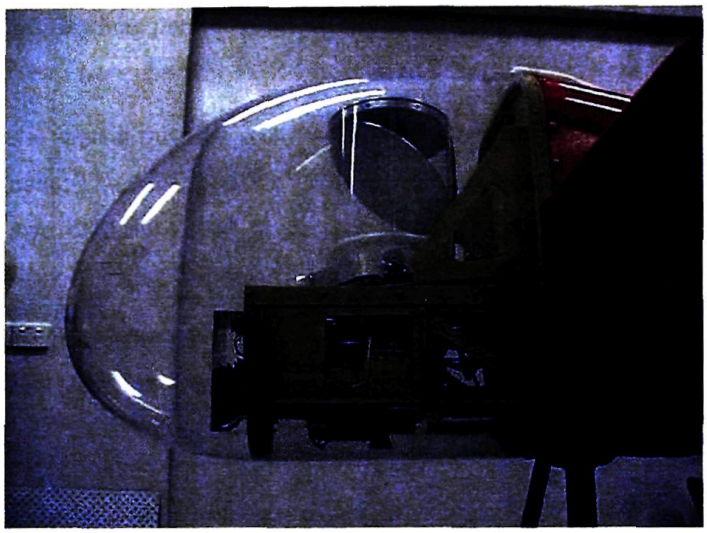


Figure 4.8: Millimetre wave radar and scanner fitted to the Brumby Mk.III nose.

to stabilise it and ensure the radar footprint is constant. Due to volume constraints within the Brumby nose, the gimbal is only able to operate to $+/-15^o$ in the pitch and roll axes. Complete technical documentation on the radar design and performance is contained in [18].

The radar is included as a payload in the ANSER system to give accurate range/bearing information for the detection and tracking of artificial ground targets. These radar targets are to be co-located with the vision targets to allow the decentralised algorithms to detect and track the same objects using multiple sensors on multiple aircraft.

4.3.6 Processing and Communications

All onboard DDF computing for the payloads is done using commercial off the shelf PC104 hardware. The PC104 stacks include modules for power, processing, CAN and PCMCIA cards. The processing modules are a combination of Pentium II 266MHz and Pentium III 700MHz CPUs. A two slot PCMCIA module is required to house a flash disk for data logging and wireless ethernet card for inter-vehicle communications.

There are also some modules which are specific to the particular sensors used. The vision node incorporates a frame grabber for the capture of camera images. Once captured,

the processing module then performs all of the feature extraction and target estimation algorithms.

The radar node uses special purpose ADC[1] and DSP[2] boards for interfacing and processing the raw radar signals. The ADC had 8 x 10MHz channels with a resolution of 12 bits. The DSP board uses two SHARC 21060 processors for the front end signal processing. This hardware is necessary in order to perform the fast fourier transforms (FFTs) in real time.

All communication within the vehicles is done using a dual CAN bus system, as illustrated in Figure 4.9. One CAN bus is used for DDF data and the other for vehicle control and management tasks. The CAN controllers were configured to use the CAN 2.0B specifications.

For inter-vehicle communications, the wireless ethernet system is used. This system has a line of sight range in the order of 4km. Although capable of a maximum bandwidth of 11Mbps, the system is manually configured to run at 2Mbps for greater robustness.

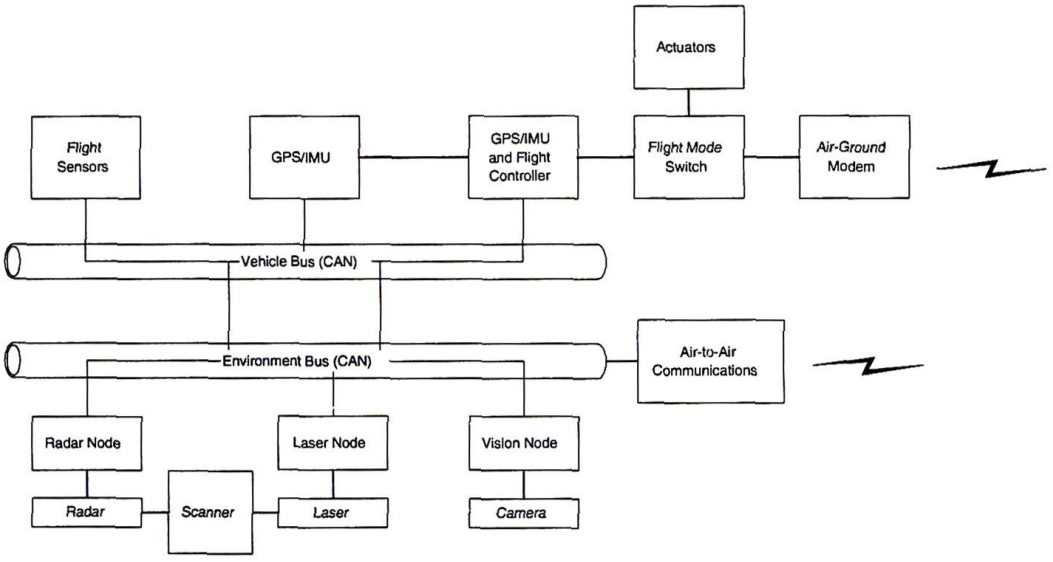


Figure 4.9: Functional diagram of the systems on the Brumby platform.

Figure 4.9 illustrates how these processing and communication systems on the Brumby are configured. The sensors and their attached processing modules are all connected to the 'environment bus' which is used to communicate DDF data. Note that only one of either

[1]Analogue to Digital Converter
[2]Digital Signal Processor

the radar or laser can be installed in the aircraft at any one time, as they would both occupy the same payload bay and use the scanner. The vehicle specific components such as the GPS/IMU localisation filter and the flight controller are all connected to the 'vehicle bus'. The commands from the pilot when being flown remotely are all received through a spread spectrum wireless modem which is used for this purpose alone.

4.4 Implementation of the Decentralised Architecture

Each aircraft is capable of carrying up to two payload sensors, each of which has its own PC104 processor and communication interface. This sensor and processing combination is grouped together to form a node in the decentralised system.

The decentralised architecture on the ANSER project is implemented as a tree connected network. The reason for selecting this configuration over a fully connected broadcast topology is that only one node per aircraft has a radio ethernet card, which means that broadcast messages do not actually reach every node. While a 'virtual' broadcast network could, in principle, be constructed by forwarding messages from the radio ethernet onto the CAN bus, this makes the node with the forwarding function critical to the operation of the system. The tree configuration does not have this problem and it is a simple matter to create virtual peer-to-peer connections using this hardware. Figure 4.10 illustrates the structure of the intra and inter-vehicle communications mediums.

4.4.1 Communications

While the tree connected communication topology requires peer-to-peer communications, both the CAN bus and radio ethernet are broadcast. Therefore, the peer-to-peer architecture is developed in software using virtual channels to connect nodes. Figure 4.11 illustrates this concept of virtual channels. The communications software can establish a virtual channel between any processes (applications) regardless of which CPU these are running on.

Within each aircraft, nodes communicate over the CAN bus. For external communications between platforms, at least one of the nodes on any given aircraft is equipped with a wireless ethernet card.

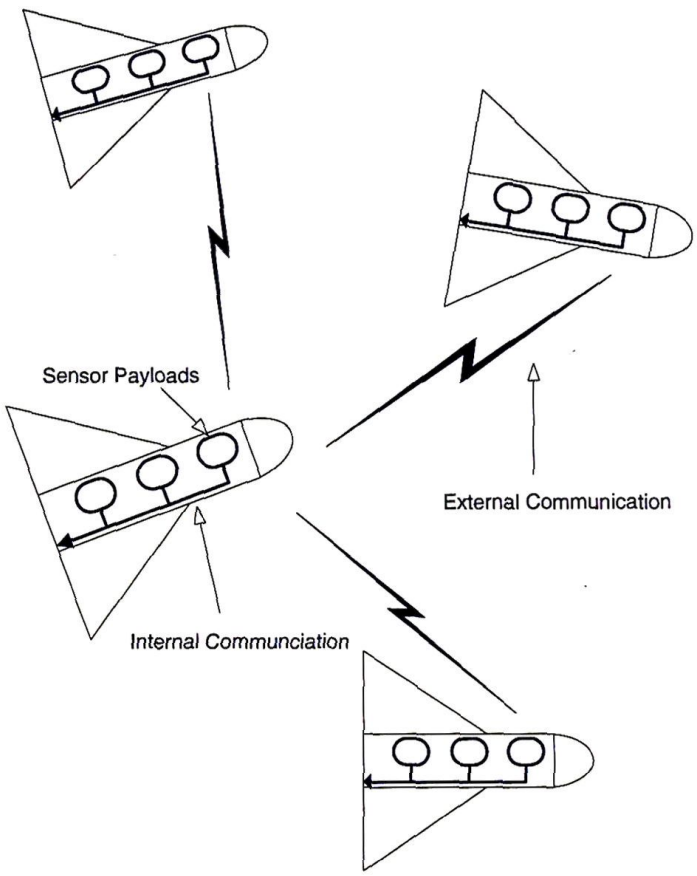


Figure 4.10: The ANSER system is configured into a tree connected communications topology. Communication between nodes on the same vehicle is done using the CAN bus. As only one node in each airframe has a radio ethernet card, it forms the communication links to other vehicles.

When a node transmits information on either the CAN bus or the wireless ethernet, only those nodes with which it is directly connected in the tree architecture will receive the message. This occurs as the communication layer uses a messaging system that creates a virtual point-to-point connection over the broadcast medium.

4.4.2 The Decentralised Filter

The decentralised information filter is implemented as a multi-threaded application, written in C++. The filter is constructed using the modular node architecture presented in

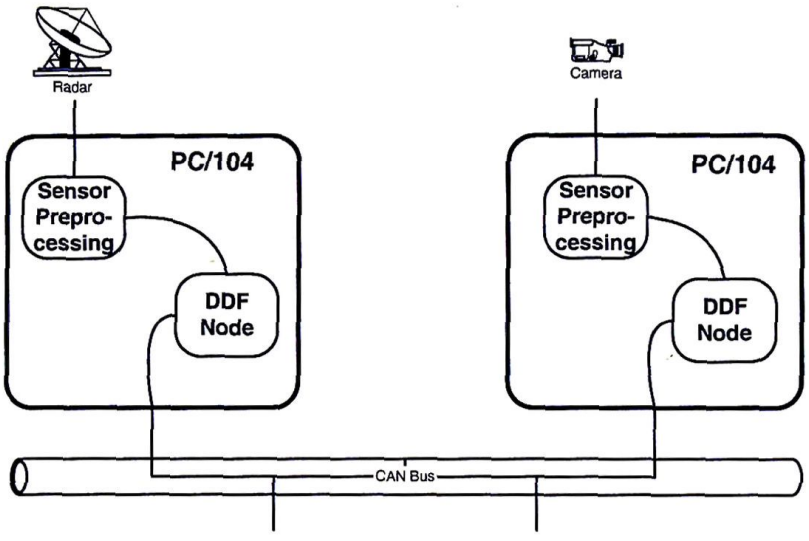


Figure 4.11: A virtual tree connection on a bus architecture. The DDF nodes are connected directly over the bus architecture using a virtual channel. Similarly, the sensor preprocessing is a separate process (application) than the DDF filter and is connected to it using another virtual channel.

Section 3.5 to ensure that each filtering node runs exactly the same software, regardless of the type of sensor it is attached to. All sensor specific calculations are performed in the sensor preprocessing stage, which is implemented as a separate process to the filter. Using this architecture, the sensor preprocessing outputs the observations to the filter in their information form and the filter needs no knowledge of the sensor itself. Figure 4.11 illustrates this connectivity between the preprocessing and the filter with the two applications running on the same PC104 module.

The internal software structure of the decentralised filtering node is shown in Figure 4.12. All communication to and from the node is handled using the CommLib software package which was developed in-house by Ali Göktoğan. The purpose of this is to allow the low level communication interfaces to be abstracted from the filter itself.

The local filter runs a prediciton/update cycle that fuses information from the locally attached sensor as it arrives, while only updating with channel information at the synchronous time horizon. After the channel update, the node transmits its own new information to its neighbours.

Each node in the system is configured in software to have a maximum of four channels.

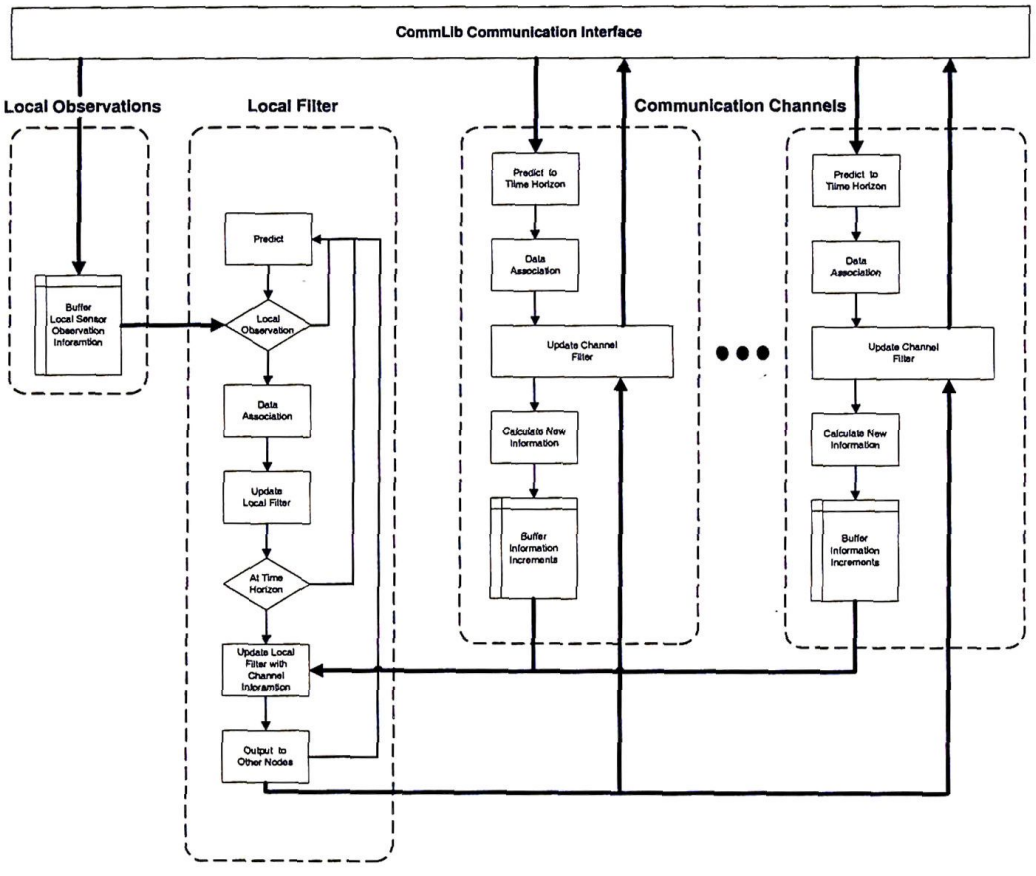


Figure 4.12: The software structure of a decentralised node. The CommLib communications software handles interfacing to both the CAN bus and the radio ethernet. Observations from the locally attached sensor are temporarily buffered and processed by the local filter as they arrive. Channel data is processed as it arrives and buffered until the next time horizon when it it fused in the local filter.

Although this can be changed, it is set at four channels as this allows nodes to have a reasonably well branched tree without any node having so many channels as to cause a computational bottleneck.

4.5 ANSER Simulator

The ANSER simulator is a software tool for developing and testing code for a real-time airborne demonstration of multi-vehicle decentralised data fusion. While standard off the shelf mathematical packages are capable of testing the validity of the algorithms, they were

not able to realistically test such practical issues as the asynchronous nature of the nodes. The ANSER simulator was therefore developed to run distributed modules on different computers in order to match the actual system as closely as possible [40].

Figure 4.13 shows the structure of the simulator and the software modules. Each block is a separate module which may be run on any computer connected on the network. The oval components are changeable plug-in modules containing models of the hardware components, and are designed to interface with a particular simulator block. It is a relatively simple matter to add a different sensor payload to the system or indeed to change the aircraft type by changing these plug-in modules.

4.5.1 SimCompiler

The simulator is run using a configuration script which is interpreted by the SimCompiler. The configuration script is used to define the number and type of each platform in the simulation, their associated waypoints, the location and type of all ground targets and any other simulation features. It is even possible to modify the length and alignment of the runway and control tower by changing a few simple parameters.

The use of the SimCompiler allows the simulator to be rapidly configured for virtually any environment. With a library of previous scripts, the development time for new simulations is extremely short.

4.5.2 Flight Simulator

The flight simulator is the module which actually simulates the aircraft motion. This is done through the use of the aircraft dynamic model provided, or alternatively, by playing back the results of actual flights. The output of the module is the pose of the platform(s), which is sent to other modules such as the sensor server to allow them to function.

The flight simulator uses a graphical display of the simulator world to convey vehicle pose information to the user. Figure 4.14 shows a typical view of the GUI when zoomed in on the aircraft.

When the aircraft motion is being simulated the control actions are generated by either the simulator control module or by inputs direct from a joystick.

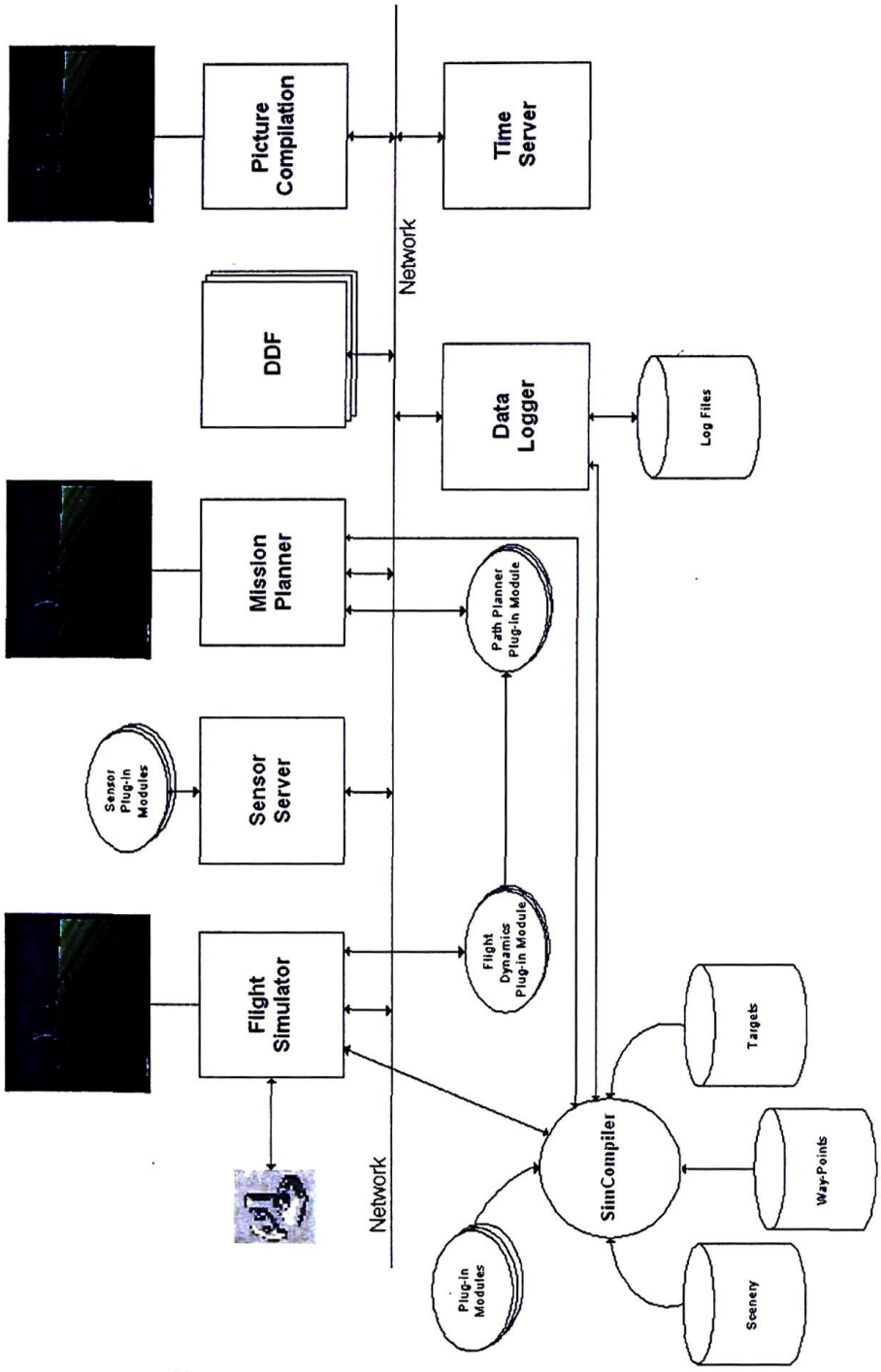


Figure 4.13: Structure of the ANSER simulator. Each box corresponds to a simulator module which may be run on any computer. The modules that include a graphical display are shown with the diagram of an airfield. The oval components are plug-in modules which contain models of components in the ANSER system. These include models for specific sensors, as well as for the vehicle itself. The cylindrical components represent information stored in files.

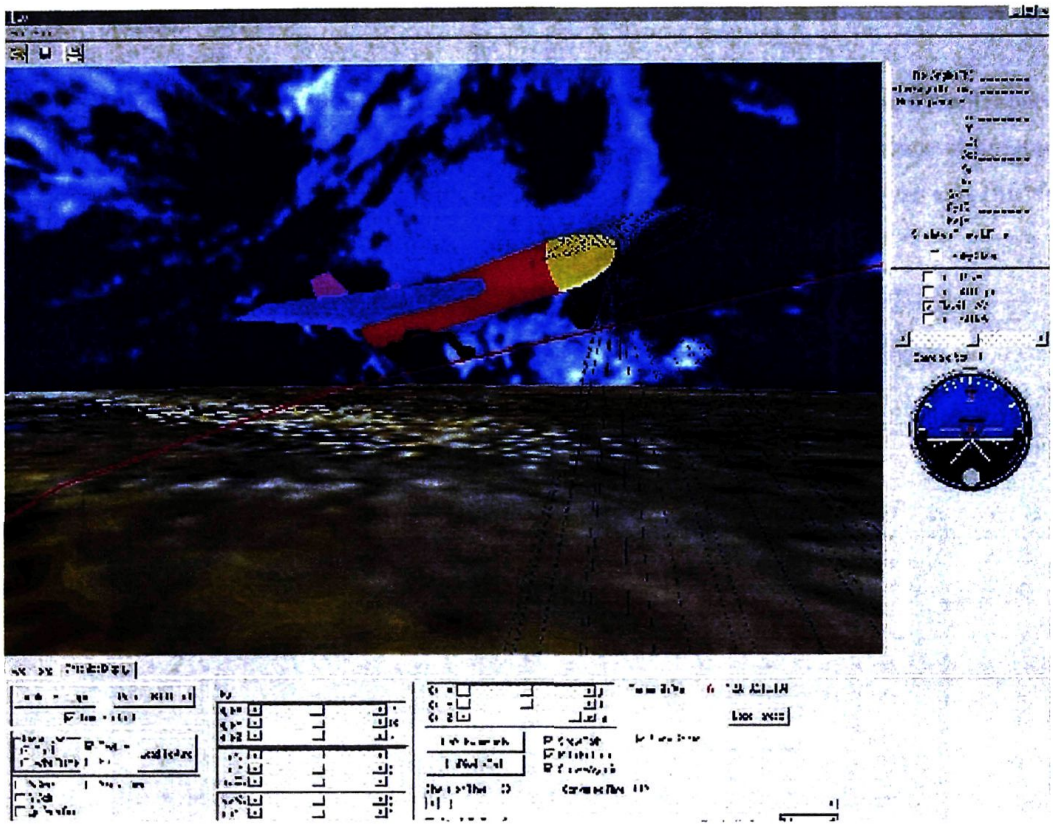


Figure 4.14: A typical screenshot from the flight simulator GUI. The aircraft model used in this version is the Brumby Mk.I, identifiable by lack of canards and regular nose. The solid red line beneath the aircraft is the flightpath the platform is following. The transparent cone extending down from the nose of the vehicle shows the field of view for the onboard sensor payload, in this case a camera.

To use the flight simulator in playback mode, the output of the GPS/IMU localisation filter is input to the flight module via a logfile. The simulator then reads the logfile and uses this as the actual platform pose.

4.5.3 Mission Planner

The mission planning module generates platforms trajectories for the simulator. It is capable of planning missions for multiple platforms anywhere in the simulation world. When operating the simulator in playback mode with logged vehicle pose data, the mission planner just uses the recorded GPS/IMU flightpath.

4.5.4 Sensor Server

The sensor server, illustrated in Figure 4.15, outputs sensor observations to the decentralised estimator. It can do this by using either a plug-in sensor model to generate simulated observations, or by outputting real flight data from a logfile. The sensor server can run up to five different sensors on each of four platforms.

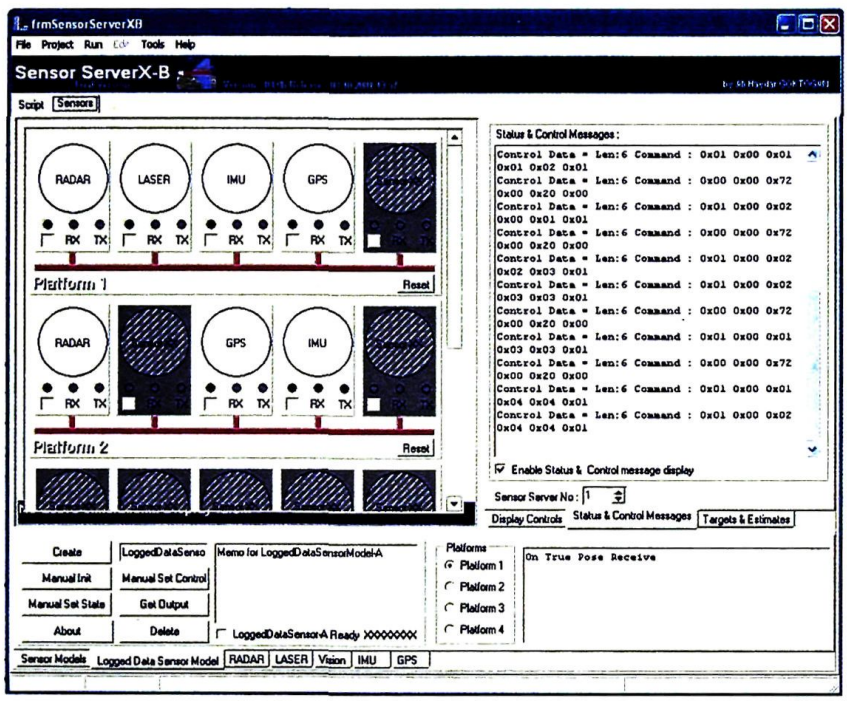


Figure 4.15: The sensor server outputs observations to the DDF module. It is capable of using either a software model of a sensor to simulate observations or of reading logged flight sensor data from file.

The simulation mode of the sensor server uses a plug-in sensor model to determine how a sensor interacts with the world. This model is some parameterised description of the sensor and its capabilities. The sensor server is designed with a fixed interface so different plug-in modules can be easily inserted into the simulator.

When in playback mode the sensor server reads the logged observation data from file and uses the datas timestamp to determine when to output information. This mode enables algorithms to be tested as rigorously as possible on real data before real-time implementation.

4.5.5 DDF

The decentralised estimators are implemented in the DDF module. This module receives the current time from the time server and observations from the sensor server. It then uses the observation information for the decentralised estimation algorithm.

There can be any number of DDF modules in any simulation, all of which communicate in a decentralised tree structure. The connectivity of the tree structure is defined by the user each time the DDF component is run. Output from the DDF module can be sent to the Picture Compilation module for display on a GUI. This output is solely for visual purposes and does not in any way affect the operation of the decentralised estimator.

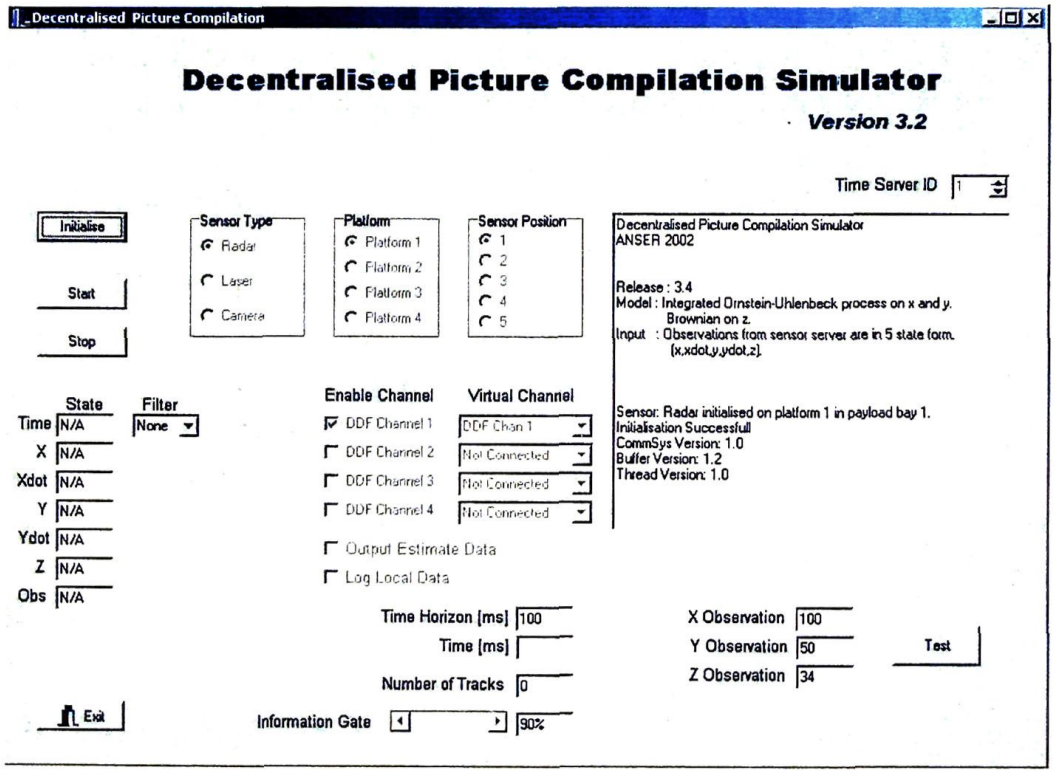


Figure 4.16: The decentralised tracking simulator. The selection boxes for which type of sensor to simulate and where the sensor is located are visible at the top of the image. The check boxes enabling DDF channels and defining their connectivity are in the middle of the picture and the timing information and data association gate are at the bottom. The text boxes on the left hand side enable the user to view the current estimate of any targets being tracked.

Figure 4.16 shows the decentralised tracking simulator. Multiple copies of this program

are run on a number of different machines simultaneously in order to simulate a real-time implementation using different computers in different aircraft. This configuration enables algorithms to be tested under conditions where nodes are completely asynchronous and operating with unknown communications latencies.

4.5.6 Picture Compilation

The picture compilation module is a graphical display for the decentralised estimator. It receives the position and 2σ variance of each target or feature being tracked on the ground and displays the result so the user can verify the correct operation of the algorithms. When operating the simulator in playback mode with logged data, it is possible to overlay the raw output from the vision sensor to give a more meaningful picture of the tracking task. Figure 4.17 shows a typical image of the display during playback mode with real data.

4.5.7 Time Server

As the simulator is run on multiple computers distributed throughout the network it is necessary generate a common time reference on all machines for the algorithms to function correctly. To do this, a single time server is run on one machine in the network and it outputs a regular count to all other modules. This count is then used as a global time source by the entire simulator. The speed of the count can be controlled to speed up or slow down the simulation time. It is important to note that although the simulator is used to test algorithms in the presence of communication delays and the like, it does not run in real-time.

4.6 Summary

This Chapter has presented a description of the equipment used in the ANSER system to demonstrate decentralised estimation. An overview of the flight vehicle and the sensor packages it carries is given. The processing and communication hardware is also presented and discussed in the context of a decentralised node. The decentralised tree connected network presented in Chapter 3 is mapped to the physical system on the aircraft.

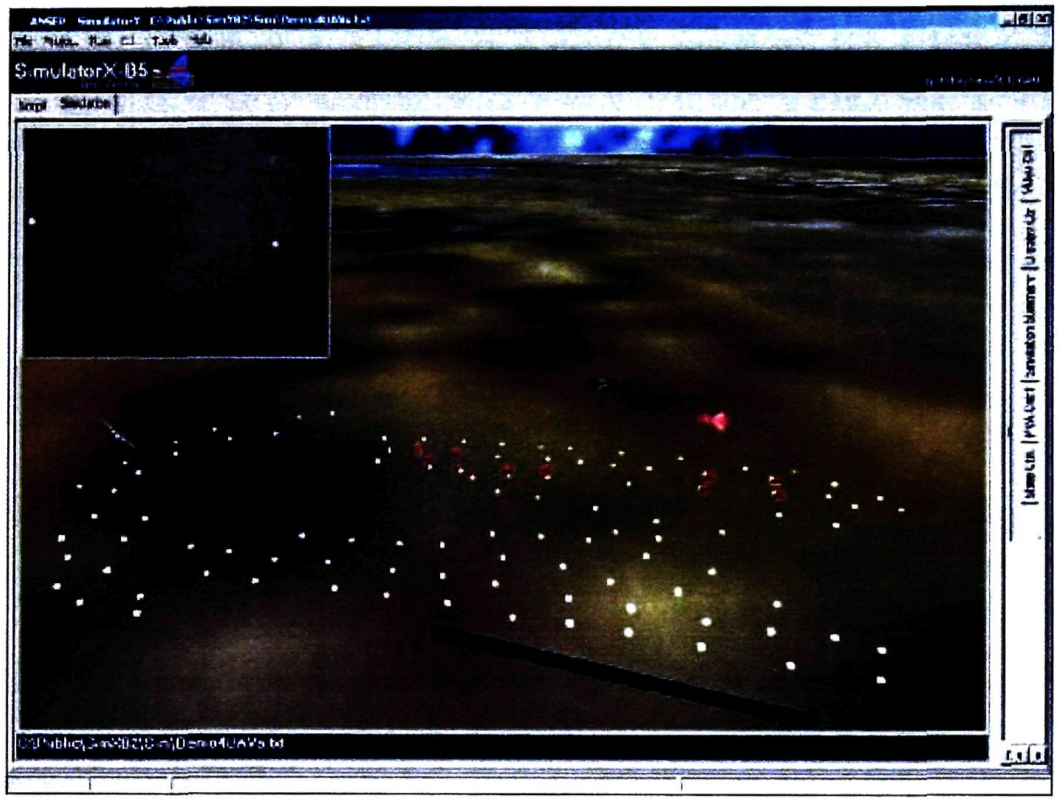


Figure 4.17: The ANSER simulator graphical display in playback mode. Note the real vision frames in the top left corner, as viewed from the red aircraft. The estimates are plotted as red spheres and the 2σ ellipse as a yellow sphere.

An overview of the ANSER simulator is given. This software is designed to develop and test the decentralised algorithms in a realistic scenario and validate the system using real flight data.

Chapters 5 and 6 develop specific decentralised tracking and navigation problems and present results of implementing the algorithms in real-time on this airborne multi-vehicle system.

Chapter 5

Decentralised Tracking

5.1 Introduction

This Chapter presents the theory and results of applying the DDF algorithms to the problem of tracking multiple ground targets from the air. This problem is often referred to as 'picture compilation' as collectively, the tracked targets form a picture or map of a region. Section 5.2 presents and discusses the problem for both single and multiple aircraft scenarios.

The implementation in this Chapter uses a combination of both the passive vision sensor and the SIS described in Sections 4.3.3 and 4.3.4. The specific sensor model and feature equations are given in Sections 5.3.1 and 5.3.2 respectively. These algorithms are used to extract point features from an image prior to calculating the observation information.

Section 5.5 describes the decentralised tracking algorithm. This covers the models used for the track motion, the equations used in the sensor preprocessing to calculate the observation information, a description of the physical targets used, the track maintenance method, data association and a description of the communication strategy between neighbouring nodes.

Section 5.6 presents results of implementing the decentralised tracking algorithm. A description of the DDF tracking system implemented in the ANSER simulator with logged flight data is presented in Section 5.6.1. The simulator enables the same data sets to be used to test the performance of the tracking system in different configurations. Results of the aircraft acting independently against results in a DDF network are presented in Section 5.6.2.

Section 5.7 details the configuration of the system for real-time implementations of the DDF architecture on multiple aircraft. Results of these real-time demonstrations are included in Sections 5.7.1–5.7.3. These results represent the first ever use of cooperative UAV's in decentralised estimation [72, 102]. Section 5.8 concludes the Chapter with a summary of the key points.

5.2 Problem Definition

This Chapter presents a classic multi-target tracking problem. The objective is to track all targets within some bounds, and therefore build a picture of all objects in that area.

There has been extensive research in the area of tracking over the last fifty years. This work has concentrated on topics such as computational efficiency, data association, model accuracy, multiple model techniques, multiple hypothesis techniques and spatial representations. Blackman [16, 17] and Bar-Shalom [5, 6, 8, 9] for example, contain very detailed descriptions of these techniques as well as results of the application of these to real problems.

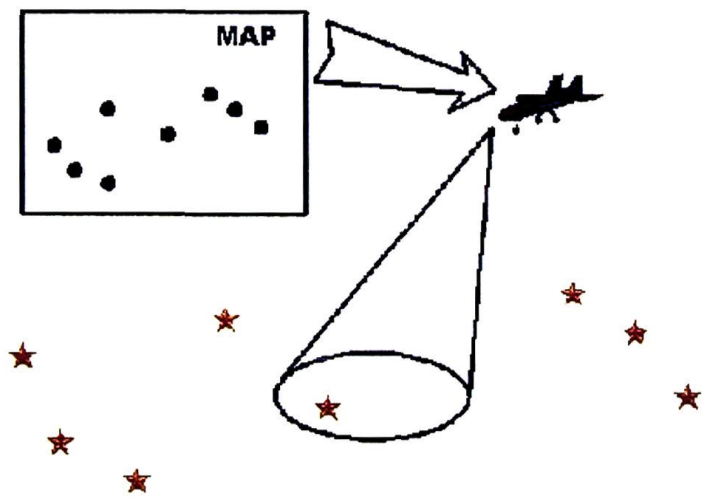


Figure 5.1: A single platform tracking multiple ground targets. The aircraft overflies some area and uses its sensors to detect and track any targets it encounters. Collectively, these targets form a picture or a map of the region.

The single platform picture compilation problem is illustrated in Figure 5.1, where an aircraft is building a map of targets in an area. The extension to the multiple vehicle scenario is, in principle, straightforward. If a central processor were used, all platforms could send

their target information to this central point for fusion. However, this architecture is neither scalable nor robust due to the reliance on a central fusion point. It is therefore desirable in the multi-platform picture compilation problem to fuse the global network information in a decentralised architecture, without a central fusion algorithm.

If the terrain sensing payload on each aircraft is integrated with local processing and communication facilities, the resulting unit can be treated as a sensing node. A decentralised network of these nodes can then be formed using the architecture and algorithms developed in Chapter 3. The sensing nodes can communicate directly in terms of target information in order to have a local estimate using global information. This concept is shown in Figure 5.2 where all platforms have the complete global picture available locally through the communication of target information. This communication has a number of positive effects:

- The extra information available to nodes through the communicated information improves the quality of the estimates.
- The robustness of a DDF architecture to 'node failure' is equivalent in the airborne tracking problem to 'aircraft failure'. If an aircraft is damaged or destroyed, the information it gathered is not lost in a DDF network as it has been communicated to all other vehicles.
- The picture of all targets in a region is available to all aircraft in the DDF network as soon as any single platform visits that area. Aircraft can therefore have a picture of targets in an area they they have not yet visited.

The framework used here leads very naturally toward directed sensing techniques. As all platforms have global knowledge and can track targets that they have not detected themselves, it is possible for a platform to enter an area for the first time and direct its attention immediately to the targets of interest. Furthermore, since the entire decentralised problem is formulated in its information form, there are readily available information metrics for predicting the information that could be gained by looking at each target. Grocholsky [44, 45] took this concept a step further by using information measures about particular targets or areas to determine control actions for the platforms themselves. Rather than directing a sensor to point at a particular area, this approach directed the platforms to locations which would maximise their information gain.

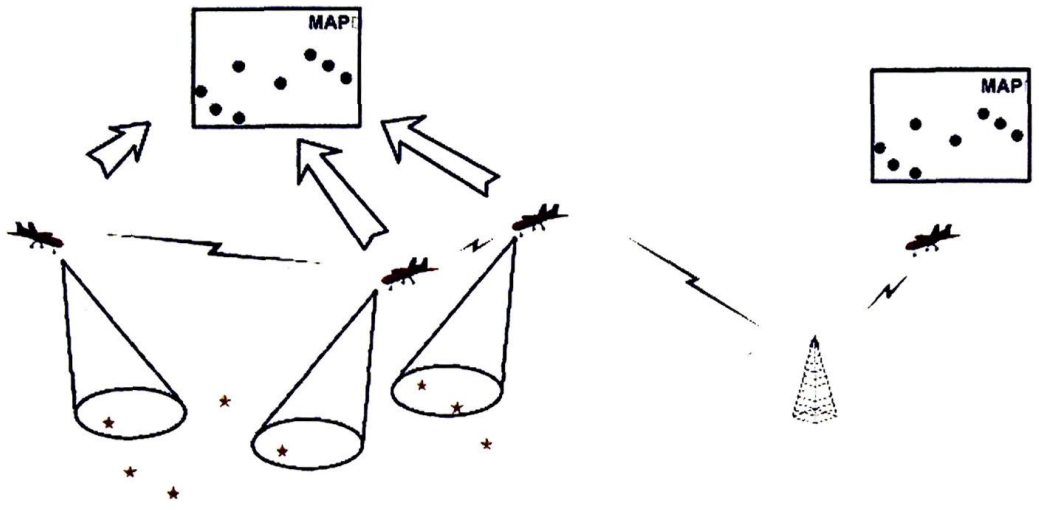


Figure 5.2: Multiple platforms performing decentralised tracking. The communication between aircraft enables all vehicles to have the same estimate of the target locations.

The targets that are tracked in this research are artificial landmarks positioned at stationary points on the ground. The targets were designed to be easily visible to the sensors used in order to focus on the decentralised nature of the task rather than areas such feature extraction.

The estimation of the target locations is done in a North, East, Down (NED) coordinate frame which is common to all platforms. These axes correspond to the x, y and z axes respectively, constructed using the right hand rule. Each platform is tracking the position of the target in all three axes, and the velocity in the North and East axes. The vertical velocity is not considered as the surface around the test facility is relatively flat and there is little to be gained from the extra state.

5.3 The Vision Sensor

This section presents the algorithms applied to the vision sensor described in Section 4.3.3.

5.3.1 Sensor Model

The camera was calibrated with methods similar to that described in [48] with a readily available tool-box. While these methods employ a fifth order polynomial for image correc-

tion, this is unsuitable for a real-time system with limited processing resources. A pinhole camera model was therefore chosen for simplicity and speed.

The calibration procedure provides the principal point (u_0, v_0) and focal lengths f_u, f_v (in pixels) for each axis and the expected error in the pinhole model. From this model a mapping from sensor coordinates (x, y, z) to image coordinates (u, v) can be defined.

$$u = \frac{yf_u}{x} + u_0 \tag{5.1}$$

$$v = \frac{zf_v}{x} + v_0 \tag{5.2}$$

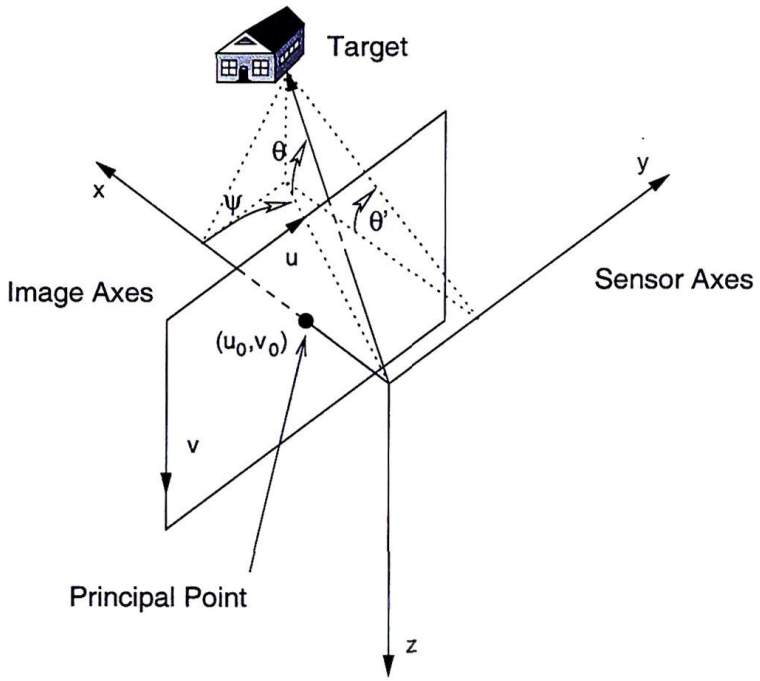


Figure 5.3: Principal point, image and sensor axes for the vision payload.

The inverse mapping cannot recover the loss of range information. However a direction to the feature can be recovered. It is convenient to define this information in a similar form to the Euler angles defining the current platform state, as shown in Figure 5.3. The relationship between the sensor and body axes is illustrated in Figure 5.4.

$$\psi = \arctan((u - u_0)f_u) \tag{5.3}$$

$$\theta' = \arctan((v - v_0)f_v) \tag{5.4}$$

$$\theta = -\arctan(\tan(\theta')\cos(\psi)) \tag{5.5}$$

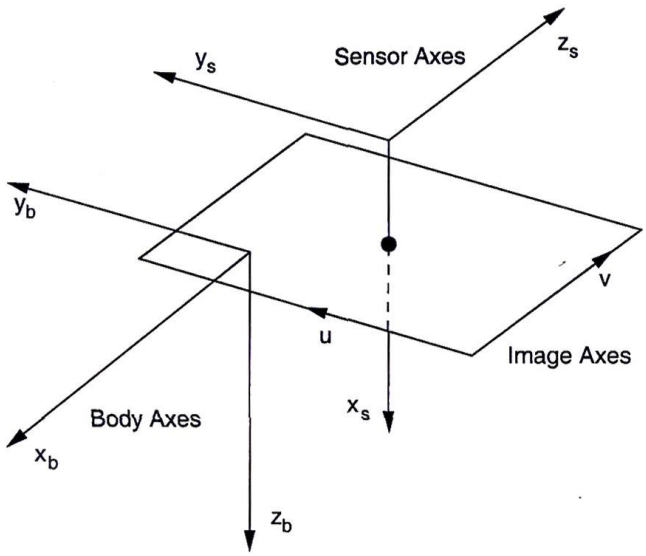


Figure 5.4: Image, sensor and body axes for the vision payload.

5.3.2 Feature Extraction

Due to limited processing resources, a simple but fast method of point based feature extraction is employed. All pixels above a threshold are converted into line segments. A range gate performs data association on these segments and the centre of mass of the pixels is obtained. The mass, aspect ratio and density of the cluster of pixels is then utilised for target identification. Figure 5.5 illustrates an image frame with an extracted target.

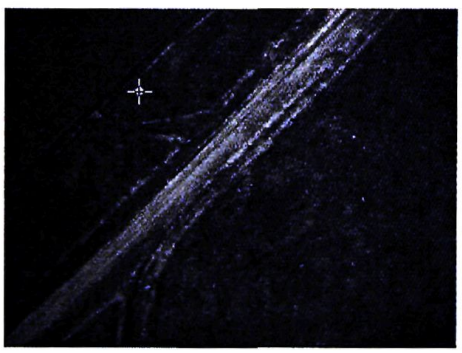


Figure 5.5: Image showing extracted feature.

Some range information can be recovered due to the fact that the beacons are of a known size. Utilising Equation 5.2 and assuming the target is parallel to the image plane, the height, width and corresponding area (in pixels) A_i of an object can be defined. This gives

$$h_u = \frac{h_y f_u}{x} \tag{5.6}$$

$$w_v = \frac{h_z f_v}{x} \tag{5.7}$$

$$A_i \approx \frac{f_u f_v A_t}{x^2}, \tag{5.8}$$

where A_t is the beacon area.

Re-arranging provides an expression for an estimate of the perpendicular height x above the target, with the area being expressed in terms of effective side length in pixels p. Differentiating with respect to p provides an expression for the error in the height estimate where ${\sigma_p}^2$ is the variance in the p. Figure 5.6 illustrates the error associated with this height estimate given the pixel resolution of the camera. The range estimate is then obtained by correcting the height estimate with the direction the the target.

$$x = \sqrt{\frac{f_u f_v A_t}{p^2}} \tag{5.9}$$

$$\sigma_r = \frac{f_u f_v A_t}{x p^3} \sigma_p \tag{5.10}$$

$$r = \frac{x}{\cos(\theta)\cos(\psi)} \tag{5.11}$$

The complete noise statistics of the vision sensor are listed in Table 5.1. The values were obtained from the camera calibration procedure and from experimental data.

Statistic	Value
σ_θ	0.0022 radians
σ_ψ	0.0028 radians
σ_p	1.5 pixels

Table 5.1: Vision sensor noise statistics.

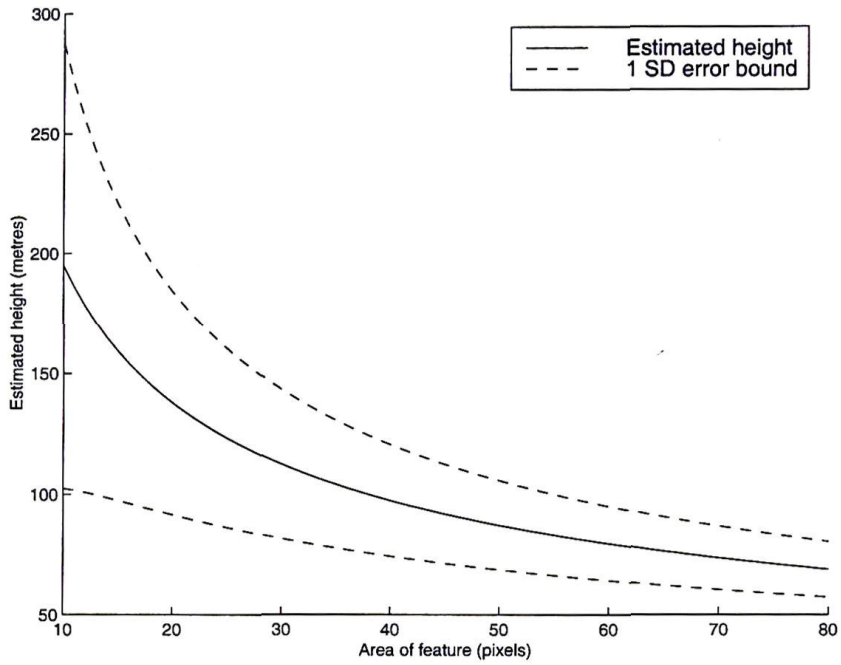


Figure 5.6: Height estimate and error bounds.

5.4 Vision/Laser Sensor: SIS

The SIS is a virtual sensor obtained by using a laser to obtain a range to targets detected by a camera. Its sensor model is identical to that of the vision system, with the exception that range is now provided by the laser rather than determined by the size of the target in the image. This virtual sensor therefore provides the same information as the vision system, but with a greater accuracy in the target range.

The noise statistics of the sensor are given in Table 5.2. As the elevation and bearing are observed by the camera, the values of σ_θ and σ_ψ are identical to those of the vision system listed in Table 5.1. The standard deviation in the laser range was determined from experimental data.

Statistic	Value
σ_θ	0.0022 radians
σ_ψ	0.0028 radians
σ_r	0.6 metres

Table 5.2: SIS noise statistics.

5.5 Track Formulation

This section describes the models, environment and management methods used in the implementation of the ground target tracking problem. These are implemented using the decentralised architecture presented in Chapter 3 on multiple uninhabited aircraft for real-time demonstrations of DDF.

The aircraft track the position of the ground targets in 3-dimensions and their associated planer velocity on the surface. The state vector for each target is given by

$$\mathbf{x}(k) \begin{bmatrix} x(k) & \dot{x}(k) & y(k) & \dot{y}(k) & z(k) \end{bmatrix}^T. \tag{5.12}$$

5.5.1 Process Model

The targets are described using the discrete model

$$\mathbf{x}_i(k) = \mathbf{F}_k \mathbf{x}_i(k-1) + \mathbf{B}_k \mathbf{u}_i(k) + \mathbf{G}_k \mathbf{w}_i(k) \quad i = 1, \ldots, n, \tag{5.13}$$

introduced in Chapter 2, where n is the number of targets being tracked. The target positions and velocities in x and y are modeled as Integrated Ornstein-Uhlenbeck (IOU) processes [94], and the z position as a simple brownian process. In the demonstrations considered in this thesis the targets are known to be stationary so the IOU process is tuned to rapidly decay target velocity towards zero.

The state transition matrix for this process model is given by

$$\mathbf{F}_k = \begin{bmatrix} 1 & \Delta T & 0 & 0 & 0 \\ 0 & F_v & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta T & 0 \\ 0 & 0 & 0 & F_v & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.14}$$

where

$$F_v = e^{-\Delta T \gamma}. \tag{5.15}$$

The process noise is written as $\mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^T$ where

$$\mathbf{Q}_k = \begin{bmatrix} q_x & 0 & 0 \\ 0 & q_y & 0 \\ 0 & 0 & q_z \end{bmatrix}, \tag{5.16}$$

and

$$\mathbf{G}_k = \begin{bmatrix} 0 & 0 & 0 \\ \sqrt{\Delta T}(1-F_v) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \sqrt{\Delta T}(1-F_v) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.17}$$

5.5.2 Sensor Preprocessing

To simplify the filter observation model, the raw sensor observations are converted to cartesian coordinates $[x, y, z]$ in a global reference frame. The transformations are applied in the sensor preprocessing stage, which outputs the observations in their information form ($\mathbf{i}(k)$ and $\mathbf{I}(k)$). When implemented in this manner, the filter observation model is reduced to a simple linear model and is the same for all sensor types. This ensures that the filter code on each node is exactly the same and that only the sensor preprocessing is sensor specific. The conversion of observations to cartesian coordinates is calculated as follows:

The Cosine Matrix relating frame i to frame j is obtained by first rotating the frame by the yaw angle ψ, then pitch θ and then roll ϕ. For notational brevity, *sin* and *cos* are written as s and c respectively.

$$\mathbf{C}_i^j = \begin{bmatrix} c(\psi)c(\theta) & c(\psi)s(\theta)s(\phi) - s(\psi)c(\phi) & c(\psi)s(\theta)c(\phi) + s(\psi)s(\phi) \\ s(\psi)c(\theta) & s(\psi)s(\theta)s(\phi) + c(\psi)c(\phi) & s(\psi)s(\theta)c(\phi) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \tag{5.18}$$

This defined, the objective is now to relate the target to the earth frame. This is calculated by relating the target to the sensor, the sensor to the body and then the body to the earth

frame. That is,

$$\mathbf{P}_t^e = \mathbf{P}_b^e + \mathbf{C}_b^e \mathbf{P}_s^b + \mathbf{C}_b^e \mathbf{C}_s^b \mathbf{P}_t^s, \tag{5.19}$$

where

- $\mathbf{P}_b^e = [x_b^e, y_b^e, z_b^e]$ is the position of the body in the earth frame and is provided by the GPS/INS filter [58]. It is important that this filter be accurate as it is used to register the relative observation generated by the sensor in the global frame. Any error in the estimate of $\mathbf{P}_b^e$ will result in a corresponding error in the converted sensor observation.
- $\mathbf{C}_b^e$ is the direction cosine matrix which relates how the body frame is rotated with respect to the earth frame.
- $\mathbf{P}_s^b = [x_s^b, y_s^b, z_s^b]$ is the position of the mission sensor to the body frame. This is determined by calibration.
- $\mathbf{C}_s^b$ 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 axes of both the sensor and body frames are parallel then the matrix is simply an identity matrix.
- $\mathbf{P}_t^s = [x_t^s, y_t^s, z_t^s]$ 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 a range measurement. θ_t^s 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 this point, the azimuth of the target ψ_t^s can be determined. Thus the position of the target becomes

$$\mathbf{P}_t^s = \begin{bmatrix} x_t^s \\ y_t^s \\ z_t^s \end{bmatrix} = \begin{bmatrix} r_t^s \cos\theta_t^s \cos\psi_t^s \\ r_t^s \cos\theta_t^s \sin\psi_t^s \\ r_t^s \sin\theta_t^s \end{bmatrix}. \tag{5.20}$$

Note that for the radar system the look down angle (grazing angle) is fixed (θ_t^s is constant). Also note that positive is looking 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 σ_r in range, σ_ψ in azimuth and σ_θ in pitch, the converted observation noise matrix $\mathbf{R}_k'$ is a function of the square of the range of the observation and the rotation matricies.

$$\mathbf{R}_k' = \left[\mathbf{C}_b^e \mathbf{C}_s^b\right] \begin{bmatrix} \sigma_r^2 & 0 & 0 \\ 0 & r^2\sigma_\psi^2 & 0 \\ 0 & 0 & r^2\sigma_\theta^2 \end{bmatrix} \left[\mathbf{C}_b^e \mathbf{C}_s^b\right]^T \tag{5.21}$$

When calculating the observation information using Equation 2.34, it is necessary to evaluate $\mathbf{R}_k'^{-1}$. This can done by recognising that the inverse of a rotation matrix $\mathbf{C}_i^j$ is its transpose $\mathbf{C}_i^{j\,T}$. The inverse of Equation 5.21 can now be written as

$$\begin{aligned} \mathbf{R}_k'^{-1} &= \left[\mathbf{C}_b^e \mathbf{C}_s^b\right]^{-T} \begin{bmatrix} \sigma_r^2 & 0 & 0 \\ 0 & r^2\sigma_\psi^2 & 0 \\ 0 & 0 & r^2\sigma_\theta^2 \end{bmatrix}^{-1} \left[\mathbf{C}_b^e \mathbf{C}_s^b\right]^{-1} \\ &= \left[\mathbf{C}_b^e \mathbf{C}_s^b\right] \begin{bmatrix} \frac{1}{\sigma_r^2} & 0 & 0 \\ 0 & \frac{1}{r^2\sigma_\psi^2} & 0 \\ 0 & 0 & \frac{1}{r^2\sigma_\theta^2} \end{bmatrix} \left[\mathbf{C}_b^e \mathbf{C}_s^b\right]^T. \end{aligned} \tag{5.22}$$

If a bearing only sensor is used (no range information is available), σ_r can be set to infinity. Although this is not numerically possible in state space, in information space $\frac{1}{\infty}$ can be written as zero. The estimated range is now represented with infinite covariance, or equivalently, zero information. Using a more accurate value of r under these conditions will simply allow the estimate to converge more quickly.

Although the bearing only formulation in Equation 5.22 is valid, the vision system in this application extracts a range estimate based on the size of the target in the image (Equation 5.10–5.11). While the variance of this range estimate is large, it is defined rather than being infinite which enables the filter to converge more quickly than for the bearing only problem.

As the raw observations are now in cartesian space, the observation model mapping them

to the target state space given by Equation 5.12 can be written as

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.23}$$

The sensor preprocessing then uses Equation 2.34 to calculate the information in the observation and transmits it onto the decentralised filtering node.

It should be noted that although the observation model has been linearised by preprocessing the raw observation to the required coordinate system, it is not a requirement of the decentralised filter. The system could just as easily function using a non-linear observation model.

It is also recognised that the coordinate transformations used are only approximations. However, the resulting accuracy is sufficient for the applications considered in this thesis. Should a more accurate result be required, the unbiased polar to cartesian transform proposed by Bar-Shalom [5] or the unscented transform proposed by Julier [53] could be used.

Results of the coordinate transformation and sensor preprocessing for the vision sensor are included in Figure 5.7. The results shown have been filtered to remove observations taken when the aircraft is banking at greater than 30^o. This was necessary as estimated ranges at high bank angles were extremely noisy and unreliable. The results after this filtering indicate that there are still several areas, particularly on the lower left side of the plot, where clusters of observations are seen away from targets. Rather than being spurious observations, these are actually natural features such as patches of sand and wombat holes [97] which appear in the images very similar to the artificial white targets. They are detected consistently during every flight.

5.5.3 Artificial Targets

Artificial targets were used in the demonstrations in this thesis as the focus of the work is on DDF rather than target detection. The white 0.9m x 0.9m artificial vision targets were positioned at the flight test facility and their location surveyed using a carrier phase differential GPS receiver to an accuracy of 2cm. The number of targets used varied depending on

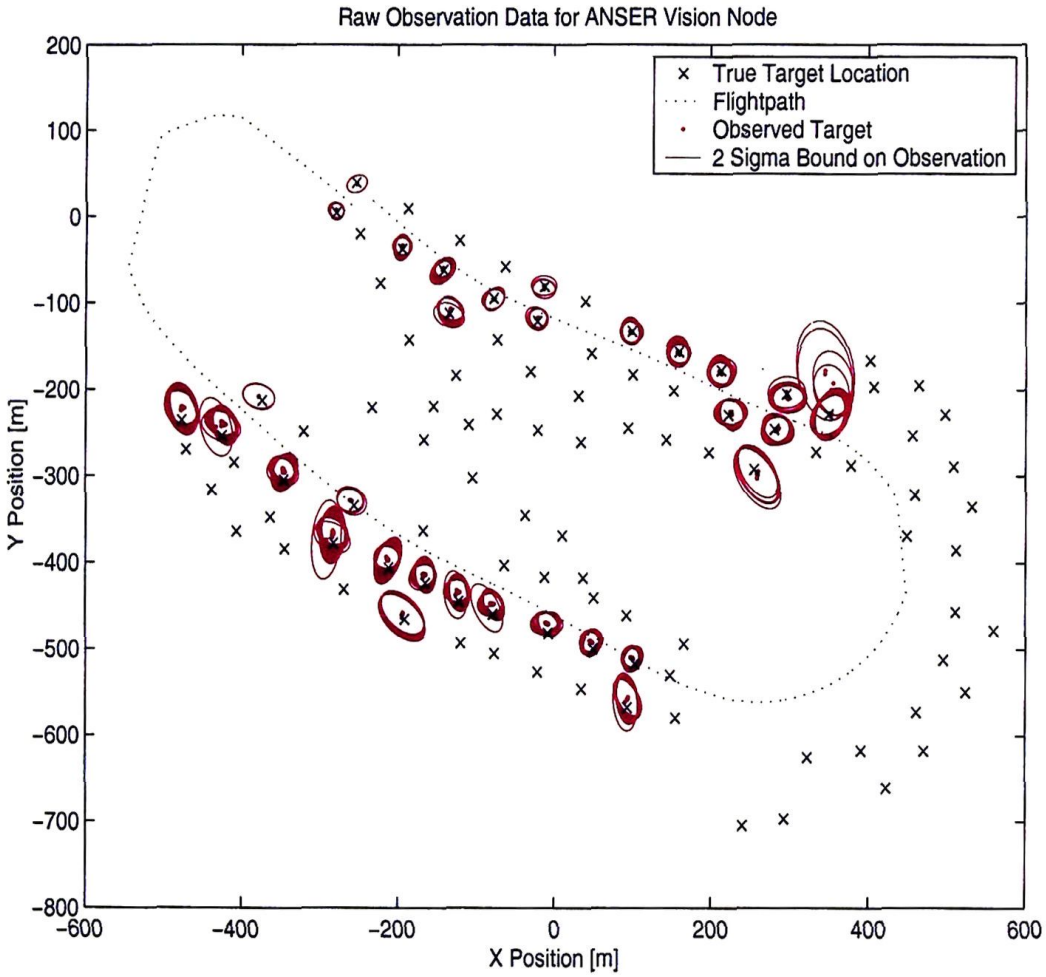


Figure 5.7: Real logged flight data recorded from the vision system after the transformations from range/bearing to cartesian coordinates. All observations were from a single flight using one aircraft. Observations taken when the aircraft is banking at greater than 30^o are ignored.

the flight trial requirements, but was never less than 50. The locations of these targets were also changed between different flights in order to maximise the number under the flightpath at all times.

At the time of writing this thesis the radar is undergoing calibration and testing trials and has not been flown as a sensing payload. However, when it is flown, tetrahedral radar beacons are to be co-located with the vision targets to ensure that both sensors can be used to track the same objects.

When 150 targets were deployed during the initial real-time flight, the target density was found to be too high for robust data association. The minimum spacing between any two targets in this demonstration was less than 20m. During poor GPS coverage, the vehicle pose uncertainty increased (particularly in attitude) and it was possible for the uncertainty in the observation covariance to encapsulate multiple targets. For later flights, approximately 50 targets were used which ensured that no two targets were closer than 50m.

5.5.4 Track Maintenance

For the initial real-time flight trials, a maximum of 17 targets were tracked using vision nodes with a 266MHz CPU. This was later changed to 15 targets on nodes with a 266MHz CPU and 25 targets on a 700MHz CPU. The number of targets was decreased on the 266MHz CPU as more complexity was added to the algorithm to improve the results of the initial real-time flights. The limiting of tracked targets is done to ensure adequate computation is available on the PC104. However, it is a conservative estimate and the number of tracks can be increased substantially using the same processors.

When a new target is detected, a filter is initialised with the first observation. To detect tracks that are initialised with spurious observations, all filters are maintained as 'tentative' tracks for the first two seconds. After this time, the tentative track must contain a minimum of 10 valid observations to be promoted to a track, otherwise it will be reaped and the filter classified as unused. This method was found to be very effective when using the vision system as each complete pass over a target at cruising speed and height results in a minimum of 20 successive frames.

No track-to-track fusion algorithm was implemented on the simulator or during the first real-time flights. Under conditions where the ANSER simulator was used to playback real

flight data this did not cause any problems. However, real-time results from the first flights do indicate that at some instances there were multiple filters allocated for the same target. A track-to-track fusion algorithm is required under these circumstances in order to fuse the two estimates into a single track. A more simplistic approach is to simply delete all but one of the tracks, however this can discard a significant amount of useful information. For the initial real-time testing the focus concentrated on the DDF algorithms rather than track maintenance so the poor allocation (and effective loss) of a small number of filters tracking the same target was accepted.

To eliminate the wasteful allocation of filters, a simple track-to-track fusion algorithm was implemented for later flights. It uses a distance measure to determine filters which are tracking the same targets, and fuses multiple estimates (for the same target) together using CI. This ensures that the resulting single track remains consistent, and was found to work very well. However, this method is conservative and other track-to-track fusion algorithms formulated in information space (such as [28]) could be implemented in this framework in a straightforward manner.

5.5.5 Data Association

The information gate, described in Section 3.5.6, is used for observation to track data association. Each observation is tested against every track, and only used if a unique assignment is made. While this method is computationally expensive, it runs in real-time provided the maximum number of targets is bounded. However, results of the initial real-time flights indicted that this algorithm alone was not sufficient when targets were grouped very closely together, as the vehicle uncertainty could sometimes be sufficiently large that multiple assignments were possible. To solve this issue, the GPS/IMU localisation filter was improved and the target density was reduced such the targets could be uniquely identified.

The information gate is also used for track to track association when DDF information is communicated between vehicles. The first real-time flights demonstrated that here too the target density was sufficiently large that multiple assignments were possible. Another issue on the initial real-time flights was that observations that were correlated through the vehicle position were treated as independent, which resulted in overly confident estimates. This gave results where the estimate was often within a few metres of the true target location,

but the covariance was overly confident and did not encompass the true location. When DDF information about these estimates was communicated, it was often impossible for the information gate to correctly associate them as the same track, instead treating them as two separate tracks. This issue was initially handled by adding a distance based gate to the information gate that matched any estimates that were located closely together. However, the issue was solved completely when the filtering algorithm was modified to correctly handle observations that are correlated (see Section 5.5.7), as this ensured that estimates did not become overly confident.

To further improve data association in the future, a frame to frame tracker for the vision node is proposed. Using this technique, the first observation by the camera would need to be associated with a target track using the information gate, but observations of the same target in subsequent sequential frames can be identified by the vision node using the frame to frame tracking algorithm. For an average target which is observed for a 2 second period at $50Hz$, this would reduce the number of applications of the information gate from 100 to 1.

5.5.6 Communication Strategy

The decentralised architecture presented in this thesis allows the communication rate between nodes to be selected as part of the system design. In the application described here, two nominal communication rates are used:

1. $2Hz$: Every half second, nodes communicate any tracks that have received new information. For example, if a node is tracking 20 targets but only three of them contain information that has not yet been communicated, then only these three will be sent. There is no need to communicate the remaining 17 as this will not give neighbouring nodes new information. New information occurs when observations are made or when information is received from another node. When information arrives through a channel, it is not propagated back to its sender at the next communication time unless the local node has received further information from another source. This ensures that nodes do not continue to communicate information gains of zero.

2. $0.1Hz$: Every 10 seconds, nodes communicate all information about all tracks. This ensures that if any of the higher frequency information messages are lost through

some communication failure, the nodes can get the information in a timely manner. Although this communication rate is not strictly necessary (the nodes will get any lost information next time that particular target is observed), it was implemented for the initial real-time flight test as the radio LAN between aircraft had not been used in this role previously. This communication rate was only used on the real-time flight and not the ANSER simulator. Results of the real-time flight also indicate that it may not be necessary to continue these $0.1Hz$ messages in future flights.

5.5.7 Data Fusion

This section describes the evolution of the data fusion algorithms for the real-time picture compilation problem. Following each set of real-time flights, the results were evaluated and any issues were addressed for future flights.

The sensor preprocessing module was implemented to convert the raw sensor observations into the global cartesian frame in an information form. This was done by registering the relative observation using the aircraft pose estimate and adding the pose uncertainty to every observation.

For the initial real-time flight, the error in the vehicle position was assumed to be zero mean and Gaussian over multiple passes of the target and the observations were simply added in the information filter update stage. However, as each pass over a target resulted in approximately 15-30 observations of each target, any error in the vehicle pose during that pass gave a corresponding error in the estimated target location. More importantly, the target covariances were found to be overly confident as all 15-30 observations on each pass were in fact correlated through the vehicle pose.

This issue was corrected following the initial flights by using the CI algorithm to fuse information from the locally attached sensors at the nodal filter. The use of this conservative method to fuse correlated observations was extremely successful, as results from recent real-time flights demonstrate. However, the CI algorithm is only used to fuse information which is correlated through the vehicle position, and the standard information filter update used at all other times. Specifically, CI is used to fuse observations from the locally attached sensor at the nodal filter, and also to fuse DDF information from another node on the same airframe. This latter case is necessary as multiple sensing nodes on the same air vehicle

use the same pose estimate to register the target location. Information received from nodes on other vehicles in the DDF network are handled using the standard additive information update to make complete use of information.

5.6 Implementation

This section presents results of implementing the DDF tracking algorithm both offline and in real-time. The offline results are obtained by post processing real flight data in the ANSER simulator. Although it is possible for the simulator to generate its own synthetic data, the use of logged flight data is more realistic. Real-time results are obtained from flight trials with the decentralised algorithm implemented on multiple aircraft.

5.6.1 Offline Implementation

The following results were obtained using the ANSER simulator in playback mode with real flight data. The data sets were from a single platform flight test using a camera payload registered with IMU and GPS data. This single 15 minute flight was then spliced into four separate 3 minute segments (with no segments overlapping) to 'simulate' the four different aircraft nodes listed in Table 5.3. The flight paths over the terrain were determined by fusing the real GPS/IMU data [58].

The camera payload which was used on this flight logged frames at 50Hz. From these images, a bearing and approximate range (based on the size of the target) was extracted. While the uncertainty in range is extremely large, range information of any sort improves track quality.

The implementation was first run with each platform acting independently. There was no communication of any sort between aircraft. In this situation, each of the four aircraft track targets using only observations from their local sensor. The ANSER simulator was then run a second time using exactly the same data sets, but with the aircraft configured in a decentralised tree network as shown in Figure 5.8. Here, aircraft communicate target information to build a single common picture of target tracks. Results of these two scenarios are presented in Section 5.6.2.

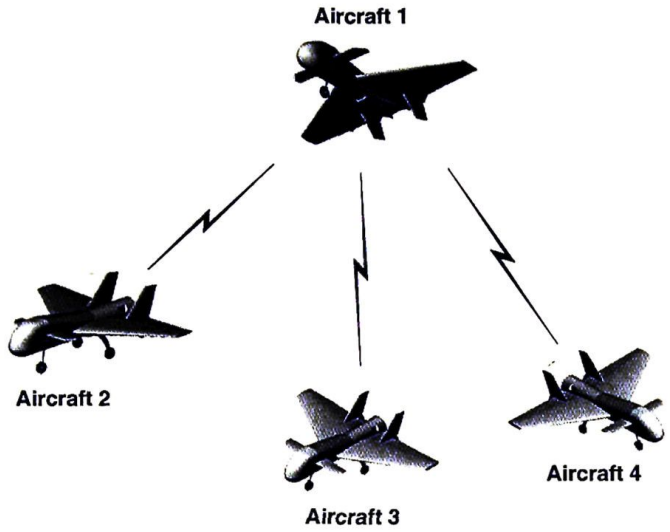


Figure 5.8: Peer-to-peer communications topology used for offline implementation.

Node	Location	Sensor
1	Aircraft 1	Vision System
2	Aircraft 2	Vision System
3	Aircraft 3	Vision System
4	Aircraft 4	Vision System

Table 5.3: Node configuration for post-processed data.

5.6.2 Offline Implementation Results

Results of tracking a typical target are presented for both the independent platform and decentralised network scenarios. The maps of tracked targets generated by the aircraft operating individually are plotted in Figures 5.9 and 5.10 and in Figures 5.11 and 5.12 for the DDF network. The maps on each aircraft when operating independently are all different. This is expected, as there is no sharing of information. Conversely, the DDF network has all aircraft operating with exactly the same map. Furthermore, the target estimates in a DDF environment have smaller errors. A zoomed in section of the DDF map is illustrated in Figure 5.13 to give a clearer indication of the estimates and the 2σ ellipses.

Consider now the results illustrated in Figures 5.14–5.19 which describe the evolution of a typical target estimate over the entire period of the flight. When the aircraft act independently, this track was only initialised on each platform when the local sensor observed

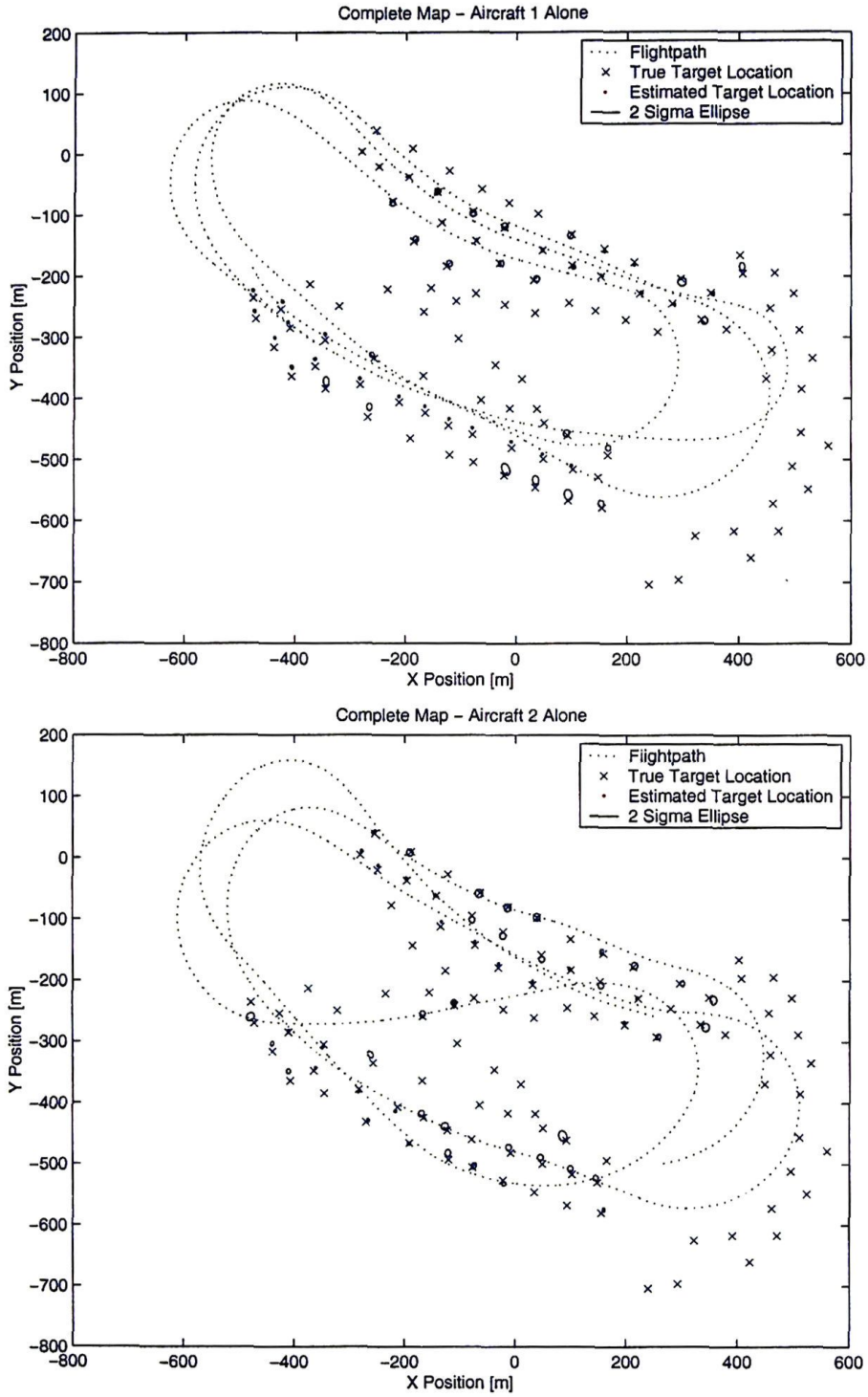


Figure 5.9: Post-processed results: When the aircraft operate independently, they each generated their own map as no information is communicated from other platforms. The maps from aircraft 1 and 2 plotted here are clearly different.

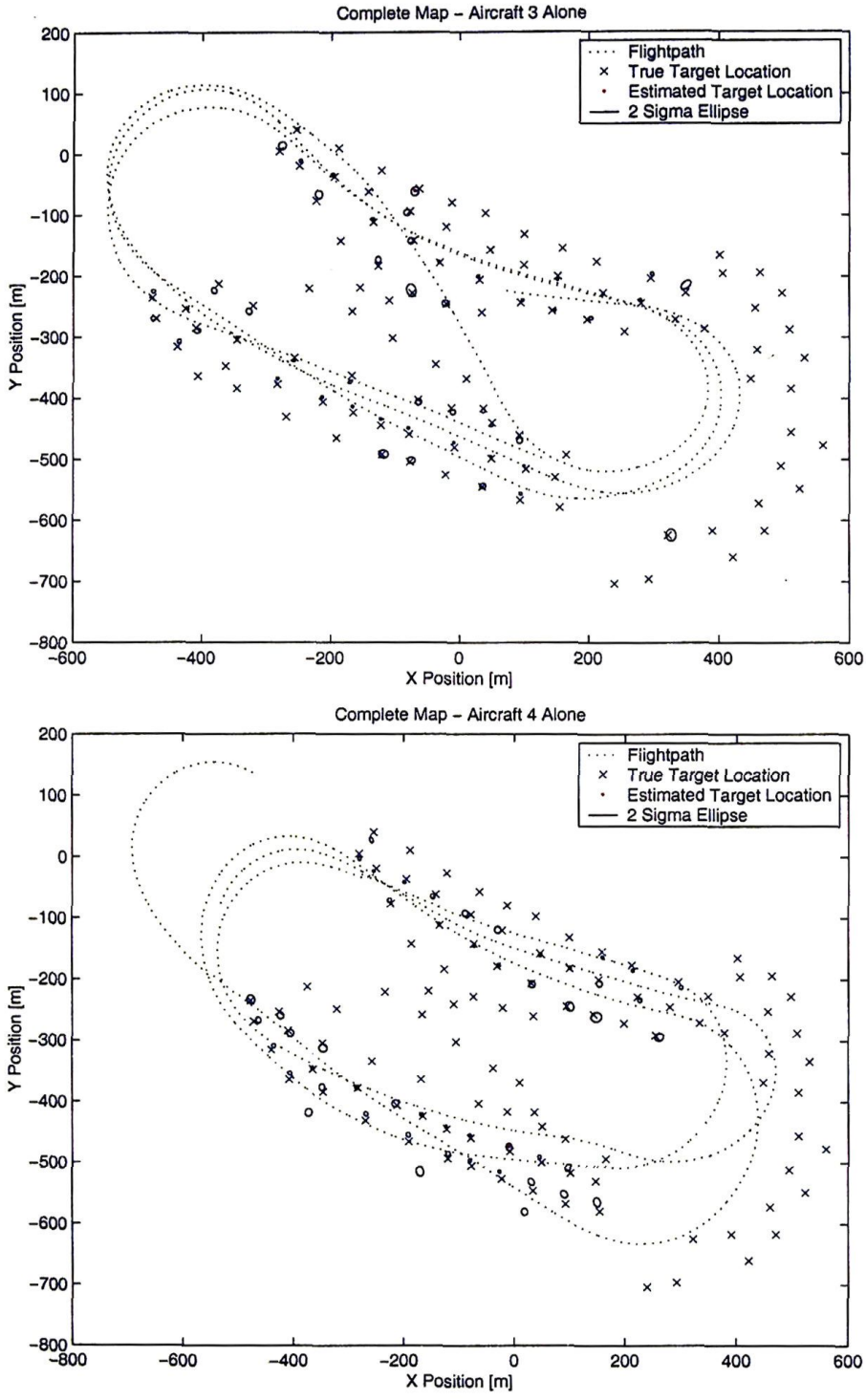


Figure 5.10: Post-processed results: The maps from aircraft 3 and 4 operating independently are different as each platform is using only locally generated information.

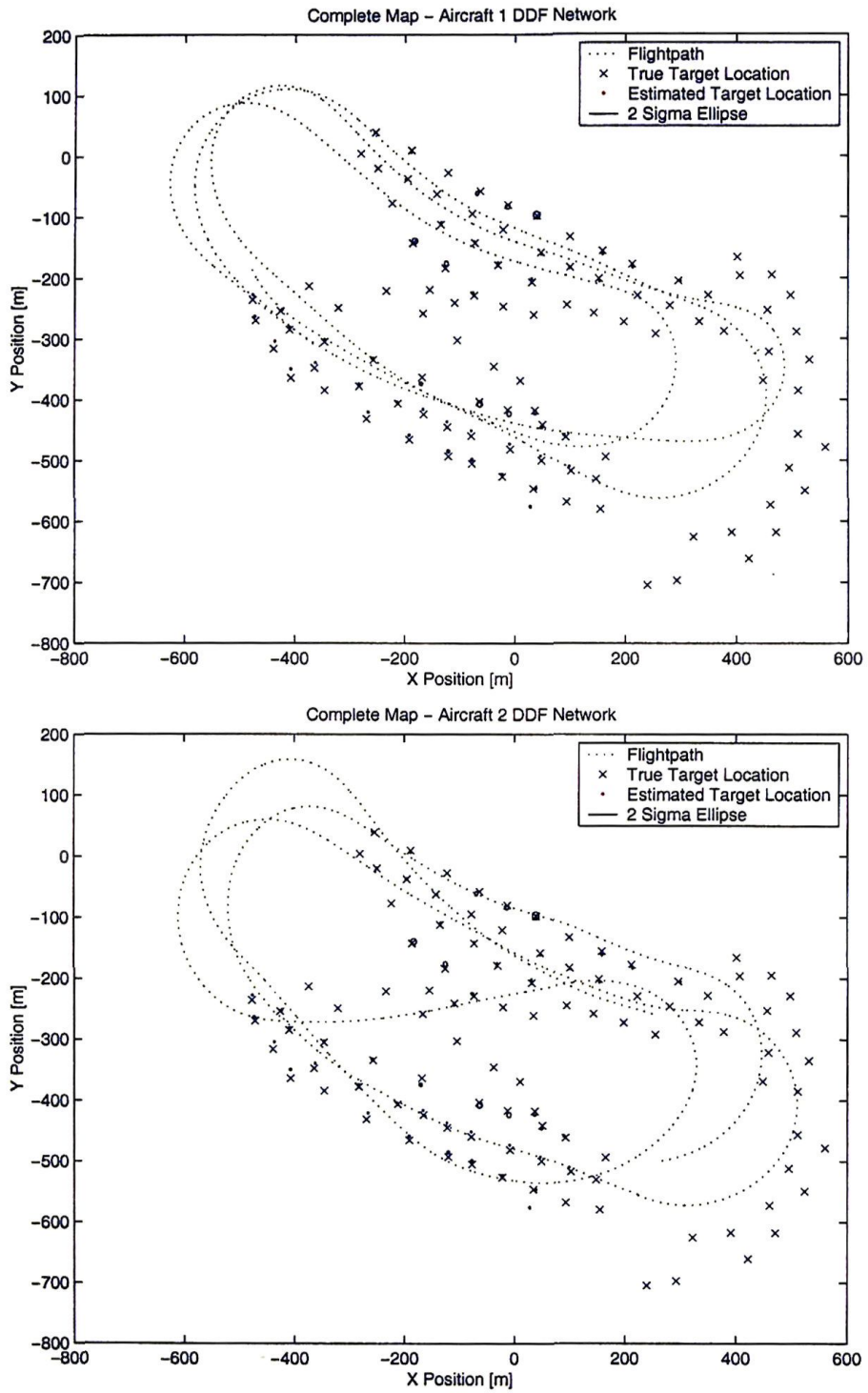


Figure 5.11: Post-processed results: The map of targets tracked by aircraft 1 and 2 are plotted with the true target locations and the respective flightpaths for the case where the platforms operated in a decentralised network. Note that the map on each platform is identical.

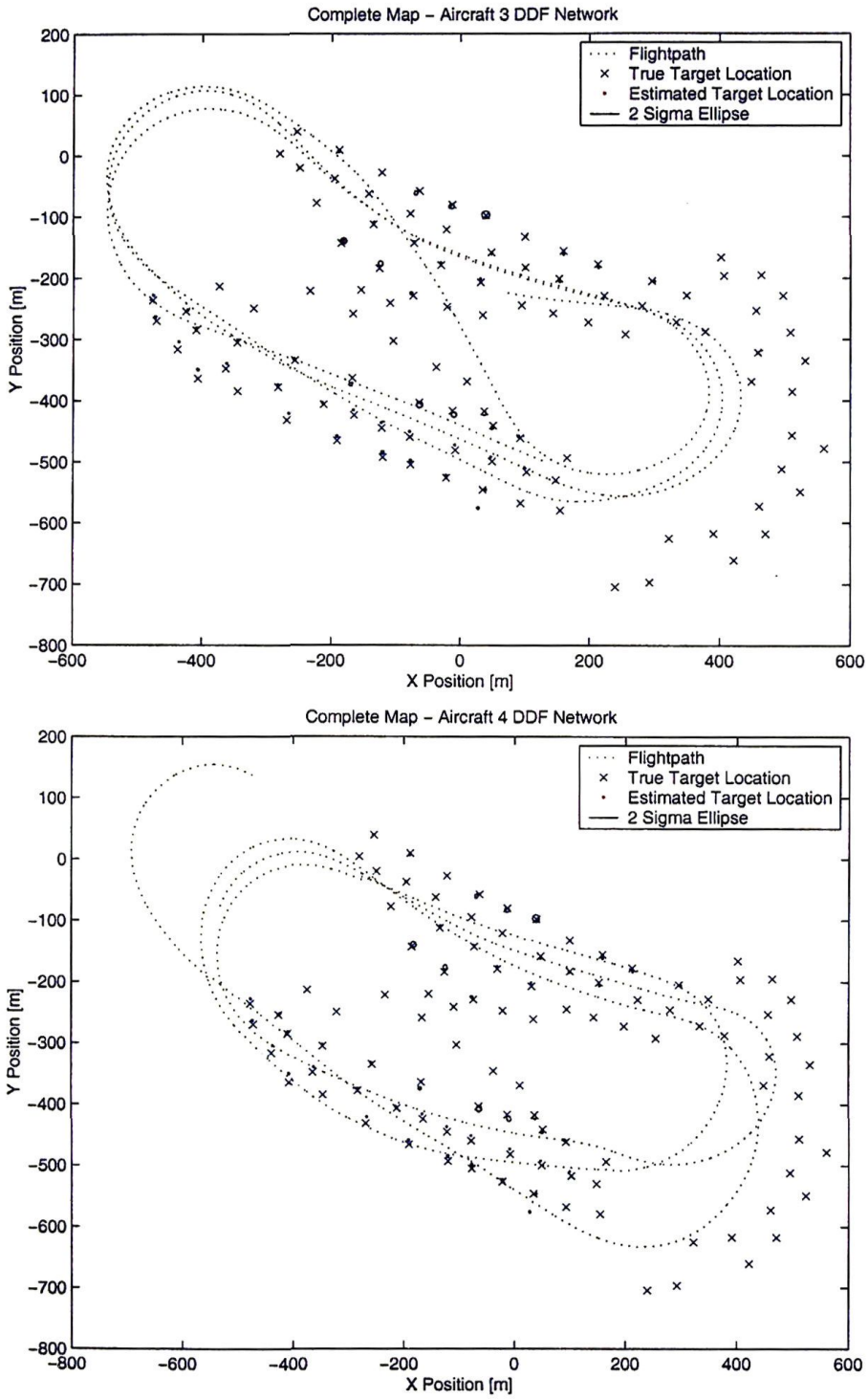


Figure 5.12: Post-processed results: The maps generated on aircraft 3 and 4 are identical to those of aircraft 1 and 2 when operating in the DDF network.

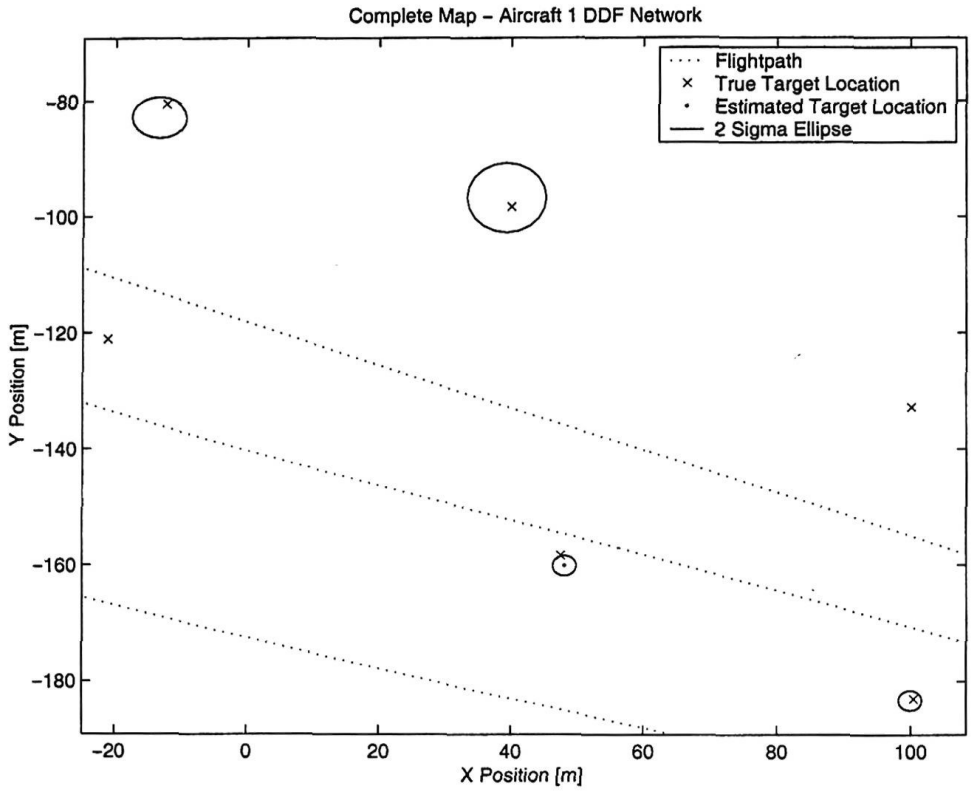


Figure 5.13: Post-processed results: A close up of the DDF network map from aircraft 1. The estimates are all very close to the true target locations.

the target for the first time. Inspection of Figure 5.14 for the case where the platforms act independently shows the track begins at approximately 25 seconds for platform 2, 28 seconds for platform 3 and 61 seconds for platform 4. As platform 1 never sees this target, it does not ever track it. In comparison, in the DDF network results shown in the same figure, all aircraft including platform 1 begin tracking this target immediately after platform 2 first observes it. This sharing of information around the network enables all platforms to maintain estimates of targets as soon as one initialises a track.

Once all platforms have information about a target, any further observations they make can be used to improve the estimate. Figures 5.14–5.16 show the position error over the entire period of the simulation. It is clear from these plots that there is a significant improvement in the quality of the estimate from the decentralised network. Figure 5.14 shows the error in the x position for each target to be approximately 5m, 2m and 12m respectively for

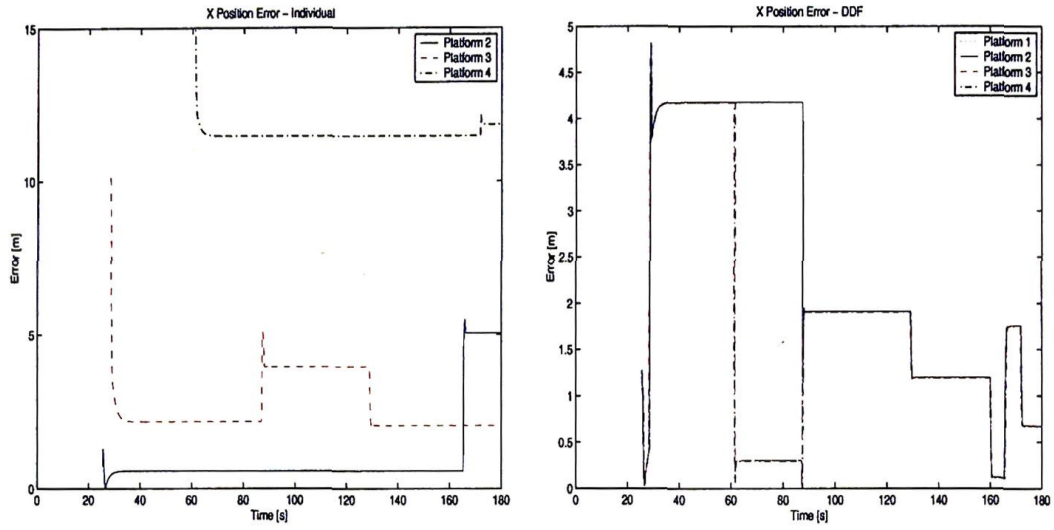


Figure 5.14: Post-processed results: x axis position error of a typical target for 4 aircraft acting individually and in a DDF network. When the platforms act independently, the errors are all independent and are different. In the DDF network, the error plots for each platform are all essentially on top of one another as all platforms are using global information. Any period where the plots are not the same is the result of a communication dropout.

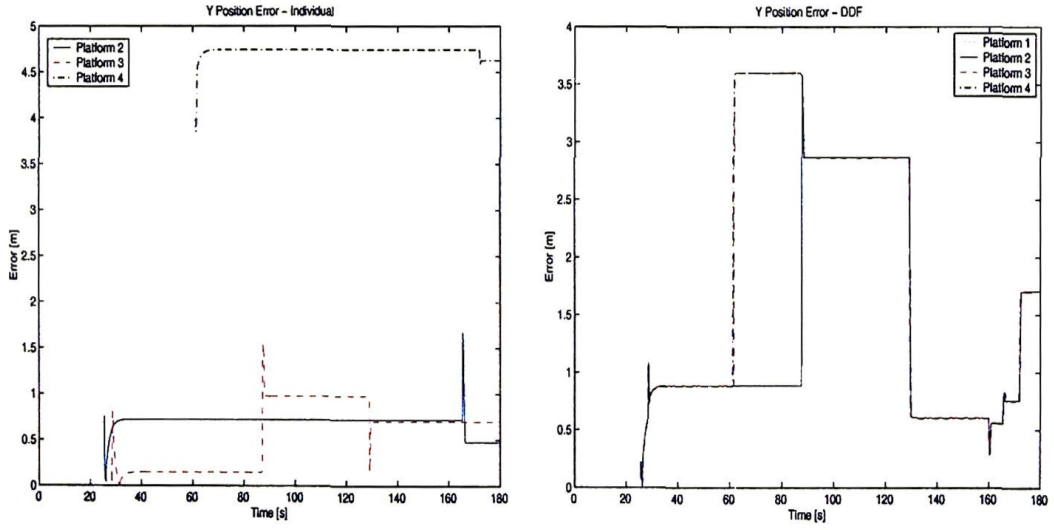


Figure 5.15: Post-processed results: y axis position error of a typical target for 4 aircraft acting individually and in a DDF network. Although two of the individual aircraft each have relatively good estimates in y, the third is less accurate and the fourth does not see the target at all. The DDF network enables all platforms, including the one that does not observe the target, to have an accurate estimate.

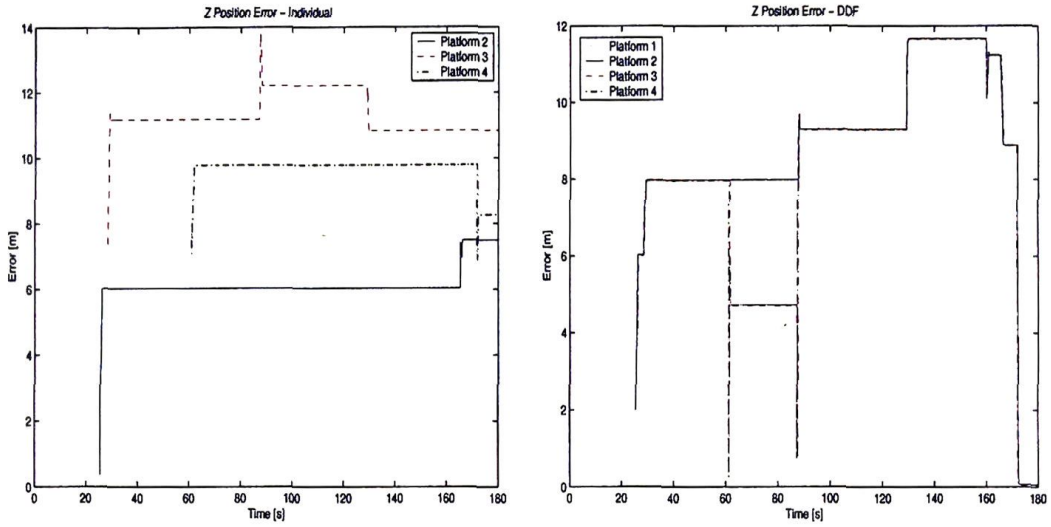


Figure 5.16: Post-processed results: z axis position error of a typical target for 4 aircraft acting individually and in a DDF network. The z axis error is significantly larger than that for x and y as the vision sensor used does not directly provide range information. In order to obtain this, an estimate of range is extracted based on the size of the target in pixels.

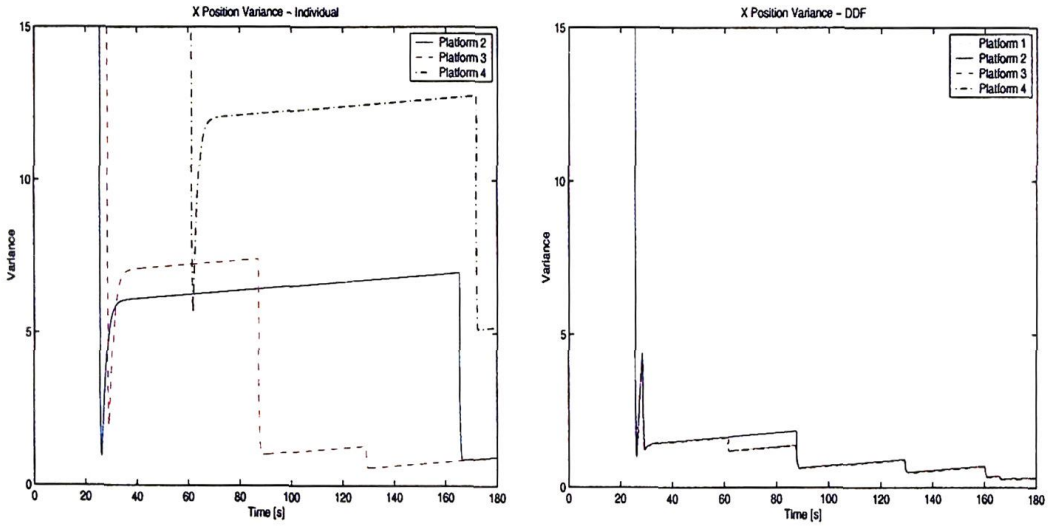


Figure 5.17: Post-processed results: x axis position variance of a typical target for 4 aircraft acting individually and in a DDF network. As the aircraft in the DDF network all have global information, this results in an estimate with a lower variance than for any of the independent single platforms.

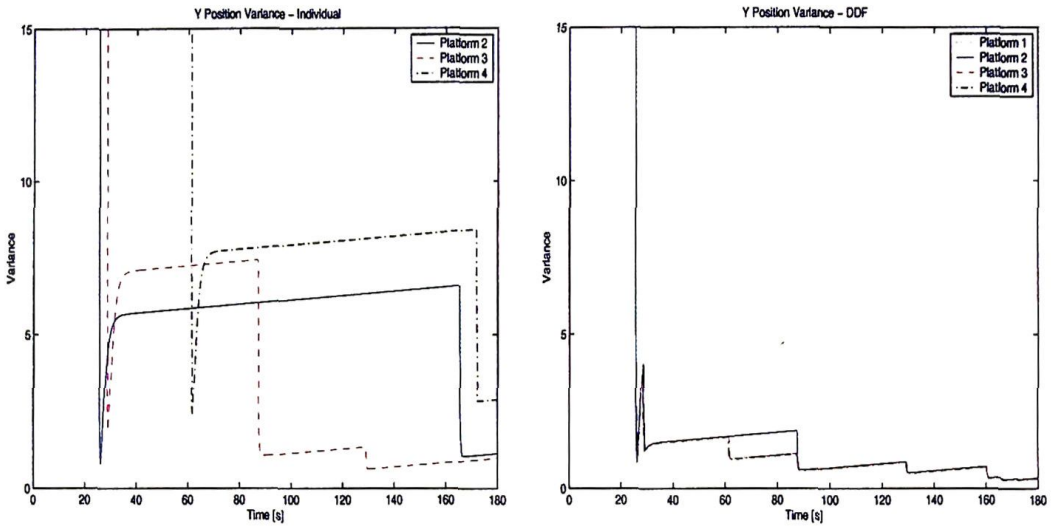


Figure 5.18: Post-processed results: y axis position variance of a typical target for 4 aircraft acting individually and in a DDF network. Note that when acting individually, all of the platforms run for a significant amount of time without observing the target as the variance gradually increases due to the process noise. However, in the DDF network the target variance on all platforms is decreased whenever any one of them makes an observation.

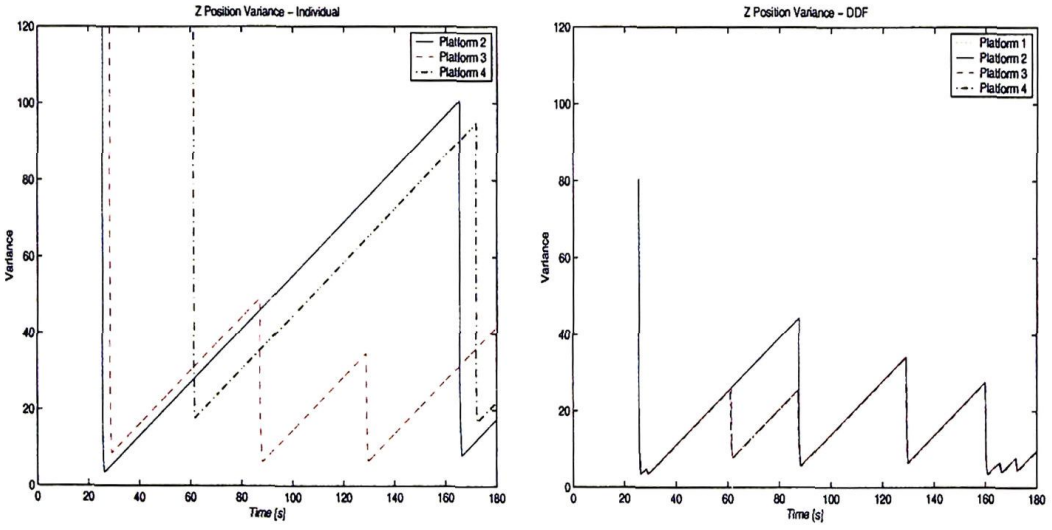


Figure 5.19: Post-processed results: z axis position variance of a typical target for 4 aircraft acting individually and in a DDF network. The periods where the independent platforms do not observe the target is very noticeable in the z position variance as it grows rapidly. The DDF network enables the growth of this variance to be managed more effectively by making use of global information.

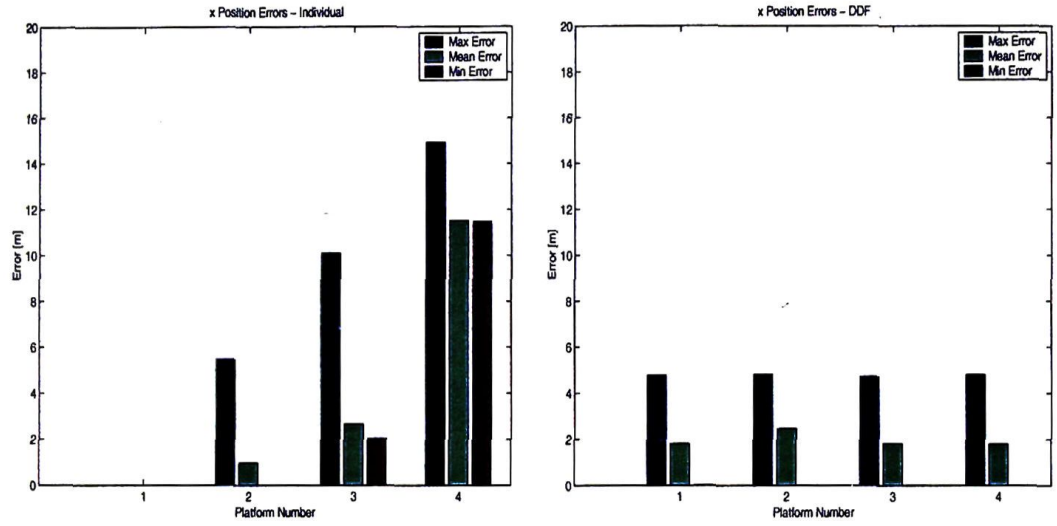


Figure 5.20: Post-processed results: The maximum, mean and minimum errors in x for a typical target are plotted over the entire flight for aircraft acting both independently and in a DDF network.

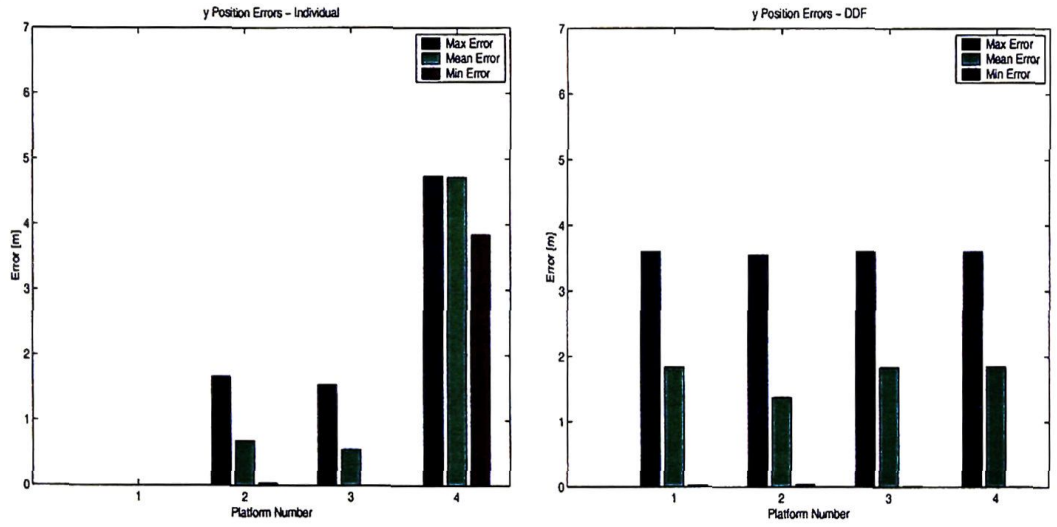


Figure 5.21: Post-processed results: The maximum, mean and minimum errors in y for a typical target are plotted over the entire flight for aircraft acting both independently and in a DDF network.

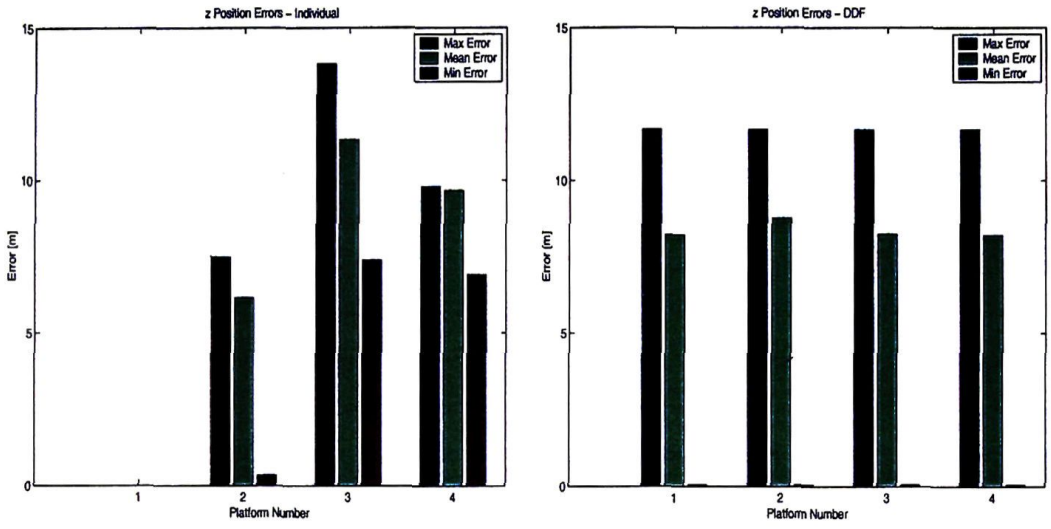


Figure 5.22: Post-processed results: The maximum, mean and minimum errors in z for a typical target are plotted over the entire flight for aircraft acting both independently and in a DDF network.

platforms 2, 3 and 4 when they are acting independently. However, when operating in a decentralised network this is reduced to under 1m for every platform.

The y axis position error in Figure 5.15 is actually very good for platforms 2 and 3 acting alone, but poor for platform 4. When configured in a DDF architecture, the use of global information results in the error at platform 4 being drastically reduced.

The z position error for the DDF architecture in Figure 5.16 results in a final error of approximately 0.2m. The information from platform 4 is highly important in this estimate as the final jump in this error plot occurs at time 170 seconds - the exact time that the target estimate on platform 4 changes when acting independently. Therefore, although the position error of the target on platform 4 is large in y, its information in the z axis is good.

The maximum, mean and minimum errors of a target over the entire duration of the simulation are shown in Figures 5.20–5.22. When there is no communication of target information, these values can be seen to vary significantly. However in the DDF network the errors are a function of global information and are, subject to communication delays, identical on all aircraft. The communication between platforms means that this common DDF error is a global minimum based on the information from all nodes in the system.

As the targets are known to be stationary, the IOU process model is tuned to decay velocity

to zero when predicting. Therefore, the errors in these states are essentially zero over the duration of the 'simulated' flight.

Figures 5.17–5.19 show the position variances over the period of the simulation. Inspection of these plots shows that the DDF network has a much lower variance than when the platforms act independently. This is to be expected as the extra information from other platforms helps to bound the growth of uncertainty. This is particularly important in this airborne environment as each aircraft is flying at approximately 100 knots (180km/hr) and is therefore only able to observe each target for a maximum of 2 seconds. By the time the aircraft continues on its orbit of the area it is not likely to revisit any target in less than 60 seconds. Even then, it was noted that targets were rarely seen on every orbit due to the difficulty the remote pilot had in following exactly the same flight path each circuit. This means that once a target is seen and the tracking filter updated, the filter must run on its process model for at least a minute, usually longer, unless information from another source is available. The target position variance plots for the individual platforms shown in Figures 5.17, 5.18 and 5.19 illustrate this as the variance can be seen to grow quite rapidly in the absence of any observation information. In the extreme case, the filters will become sufficiently uncertain that they will have to be discarded. The decentralised network on the other hand uses information from all platforms to continually bound the estimate to reasonable values. Therefore, not only is the physical error in the target location improved using the DDF architecture, but the estimate of the uncertainty in the filter is also reduced.

It is important to note that these results were obtained using real flight data from a vision sensor. The results highlight the effectiveness of the DDF architecture in improving the accuracy of target estimates. The final target position error for all platforms is significantly improved using the decentralised architecture. This follows logically as the use of more information in a filter should indeed allow for a better track estimate.

5.7 Real-Time Implementation

After initial testing using the ANSER simulator, the decentralised architecture was implemented in real-time on multiple airborne platforms. The system used the output of the real-time GPS/IMU navigation filter [58] to give the platform pose, and a combination of vision and SIS payloads to track the ground targets.

Section 5.7.1 presents the results of the initial real-time flight. While the results of this demonstration illustrated that the DDF algorithm communicated information robustly and correctly, it also highlighted some serious issues with the tracking algorithm and target registration. These issues were subsequently addressed and Sections 5.7.2 and 5.7.3 present results of the upgraded system.

5.7.1 Results of First Real-Time Flight Test - Multiple Aircraft DDF

Results are presented for a 2 aircraft, 4 node system running in real-time. Table 5.4 lists the different nodes used in this demonstration.The DDF network was configured as shown in Figure 5.23. A ground node with no sensor attached was connected to each aircraft in order to extend the size of the network and to show the full information states on a non-critical node. As the ground nodes have no sensor, they do not contribute any information to the network and therefore have exactly the same estimates as the aircraft to which they are connected. These ground nodes then output their tracking estimates to a graphical interface.

Node	Location	Sensor
1	Ground	None
2	Aircraft 1	Vision System
3	Aircraft 2	Vision System
4	Ground	None

Table 5.4: Configuration for the 4 node real-time DDF flight.

Both aircraft in this real-time demonstration use an identical vision system as their external sensing payload. The cameras process images at 25Hz, half their maximum rate, due to computational limitations. Each DDF node is limited in software to track a maximum of 17 targets. As the targets are known to be stationary, the IOU target process model is tuned to decay target velocity to zero.

The resulting maps of ground targets when running the DDF tracking algorithm in real-time are presented in Figures 5.24, 5.25 and 5.26. Figure 5.24 presents the complete map of targets from aircraft 1, along with the true target locations and the flight path of the aircraft. The area containing the majority of the targets has been enlarged in the lower plot. Similarly, Figure 5.25 contains the same information from the second aircraft. The results generated by the ground nodes are included in Figure 5.26.

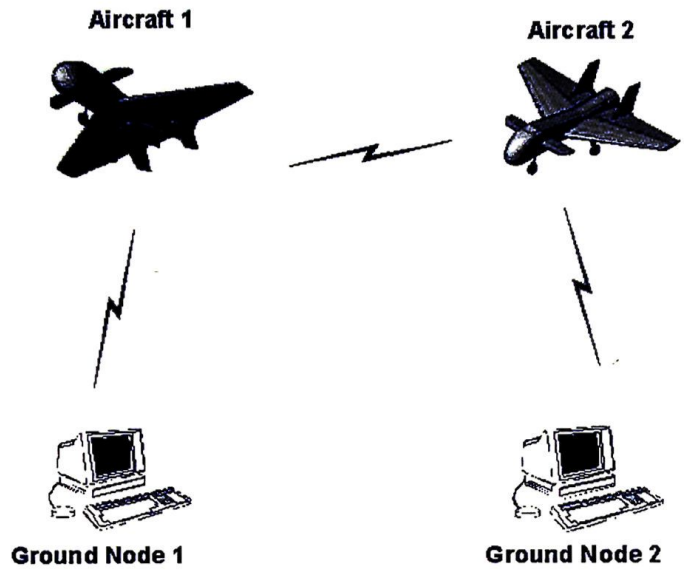


Figure 5.23: Peer-to-peer communications architecture used when flying 2 aircraft for real-time tracking. The aircraft communicate not only between themselves, but to DDF ground nodes as well. As these ground nodes have no sensor attached they do not contribure any information to the network. Their purpose is to provide information to the ground based GUI.

An inspection of the results illustrates that the maps on aircraft 1 and ground node 1 are both identical, as are the maps on aircraft 2 and ground node 2. This is the expected result as the DDF information from the respective aircraft was the only source on information for each of the ground nodes. However, there is a slight difference between the maps on the two aircraft. While the majority of targets in the maps are common, there are a small number of targets that aircraft 1 tracks that 2 does not, and vise versa. This result is also expected in a real system, and occurs due to the delay in transmitting target information between nodes. Each node will track the first 17 targets it encounters, whether they be targets the node observes directly or tracks received from other nodes via DDF information.

What occurs in practice is that each of the nodes observes targets directly and initiates tracks, then communicates these tracks to neighbouring nodes at the next communication step. If the neighbouring nodes have unused tracking filters they can allocate one of these and maintain a track estimate. This process works up until the point where each of the nodes has only a small number of filters free. When this happens, it is not unusual for each node to allocate its last remaining filters to targets it directly observes. If all nodes do this,

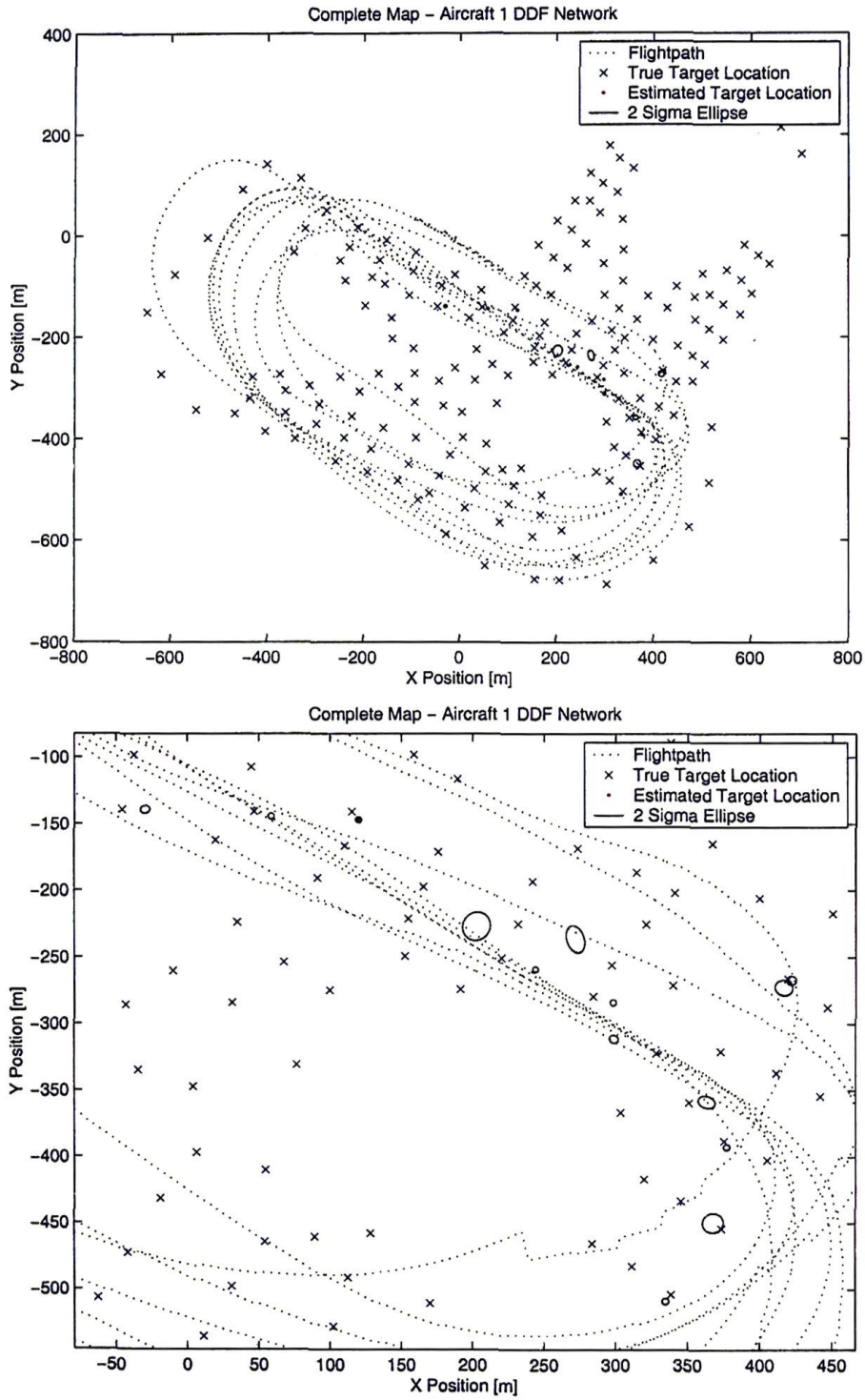


Figure 5.24: Real-time results: The map generated by aircraft 1 is shown with the true target locations and the platform flightpath. Note the jumps in the flightpath which are a result of poor GPS coverage during the flight. This has a significant impact on the registration of the target estimates, as they are generally offset from the true target locations.

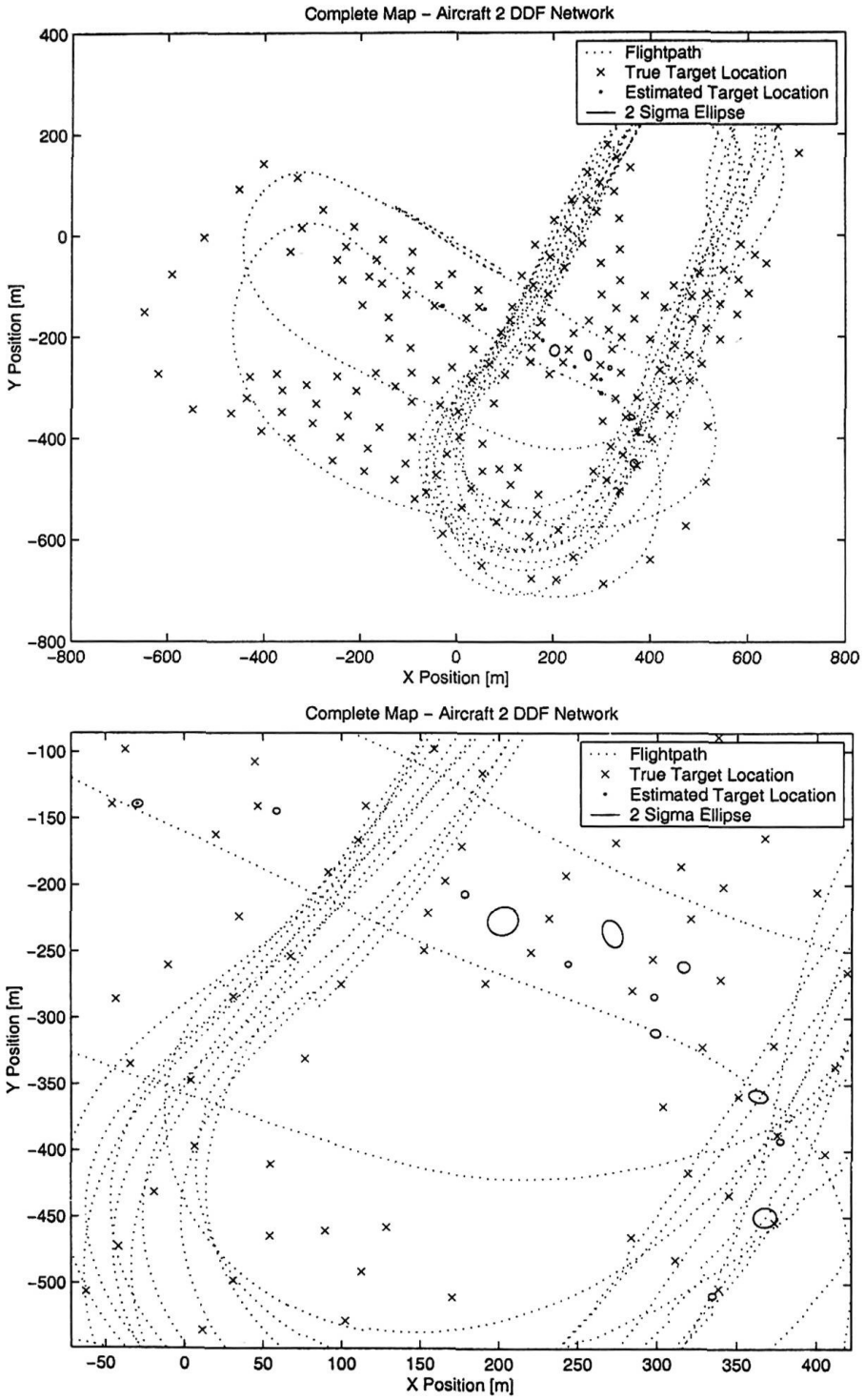


Figure 5.25: Real-time results: The map generated by aircraft 2 differs from that on aircraft 1 for only a few targets. This is an expected result which occurs due to limiting the number of tracks that a node can maintain. The poor GPS coverage during the flight affects the registration of target estimates at this node, as with aircraft 1.

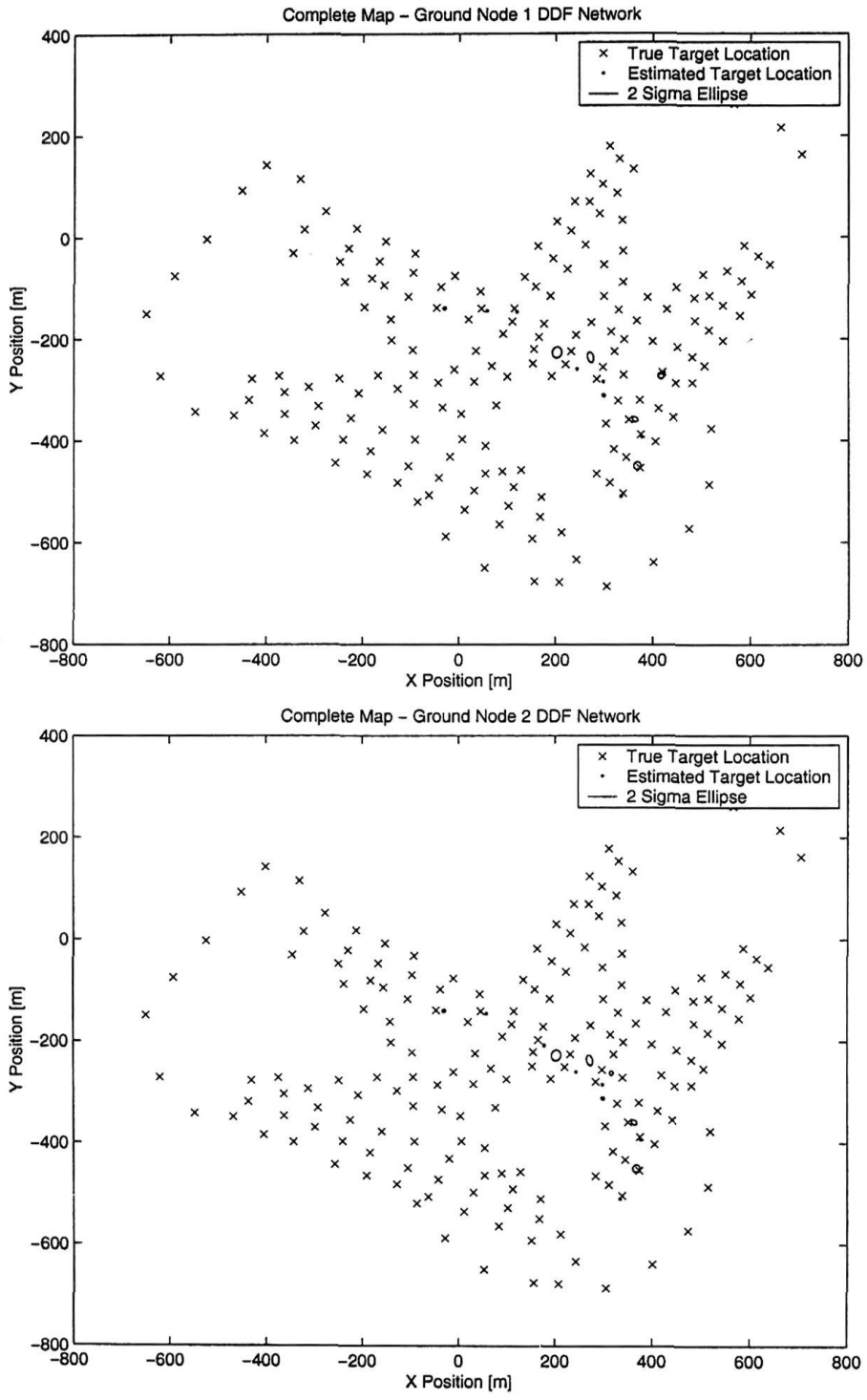


Figure 5.26: Real-time results: The maps generated on each of the ground nodes are exactly the same as those on the aircraft they were connected to in the DDF network. Ground node 1 matches aircraft 1 and ground node 2 matches aircraft 2.

there are no spare filters to use for estimates communicated from neighbouring nodes. This gives the result illustrated in this implementation where there are a small number of targets that are not common between connected nodes.

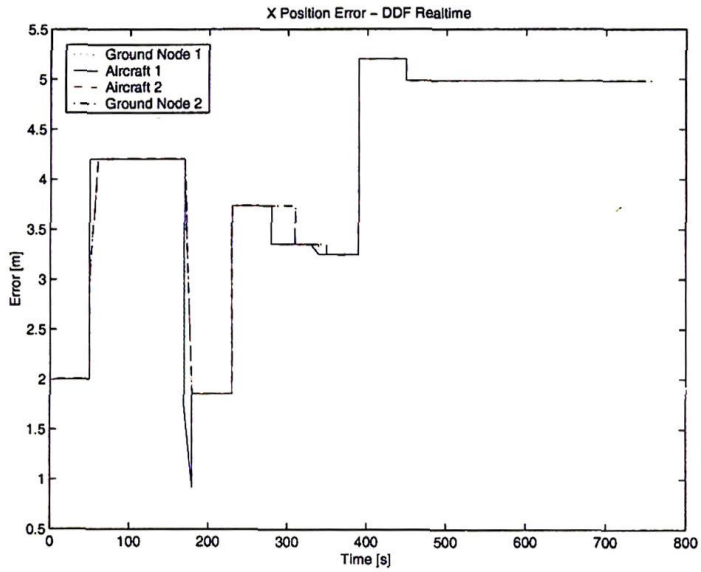


Figure 5.27: Real-time results: x axis position error for a typical target during real-time tracking. As the aircraft are operating in a DDF network, the estimates on each node are all identical.

As the target estimates are generated relative to the aircraft pose, an error in the location of the platform will result in a corresponding error in the target location. Over multiple passes of a target this error is assumed to be zero mean. However, as the real-time flight was only of a relatively short duration and not all targets were observed on each circuit, the errors in the estimated platform location did result in an offset from the true target location. This can be seen in the results as the 2σ ellipses do not encapsulate the true target locations. The effect was amplified during the flight as there was poor GPS coverage at the time, which degraded the estimate of platform pose significantly. The estimated flightpath of aircraft 1, illustrated in Figure 5.24, shows jumps of up to $10m$ occur periodically. The errors in the height estimate of the platform were significantly worse than this, and were often in the region of $10 - 20m$. As a result, there are some estimates which sit midway between two real targets as the platform uncertainty was sufficiently large to allow two different targets to be associated with a single track. This issue was addressed in later flights by improving the GPS/IMU navigation loop, decreasing the target density and by

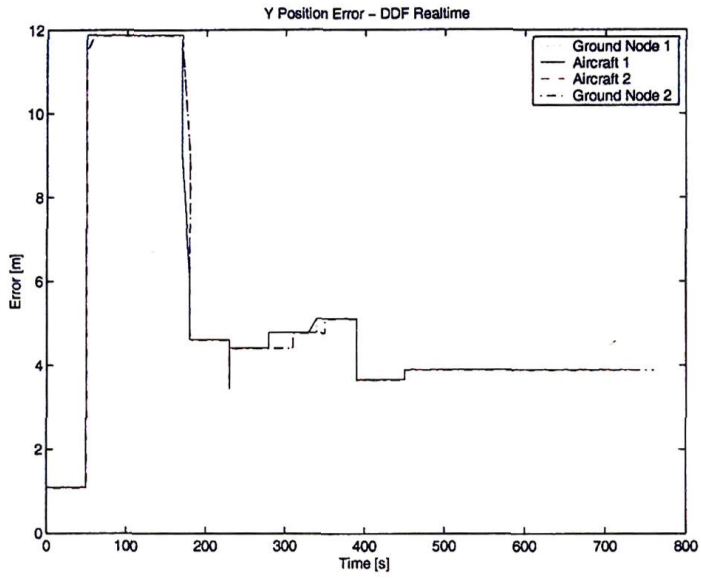


Figure 5.28: Real-time results: y axis position error for a typical target during real-time tracking. As the camera is pointing straight down, there is a large amount of information in horizontal plane (x-y axes) which in turn results in accurate estimates.

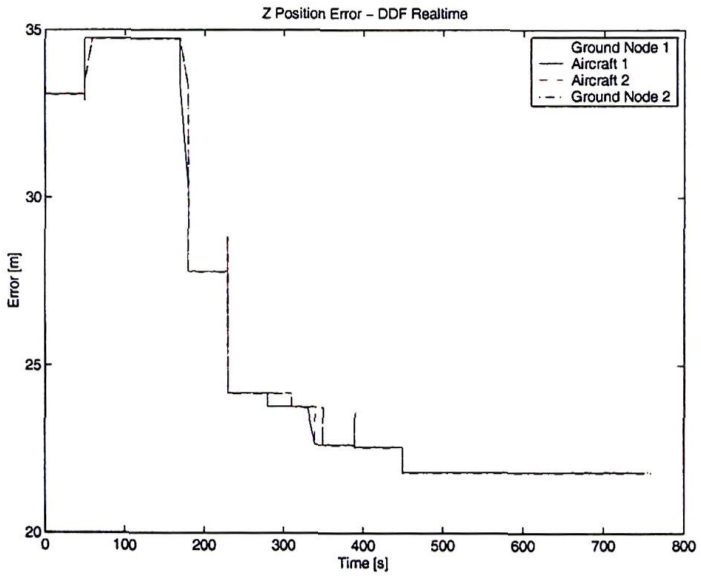


Figure 5.29: Real-time results: z axis position error for a typical target during real-time tracking. Due to poor GPS coverage at the time of the flight, the aircraft pose had significant errors, particularly in the z axis. This error, combined with the vision sensors intrinsicly poor range estimate combine to give a large error in the z position of the target.

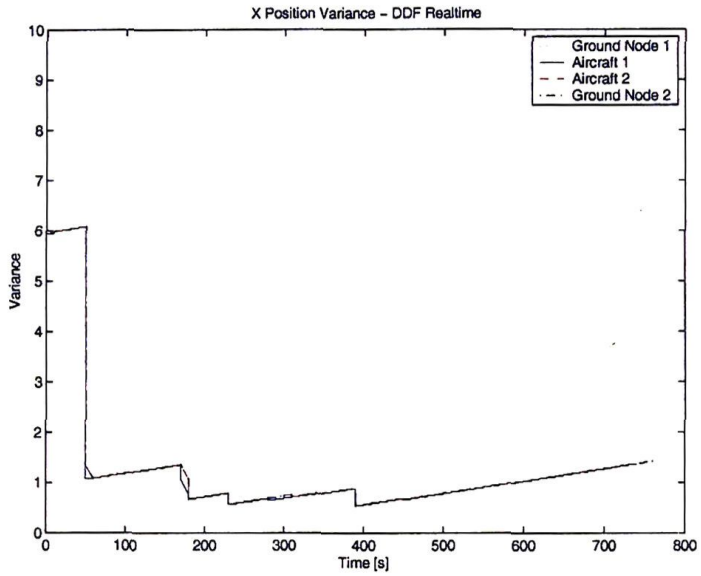


Figure 5.30: Real-time results: The x axis position variance for a typical target during real-time tracking is quite low as the platforms are sharing their information. With the camera mounted in the aircraft pointing directly down, the x axis is also in a plane which is directly observable.

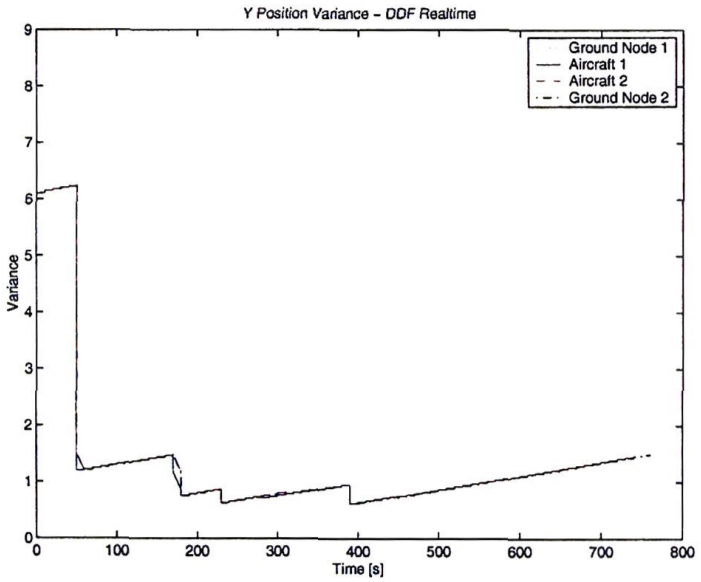


Figure 5.31: Real-time results: The y axis position variance for a typical target during real-time tracking exhibits the same properties as the x axis error. As it is in the image plane, observations of the target give good information in this axis.

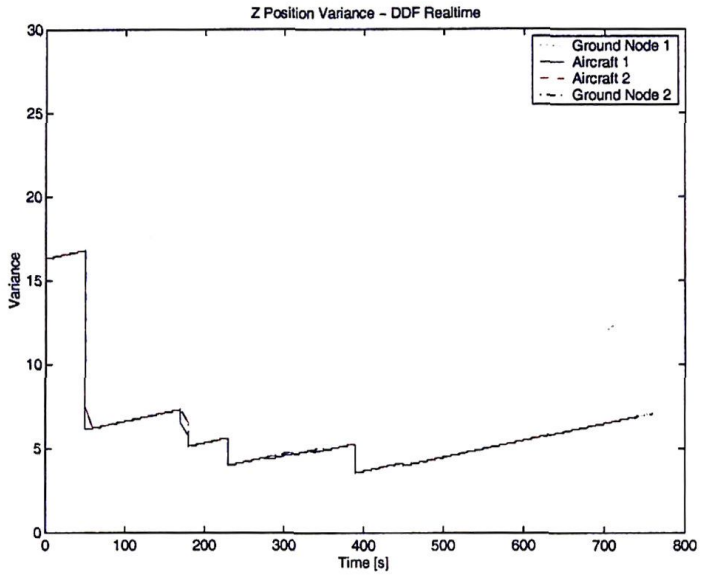


Figure 5.32: Real-time results: z axis position variance for a typical target during real-time tracking. Although the variance in this axis is greater than the others, it is not as high as it should be as there were large errors on the platform height when observations were made.

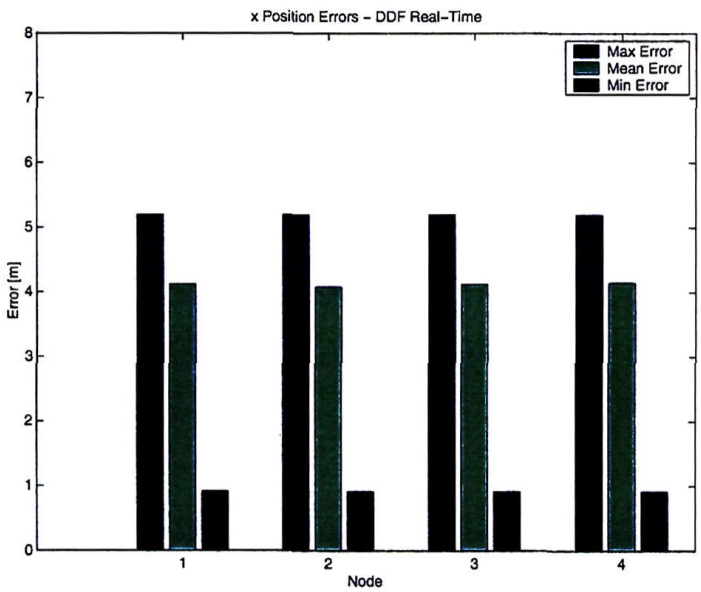


Figure 5.33: Real-time results: The maximum, mean and minimum errors in x for a typical target over the entire flight duration.

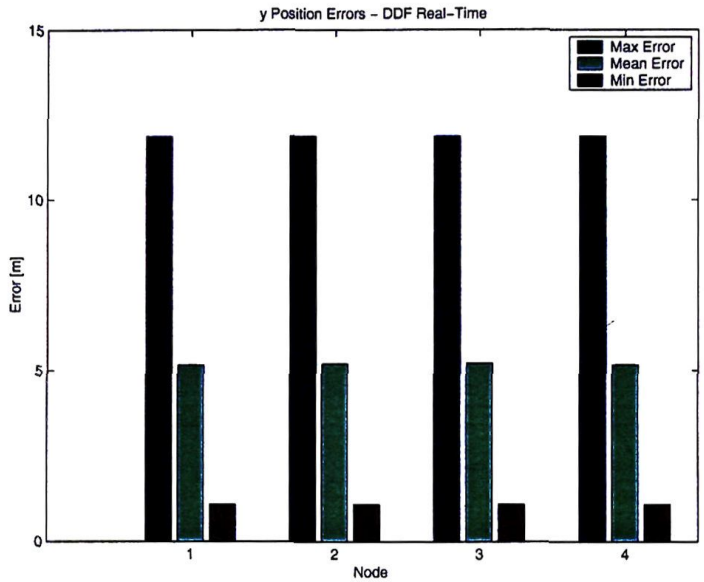


Figure 5.34: Real-time results: The maximum, mean and minimum errors in y for a typical target over the entire flight duration.

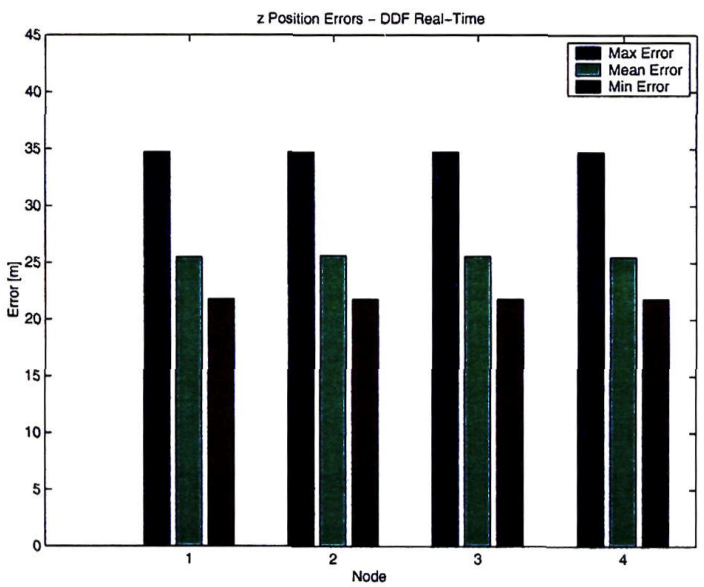


Figure 5.35: Real-time results: The maximum, mean and minimum errors in z for a typical target over the entire flight duration.

making the data association more stringent. Notwithstanding these problems, the majority of estimates were still within $10m$ of the true location.

Although the target estimates are close to the true target locations, the 2σ covariance ellipses are clearly overly confident. The primary reason for this is that successive observations of a target are correlated through the vehicle pose estimate which is used to register the observation, and this is not accounted for.

As the focus of this initial demonstration was on the DDF architecture, there was no track-to-track fusion implemented. This can be seen in the results from aircraft 1 (Figure 5.24) as there are two tracks for one of the targets. While this does result in a poor allocation of resources, it does not hinder the demonstration of the DDF technology. As the ANSER project evolved, the inclusion of a simple track-to-track fusion algorithm did occur.

The detailed tracking results for a single target over the duration of the flight are shown in Figures 5.27–5.35. These plots show the position error and variance of a typical target estimate on all nodes over the entire period of the flight. Note that as the decentralised network shares all information between all nodes, the plots are almost on top of one another. Any period where the plots are not exactly the same is a result of a temporary communication failure between nodes. The robustness of the decentralised architecture allows the network to realign the estimates in all nodes when communications are restored.

The position error in x and y shown in Figures 5.27 and 5.28 illustrate the logical result that the accuracy of the track in the camera plane is quite good. Figures 5.33 and 5.34 illustrate that the mean error in these axes over the entire flight was approximately $5m$. The results in height shown in Figures 5.29 and 5.35 are not as accurate. There are two main factors contributing to this error. First, the only range information being extracted from the vision sensor is based on the size of the target. This gives extremely poor quality information and requires a large number of observations to converge to an accurate estimate. Second, at the time of the experiments the GPS satellite coverage was quite poor, with only 6 satellites in view. When the aircraft banked it often lost sight of some of these satellites which degraded the platform height estimate, and subsequently the estimated height of the targets tracked by the platforms. This highlights the importance of an accurate estimate of the platform pose in picture compilation as any error here will result in an error in the target location. In practice this error in platform pose is assumed to be a zero mean Gaussian over multiple passes of the target. The more often the target is visited, the more accurate this assumption

and the better the resulting estimate.

As the IOU process model was tuned with knowledge that the targets are stationary, the velocity errors and variances are both near zero throughout the duration of the flight.

The decentralised architecture can be seen to communicate effectively as the real-time results presented in this section are the same for all nodes. However, the registration of targets and overly confident estimates were a serious problem and were addressed in later flights.

5.7.2 Results of Second Real-Time Flight Test - Single Vehicle DDF

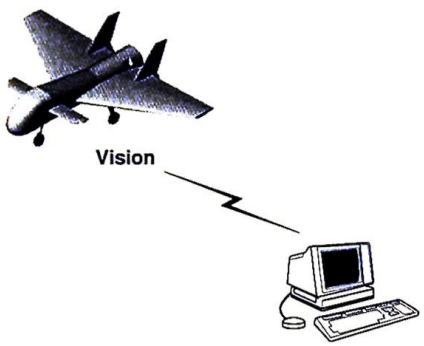


Figure 5.36: 1 aircraft, 2 node DDF network.

Following the initial real-time flight, the DDF algorithm was modified to address the issues that arose. These changes included:

- The GPS/IMU localisation filter was improved, ensuring the relative observations were registered with greater accuracy.
- The GPS coverage at the time of this flight was sufficient to ensure that the aircraft was able to maintain a differential RTK solution.
- The target density was reduced to allow the data association algorithm to uniquely identify targets.
- The nodal filter used the CI update to fuse observations from the locally attached sensor.

Of these changes, the first three contribute primarily to ensuring the estimated target location is more accurate, while the fourth ensures that the estimate covariance is consistent.

The first flight after making these changes involved a single aircraft with a vision node connected to a ground node. Table 5.5 lists the different node types used. Observations from the vision sensor were converted to their information form using the equations given in Section 5.5.2, and fused in the nodal filter with the CI update maximising the determinate of the information matrix (equivalent to minimising the determinate of the covariance). The DDF information was then communicated to the ground node where it was handled by a channel filter using the full information update.

Node	Location	Sensor
1	Ground	None
2	Aircraft 1	Vision System

Table 5.5: Configuration for the 2 node real-time DDF flight.

The results of this real-time demonstration are illustrated in Figure 5.37. The vision node was limited to tracking 15 targets for this flight as it was using the 266MHz CPU configuration.

The map generated on both the ground node and the vision node are identical as a result of communicating DDF information, as was the case in the initial real-time demonstrations described in Section 5.7.1. However, unlike the earlier results, the estimates are consistent with the 2σ ellipses clearly encapsulating the true target location.

It is also worth noting that these results were obtained with the aircraft flying autonomously rather than being remotely flown. The circuits are therefore largely on top of one another as the flight controller was able to follow the desired flightpath significantly better than a remote pilot. This ensured that the same targets were repeatedly observed on every circuit.

5.7.3 Results of Third Real-Time Flight Test - Multiple Vehicle DDF

Following the successful single vehicle real-time demonstration in Section 5.7.2, the DDF algorithm was flown on a 5 node, 2 vehicle system. Table 5.6 lists the different nodes used in this demonstration. This network, illustrated in Figure 5.38, has one aircraft with both vision and SIS nodes, the other with a vision node only, and two ground nodes with no sensors attached.

The results of this flight test are illustrated in Figures 5.39, 5.40 and 5.41. All 5 nodes in this network have common target estimates, as a result of the DDF information. As

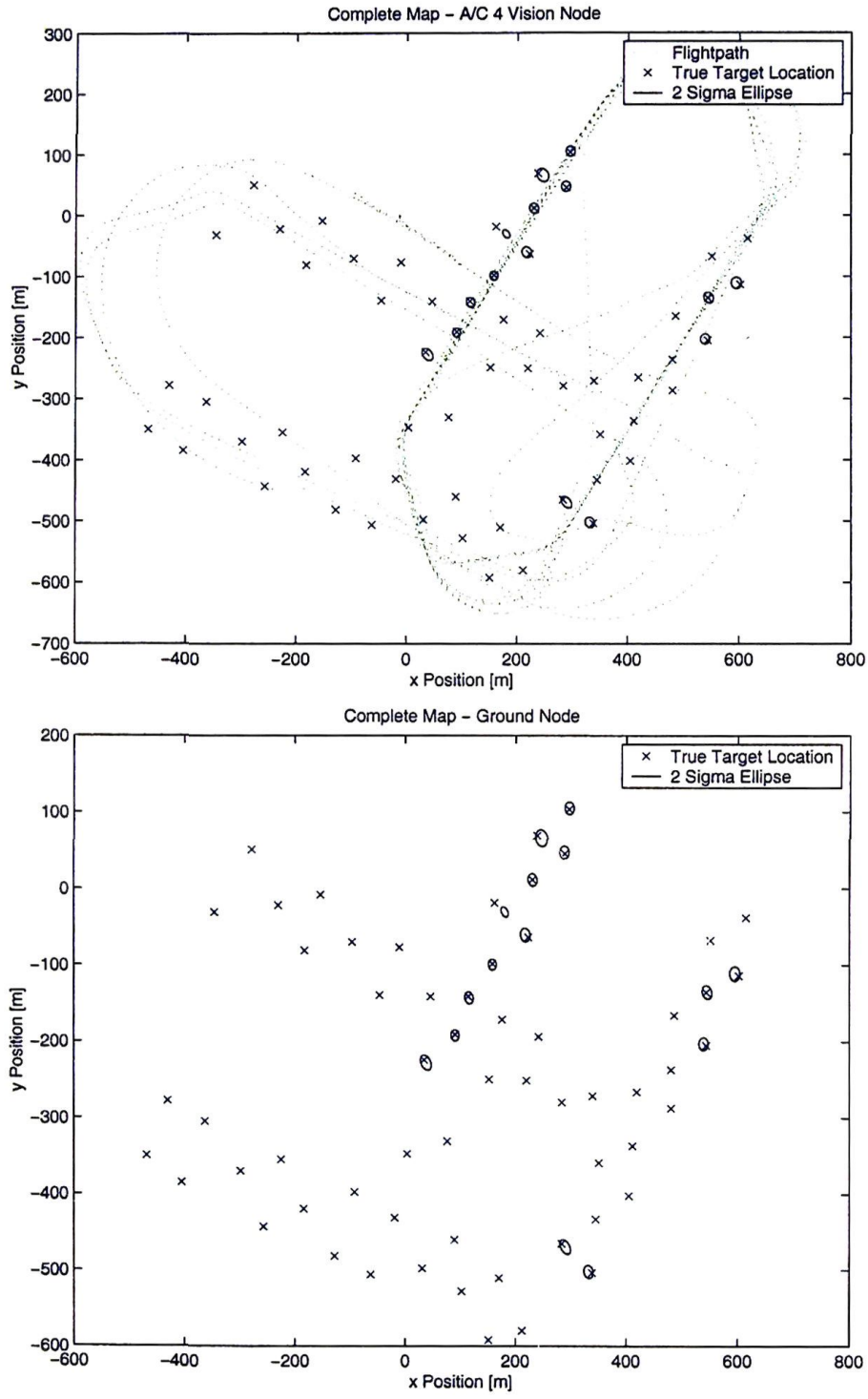


Figure 5.37: Real-time results: Map generated by airborne vision node and the ground node it was connected to.

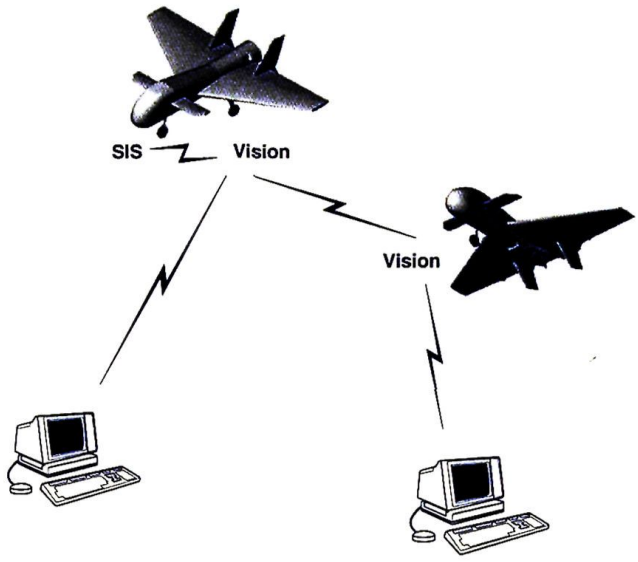


Figure 5.38: 2 aircraft, 5 node DDF network.

explained in Section 5.7.1, there are a small number of targets that are tracked on one node but not another as a result of the timing in how filters are allocated.

Node	Location	Sensor
1	Ground	None
2	Aircraft 1	Vision System
3	Aircraft 1	SIS
4	Aircraft 2	Vision System
5	Ground	None

Table 5.6: Configuration for the 5 node real-time DDF flight.

The results of this flight were obtained using the CI update on observation information at the nodal filter, as well as a CI update on DDF information between sensors on the same platform. As explained in Section 5.5.7, this is necessary as the sensors on the same platform are also correlated through the vehicle pose in the same way as consecutive observations from the same sensor. A standard information filter update is performed on all DDF data to other vehicles and to the ground station.

The results in the maps of Figures 5.39, 5.40 and 5.41 demonstrate that the algorithm worked successfully. All nodes have a common global tactical picture of the targets. Furthermore, the target estimates in these maps are clearly consistent as the 2σ error bounds

enclose the true target location. The few estimates plotted on the map that do not correspond to a surveyed target were all investigated post-flight and found to be natural features. These include a number of wombat holes, a rock outcrop and the roof of a small shed.

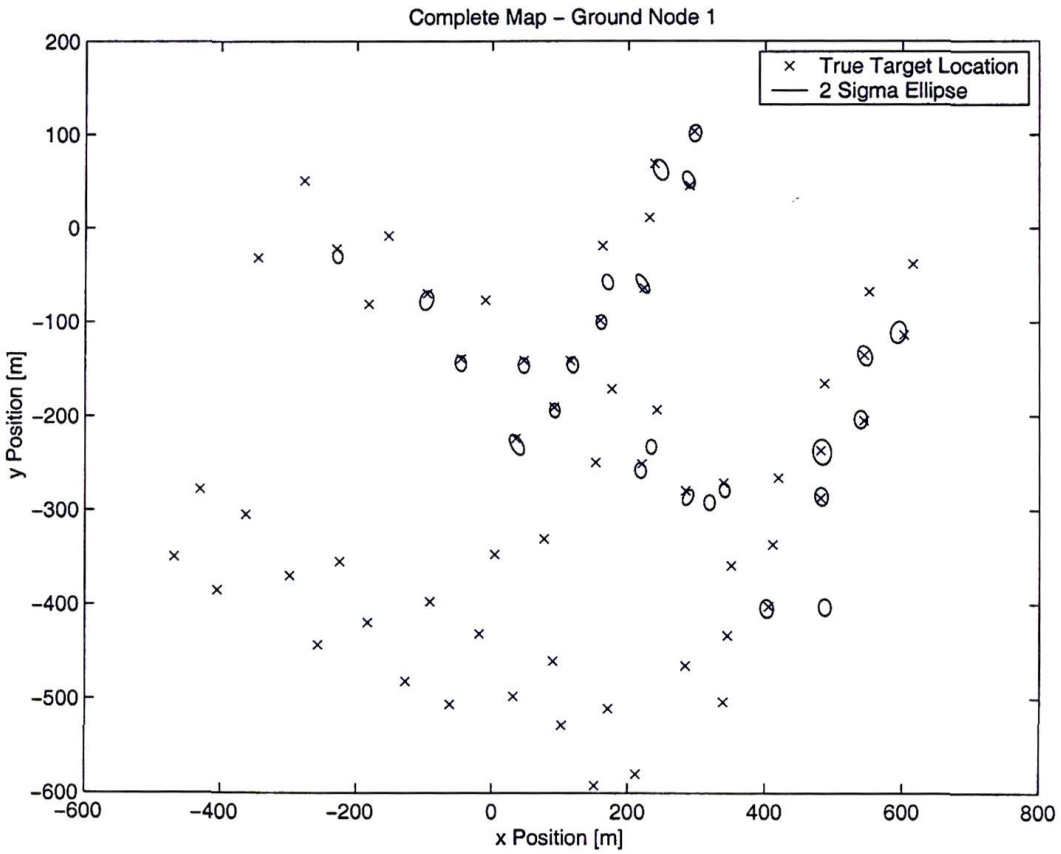


Figure 5.39: Real-time results: Map at ground node connected to aircraft 3.

5.8 Summary

This Chapter has presented the formulation of a decentralised multi-target tracking system using the architecture of Chapter 3. The problem involves using multiple aircraft to track ground based targets to build a composite picture of an environment.

The decentralised tracking algorithm was initially demonstrated using the ANSER simulator in playback mode with logged flight data from a vision sensor. These results were used to

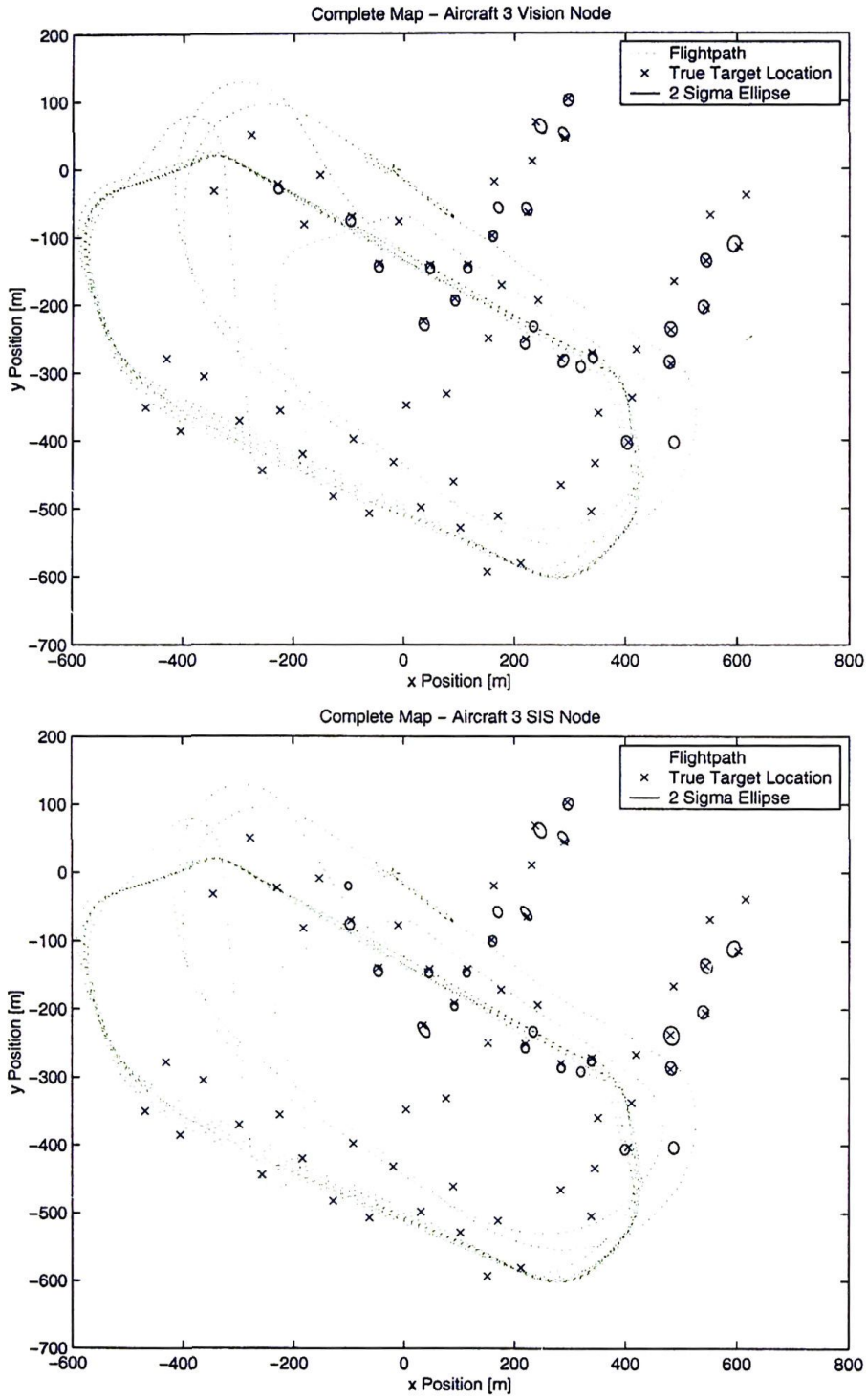


Figure 5.40: Real-time results: Map generated by airborne vision and sis nodes on aircraft 3.

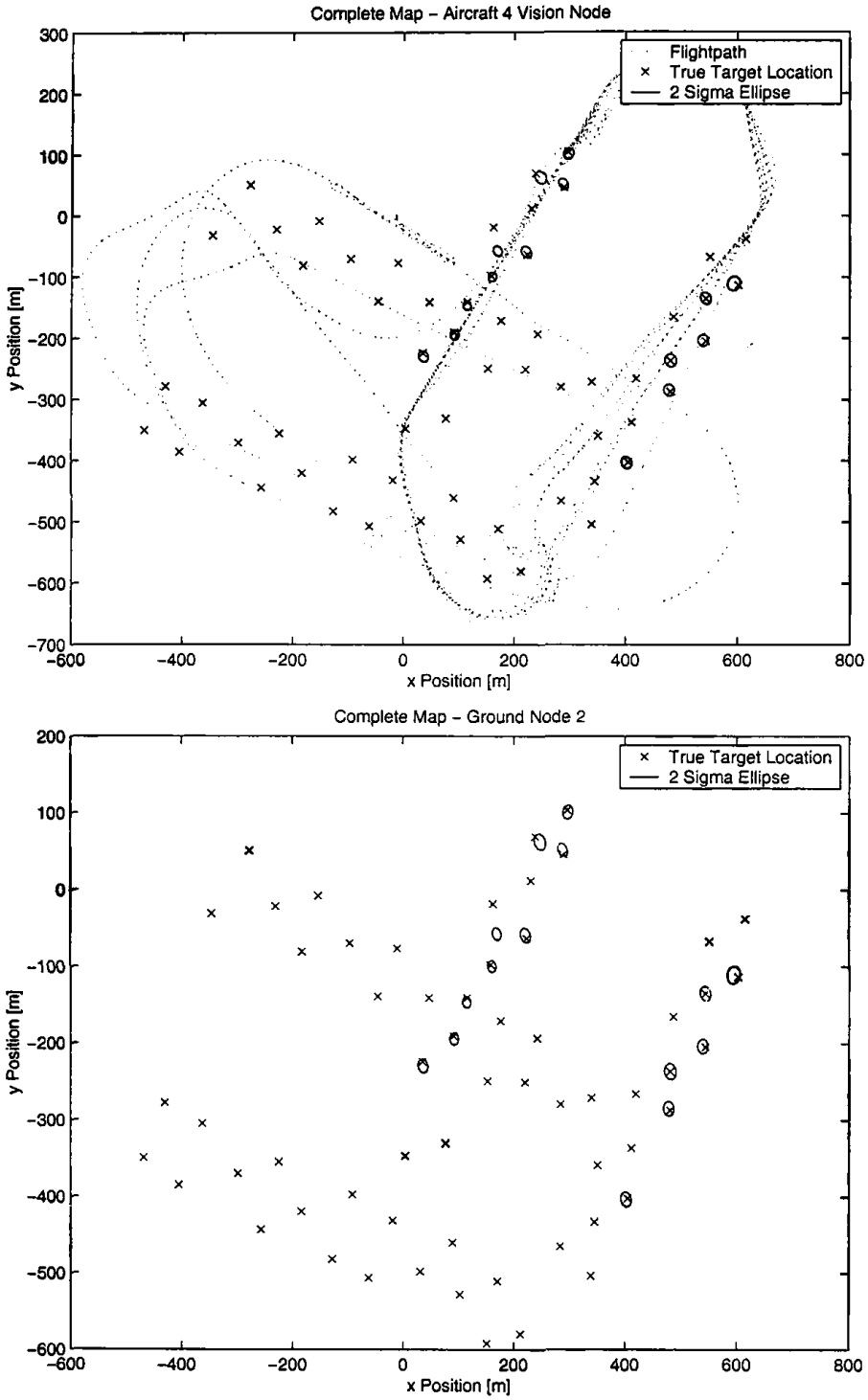


Figure 5.41: Real-time results: Map generated by airborne vision and ground nodes of aircraft 4.

validate the algorithms under controlled conditions and are included to highlight the key advantages of the decentralised architecture over a system which does not share information.

Results are also presented of real-time demonstrations of the decentralised tracking system on multiple UAVs. Although information was propagated successfully in the initial real-time demonstration, the registration of target locations was very poor and the resulting target estimates inconsistent. However, this issue was then addressed by improving the vehicle localisation filter, decreasing the target density and by allowing for the correlation between successive observations from the same sensor. Following these improvements, subsequent real-time DDF flights were highly successful and provided accurate and consistent target maps on all nodes.

Chapter 6

Simultaneous Localisation and Map Building

6.1 Introduction

This Chapter develops the Simultaneous Localisation and Mapping Building (SLAM) algorithm in a decentralised multi-platform environment. SLAM is a method for building a navigation map of an environment from a moving platform while at the same time estimating the location and trajectory of this platform. Essentially it provides a navigation solution for a vehicle without access to external map, beacon or landmark information. While the single platform SLAM algorithm has received considerable attention in the past five years [23, 59, 63], the multiple platform problem has only recently been addressed and this Chapter represents one of the first developments in this area. The Decentralised SLAM problem, D-SLAM, has a number of additional features which make it much more than simply a novel application. In particular it shows how the information filter concepts developed for the DDF algorithms find a natural application in map building. Practically, it also permits a scalable number of platforms to build, fuse and share a common map.

Section 6.2 describes the SLAM problem for the single vehicle case. The augmented state vector, vehicle model, observation model, and state and information space feature initialisation processes are described in Sections 6.2.1, 6.2.2, 6.2.3, 6.2.4 and 6.2.5 respectively. These models are used throughout this Chapter and are applied in simulation to demonstrate the essential SLAM and later D-SLAM algorithms.

Section 6.3 presents closed form solutions to the continuous time single degree of freedom SLAM problem both in state-space and information-space form. These closed-form solutions are a substantial aid in understanding the performance of the SLAM algorithm and the efficacy of the information form in multiple platform algorithms. These solutions are obtained by solving the continuous time Riccati equation in both its information and state space forms using the method presented in Section 6.3.2. The multiple platform problem is initially presented in state space in Section 6.3.3 before Sections 6.3.4 and 6.3.5 present the information form for single and multiple platforms respectively.

One of the key results from the closed form solutions of D-SLAM is that the total map in information space is equal to the sum of the maps on each of the individual platforms. Therefore, it is only the map information that platforms need to communicate to each other in the D-SLAM algorithm. Section 6.4 considers this map information and how it can be extracted for communication.

Section 6.5 presents algorithms for D-SLAM. The algorithm is initially presented in its information form in Section 6.5.1. This formulation makes use of the fact that vehicles only need to communicate map information in order to solve the multiple vehicle SLAM problem. Using the vehicle and observation models generated in Sections 6.2.2 and 6.2.3, the operation of the prediction and update stages of the filter are discussed. The update with map information from other vehicles, which is the critical part of the D-SLAM algorithm, is also presented.

Although the formulation of the D-SLAM problem in information space is more natural for communication between vehicles, it requires significant computation to decouple the vehicle state estimate from the map when the vehicle is described by a non-linear motion model. For this reason, a hybrid state/information space formulation of the D-SLAM algorithm is presented in Section 6.5.2. This formulation is numerically identical to the information space algorithm of Section 6.5.1, but maintains the vehicle and map estimates in state space form while leaving the channel filter, through which map estimates are communicated, in information space. As inter-vehicle communication will usually occur at a significantly slower rate than the local prediction stage of the SLAM filter, this formulation requires less computation than the information space formulation. A discussion of the two D-SLAM formulations is given in Section 6.5.3.

In order for D-SLAM to work in any large scale problem, it is necessary to define a com-

munications scheme which is flexible to the size of maps transmitted between platforms. Section 6.6 discusses this issue and presents a constant time communication scheme which uses a hybrid information filter/covariance intersect algorithm. Using this method, vehicles can form a sub-map of an arbitrary size which, when communicate to other nodes, guarantees the system remains scalable. Section 6.6.1 shows how the sub-map is extracted. Sections 6.6.2 and 6.6.3 respectively describe the channel filter update and the method of updating the SLAM filter with information from other nodes.

Section 6.7 describes a simulation of the different D-SLAM algorithms presented in this Chapter. The simulation compares three different DDF communication methods to demonstrate their performance. The methods compared were:

1. No communication between platforms. The platform simulations run a SLAM filter using only locally generated information.

2. Complete communication. The complete map of N^2 elements is communicated between platforms every 10 seconds.

3. Constant time communications strategy. A sub-map of at most 5 features is communicated every 10 seconds. The features selected for communication are those with the greatest information gain that had not yet been transmitted.

The results of these simulations are presented in Section 6.7.1. Section 6.8 concludes this Chapter with a summary of the key issues in D-SLAM.

6.2 Problem Definition

The SLAM algorithm involves a platform building a map of an environment while simultaneously using this map to localise. Figure 6.1 shows this process graphically. A platform uses some sensor to make relative observations to stationary features or landmarks in the environment. Once observed, the features are augmented with the vehicle states into a single combined state vector. As these observations are relative to the vehicle, an error in the platform pose results in a corresponding error in the landmark location. This is clearly shown in the figure, where the target is offset from its true location by the same amount as the vehicle error. Over time, the entire map becomes correlated through the motion of the

vehicle. This and other convergence properties of the SLAM algorithm were investigated by Newman [80], who proved the following:

1. The determinate of any submatrix of the map covariance matrix decreases monotonically with every observation.
2. In the limit, as the number of observations increases, the map becomes fully correlated.
3. The lower limit of accuracy of the map is given by the vehicle uncertainty when the first landmark is observed.

These proofs were a significant advance in the understanding of the SLAM problem as they show that the algorithm will force convergence of the map error to a fixed lower bound.

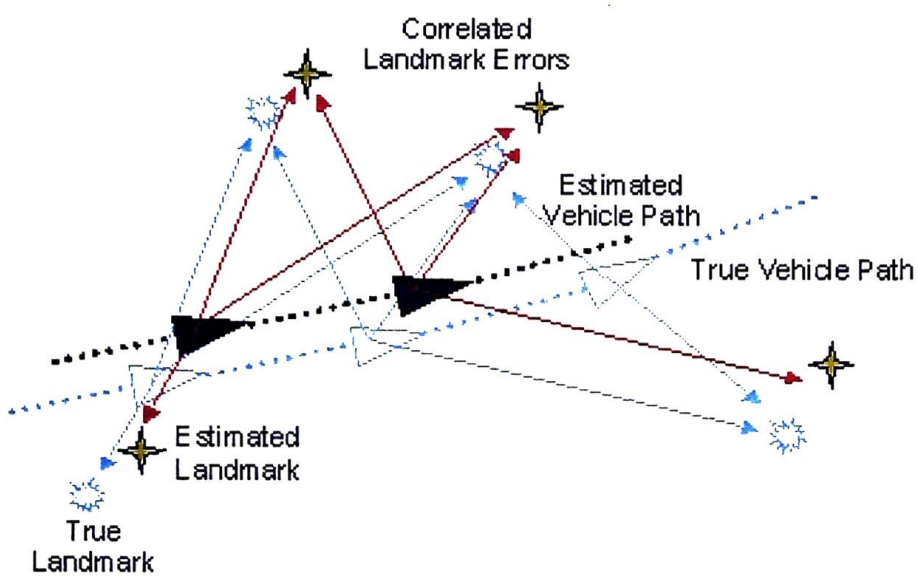


Figure 6.1: The SLAM problem.

A large amount of research has been undertaken in the SLAM problem in recent years [61, 90, 92]. There has been particular interest and progress in areas such as large scale mapping problems [54, 60], computational efficiency [62, 73] and convergence properties [35, 39, 80]. However, to date, this research has concentrated on using conventional state space formulations of the SLAM algorithm.

In this thesis, the SLAM problem is cast in an information space form. This enables the SLAM algorithm to exploit the properties of the information filter in being decentralised and allows the multiple platform SLAM problem to be developed in a fully decentralised, information theoretic form.

The benefits of decentralising SLAM are potentially enormous. For a single platform using a SLAM algorithm to decrease its location covariance, it must revisit features it has seen previously. However, with multiple platforms it is possible to share information and so generate a more accurate map and location estimate. Indeed, if certain platforms are directed to revisit shared features, then there will be significant location estimate improvements for all platforms.

Solutions to the multi-vehicle SLAM problem have been investigated in a number of different forms. Thrun [95] demonstrated a robust implementation for cyclic environments using a combination of maximum likelihood and posterior estimation techniques on both single and multiple mobile robots. However, the multiple robot version of this algorithm is restrictive in that it requires one robot to be a "team leader" to initialise all other platforms.

A Kalman filtering approach was taken by Williams [103] who generated local submaps which were uncorrellated with the global map on each vehicle. These local submaps could then be transmitted to other vehicles and periodically fused into a local global map. While this approach manages the amount of communications very well (the size of the submap is not fixed), it is not robust to communications failure. The submaps are essentially an increment of information, and if one of these messages is corrupted or not received, the information it contains is not encapsulated in any future transmissions.

When formulated in information space it was shown in [79] that the global map with multiple platforms is simply the sum of the map information at each platform. This follows logically from the update stage of the information filter which is additive and the fact that the process noise acts only on the vehicle. Further research into information space SLAM can be found in [32, 34, 78, 96].

The communications management scheme presented in this thisis uses a hybrid Covariance Intersect/Information filter algorithm to manage the complexity of large maps. Julier [54] also used Covariance Intersection for large scale SLAM, and in the process noted that the algorithm could also be used for the multiple vehicle problem. However the communications

management algorithm presented here differs in that it utilises the channel filter as a means of reducing the information loss in a conservative CI update.

When implemented using a Kalman filter, the SLAM algorithm operates using a recursive prediction and update cycle. The prediction stage propagates the platform estimate and covariance over time, and is updated when observations to landmarks are made. When considered in its equivalent information form, the SLAM filter again uses the recursive prediction and cycle, but maintains the estimate in terms of information. The prediction cycle serves to decrease the amount of information about the platform position, while observations add new information and hence increase the certainty of the estimate.

6.2.1 The Augmented State

Consider the problem of estimating the position and orientation of a platform in two degrees of freedom. The state vector can be written

$$\mathbf{x}_v(k) = \begin{bmatrix} x_v(k), & y_v(k), & \psi_v(k) \end{bmatrix}^T. \tag{6.1}$$

The platform now moves through an environment observing stationary landmarks which are augmented into a combined state vector with these vehicle states. Assuming the platform operates in a plane and the landmarks are point features, the landmark states are their position

$$\mathbf{x}_f(k) = \begin{bmatrix} x_f, & y_f \end{bmatrix}^T. \tag{6.2}$$

Augmenting these with the vehicle states results in a combined state vector of the form

$$\mathbf{x}(k) = \begin{bmatrix} x_v(k), & y_v(k), & \psi_v(k), & x_{f1}(k), & y_{f1}(k), & \cdots & x_{fn}(k), & y_{fn}(k) \end{bmatrix}^T. \tag{6.3}$$

As the combined state vector in Equation 6.3 is quite cumbersome, it is abbreviated to

$$\mathbf{x}(k) = \begin{bmatrix} \mathbf{x}_v(k), & \mathbf{x}_m(k) \end{bmatrix}^T \tag{6.4}$$

where $\mathbf{x}_v(k)$ corresponds to the vehicle states and $\mathbf{x}_m(k)$ a combined vector of all landmarks. The subscript m is used as collectively, the landmarks are referred to as a map.

When formulated using the Kalman filter, the estimate and covariance of the combined state vector are defined using the standard definitions given in Equation 2.11. In discrete time notation, these are written as

$$\hat{\mathbf{x}}(k \mid k) = \begin{bmatrix} \hat{\mathbf{x}}_v(k \mid k) \\ \hat{\mathbf{x}}_m(k \mid k) \end{bmatrix} \tag{6.5}$$

$$\mathbf{P}(k \mid k) = \begin{bmatrix} \mathbf{P}_{vv}(k \mid k) & \mathbf{P}_{vm}(k \mid k) \\ \mathbf{P}_{vm}^T(k \mid k) & \mathbf{P}_{mm}(k \mid k) \end{bmatrix}. \tag{6.6}$$

6.2.2 Process Model

The process model describes how the states evolve over time. In the work described here, the platform is assumed to operate at constant (or at least known) altitude. The platform model is therefore two dimensional. The landmarks are considered stationary and no noise is added to their location. The process model for this system is therefore written in the form

$$\begin{bmatrix} \mathbf{x}_v(k) \\ \mathbf{x}_m(k) \end{bmatrix} = \begin{bmatrix} \mathbf{f}\left(\mathbf{x}_v(k-1), \mathbf{u}(k)\right) \\ \mathbf{x}_m(k-1) \end{bmatrix} + \begin{bmatrix} \mathbf{w}_v(k) \\ \mathbf{0} \end{bmatrix}, \tag{6.7}$$

where the platform model is described by the equations

$$\begin{bmatrix} x_v(k) \\ y_v(k) \\ \psi_v(k) \end{bmatrix} = \begin{bmatrix} x_v(k-1) + \Delta t V(k) \cos\left(\psi_v(k-1) + \gamma(k)\right) \\ y_v(k-1) + \Delta t V(k) \sin\left(\psi_v(k-1) + \gamma(k)\right) \\ \psi_v(k-1) + \Delta t V(k) \frac{\sin(\gamma(k))}{B} \end{bmatrix} + \begin{bmatrix} w_x(k) \\ w_y(k) \\ w_\psi(k) \end{bmatrix}, \tag{6.8}$$

and $V(k)$ and $\gamma(k)$ are the control inputs of velocity and turning angle respectively and B is the minimum turning radius of the platform. Equation 6.8 propagates the vehicle at time $k-1$ to time k.

Consider now the prediction of the covariance matrix from $k-1$ to k. As discussed in Section 2.3.3, the EKF linearises the process model around the current state to propagate the covariance matrix. The Jacobians $\nabla_{\mathbf{x}}\mathbf{f}(k)$ and $\nabla_{\mathbf{u}}\mathbf{f}(k)$ of $\mathbf{f}$, taken with respect to $\mathbf{x}(k-1)$ and the control inputs $\mathbf{u}(k)$ respectively and evaluated at $\mathbf{x}(k-1) = \hat{\mathbf{x}}(k-1 \mid k-1)$, are

given by

$$\nabla_{\mathbf{x}}\mathbf{f}(k) = \begin{bmatrix} 1 & 0 & -\Delta tV(k)\sin(\hat{\psi}_v(k-1 \mid k-1)+\gamma(k)) & \mathbf{0} \\ 0 & 1 & \Delta tV(k)\cos(\hat{\psi}_v(k-1 \mid k-1)+\gamma(k)) & \mathbf{0} \\ 0 & 0 & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_m \end{bmatrix} \tag{6.9}$$

$$\nabla_{\mathbf{u}}\mathbf{f}(k) = \begin{bmatrix} \Delta t\cos(\hat{\psi}_v(k-1 \mid k-1)+\gamma(k)) & -\Delta tV(k)\sin(\hat{\psi}_v(k-1 \mid k-1)+\gamma(k)) \\ \Delta t\sin(\hat{\psi}_v(k-1 \mid k-1)+\gamma(k)) & \Delta tV(k)\cos(\hat{\psi}_v(k-1 \mid k-1)+\gamma(k)) \\ \frac{\Delta t\sin(\gamma(k))}{B} & \frac{\Delta tV(k)\cos(\hat{\psi}_v(k-1 \mid k-1))}{B} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{6.10}$$

As the map features are stationary, the identity $\mathbf{I}_m$ is used over all map features in Equation 6.9. The platform to map terms in both Equation 6.9 and 6.10 are filled using a vector of zeros ($\mathbf{0}$).

The prediction equations for the EKF can now be written in the form

$$\hat{\mathbf{x}}(k \mid k-1) = \mathbf{f}(\hat{\mathbf{x}}(k-1 \mid k-1), \mathbf{u}(k)) \tag{6.11}$$

$$\begin{aligned} \mathbf{P}(k \mid k-1) &= \nabla_{\mathbf{x}}\mathbf{f}(k)\mathbf{P}(k-1 \mid k-1)\nabla_{\mathbf{x}}^{\mathbf{T}}\mathbf{f}(k) + \nabla_{\mathbf{u}}\mathbf{f}(k)\mathbf{U}_k\nabla_{\mathbf{u}}^{\mathbf{T}}\mathbf{f}(k) \\ &\quad +\mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^T, \end{aligned} \tag{6.12}$$

where $\mathbf{U}_k$ is the uncertainty in the control inputs $\mathbf{u}(k)$, and

$$\mathrm{E}\{\mathbf{u}(k)\} = \mathbf{0} \tag{6.13}$$

$$\mathrm{E}\{\mathbf{u}(k), \mathbf{u}^T(k)\} = \mathbf{U}_k \tag{6.14}$$

$$= \begin{bmatrix} \sigma_V^2 & 0 \\ 0 & \sigma_\gamma^2 \end{bmatrix}. \tag{6.15}$$

The process noise $\mathbf{w}_v(k)$ is zero mean and white, with covariance $\mathbf{Q}_k$ given by

$$\mathbf{Q}_k = \mathrm{E}\{\mathbf{w}_v(k), \mathbf{w}_v^T(k)\} \tag{6.16}$$

$$= \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\psi^2 \end{bmatrix}. \tag{6.17}$$

When formulating this problem in information space, the propagation equations become

$$\mathbf{Y}(k \mid k-1) = \mathbf{M}_k - \mathbf{M}_k \mathbf{G}_k \boldsymbol{\Sigma}_k^{-1} \mathbf{G}_k^T \mathbf{M}_k \tag{6.18}$$

$$\hat{\mathbf{y}}(k \mid k-1) = \mathbf{Y}(k \mid k-1)\mathbf{f}\left(\hat{\mathbf{x}}(k-1 \mid k-1), \mathbf{u}(k)\right), \tag{6.19}$$

where

$$\boldsymbol{\Sigma}_k = \mathbf{G}_k^T \mathbf{M}_k \mathbf{G}_k + \mathbf{Q}_k^{-1}, \tag{6.20}$$

and

$$\mathbf{M}_k = \mathbf{A}_k - \mathbf{A}_k \nabla_{\mathbf{u}} \mathbf{f}(k) \left[\nabla_{\mathbf{u}}^{\mathbf{T}} \mathbf{f}(k) \mathbf{A}_k \nabla_{\mathbf{u}} \mathbf{f}(k) + \mathbf{U}_k^{-1}\right]^{-1} \nabla_{\mathbf{u}}^{\mathbf{T}} \mathbf{f}(k) \mathbf{A}_k \tag{6.21}$$

$$\mathbf{A}_k = \nabla_{\mathbf{x}}^{-\mathbf{T}} \mathbf{f}(k) \mathbf{Y}(k-1 \mid k-1) \nabla_{\mathbf{x}}^{-\mathbf{1}} \mathbf{f}(k). \tag{6.22}$$

6.2.3 Observation Model

Relative observations to landmarks are made using an observation model in the form

$$\mathbf{z}(k) = \mathbf{h}\left(\mathbf{x}(k)\right) + \mathbf{v}(k). \tag{6.23}$$

With a range/bearing sensor, Equation 6.23 can be written as

$$\mathbf{z}(k) = \begin{bmatrix} z_R(k) \\ z_\theta(k) \end{bmatrix} \tag{6.24}$$

$$= \begin{bmatrix} \sqrt{x_v(k) - x_i(k))^2 + (y_v(k) - y_i(k))^2} \\ \arctan\left(\frac{(y_v(k) - y_i(k))}{(x_v(k) - x_i(k))}\right) - \psi_v(k)) \end{bmatrix} + \begin{bmatrix} v_R(k) \\ v_\theta(k) \end{bmatrix}, \tag{6.25}$$

where $[x_i(k), y_i(k)]$ is the location of the feature being observed.

The Jacobian of this observation model with respect to the current state is required by the EKF for the update of the covariance matrix. As the observation is only a function of the

vehicle states and the those of the observed landmark, the result is a sparse matrix and can be written as [103]

$$\nabla_{\mathbf{x}}\mathbf{h}(k) = \left[\nabla_v\mathbf{h}(k), \;\; 0, \;\; \cdots \;\; 0, \;\; \nabla_i\mathbf{h}(k), \;\; 0, \;\; \cdots \right], \tag{6.26}$$

where $\nabla_v\mathbf{h}(k)$ is the Jacobian with respect to the vehicle states and $\nabla_i\mathbf{h}(k)$ the Jacobian with respect to the observed feature states. Evaluated at the current state estimate $\mathbf{x}(k) = \hat{\mathbf{x}}(k \mid k-1)$, these are given as

$$\nabla_v\mathbf{h}(k) = \begin{bmatrix} \frac{\hat{x}_v(k \mid k-1) - \hat{x}_i(k \mid k-1)}{\hat{d}(k)} & \frac{\hat{y}_v(k \mid k-1) - \hat{y}_i(k \mid k-1)}{\hat{d}(k)} & 0 \\ -\frac{\hat{y}_v(k \mid k-1) - \hat{y}_i(k \mid k-1)}{\hat{d}(k)^2} & \frac{\hat{x}_v(k \mid k-1) - \hat{x}_i(k \mid k-1)}{\hat{d}(k)^2} & -1 \end{bmatrix} \tag{6.27}$$

$$\nabla_i\mathbf{h}(k) = \begin{bmatrix} -\frac{\hat{x}_v(k \mid k-1) - \hat{x}_i(k \mid k-1)}{\hat{d}(k)} & -\frac{\hat{y}_v(k \mid k-1) - \hat{y}_i(k \mid k-1)}{\hat{d}(k)} \\ \frac{\hat{y}_v(k \mid k-1) - \hat{y}_i(k \mid k-1)}{\hat{d}(k)^2} & -\frac{\hat{x}_v(k \mid k-1) - \hat{x}_i(k \mid k-1)}{\hat{d}(k)^2} \end{bmatrix}, \tag{6.28}$$

where the estimated distance to the feature is

$$\hat{d}(k) = \sqrt{(\hat{x}_v(k \mid k-1) - \hat{x}_i(k \mid k-1))^2 + (\hat{y}_v(k \mid k-1) - \hat{y}_i(k \mid k-1))^2}. \tag{6.29}$$

The Kalman filter update stage can now be written as

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}_k \left[\mathbf{z}(k) - \mathbf{h}\left(\hat{\mathbf{x}}(k \mid k-1)\right) \right] \tag{6.30}$$

$$\mathbf{P}(k \mid k) = \mathbf{P}(k \mid k-1) - \mathbf{W}_k\mathbf{S}_k\mathbf{W}_k^T, \tag{6.31}$$

where the weighting is given by

$$\mathbf{W}_k = \mathbf{P}(k \mid k-1)\nabla_{\mathbf{x}}^{\mathbf{T}}\mathbf{h}(k)\mathbf{S}_k^{-1}, \tag{6.32}$$

and the innovation covariance is

$$\mathbf{S}_k = \nabla_{\mathbf{x}}\mathbf{h}(k)\mathbf{P}(k \mid k-1)\nabla_{\mathbf{x}}^{\mathbf{T}}\mathbf{h}(k) + \mathbf{R}_k. \tag{6.33}$$

In information space, the observation information is calculated using

$$\mathbf{i}(k) = \nabla_{\mathbf{x}}^{\mathrm{T}}\mathbf{h}(k)\mathbf{R}^{-1}(k)\left[\mathbf{z}(k) - \left(\mathbf{h}(\hat{\mathbf{x}}(k \mid k-1)) - \nabla_{\mathbf{x}}^{\mathrm{T}}\mathbf{h}(k)\hat{\mathbf{x}}(k \mid k-1)\right)\right] \quad (6.34)$$

$$\mathbf{I}(k) = \nabla_{\mathbf{x}}^{\mathrm{T}}\mathbf{h}(k)\mathbf{R}^{-1}(k)\nabla_{\mathbf{x}}\mathbf{h}(k), \quad (6.35)$$

and is fused with the information estimate as

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k) \quad (6.36)$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \mathbf{I}(k). \quad (6.37)$$

6.2.4 Feature Initialisation in State Space

When a feature is first observed, it must be initialised into the augmented state vector in such as way as to account for the platform uncertainty at that time. This is necessary to ensure the estimate remains consistent.

Using the range/bearing observation model given in Section 6.2.3, the initial estimate of a feature location is calculated using an initialisation function $\mathbf{g}(\hat{\mathbf{x}}(k \mid k-1), \mathbf{z}(k))$, shown in Equation 6.38.

$$\begin{bmatrix} \hat{x}_i(k) \\ \hat{y}_i(k) \end{bmatrix} = \begin{bmatrix} \hat{x}_v(k \mid k-1) + z_R(k)\cos(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) \\ \hat{y}_v(k \mid k-1) + z_R(k)\sin(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) \end{bmatrix} \quad (6.38)$$

This estimate is then augmented with the state vector as a new feature.

To initialise the covariance matrix, it is necessary to account for both the observation uncertainty and the vehicle uncertainty at the time of augmentation. Observation uncertainty is accounted for by simply adding the observation covariance to the diagonal of the augmented state covariance matrix as

$$\mathbf{P}'(k \mid k-1) = \begin{bmatrix} \mathbf{P}_{vv}(k \mid k-1) & \mathbf{P}_{vm}(k \mid k-1) & 0 \\ \mathbf{P}_{vm}^{T}(k \mid k-1) & \mathbf{P}_{mm}(k \mid k-1) & 0 \\ 0 & 0 & \mathbf{R}_k \end{bmatrix}. \quad (6.39)$$

The modified state covariance must then be projected through the the Jacobian of the

initialisation function $\mathbf{g}$ with respect to the augmented states. This gives

$$\mathbf{P}(k \mid k) = \nabla_{\mathbf{x}}\mathbf{g}(k)\mathbf{P}'(k \mid k-1)\nabla_{\mathbf{x}}^{\mathbf{T}}\mathbf{g}(k), \tag{6.40}$$

where

$$\nabla_{\mathbf{x}}\mathbf{g}(k) = \begin{bmatrix} \mathbf{I}_v & 0 & 0 \\ 0 & \mathbf{I}_m & 0 \\ \nabla_v\mathbf{g}(k) & 0 & \nabla_z\mathbf{g}(k) \end{bmatrix}, \tag{6.41}$$

and

$$\nabla_v\mathbf{g}(k) = \begin{bmatrix} 1 & 0 & -z_R(k)\sin(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) \\ 0 & 1 & z_R(k)\cos(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) \end{bmatrix} \tag{6.42}$$

$$\nabla_z\mathbf{g}(k) = \begin{bmatrix} \cos(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) & -z_R\sin(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) \\ \sin(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) & z_R\cos(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) \end{bmatrix}. \tag{6.43}$$

$\mathbf{I}_v$ and $\mathbf{I}_m$ in Equation 6.41 are identity matricies of the dimension of the vehicle and map states respectively.

Given the sparse nature of $\nabla_{\mathbf{x}}\mathbf{g}(k)$, it is possible to simplify the update of the covariance substantially by multiplying out Equation 6.40. This gives

$$\mathbf{P}(k \mid k) = \begin{bmatrix} \mathbf{P}_{vv}(k \mid k-1) & \mathbf{P}_{vm}(k \mid k-1) & (\nabla_{\mathbf{v}}\mathbf{g}(k)\mathbf{P}_{vv}(k \mid k-1))^T \\ \mathbf{P}_{vm}^T(k \mid k-1) & \mathbf{P}_{mm}(k \mid k-1) & (\nabla_{\mathbf{v}}\mathbf{g}(k)\mathbf{P}_{vm}(k \mid k-1))^T \\ \nabla_v\mathbf{g}(k)\mathbf{P}_{vv}(k \mid k-1) & \nabla_v\mathbf{g}(k)\mathbf{P}_{vm}(k \mid k-1) & \mathbf{\Omega}(k) \end{bmatrix}, \tag{6.44}$$

where

$$\mathbf{\Omega}(k) = \nabla_v\mathbf{g}(k)\mathbf{P}_{vv}(k \mid k-1)\nabla_v^T\mathbf{g}(k) + \nabla_z\mathbf{g}(k)\mathbf{R}(k)\nabla_z^T\mathbf{g}(k). \tag{6.45}$$

Equations 6.38 and 6.44 can be applied directly to initialise a new feature in the map. After initialisation, the filter can run using the EKF prediction and update cycles given by Equations 6.11–6.12 and 6.30–6.33 respectively.

6.2.5 Feature Initialisation in Information Space

The initialistion of features in information space can be done by transforming the state space initialisation equations in Section 6.2.4 to their inverse covariance form.

The initial step involves augmenting the diagonal of the information matrix for the map with the inverse of the observation variance $\mathbf{R}_k$ as

$$\mathbf{Y}'(k \mid k-1) = \begin{bmatrix} \mathbf{Y}_{vv}(k \mid k-1) & \mathbf{Y}_{vm}(k \mid k-1) & 0 \\ \mathbf{Y}_{vm}^T(k \mid k-1) & \mathbf{Y}_{mm}(k \mid k-1) & 0 \\ 0 & 0 & \mathbf{R}_k^{-1} \end{bmatrix}. \tag{6.46}$$

The modified information matrix must then be transformed using the Jacobian of the initialisation function $\mathbf{g}$ with respect to the augmented states. This can be written

$$\begin{aligned} \mathbf{Y}(k \mid k) &= \left(\nabla_{\mathbf{x}}\mathbf{g}(k)\mathbf{Y}'^{-1}(k \mid k-1)\nabla_{\mathbf{x}}^{\mathbf{T}}\mathbf{g}(k)\right)^{-1} \\ &= \nabla_{\mathbf{x}}^{-\mathbf{T}}\mathbf{g}(k)\mathbf{Y}'(k \mid k-1)\nabla_{\mathbf{x}}^{-1}\mathbf{g}(k), \end{aligned} \tag{6.47}$$

where

$$\nabla_{\mathbf{x}}\mathbf{g}(k) = \begin{bmatrix} \mathbf{I}_v & 0 & 0 \\ 0 & \mathbf{I}_m & 0 \\ \nabla_v\mathbf{g}(k) & 0 & \nabla_z\mathbf{g}(k) \end{bmatrix}, \tag{6.48}$$

and

$$\nabla_v\mathbf{g}(k) = \begin{bmatrix} 1 & 0 & -z_R(k)\sin(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) \\ 0 & 1 & z_R(k)\cos(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) \end{bmatrix} \tag{6.49}$$

$$\nabla_z\mathbf{g}(k) = \begin{bmatrix} \cos(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) & -z_R\sin(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) \\ \sin(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) & z_R\cos(\hat{\psi}_v(k \mid k-1) + z_\theta(k)) \end{bmatrix}. \tag{6.50}$$

Using the matrix inversion lemma for partitioned matrixies in the form

$$\begin{bmatrix} A & 0 \\ C & B \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & 0 \\ -B^{-1}CA^{-1} & B^{-1} \end{bmatrix}, \tag{6.51}$$

and making the substitutions

$$A = \begin{bmatrix} \mathbf{I}_v & 0 \\ 0 & \mathbf{I}_m \end{bmatrix}, \qquad B = [\nabla_z \mathbf{g}(k)], \qquad C = \begin{bmatrix} \nabla_v \mathbf{g}(k) & 0 \end{bmatrix}, \tag{6.52}$$

the inverse of $\nabla_{\mathbf{x}} \mathbf{g}(k)$ can be written as

$$\nabla_{\mathbf{x}}^{-1} \mathbf{g}(k) = \begin{bmatrix} \mathbf{I}_v & 0 & 0 \\ 0 & \mathbf{I}_m & 0 \\ -\nabla_z^{-1} \mathbf{g}(k) \nabla_v \mathbf{g}(k) & 0 & \nabla_z^{-1} \mathbf{g}(k) \end{bmatrix}. \tag{6.53}$$

Substituting Equation 6.53 into Equation 6.47 and multiplying gives

$$\mathbf{Y}(k \mid k) = \begin{bmatrix} \mathbf{Y}_{vv}(k \mid k-1) & \mathbf{Y}_{vm}(k \mid k-1) & (\lambda(k)\mathbf{Y}_{vv}(k \mid k-1))^T \\ \mathbf{Y}_{vm}^T(k \mid k-1) & \mathbf{Y}_{mm}(k \mid k-1) & (\lambda(k)\mathbf{Y}_{vm}(k \mid k-1))^T \\ \lambda(k)\mathbf{Y}_{vv}(k \mid k-1) & \lambda(k)\mathbf{Y}_{vm}(k \mid k-1) & \mathbf{\Lambda}(k) \end{bmatrix}, \tag{6.54}$$

where

$$\lambda(k) = -\nabla_z^{-1} \mathbf{g}(k) \nabla_v \mathbf{g}(k) \tag{6.55}$$

$$\mathbf{\Lambda}(k) = \lambda(k)\mathbf{Y}_{vv}(k \mid k-1)\lambda^T(k) + \nabla_z^{-1} \mathbf{g}(k) \mathbf{R}^{-1}(k) \nabla_z^{-T} \mathbf{g}(k). \tag{6.56}$$

The states can be initialised by augmenting the state vector with the new feature estimates computed using Equation 6.38. The information vector can be obtained by premultiplying the state vector with the updated information matrix in Equation 6.54. The information filter can then continue to operate using Equations 6.18–6.21 for predictions and Equations 6.34–6.37 for updates.

6.3 Closed Form Solutions

This section presents closed form solutions to a simplified form of the single and multiple platform continuous time SLAM problem [79]. While the solutions are not practically useful, they do provide insight into the underlying structure of the problem.

6.3.1 Platform Models

All platforms $i = 1, \ldots, m$ use the same linear one degree of freedom model given by

$$\dot{x}_i(t) = u_i(t) + w_i, \quad i = 1, \ldots, m, \tag{6.57}$$

where $u_i(t)$ is the measured velocity of each platform i. The error w_i in this measurement is modeled as a zero mean Gaussian with variance $Q_i = E[w_i, w_i^T] = q_i$.

Features $j = i, ..., n$ are located throughout the environment at fixed locations p_j such that $\dot{p}_j = 0$. An augmented state vector containing the estimates of the states of all m platforms and the feature locations is used, written in the form

$$\mathbf{x}(t) = [x_1(t), ..., x_m(t), p_1, ..., p_n]^T. \tag{6.58}$$

As all platforms are augmented with a common map, this formulation simulates a fully connected network of platforms sharing their observations.

The state transition equation then becomes

$$\dot{\mathbf{x}}(t) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}(\mathbf{u}(t) + \mathbf{w}(t)), \tag{6.59}$$

where

$$\mathbf{F} = \begin{bmatrix} \mathbf{0} \end{bmatrix} \tag{6.60}$$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & & \vdots & \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 & 0 & \cdots & \cdots & 0 \end{bmatrix}^T. \tag{6.61}$$

Observations are made of the range to each feature using a suitable sensor on each platform. An observation of feature j from platform i is written

$$z_{ij}(t) = p_j - x_i(t) + v_j, \tag{6.62}$$

where v_j is the observation error and is assumed zero mean Gaussian with variance r_j. The

total observation noise can then be constructed in the form

$$\mathbf{R} = diag(r_1, ..., r_n). \tag{6.63}$$

The observation model is written

$$\mathbf{z}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{v}, \tag{6.64}$$

where

$$\mathbf{z}(t) = \left[\; z_{11}(t), \;\; \ldots, \;\; z_{1n}(t), \;\; z_{21}(t), \;\; \ldots, \;\; z_{mn}(t) \; \right]^T \tag{6.65}$$

$$\mathbf{v} = \left[\; v_{11}, \;\; \ldots, \;\; v_{1n}, \;\; v_{21}, \;\; \ldots, \;\; v_{mn} \; \right]^T, \tag{6.66}$$

and

$$\mathbf{H} = \begin{bmatrix} -1 & 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & \cdots & 0 & 0 & \cdots & 1 \\ 0 & -1 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -1 & \cdots & 0 & 0 & \cdots & 1 \\ \\ 0 & 0 & \cdots & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & 0 & \cdots & 1 \end{bmatrix}. \tag{6.67}$$

The required estimate is the conditional mean

$$\hat{\mathbf{x}}(t) = \mathrm{E}\{\mathbf{x}(t)|\mathbf{Z}^t\} \; , \tag{6.68}$$

with mean squared error

$$\mathbf{P}(t) = \mathrm{E}\{\tilde{\mathbf{x}}^2(t)|\mathbf{Z}^t\} \; , \tag{6.69}$$

where

$$\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(\mathbf{t}). \tag{6.70}$$

In information space, the inverse of the state variance is maintained, and is denoted $\mathbf{Y}(t) = \mathbf{P}^{-1}(t)$.

6.3.2 Solutions to the Riccati Equation

The evolution of the SLAM covariance matrix is described by the algebraic Riccati equation

$$\dot{P}(t) = FP(t) + P(t)F^T + GQG^T - P(t)H^T R^{-1} HP(t). \tag{6.71}$$

Similarly, the evolution of the information matrix is described by the inverse of Equation 6.71, written in the form

$$\dot{Y}(t) = -Y(t)F - F^T Y(t) - Y(t)GQG^T Y(t) + H^T R^{-1} H. \tag{6.72}$$

Further information on the Riccati equation and its applications can be found in [2, 3, 15, 89].

Solving Equations 6.71 and 6.72 provides a closed form solution to the covariance and information matricies. These solutions are potentially extremely valuable as they allow the performance of the problem to be evaluated for any set of system parameters.

The method used to solve the Riccati equation in this thesis is to represent it as a fraction decomposition in the form

$$P(t) = U(t)V^{-1}(t), \tag{6.73}$$

for a state space solution, and

$$Y(t) = V(t)U^{-1}(t), \tag{6.74}$$

for an information space solution, where U(t) and V(t) are the solution to the Hamiltonian described by the linear dfferential equation [2, 41]

$$\begin{bmatrix} \dot{U}(t) \\ \dot{V}(t) \end{bmatrix} = \begin{bmatrix} F & Q \\ HR^{-1}H & -F \end{bmatrix} \begin{bmatrix} U(t) \\ V(t) \end{bmatrix}, \tag{6.75}$$

with initial conditions

$$\begin{bmatrix} U(0) \\ V(0) \end{bmatrix} = \begin{bmatrix} P(0) \\ I \end{bmatrix}. \tag{6.76}$$

As $F = 0$ (Equation 6.60), Equation 6.75 simplifies to

$$\begin{bmatrix} \dot{U}(t) \\ \dot{V}(t) \end{bmatrix} = \begin{bmatrix} 0 & Q \\ HR^{-1}H & 0 \end{bmatrix} \begin{bmatrix} U(t) \\ V(t) \end{bmatrix}, \tag{6.77}$$

which can be solved using a commercial symbolic algebra package. The results presented in this thesis are obtained by solving Equation 6.77 for $n = 1, 2, 3$ features and $m = 1, 2, 3$ platforms. Using these solutions, the general case was deduced and checked by substitution back into Equations 6.71 and 6.72.

6.3.3 State Space Solution

A closed form solution to the single degree of freedom SLAM covariance matrix in continuous time was first presented for the single vehicle case in [39]. This investigated the convergence properties of the SLAM algorithm for various combinations of process and observation noise. In this thesis, development is focused on information filter formulations of multiple vehicle SLAM problems.

A general form of covariance matrix $P(t)$ for the of the n feature, m platform scalar SLAM problem is presented in Equations 6.78–6.86. The solution was obtained by solving Equation 6.71 using the method described in Section 6.3.2. Multiple platforms each observe the same n continuously observable, viewpoint invariant features at all times. These features are used for SLAM. Data association is not considered in this problem and platforms only sense the features and not the relative location of each other. The process noise q is the same for all platforms.

The solution to the covariance matrix is decomposed into three different sub-matrices due to size constraints. These are the platform to platform $P_{PP}(t)$, platform to map $P_{PM}(t)$

and map to map $P_{MM}(t)$ components. The structure of the full matrix is given by

$$P(t) = \begin{bmatrix} P_{PP}(t) & P_{PM}(t) \\ P_{PM}^T(t) & P_{MM}(t) \end{bmatrix}. \tag{6.78}$$

The platform to platform covariance $P_{PP}(t)$ is given by

$$\begin{bmatrix} \frac{q(1-e^{-2\alpha t})}{D(t)} + a + (m-1)b & a-b & \cdots & a-b \\ a-b & \frac{q(1-e^{-2\alpha t})}{D(t)} + a + (m-1)b & \vdots & a-b \\ \vdots & \cdots & \ddots & \vdots \\ a-b & a-b & \cdots & \frac{q(1-e^{-2\alpha t})}{D(t)} + a + (m-1)b \end{bmatrix}, \tag{6.79}$$

where

$$a = \frac{\frac{2q}{\alpha}(1-e^{-\alpha t})^2}{D(t)}, \tag{6.80}$$

$$b = \frac{\frac{q}{\alpha}(1-e^{-2\alpha t})^2}{D(t)(1+e^{-2\alpha t})}. \tag{6.81}$$

The platform to map $P_{PM}(t)$ covariance is

$$\begin{bmatrix} \frac{\frac{q}{\alpha}(1-e^{-\alpha t})^2}{D(t)} & \cdots & \cdots & \frac{\frac{q}{\alpha}(1-e^{-\alpha t})^2}{D(t)} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\frac{q}{\alpha}(1-e^{-\alpha t})^2}{D(t)} & \cdots & \cdots & \frac{\frac{q}{\alpha}(1-e^{-\alpha t})^2}{D(t)} \end{bmatrix}, \tag{6.82}$$

and the map to map component $P_{MM}(t)$ is

$$\begin{bmatrix} \frac{r_1(I_T - r_1^{-1})}{I_T(mt+1)} + \frac{\frac{q}{\alpha}(1+e^{-2\alpha t})}{D(t)} & \frac{-1}{I_T(mt+1)} + \frac{\frac{q}{\alpha}(1+e^{-2\alpha t})}{D(t)} & \cdots & \frac{-1}{I_T(mt+1)} + \frac{\frac{q}{\alpha}(1+e^{-2\alpha t})}{D(t)} \\ \frac{-1}{I_T(mt+1)} + \frac{\frac{q}{\alpha}(1+e^{-2\alpha t})}{D(t)} & \frac{r_2(I_T - r_2^{-1})}{I_T(mt+1)} + \frac{\frac{q}{\alpha}(1+e^{-2\alpha t})}{D(t)} & \vdots & \frac{-1}{I_T(mt+1)} + \frac{\frac{q}{\alpha}(1+e^{-2\alpha t})}{D(t)} \\ \vdots & \cdots & \ddots & \vdots \\ \frac{-1}{I_T(mt+1)} + \frac{\frac{q}{\alpha}(1+e^{-2\alpha t})}{D(t)} & \cdots & \cdots & \frac{r_n(I_T - r_n^{-1})}{I_T(mt+1)} + \frac{\frac{q}{\alpha}(1+e^{-2\alpha t})}{D(t)} \end{bmatrix} \tag{6.83}$$

The dominant time constant for the system is given by

$$\alpha = \sqrt{qI_T}, \tag{6.84}$$

and

$$I_T = \sum_{i=1}^{N} r_i^{-1}, \tag{6.85}$$

is the total Fisher information. The characteristic equation of the system is

$$D(t) = (\alpha + m) + (\alpha - m)e^{-2\alpha t}. \tag{6.86}$$

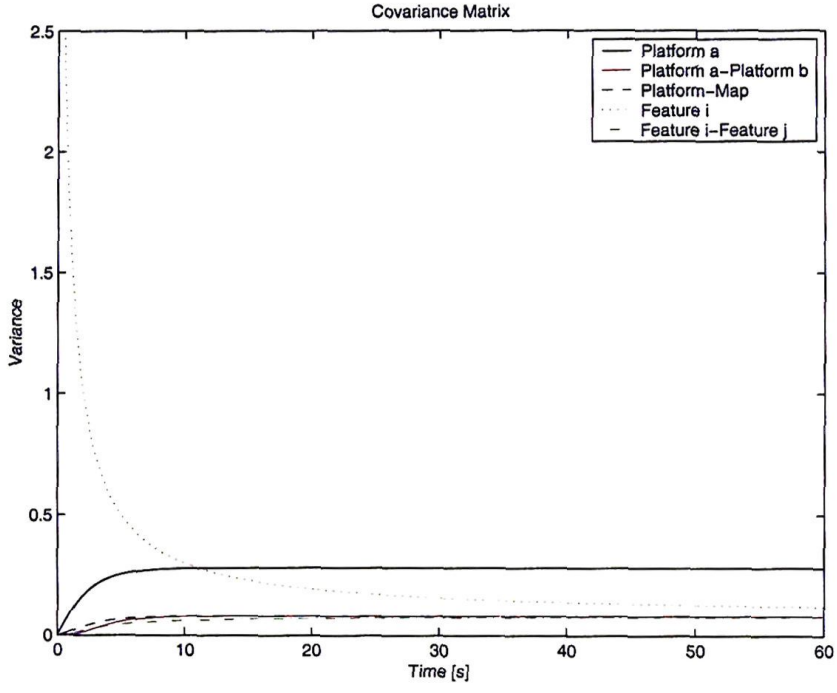


Figure 6.2: The evolution of the continuous time SLAM covariance matrix for a two platform, two feature problem. The results were obtained with $q = 0.1$ for both platforms and $r = 5$ for both features.

The characteristic equation of the system, given by Equation 6.86, is clearly dependent on the number of platforms. This occurs as all platforms are correlated through the use of the same map, and subsequently affect one another.

The structure of the platform to platform elements are somewhat simplified due to each platform having the same process noise covariance q. The term beginning $(m-1)\frac{q}{\alpha}$ in Equation 6.79 would in practice change if q were different for each platform. However, it is important to note that the platform to platform cross correlations are all non-zero and equal, as each platform is correlated through the use of the common map.

The time t on the denominator of the first term of each element in Equation 6.83 is multiplied by the number of platforms m. This occurs as all platforms, using the same process noise q, are observing the same features with the same observation noise r. The net result is that this can be thought of as one sensor observing the same features for m times as long (where m is the number of platforms).

In the limit $t \to \infty$, the platform to platform cross correlations become equal to the platform to map cross correlations and also the map to map cross correlations. The reason for this is that all features and platforms are correlated through the use of the same map. This is illustrated in Figure 6.2 which plots the evolution of the covariance matrix over time.

6.3.4 Information Space Solution for a Single Platform

The information filter form of the SLAM problem can also be found using an identical solution procedure to that used for the Kalman filter form. The inverse covariance matrix for general form of the n feature scalar SLAM problem is given by

$$\begin{bmatrix} \frac{\alpha \coth(\alpha t)}{q} & \cdots & -\frac{\tanh(\frac{\alpha t}{2})}{r_i \alpha} & \cdots & -\frac{\tanh(\frac{\alpha t}{2})}{r_j \alpha} \\ \vdots & \ddots & \vdots & & \vdots \\ -\frac{\tanh(\frac{\alpha t}{2})}{r_i \alpha} & \cdots & Y_{ii}(0) + \frac{t}{r_i} - \frac{t}{r_i^2 I_T} + \frac{2\tanh(\frac{\alpha t}{2})}{\alpha r_i^2 I_T} & \cdots & -\frac{t}{r_i r_j I_T} + \frac{2\tanh(\frac{\alpha t}{2})}{\alpha r_i r_j I_T} \\ \vdots & & \vdots & \ddots & \vdots \\ -\frac{\tanh(\frac{\alpha t}{2})}{r_j \alpha} & \cdots & -\frac{t}{r_i r_j I_T} + \frac{2\tanh(\frac{\alpha t}{2})}{\alpha r_i r_j I_T} & \cdots & Y_{jj}(0) + \frac{t}{r_j} - \frac{t}{r_j^2 I_T} + \frac{2\tanh(\frac{\alpha t}{2})}{\alpha r_j^2 I_T} \end{bmatrix}, \tag{6.87}$$

where

$$\alpha = \sqrt{q I_T}, \tag{6.88}$$

is the dominant time constant for the system and

$$I_T = \sum_{i=1}^{N} r_i^{-1}, \tag{6.89}$$

is the total Fisher information.

In continuous time, the term $\frac{1}{r_i}$ in each element on the diagonal of the map component of Equation 6.87 can be thought of as the amount of information per unit time added to the

system by observing feature i. The term $\frac{t}{r_i}$ is thus the information added by observing the i^{th} feature for t seconds.

The off diagonal feature i to feature j cross-information is structured very closely to that of the diagonal elements.

$$-\frac{t}{r_i r_j I_T} + \frac{2\tanh(\frac{\alpha t}{2})}{\alpha r_i r_j I_T} \tag{6.90}$$

The primary difference is that the off diagonal terms do not contain the $\frac{t}{r_i}$ term described above. This is to be expected as this term relates to a continuous observation of a feature rather than the cross information between features.

The feature i to platform cross information is in the form

$$-\frac{\tanh(\frac{\alpha t}{2})}{r_i \alpha}. \tag{6.91}$$

The cross information for a feature i is clearly dependent on r_i. This differs significantly from the state space case in Equation 6.82 where all cross-correlations are the same [39].

In the limit $t \to \infty$, the platform to platform information approaches $\frac{\alpha}{q}$. The limits of the vehicle to map components are different for each feature, and are governed by r_i for that feature. This differs from the state space equivalent where the feature to feature and feature to map terms were all equivalent in the limit (see Figure 6.2). Also, the information term for a particular feature grows without bound. This is reasonable because as the platform continuously observes a feature, information continues to accumulate.

6.3.5 Information Space Solution for Multiple Platforms

The information filter solution to the multiple platform SLAM problem is now considered. The same models and assumptions stated in Section 6.3.3 are employed save that different process noise covariance values, q_i, are used for each platform.

The information matrix $Y(t)$ for the n feature, m platform scalar SLAM problem is decomposed into three sub-matrices. These are platform to platform $Y_{PP}(t)$, platform to map $Y_{PM}(t)$ and map to map $Y_{MM}(t)$ components. The structure of the full matrix is

$$Y(t) = \begin{bmatrix} Y_{PP}(t) & Y_{PM}(t) \\ Y_{PM}^T(t) & Y_{MM}(t) \end{bmatrix}. \tag{6.92}$$

The platform to platform $Y_{PP}(t)$ component is given by

$$\begin{bmatrix} \frac{\alpha_1 \coth(\alpha_1 t)}{q_1} & 0 & \cdots & 0 \\ 0 & \frac{\alpha_2 \coth(\alpha_2 t)}{q_2} & \vdots & 0 \\ \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\alpha_m \coth(\alpha_m t)}{q_m} \end{bmatrix}, \tag{6.93}$$

the platform to map $Y_{PM}(t)$ by

$$\begin{bmatrix} \frac{-\tanh(\frac{\alpha_1 t}{2})}{r_1 \alpha_1} & \frac{-\tanh(\frac{\alpha_1 t}{2})}{r_2 \alpha_1} & \cdots & \frac{-\tanh(\frac{\alpha_1 t}{2})}{r_n \alpha_1} \\ \frac{-\tanh(\frac{\alpha_2 t}{2})}{r_1 \alpha_2} & \frac{-\tanh(\frac{\alpha_2 t}{2})}{r_2 \alpha_2} & \vdots & \frac{-\tanh(\frac{\alpha_2 t}{2})}{r_n \alpha_2} \\ \vdots & \cdots & \ddots & \vdots \\ \frac{-\tanh(\frac{\alpha_m t}{2})}{r_1 \alpha_m} & \frac{-\tanh(\frac{\alpha_m t}{2})}{r_2 \alpha_m} & \cdots & \frac{-\tanh(\frac{\alpha_m t}{2})}{r_n \alpha_m} \end{bmatrix}, \tag{6.94}$$

and the map to map $Y_{MM}(t)$ component by

$$\begin{bmatrix} Y_{11}(0) + \sum_{i=1}^{m} \left[\frac{t(I_T - r_1^{-1})}{r_1 I_T} + \frac{b_i}{r_1^2} \right] & \sum_{i=1}^{m} \left[\frac{-t}{r_1 r_2 I_T} + \frac{b_i}{r_1 r_2} \right] & \cdots \\ \sum_{i=1}^{m} \left[\frac{-t}{r_2 r_1 I_T} + \frac{b_i}{r_2 r_1} \right] & Y_{22}(0) + \sum_{i=1}^{m} \left[\frac{t(I_T - r_2^{-1})}{r_2 I_T} + \frac{b_i}{r_2^2} \right] & \vdots \\ \vdots & \cdots & \ddots \\ \sum_{i=1}^{m} \left[\frac{-t}{r_n r_1 I_T} + \frac{b_i}{r_n r_1} \right] & \sum_{i=1}^{m} \left[\frac{-t}{r_n r_2 I_T} + \frac{b_i}{r_n r_2} \right] & \cdots \end{bmatrix}$$

$$\begin{bmatrix} \sum_{i=1}^{m} \left[\frac{-t}{r_1 r_n I_T} + \frac{b_i}{r_1 r_n} \right] \\ \sum_{i=1}^{m} \left[\frac{-t}{r_2 r_n I_T} + \frac{b_i}{r_2 r_n} \right] \\ \vdots \\ \mathbf{Y}(_{|}nn)(0) + \sum_{i=1}^{m} \left[\frac{t(I_T - r_n^{-1})}{r_n I_T} + \frac{b_i}{r_n^2} \right], \end{bmatrix}, \tag{6.95}$$

where

$$b_i = \frac{2 \tanh(\frac{\alpha_i t}{2})}{\alpha_i I_T}. \tag{6.96}$$

The dominant time constant of the system is given by

$$\alpha_i = \sqrt{q_i I_T}, \tag{6.97}$$

and

$$I_T = \sum_{i=1}^{n} r_i^{-1}, \tag{6.98}$$

is the total Fisher information.

The diagonal elements of the platform to platform sub-matrix in Equation 6.93 are

$$\frac{\alpha_i \coth(\alpha_i t)}{q_i}, \tag{6.99}$$

and are clearly the same as for the single platform case shown in Equation 6.87. All off-diagonal elements (platform i to platform j cross information) are zero. Although the assumption was made that the platforms do not sense each other, they are correlated in state space as they are navigating using the same map. The state space platform to platform cross correlations are clear in Equation 6.79 as the matrix is fully populated. In information space the platform information vectors are constructed to be orthogonal to one another, which results in the cross information being zero. For a more complex system where each platform has multiple states, the full platform to platform matrix should be block diagonal. This is a very important result as it demonstrates that having information about one platform does not provide any information about any other platforms.

The platform i to feature j cross information is in the form

$$\frac{-\tanh(\frac{\alpha_i t}{2})}{r_j \alpha_i}, \tag{6.100}$$

which is also the same as for the single platform case. This arises as all platform information vectors are orthogonal to one another, and therefore the platform to map cross information from one platform will not contribute to another.

All elements in the map sub-matrix (Equation 6.95) are simply the sum of the contributions from each platform, plus the initial condition. This can be seen as the feature i information is

$$Y_{ii}(0) + \sum_{k=1}^{m} \left[\frac{t(I_T - r_i^{-1})}{r_i I_T} + \frac{b_k}{r_i^2} \right], \tag{6.101}$$

and the feature i to feature j cross information is

$$\sum_{k=1}^{m} \left[\frac{-t}{r_i r_j I_T} + \frac{b_k}{r_i r_j} \right], \tag{6.102}$$

where $b_k = \frac{2\tanh(\frac{\alpha_k t}{2})}{\alpha_k I_T}$. This follows logically from the structure of the information filter update stage which is additive. The limits of the feature to feature elements are all unbounded as $t \rightarrow \infty$. This occurs as extra information is continually being added to the map information matrix.

6.4 Map Information

There are two important results that the continuous time solution to the multiple platform information matrix in Section 6.3.5 provides:

1. The platform to platform cross information is zero provided the vehicles do not sense each other.
2. The global map information is the sum of the map information on each platform in the system.

These illustrate that when the multi-platform SLAM problem is formulated in information space, the only information that platforms have in common is map information. Provided platforms do not use relative observations of each other for localisation, the cross information between vehicles is always zero. As the global map information is simply the sum of the contributions from each vehicle, it is possible to construct a decentralised system where the map information is all that needs to be communicated between platforms.

Each platform in the decentralised network now maintains a combined state estimate in information form of its own states and the map of landmarks. Crucially, the fact that the cross information between multiple platforms is zero means that it is not necessary for a platform to augment the vehicle states of any other node it is communicating with. Platforms merely receive map information from others in the network and fuse this with their own local map.

6.4.1 The Information Map

The D-SLAM problem can be depicted as shown in Figure 6.3 where multiple aircraft are all communicating map information and localising themselves within the common map.

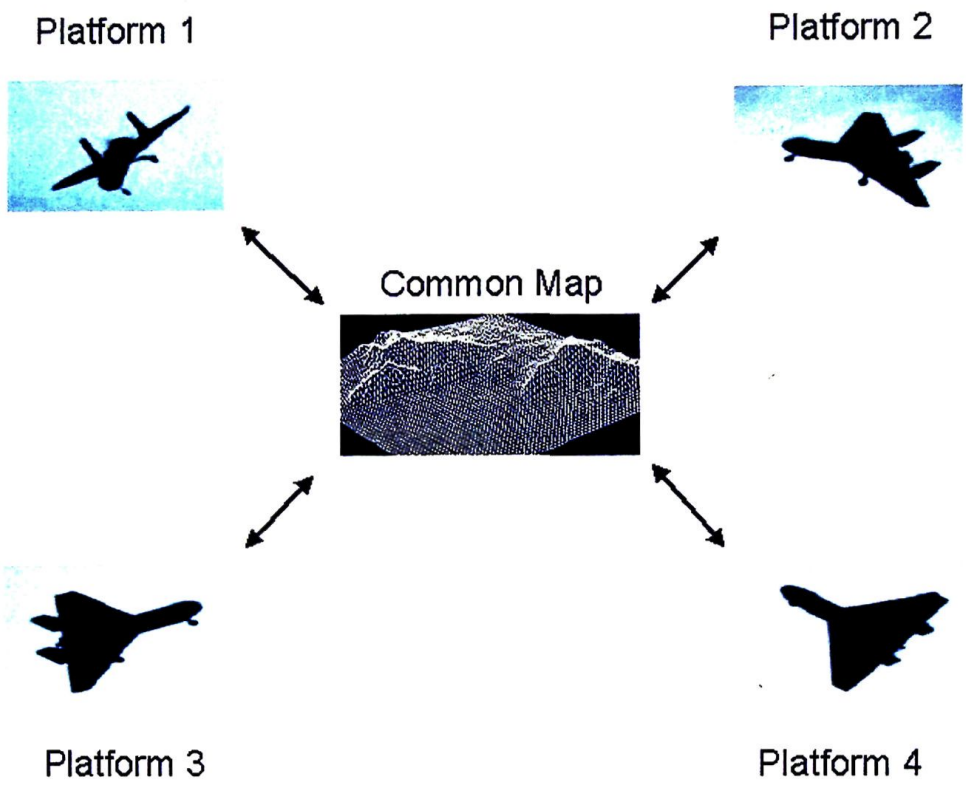


Figure 6.3: Multiple platforms using SLAM in a decentralised network all contribute to a common map and use it to navigate.

In the discrete time problem, the map information that is communicated between nodes is defined as

$$\mathbf{Y}^*_{mm}(k \mid k) = \mathbf{P}^{-1}_{mm}(k \mid k) \tag{6.103}$$

$$\hat{\mathbf{y}}^*_m(k \mid k) = \mathbf{P}^{-1}_{mm}(k \mid k)\hat{\mathbf{x}}_m(k \mid k), \tag{6.104}$$

where $\mathbf{P}_{mm}(k \mid k)$ is the map component of the covariance matrix.

$$\mathbf{P}(k \mid k) = \begin{bmatrix} \mathbf{P}_{vv}(k \mid k) & \mathbf{P}_{vm}(k \mid k) \\ \mathbf{P}^T_{vm}(k \mid k) & \mathbf{P}_{mm}(k \mid k) \end{bmatrix} \qquad \hat{\mathbf{x}}(k \mid k) = \begin{bmatrix} \hat{\mathbf{x}}_v(k \mid k) \\ \hat{\mathbf{x}}_m(k \mid k) \end{bmatrix} \tag{6.105}$$

This is the information form of the map alone. It is important to note that $\mathbf{Y}^*_{mm}(k \mid k)$ and

$\hat{\mathbf{y}}^*_m(k \mid k)$ are not simply the map component of the information matrix.

$$\mathbf{Y}^*_{mm}(k \mid k) \neq \mathbf{Y}_{mm}(k \mid k) \tag{6.106}$$

$$\hat{\mathbf{y}}^*_m(k \mid k) \neq \hat{\mathbf{y}}_m(k \mid k) \tag{6.107}$$

where

$$\mathbf{Y}(k \mid k) = \begin{bmatrix} \mathbf{Y}_{vv}(k \mid k) & \mathbf{Y}_{vm}(k \mid k) \\ \mathbf{Y}^T_{vm}(k \mid k) & \mathbf{Y}_{mm}(k \mid k) \end{bmatrix} \qquad \hat{\mathbf{y}}(k \mid k) = \begin{bmatrix} \hat{\mathbf{y}}_v(k \mid k) \\ \hat{\mathbf{y}}_m(k \mid k) \end{bmatrix}. \tag{6.108}$$

This is because the vehicle terms have an effect on the map which must be correctly accounted for. In order to remove the effect of the vehicle and get the map information the following relationships can be used

$$\mathbf{Y}^*_{mm}(k \mid k) = \mathbf{Y}_{mm}(k \mid k) - \mathbf{Y}_{vm}(k \mid k)\mathbf{Y}^{-1}_{vv}(k \mid k)\mathbf{Y}^T_{vm}(k \mid k) \tag{6.109}$$

$$\hat{\mathbf{y}}^*_m(k \mid k) = \hat{\mathbf{y}}_m(k \mid k) - \mathbf{Y}_{vm}(k \mid k)\mathbf{Y}^{-1}_{vv}(k \mid k)\hat{\mathbf{y}}_v(k \mid k). \tag{6.110}$$

6.5 Algorithms for Decentralised SLAM

This section describes practical algorithms for D-SLAM using the decentralised architecture presented in Chapter 3. Multiple platforms are now considered as nodes communicating map information in a decentralised network. However, to use the communicated map information, it is assumed that all platforms are initialised in a global coordinate frame. For the UAV applications considered in this thesis, the D-SLAM algorithms can be registered in a global frame by using the IMU/GPS solution on each aircraft for initialisation.

6.5.1 Decentralised Information Space Formulation

In the decentralised information form of the SLAM problem each node maintains an estimate of its own platform states together with a common information map. At no time does any node need to estimate the states of any other platform. This augmented state estimate maintained using the information filter prediction and update equations is derived in Section 6.2 (Equations 6.18–6.21 for predictions and Equations 6.34–6.37 for updates).

When computing the prediction stage of the information filter, it is important to recognise that the information about every state (including the map elements) will change even though the model only propagates the vehicle states. The result of this is that the prediction stage of SLAM is more computationally expensive in an information form as the entire SLAM matrix will change instead of just the vehicle submatrix. However, the the SLAM update in information form, given by Equations 6.34–6.37, very simple as it only affects the submatricies of the vehicle and feature being observed.

The difference between a network of independent platforms and a D-SLAM implementation is the incorporation of map information from other nodes in the DDF network. Whereas an independent platform will only update with information obtained from locally attached sensors, the decentralised version has an extra update stage which is used to fuse this extra map information.

It is the map information quantities $\mathbf{Y}^*_{mm}(k \mid k)$ and $\hat{\mathbf{y}}^*_m(k \mid k)$ given by Equations 6.109 and 6.110 that are communicated between nodes. A channel filter is used to maintain communications and keep a record of the map information nodes have in common. When information is received at a node, the channel filter determines the new information by subtracting the existing channel estimate from the newly arrived data using the algorithm described in Section 3.4. This difference or new information, written as $\mathbf{I}^*_{mm}(k)$ and $\mathbf{i}^*_m(k)$, is then output by the channel filter to the local filter for update.

An important property of the channel filter in decentralised SLAM is that there is no time alignment problem as all features are, by definition, stationary. This results in a simple channel algorithm that literally only performs addition and subtraction operations for updating and calculating new map information.

After the channel filter has computed the new information $\mathbf{I}^*_{mm}(k)$ and $\mathbf{i}^*_m(k)$, it passes these increments on to the local SLAM filter. These increments are essentially a pseudo-observation of the entire map. Therefore, the associated observation matrix $\mathbf{H}(k)$ is simply

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{0}_v & 0 \\ 0 & \mathbf{I}_m \end{bmatrix}, \tag{6.111}$$

which is the identity in the map section and zeros in the vehicle section. The inverse of the

variance for this pseudo-observation is given by $\mathbf{I}^*_{mm}(k)$, so

$$\begin{aligned}\mathbf{I}(k) &= \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k) \\ &= \mathbf{H}^T(k)\mathbf{I}^*_{mm}(k)\mathbf{H}(k).\end{aligned} \tag{6.112}$$

Since H is the identity over the map, and zero everywhere else, this will give

$$\mathbf{I}(k) = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}^*_{mm}(k) \end{bmatrix}. \tag{6.113}$$

Similarly, looking at the information vector $\mathbf{i}^*_m(k)$, the actual observation in state space can be written as

$$\mathbf{z}(k) = \mathbf{I}^{*-1}_{mm}(k)\mathbf{i}^*_m(k). \tag{6.114}$$

The observation information is then given by

$$\begin{aligned}\mathbf{i}(k) &= \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{z}(k) \\ &= \mathbf{H}^T(k)\mathbf{I}^*_{mm}(k)\mathbf{z}(k) \\ &= \mathbf{H}^T(k)\mathbf{I}^*_{mm}(k)\mathbf{I}^{*-1}_{mm}(k)\mathbf{i}^*_m(k) \\ &= \mathbf{H}^T(k)\mathbf{i}^*_m(k).\end{aligned} \tag{6.115}$$

Using the fact that H is the identity over the map, and zero everywhere else gives

$$\mathbf{i}(k) = \begin{bmatrix} 0 \\ \mathbf{i}^*_m(k) \end{bmatrix}. \tag{6.116}$$

The update of the local filter using the increments of map information from other nodes in the DDF network is therefore given by

$$\mathbf{Y}(k \mid k) = \begin{bmatrix} \mathbf{Y}_{vv}(k \mid k-1) & \mathbf{Y}_{vm}(k \mid k-1) \\ \mathbf{Y}^T_{vm}(k \mid k-1) & \mathbf{Y}_{mm}(k \mid k-1) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}^*_{mm}(k) \end{bmatrix} \tag{6.117}$$

$$\hat{\mathbf{y}}(k \mid k) = \begin{bmatrix} \hat{\mathbf{y}}_v(k \mid k-1) \\ \hat{\mathbf{y}}_m(k \mid k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{i}^*_m(k) \end{bmatrix}. \tag{6.118}$$

6.5.2 Decentralised Hybrid Space Formulation

Although the information formulation of decentralised SLAM has the map in a form which is easily communicated, it requires significant computation to invert the information matrix and extract the states if non-linear platforms models are employed. This occurs as the models are linearised around the current state estimate which is not readily available in an information form.

One method of reducing the computation is to note that the prediction stage of the filter will invariably be more frequent than an update with information from other nodes in the system. As a non-linear process model would require the inversion of the information matrix for every prediction, it is therefore usually simpler to formulate the SLAM problem in state space and only perform an inversion to information space when communicating with other nodes. The channel filter in this formulation is still left in information space. Using this method, the prediction and update with local observation can be done using the EKF Equations 6.11–6.12 and 6.30–6.33 respectively.

When information from another node is received, the channel filter operates using the algorithm defined in Section 3.4 and calculates the new map information increment $\mathbf{I}^*_{mm}(k)$ and $\mathbf{i}^*_m(k)$. At this time, the local SLAM estimator is converted from state space to information space and the update is performed using the information update given in Equations 6.117 and 6.118. After the update step, the local SLAM estimate is converted back to state space where the EKF continues to run.

6.5.3 Information Space SLAM Vs Hybrid Space SLAM

The two D-SLAM formulations given in Sections 6.5.1 and 6.5.2 are both numerically identical. The former, developed entirely in information space, allows the inter-vehicle communications to be integrated easily with the rest of the system as the map is always available in its information form. For a linear system, this formulation is ideal as it is not necessary to convert to state space at any time. However, if the system uses non-linear models as is usually the case, the information matrix must be constantly inverted to extract the state estimate. Given that the prediction stage on platform navigation filters often run at quite high rates, this can cause significant computational problems.

Recognising that inter-vehicle communication will usually be must less frequent than local predictions enables the D-SLAM filter to be recast into a hybrid state/information space filter which has the state estimates readily available. However, this does have the disadvantage that at every inter-vehicle communication step the state estimate is converted to information space and updated before returning to state space. Notwithstanding this, the hybrid form given in Section 6.5.2 is typically more efficient for a non-linear system as these inter-vehicle updates occur much less often than predictions.

The selection of which form of D-SLAM to use therefore depends on whether the system is linear. If it is, the entire problem can be left in information space using the method in Section 6.5.1. Otherwise, the hybrid form given in Section 6.5.2 is typically more efficient.

6.6 Scalable Communication with Large Maps

While the preceding sections have demonstrated that communicating map information results in a natural and elegant D-SLAM the problem remains of managing the complexity of communicating large maps between platforms. This section discusses this issue and presents a realistic and scalable solution to the problem.

The heart of a D-SLAM algorithm is the communication of map information between platforms. The simplest and most obvious method is to send the entire map. However, this leads to communication requirements of order N^2, where N is the number of features in the map. As the map size increases, this method will rapidly become impossible to maintain over any communication medium with a finite bandwidth. Some alternative method that communicates the 'most informative' map information given communication bandwidth limitations is therefore appropriate.

One possible algorithm is to simply send a subset of the total map (a submap) and to fuse this at the receiving node as if it were the entire map. However such communications ignore the correlation between this submap and the remaining map. If future communications only refer to this submap, then all will remain consistent. However if, in future other submaps are communicated, the correlation between submaps must be known if a consistent estimate is to be obtained. The problem is therefore how to send different submaps and fuse them consistently. The solution to this problem lies in the implementation of the channel filter. Since the channel filter keeps track of common information between nodes, it is necessary to

implement it in such a way that it does not pass inconsistent information from the network to the local filter. As it is the unknown cross-correlations that cause the problem, it is useful to exploit the well known advantage of Covariance Intersect (CI) to update pieces of information whose degree of correlation is unknown. By using a CI update in the channel filter, it is possible to handle different submaps simply and effectively.

The essential problem is to implement a a channel filter which is able to communicate and fuse submaps consistently. The approach taken here is to use the Covariance Intersect (CI). The CI method allows data with unknown correlation to be fused in a conservative and consistent manner. Implementing the CI method in the channel filter gurantees that the node filter only ever fuses consistent estimates. This structure allows submaps of arbitrary size to be consistently communicated between nodes.

6.6.1 Extracting the Submap to Communicate

An important consideration of decentralised SLAM is how to dynamically select which submaps to communicate. In information form, this problem has a natural solution in terms of information gain. Practically, map information in the channel filter is substracted from information at the local filter and those map elements whose information residual is the greatest are communicated. Doing this in the channel filter, where common information is maintained, has the effect of maximising information gain at the node where map data is to be sent. Other possible submap selection strategies are possible including, for example, requesting a submap in a specific area.

Once the submap for transmission have been selected, information from all of the other states in the map must be removed. This can be done by defining a projection matrix G_m such that $G_m\hat{\mathbf{y}}(k \mid k)$ extracts only those features in the map which are to be transmitted, and another projection G_v such that $G_v\hat{\mathbf{y}}(k \mid k)$ contains all other states. Using the notation $\mathbf{y}^*(k \mid k)$ and $\mathbf{Y}^*(k \mid k)$ to define the information vector and information matrix of the submap to be sent gives

$$\mathbf{y}^*(k \mid k) = G_m \left[\hat{\mathbf{y}}(k \mid k) - \mathbf{Y}(k \mid k)G_v^T \left(G_v\mathbf{Y}(k \mid k)G_v^T\right)^{-1} G_v\hat{\mathbf{y}}(k \mid k)\right] \quad (6.119)$$

$$\mathbf{Y}^*(k \mid k) = G_m \left[\mathbf{Y}(k \mid k) - \mathbf{Y}(k \mid k)G_v^T \left(G_v\mathbf{Y}(k \mid k)G_v^T\right)^{-1} G_v\mathbf{Y}(k \mid k)\right] G_m^T. \quad (6.120)$$

The small information submap given by $\mathbf{y}^*(k \mid k)$ and $\mathbf{Y}^*(k \mid k)$ is now sent to the channel filters prior to transmission to another node.

6.6.2 Channel Update

The channel filter maintains an estimate of the common information between nodes. In SLAM, this common information is simply those map elements that have been communicated. As map features are assumed stationary, the prediction step in the channel filter is redundant.

The channel filter update is computed using the CI algorithm to maintain consistency when communicating submaps without additional map coorelations. When a submap $\mathbf{y}^*(k \mid k)$ and $\mathbf{Y}^*(k \mid k)$ is received in a channel filter, the CI update may be accomplished by first back-projecting through the matrix G_m to the global map, and then updating according to

$$\mathbf{Y}_{Chan_i}(k \mid k) = \omega \mathbf{Y}_{Chan_i}(k \mid k-1) + (1-\omega)\, G_m^T \mathbf{Y}^*(k \mid k) G_m \tag{6.121}$$

$$\hat{\mathbf{y}}_{Chan_i}(k \mid k) = \omega \hat{\mathbf{y}}_{Chan_i}(k \mid k-1) + (1-\omega)\, G_m^T \mathbf{y}^*(k \mid k), \tag{6.122}$$

where ω, as in Section 2.5, is chosen to minimize the determinant of the updated information matrix.

The channel filter update is used to compute a new information increment resulting from a transmission as

$$\mathbf{I}_{ij}^*(k) = \mathbf{Y}_{Chan_j}(k \mid k) - \mathbf{Y}_{Chan_j}(k \mid k-1) \tag{6.123}$$

$$\mathbf{i}_{ij}^*(k) = \hat{\mathbf{y}}_{Chan_j}(k \mid k) - \hat{\mathbf{y}}_{Chan_j}(k \mid k-1). \tag{6.124}$$

These increments are then sent to the local node filter for fusion.

6.6.3 Fusing Information from Other Nodes at the Local Filter

At the local filter, the information increments $\mathbf{I}_{ij}^*(k)$ and $\mathbf{i}_{ij}^*(k)$ from the channel filter are first projected, through a matrix G_s, into the complete information state dimension. The

update (fusion) is then performed in the standard manner:

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + G_s \mathbf{i}^*_{ij}(k) \quad (6.125)$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + G_s \mathbf{I}^*_{ij}(k) G_s^T. \quad (6.126)$$

This update of the local filter is essentially the same as that given by Equations 6.117 and 6.118 for the case where the full map is sent. Both updates contain zero information for the vehicle and simply add in the map information increments. The only difference is the number of landmarks that are updated with new information.

6.7 Simulation

The D-SLAM algorithm developed in this thesis is demonstrated in simulation with a number of different communication strategies. These are:

1. No communication between platforms - each vehicle operates independently using only its own observations.
2. Transmission of the complete information map of N^2 elements at every communication step.
3. The constant time communications strategy where a submap of only 5 features is transmitted at each communication step. The features to include in the transmitted submap are selected at transmission time to be those which had the greatest amount of information not yet sent.

The simulation uses 100 vehicles moving in an environment of 100 features. Table 6.1 lists the bandwidth required by each of the communication strategies in this operating environment. Figure 6.4 illustrates this simulation world, marking the feature locations and a number of the vehicle paths. Platforms use the non-linear motion model presented in Section 6.2.2 to estimate their position and orientation $\begin{bmatrix} x & y & \phi \end{bmatrix}$. Each vehicle is equipped with a range/bearing sensor which gives observations to features in the forward hemisphere of the vehicle at a frequency of 1Hz. Vehicles are able to communicate map information to their neighbours in the network at a frequency of 0.1Hz.

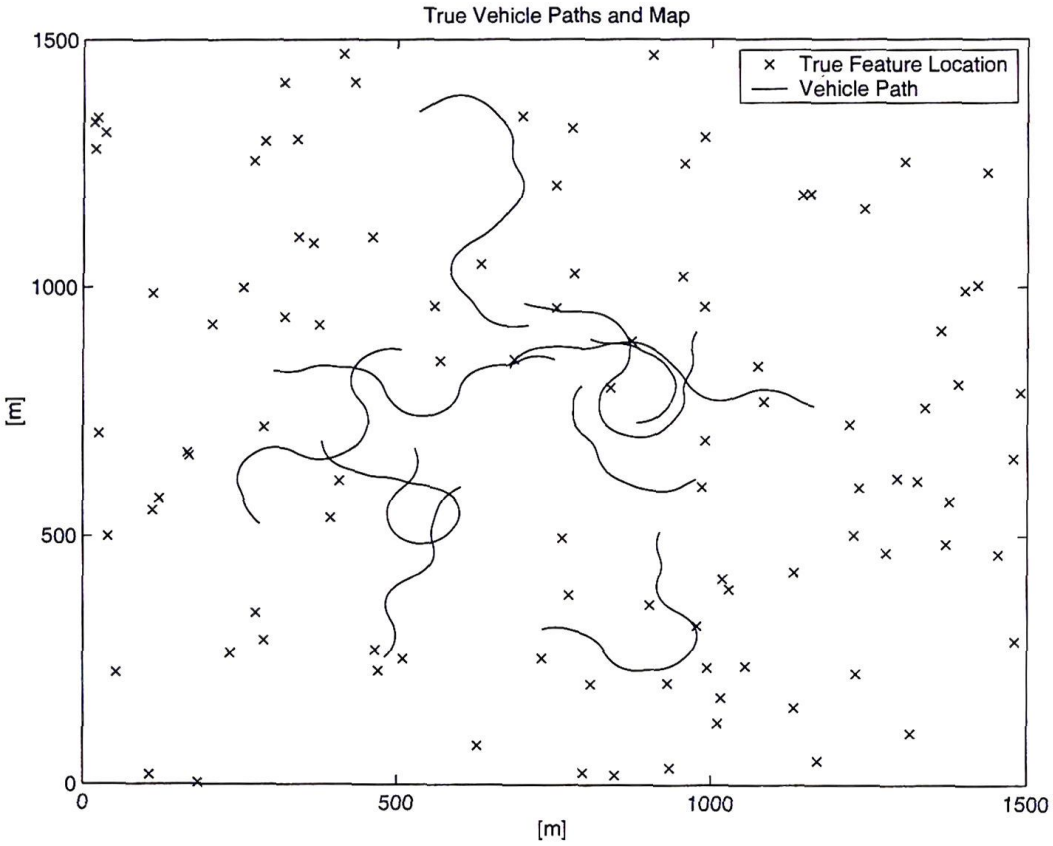


Figure 6.4: The simulation world. For clarity, only 10 of the 100 vehicle paths are shown.

Strategy	States to Communicate	Map Elements to Communicate
No Communication	0	0
Complete Map	200^2	40000
Submap of 5 features	10^2	100

Table 6.1: Number of elements to communicate in D-SLAM. For the problem presented here, both the x and y location of each landmark are estimated so there are 200 states for the map of 100 features.

6.7.1 Results

The results of the simulation are illustrated in Figures 6.5–6.12. Platform and landmark error and variance results are shown for a typical vehicle and feature in Figures 6.5–6.10. These illustrate that the different communication algorithms all operate logically and consistently. Unsurprisingly, platform and feature location errors are significantly lower when platforms communicate map information than when they do not. This can be seen as the error and 2σ bounds of both the vehicle and feature estimates are significantly smaller for the constant time method than when the platforms act independently. The N^2 communication strategy communicates more information again, so it follows that it is better again. Importantly, the constant time method is only marginally less accurate than the full N^2 algorithm, even though it communicates only 0.25% the data.

Figure 6.11 superimposes the estimate of a typical feature from each of the different communication strategies. Clearly, the covariances associated with this estimate differ significantly with different communication strategies. As expected, the covariance ellipse for the platforms acting independently is the largest, while the constant time strategy and N^2 strategy are both significantly better. Of these, the latter has the smallest error ellipse as it communicates the most information. However, it is again important to stress that the marginal improvement in the feature estimate using the N^2 algorithm over the constant time method comes at an enormous expense in terms of communication.

The results indicate that the constant time algorithm approaches the N^2 strategy quite quickly even though it is not transmitting as much information. This occurs as the submap that is transmitted is dynamically selected to contain those features which will give the greatest information gain. This trend illustrates the idea that good information about a small number of features increases the map accuracy significantly. This concept is further illustrated in Figure 6.12 which shows the mean variance of the vehicle states over the

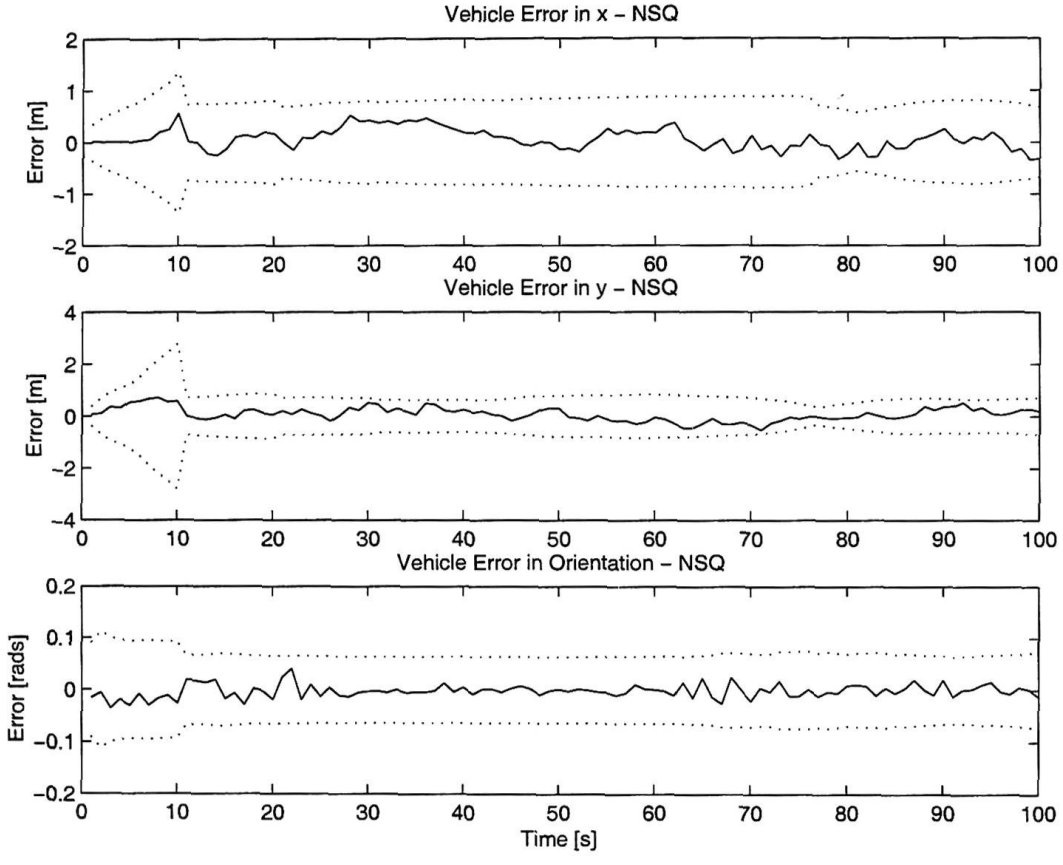


Figure 6.5: The error and 2σ bounds for the vehicle states for the N^2 communication strategy.

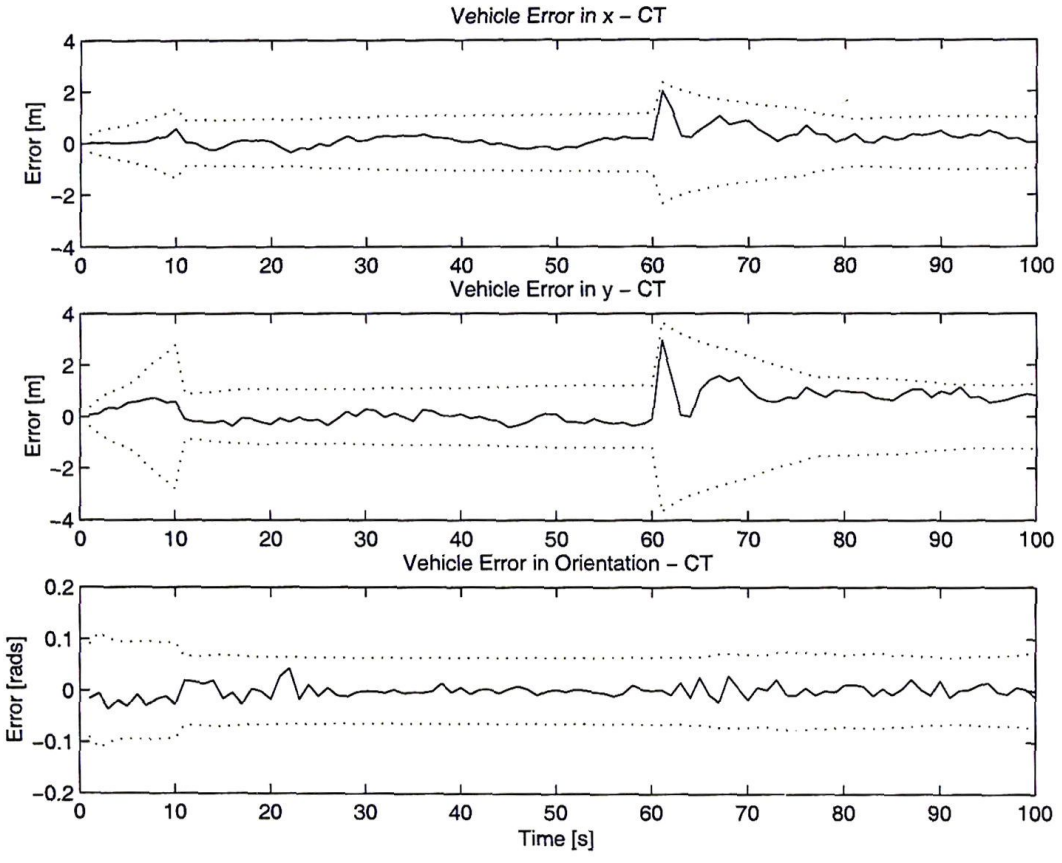


Figure 6.6: The error and 2σ bounds for the vehicle states for the constant time communication strategy.

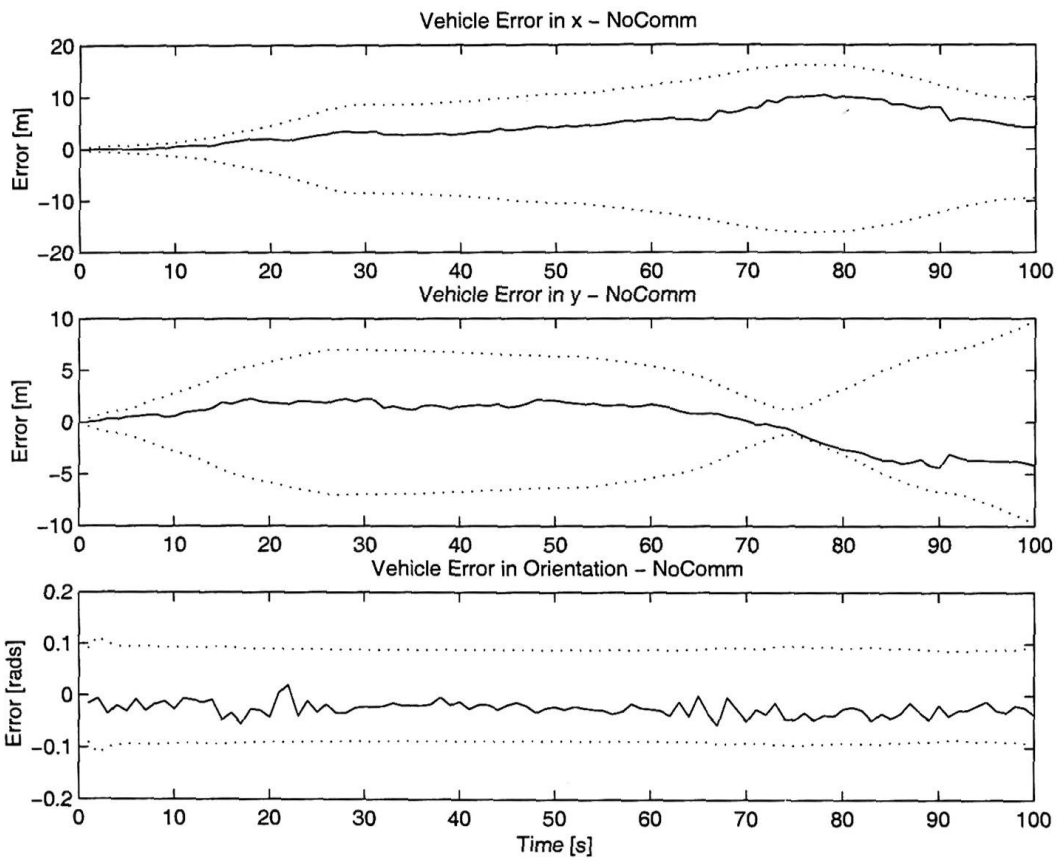


Figure 6.7: The error and 2σ bounds for the vehicle states when there was no communication between any vehicles.

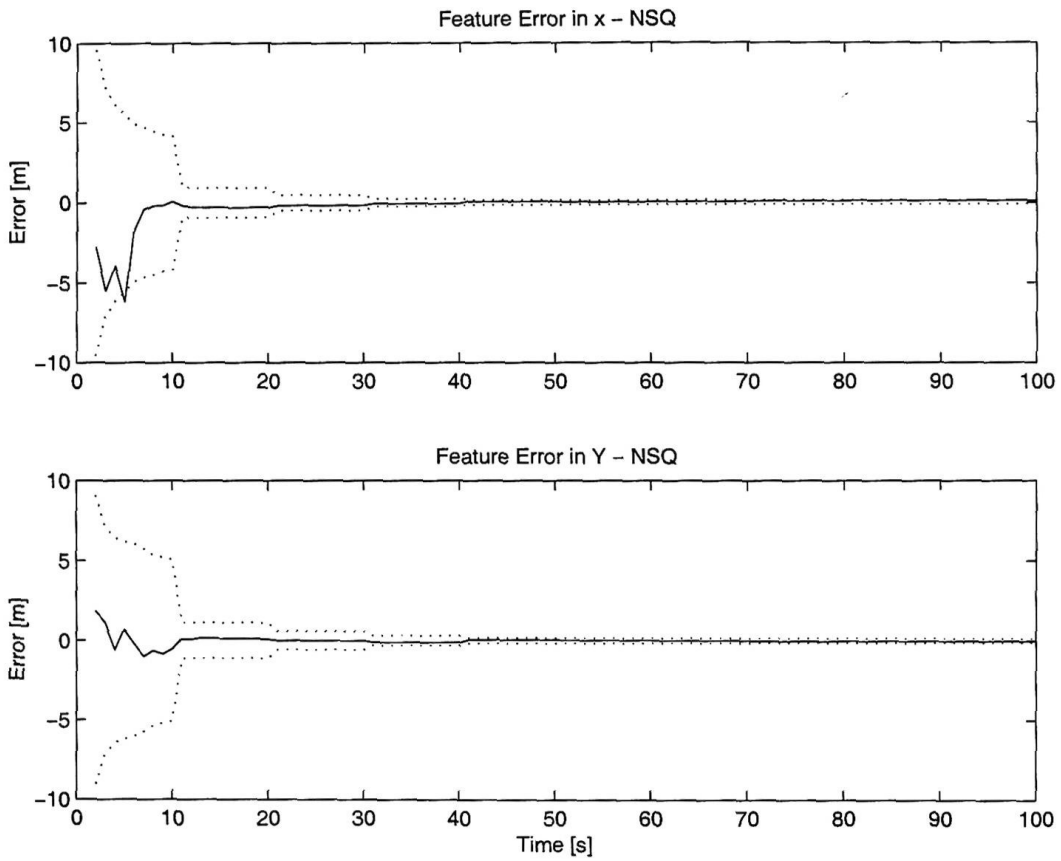


Figure 6.8: The error and 2σ bounds for a typical feature for the N^2 communication strategy.

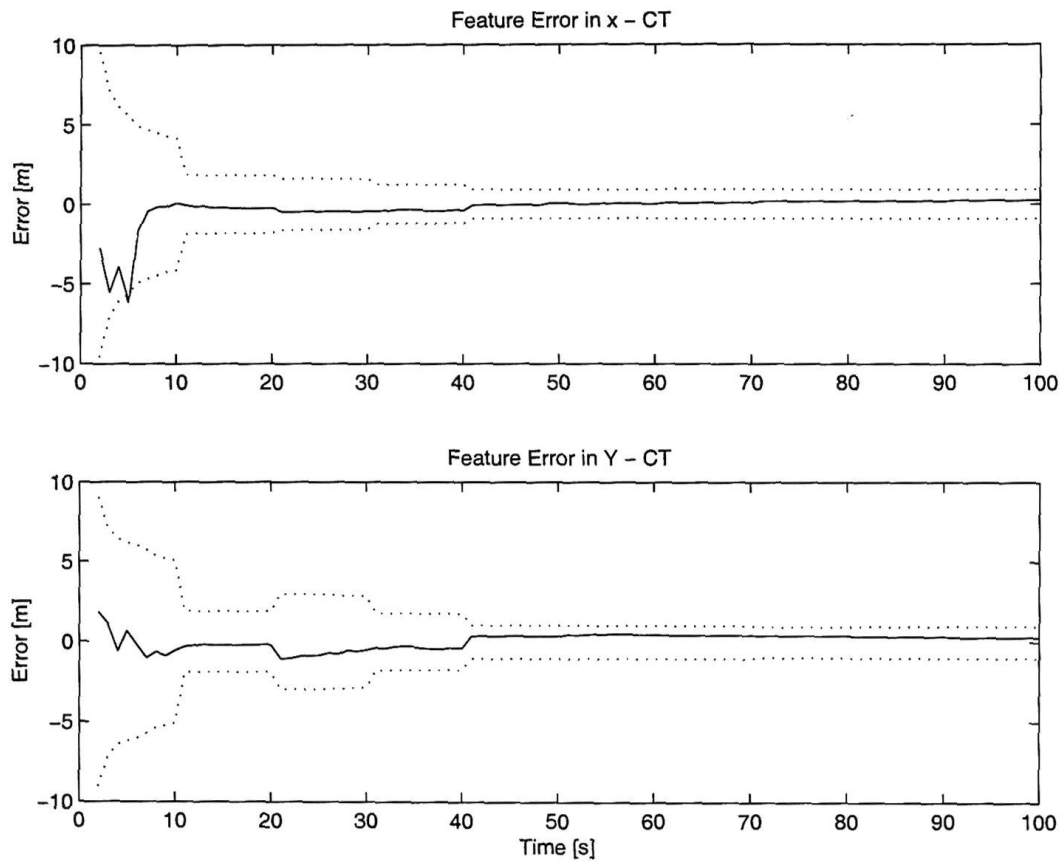


Figure 6.9: The error and 2σ bounds for a typical feature for the constant time communication strategy.

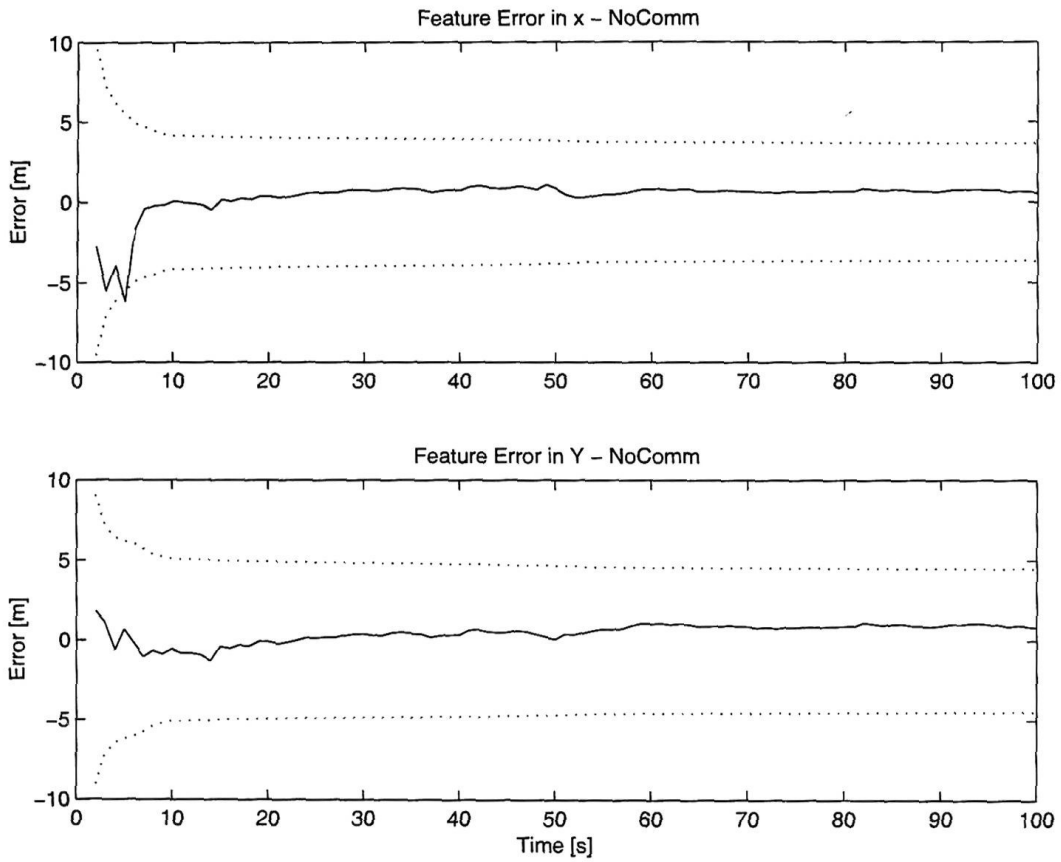


Figure 6.10: The error and 2σ bounds for a typical feature when there was no communication between vehicles.

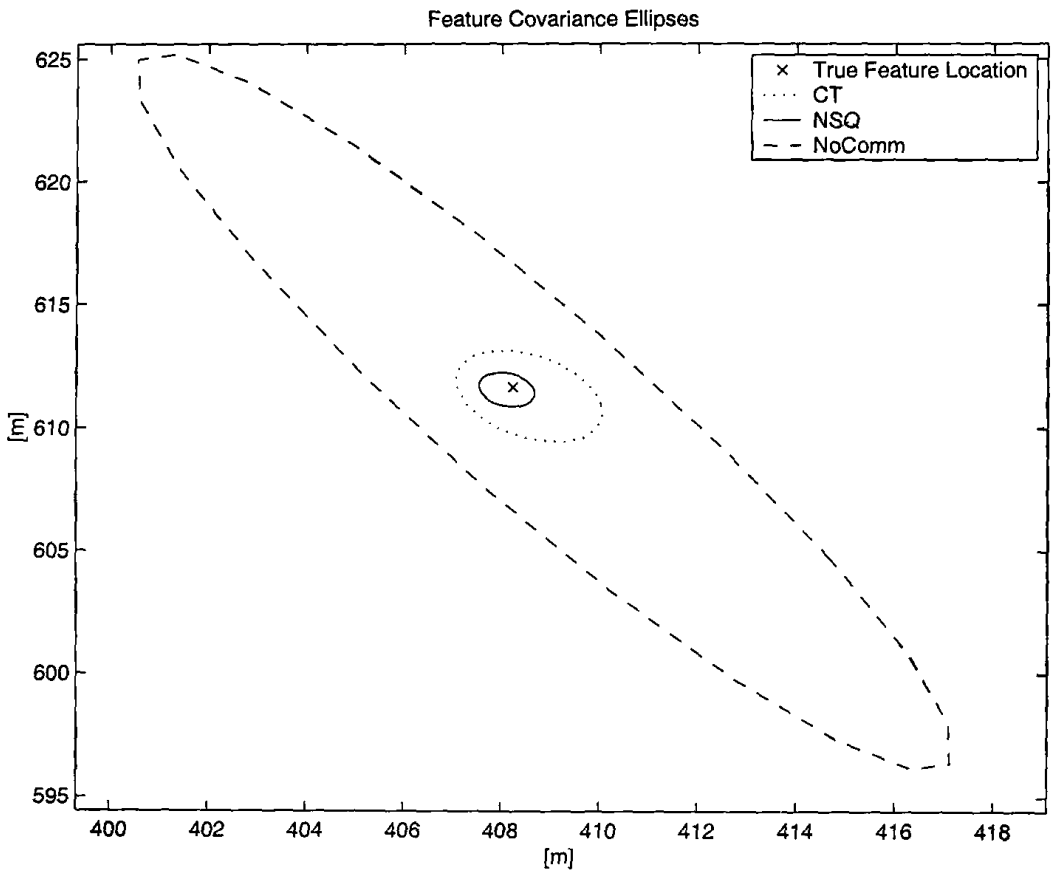


Figure 6.11: The covariance ellipse for a feature under the different communication strategies.

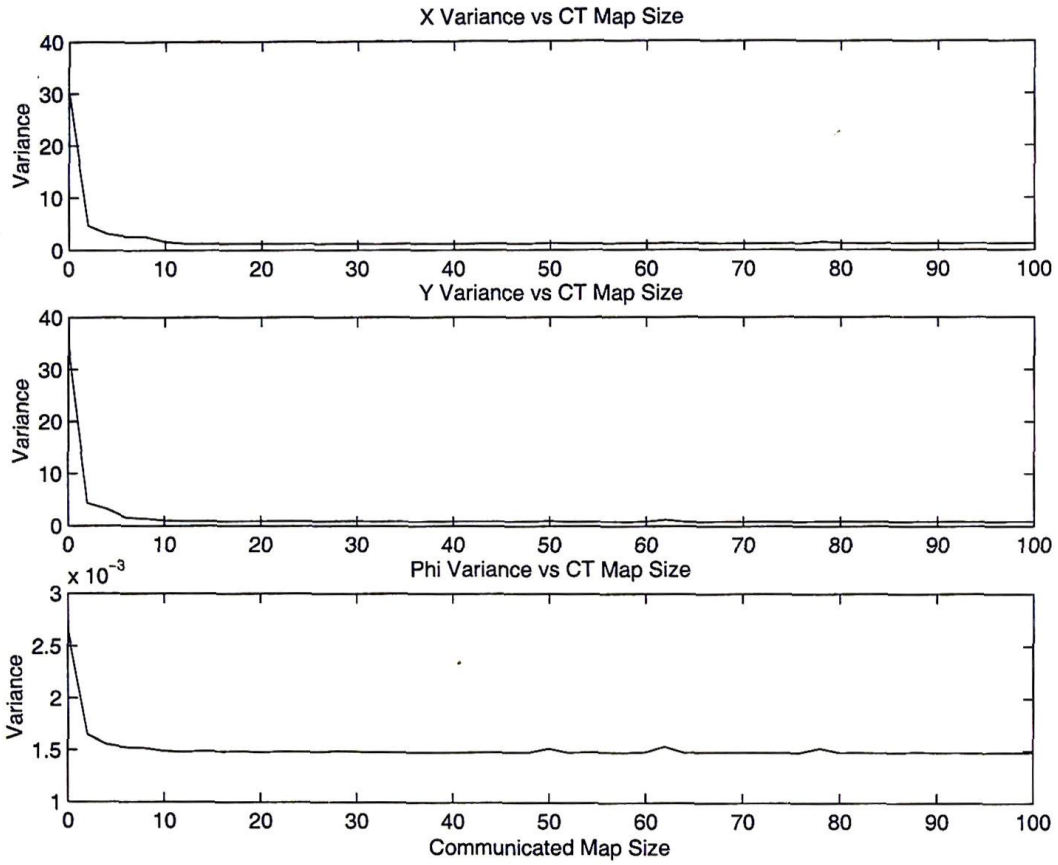


Figure 6.12: The mean variance in x, y and ϕ from all vehicles over the period of the simulation for different maximum map sizes using the constant time communications algorithm. This illustrates that the information gain obtained by sending greater map sizes decreases as the maps get larger.

entire simulation period for different communicated map sizes. While the vehicle variances are significantly lowered by communicating map information, the benefit of sending larger maps decreases asymptotically. This is to be expected as communication of the whole map will include features that have not been observed recently and which will therefore not contribute large amounts of new information. Communicating larger submaps using this constant time algorithm simply has the effect of allowing the estimate to converge toward the N^2 result faster.

By communicating map information, platforms can improve their self-location estimates. This has the knock-on effect of allowing new features to be initialised in to the map with lower uncertainty.

6.8 Summary

This Chapter has investigated the SLAM problem as an application of DDF. To provide a better understanding of the problem, closed form solutions to the continuous time single and multiple platform SLAM problem in both information and state-covariance form were presented. These solutions showed that the information form of the SLAM problem has two advantages: the cross-information between multiple platforms is zero and the global map estimate can be obtained simply by summing the individual maps from each platform. Using these results, a decentralised architecture for SLAM was developed that allows platforms to only communicate map information.

The D-SLAM architecture was formulated in both information space and a hybrid state/ information space. The former allows inter-platform communication to occur more easily, but can become computationally expensive when the system uses non-linear models. To reduce this problem, the hybrid space formulation was presented which allows non-linear systems to operate in state space for local observation and prediction operations, but performs inter-platform updates in information space.

To manage communications in D-SLAM systems with large maps, a communication scheme was presented which allows nodes to communicate submaps of an arbitrary size. Using this method allows platforms to continue sending submaps of a manageable size even as the map grows extremely large. Furthermore, the strategy uses information gains to dynami-

cally determine the features to transmit to other nodes at each communication step which maximises the amount of information in every message.

Simulated results were presented of a 100 platform D-SLAM system using different communication schemes. The results illustrate that using the DDF architecture improves the estimates of both the platform and the map when compared to the case where platforms act independently. The constant time communication scheme performed extremely well, producing estimates substantially better than independent operation and only marginally worse than communication of the complete map. It was also demonstrated that the value of sending larger submaps decreases asymptotically.

Chapter 7

Conclusions and Future Work

7.1 Introduction

This Chapter briefly summarises the principle contributions of this thesis and provides suggestions for future research directions. Section 7.2 summarises the main contributions in developing and demonstrating decentralised data fusion architectures. Section 7.3 describes possible new research directions.

7.2 Summary of Contributions

This thesis is concerned with the development and demonstration of decentralised data fusion algorithms for tracking and feature based navigation problems in an airborne environment. The principle contributions of the thesis are in the development of the DDF architecture and its application to real problems.

7.2.1 Decentralised Architectures

A substantial contribution of this thesis is the development of a general-purpose architecture for decentralised data fusion systems. This architecture focused on developing a structure for sensing nodes allowing local fusion of information and efficient and consistent communication in different networks. The architecture described provides for a modular design and permits unlimited scalability in a fully decentralised manner.

Both structured and unstructured communication topologies were considered. In structured networks, optimal data transmission and fusion algorithms were described. In unstructured networks it is known that no finite realisable method exists which can guarantee optimal decentralised estimates. A number of suboptimal unstructured network algorithms were developed, and in particular, a scalable architecture for a dynamic tree connected DDF system was presented. This architecture combines the advantages of the tree topology with the robustness and flexibility of a dynamic network. Nodes can therefore form into a topology which will make maximum use of network information wherever possible.

7.2.2 Delayed and Asequent Data

Exact solutions to both the delayed and asequent data problems were presented for the information form of the Kalman filter. These solutions allow the information filter to consistently use observation information that arrives both late and out of sequence. These algorithms are essential in realistic operational environments, such as the airborne domain considered in this thesis, where transmission delays and intermittent communication occur.

7.2.3 Decentralised Tracking

A decentralised tracking and tactical picture compilation system was presented and implemented in real-time on multiple UAVs. This implementation is probably one of the most complex and demanding operating environments in which to develop and demonstrate decentralised data fusion methods. The implementation uses a full six-degree of freedom platform model together with range and bearing information from an on-board visual sensor. The platforms themselves are registered with a separate GPS/INS navigation system.

The system is demonstrated in real-time using two aircraft and two ground stations in a five node DDF configuration. The aircraft, using the onboard vision and laser payload sensors, are able to maintain tracks to multiple ground targets and are able to communicate to and fuse information from other nodes in the network. This ensures that even the ground nodes, which have no sensor attached, benefit from having real-time estimates of target states. The results of this demonstration illustrate and confirm the properties of the DDF architecture in correctly propagating and fusing information in a network. The real-time implementation of the DDF method is a substantial contribution of this thesis. The results

are believed to be the first ever flight of a set of cooperative UAVs. Registration and bias problems were identified in the initial tracking results. However, these were identified and corrected and subsequent real-time flights performed extremely well.

7.2.4 Decentralised SLAM

In this thesis the SLAM problem is considered as an application of DDF. Closed form solutions to the one degree of freedom continuous time information matrix were presented for both single and multiple platform scenarios. A closed form solution to the multiple vehicle covariance matrix for the same problem was also described. The structure of the problem revealed by these solutions illustrates two important points:

1. The cross information between multiple vehicles is zero provided the vehicles do not sense each other.
2. The global map in D-SLAM is simply the sum of the map information from each node within the system.

These two points enabled a D-SLAM architecture to be developed in which vehicles communicate map information between one another. The decentralised architecture was initially formulated in information space enabling map information to be maintained in a readily available form for communication. However, as this formulation requires a computationally expensive inversion operation to obtain state estimates when non-linear models are employed, an equivalent hybrid state/information space formulation is also presented.

To ensure that communications in the D-SLAM algorithm remain scalable, a constant time communication algorithm was presented. This uses a hybrid information filter/covariance intersect method to ensure consistent sub-map fusion. This method allows nodes to dynamically select a submap of features to communicate to other nodes. The size of this submap is not fixed and can be dynamically selected based on the requirements of the system and the available bandwidth. The selection of landmarks to be included in these submaps is based on maximising the information value of communicated messages.

The D-SLAM method is one of the first consistent algorithms for performing multiple vehicle SLAM and is a substantial contribution made in this thesis. It shows the enormous

potential value in communicating map information between platforms. It also shows the fundamental nature of using information measures as the primary means of communicating and fusing map estimates. While the implementation and simulations described are somewhat preliminary, they show the way to future development of multi-platform SLAM systems.

7.3 Further Research

This thesis has presented a comprehensive theory and experimental results supporting the use of decentralised estimation methods in airborne sensor networks. The results in this thesis are limited, especially in regard of up-front sensor processing, and could be significantly improved in future demonstrations. Some of these improvements are described here. The decentralised data fusion method also gives rise to related questions in the development of analogous decentralised control and network management issues. Some of these future directions are also discussed.

7.3.1 System Complexity

The focus of the work presented in this thesis is on the design and implementation of a robust real time decentralised system in an airborne environment. As such, most of the complexity is directed at architecture and communications systems rather than target registration, models, data association and feature extraction. However, with the robustness and integrity of the decentralised architecture now demonstrated, future efforts should be made to add more detail and complexity into these areas.

7.3.2 Real Time Decentralised SLAM

The D-SLAM algorithm which is presented and shown in simulation can be implemented in real time. At the time of writing this thesis, single vehicle airborne SLAM on the ANSER project is undergoing testing prior to a real-time flight. Once this has been accomplished, the D-SLAM algorithm is to be implemented.

7.3.3 Decentralised Control

A decentralised control layer would integrate neatly with the decentralised estimation problems presented in this thesis. Currently, the vehicles use each others information to obtain a better, more robust estimate of some state. However, it is possible to integrate this with a decentralised control strategy to determine platform control actions which maximise some criteria specific to the common task. For example, in an information gathering task platforms could be directed to regions which maximise the information gain of the system. Some work has been done on this by Grocholsky [45], however it remains an open issue.

7.3.4 Dynamic Networks

A demonstration of a dynamic network architecture is an important step which should be included in the next generation DDF implementation. The concept has already been designed in theory and tested in simulation (see Section 3.6), and the next logical progression is to run such a system in real-time.

The implementation of this type of architecture is significant as it maintains all of the advantages of the DDF architecture demonstrated in this thesis, with the additional benefit of complete flexibility in network topology. This architecture would be the most useful in any large scale operational environment as it allows nodes to come and go freely.

7.4 Summary

This thesis has provided a significant contribution to the development and eventual deployment of decentralised data fusion systems in real problems. With an increasingly large number of applications needing to realise the benefits of large numbers of distributed sensors, the DDF method provides an important starting point for designing and implementing algorithms capable of delivering robust and scalable sensor networks.

Appendix A

Normalised Innovation: The Information Gate

The information gate is the information space equivalent of the state space innovation gate. The information observation quantities $\mathbf{i}(k)$ and $\mathbf{I}(k)$ are of dimension the state space, whereas the innovation and innovation variance are of dimension the observation space. In the innovation gate, the inverse innovation covariance is used to normalise the gate. In the information gate, the inverse of the corresponding information matrix is required. This, however will generally be singular as it is of dimension the state but has rank of only observation dimension. A generalised inverse $\mathbf{I}^+(k)$ is therefore defined in the following manner.

$$\mathbf{I}(k)\mathbf{I}^+(k) = \mathbf{E}, \tag{A.1}$$

where $\mathbf{E}$ is an idempotent matrix which acts as the identity for both $\mathbf{I}(k)$ and $\mathbf{I}^+(k)$,

$$\mathbf{I}(k)\mathbf{E} = \mathbf{I}(k), \qquad \mathbf{I}^+(k)\mathbf{E} = \mathbf{I}^+(k), \tag{A.2}$$

and

$$\mathbf{I}(k)\mathbf{I}^+(k)\mathbf{I}(k) = \mathbf{I}(k), \qquad \mathbf{I}^+(k)\mathbf{I}(k)\mathbf{I}^+(k) = \mathbf{I}^+(k). \tag{A.3}$$

One generalised inverse which satisfies these requirements is calculated by exploiting the observation model $\mathbf{H}_k$ as a projection operator. This matrix projects a state space into an observation space and conversely it's transpose projects observations back to state space.

The generalised inverse is then

$$\mathbf{I}^{+}(k) = \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{I}(k) \mathbf{H}_k^T\right]^{-1} \mathbf{H}_k. \tag{A.4}$$

The appropriateness of this selection of projection matrix is apparent as

$$\mathbf{H}_k \mathbf{I}(k) \mathbf{I}^{+}(k) = \mathbf{H}_k \mathbf{I}(k) \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{I}(k) \mathbf{H}_k^T\right]^{-1} \mathbf{H}_k = \mathbf{H}_k. \tag{A.5}$$

The innovation, defined as the the difference between the observed and predicted observation and given in Equation A.6 below, is used for data association

$$\nu(k) = \mathbf{z}(k) - \mathbf{H}_k \hat{\mathbf{x}}(k \mid k-1). \tag{A.6}$$

Written in its equivalent information form, this information residual vector is given by

$$\mathbf{v}(k) \triangleq \mathbf{H}_k^T \mathbf{R}_k^{-1} \nu(k). \tag{A.7}$$

Substituting Equation A.6 into Equation A.7 gives

$$\begin{aligned} \mathbf{v}(k) &= \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{z}(k) - \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \hat{\mathbf{x}}(k \mid k-1) \\ &= \mathbf{i}(k) - \mathbf{I}(k) \mathbf{Y}^{-1}(k \mid k-1) \hat{\mathbf{y}}(k \mid k-1). \end{aligned} \tag{A.8}$$

The covariance of this information residual is now calculated from $\mathrm{E}\{\mathbf{v}(k), \mathbf{v}^T(k) \mid \mathbf{Z}^{k-1}(k)\}$ as

$$\begin{aligned} \mathbf{B}(k) &= \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathrm{E}\{\nu(k) \nu^T(k) \mid \mathbf{Z}^k\} \, \mathbf{R}_k^{-1} \mathbf{H}_k \\ &= \mathbf{H}_k^T \mathbf{R}_k^{-1} \left[\mathbf{H}_k^T \mathbf{P}(k \mid k-1) \mathbf{H}_k + \mathbf{R}_k\right] \mathbf{R}_k^{-1} \mathbf{H}_k \\ &= \mathbf{I}(k) + \mathbf{I}(k) \mathbf{Y}^{-1}(k \mid k-1) \mathbf{I}(k) \\ &= \mathbf{I}(k) \left[\mathbf{I}^{+}(k) + \mathbf{Y}^{-1}(k \mid k-1)\right]^{-1} \mathbf{I}(k). \end{aligned} \tag{A.9}$$

The normalised information residual is now given by

$$\boldsymbol{\Gamma}(k) = \mathbf{v}^T(k) \mathbf{B}^{+}(k) \mathbf{v}(k), \tag{A.10}$$

where $\mathbf{B}^+(k)$ is the generalised inverse of $\mathbf{B}(k)$ and is again calculated using the projection operation $\mathbf{H}_k$ as

$$\mathbf{B}^+(k) = \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{B}(k) \mathbf{H}_k^T\right]^{-1} \mathbf{H}_k. \tag{A.11}$$

The relationship between the information gate to the state space innovation gate can be illustrated by noting that Equation A.11 can also be written

$$\begin{aligned} \mathbf{B}^+(k) &= \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{B}(k) \mathbf{H}_k^T\right]^{-1} \mathbf{H}_k \\ &= \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{S}_k \mathbf{R}_k^{-1} \mathbf{H}_k \mathbf{H}_k^T\right]^{-1} \mathbf{H}_k \\ &= \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{H}_k^T\right]^{-1} \mathbf{R}_k \mathbf{S}_k^{-1} \mathbf{R}_k \left[\mathbf{H}_k \mathbf{H}_k^T\right]^{-1} \mathbf{H}_k. \end{aligned} \tag{A.12}$$

Substituting Equations A.7 and A.12 into Equation A.10 gives

$$\begin{aligned} \mathbf{v}^T(k) \mathbf{B}^+(k) \mathbf{v}^T(k) &= \nu^T(k) \mathbf{R}_k^{-1} \mathbf{H}_k \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{H}_k^T\right]^{-1} \mathbf{R}_k \mathbf{S}_k^{-1} \mathbf{R}_k \left[\mathbf{H}_k \mathbf{H}_k^T\right]^{-1} \mathbf{H}_k \mathbf{H}_k^T \mathbf{R}_k^{-1} \nu(k) \\ &= \nu^T(k) \mathbf{S}_k^{-1} \nu(k). \end{aligned} \tag{A.13}$$

The normalised information residual is thus an information form of the conventional innovation residual.

Bibliography

[1] D. Alberts, J. Garstka, and P. Stein. *Network Centric Warfare: Developing and Leveraging Information Superiority.* C4ISR Cooperative Research Program (CCRP), http://www.dodccrp.org/NCW/ncw.html, 1999.

[2] B. Anderson. *Optimal Control.* Prentice Hall International, Englewood Cliffs, 1990.

[3] B. Anderson and J. Moore. *Optimal Filtering.* Prentice Hall International, Englewood Cliffs, 1979.

[4] Y. Bar-Shalom. On the track-to-track correlation problem. In *IEEE Trans. Automatic Control*, volume 25(8), pages 802–807, 1981.

[5] Y. Bar-Shalom. *Multitarget-Multisensor Tracking: Advanced Applications.* Artech House, Norwood, 1990.

[6] Y. Bar-Shalom. *Multitarget-Multisensor Tracking: Applications and Advances Volume 2.* Artech House, Norwood, 1992.

[7] Y. Bar-Shalom. Update with out-of-sequence measurements in tracking: Exact solution. In O.E. Drummond, editor, *Signal and Data Processing of Small Targets 2000*, volume 4048, pages 541–556, Bellingham, 2000.

[8] Y. Bar-Shalom and W. Blair. *Multitarget-Multisensor Tracking: Applications and Advances Volume 3.* Artech House, Norwood, 2000.

[9] Y. Bar-Shalom and X. Li. *Multitarget-Multisensor Tracking: Principles and Techniques.* Artech House, Norwood, 1995.

[10] P. Bauer, M. Sichitiu, R. Istepanian, and K. Premaratne. The mobile patient: Wireless distributed sensor networks for patient monitoring and care. In *Proc. of the IEEE Conference on Information Technology Applications in Biomedicine*, pages 17–21, 2000.

[11] T. Berg. *Model Distribution in Decentralised Multi-Sensor Data Fusion.* Ph.D Thesis, The University of Oxford, 1993.

[12] T. Berg and H. Durrant-Whyte. Model distribution in decentralised multi-sensor data fusion. In *Proceedings of the American Control Conference*, pages 2292–94, 1991.

[13] T. Berg and H. Durrant-Whyte. Model distribution in decentralised sensor networks. In *IEEE International Conference on Intelligent Control and Instrumentation*, 1992.

[14] T. Berg and H. Durrant-Whyte. On distributed and decentralised estimation. In *Proceedings of the American Control Conference*, pages 2273–4, 1992.

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

[16] S. Blackman. *Multiple-Target Tracking with Radar Application.* Artch House, Norwood, 1986.

[17] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems.* Artch House, Norwood, 1999.

[18] G. Brooker. *Technical Manual: ANSER Radar Node.* BAE Systems Australia, 2001.

[19] R. Brown and P. Hwang. *Introduction to Random Signals and Applied Kalman Filtering, Third Edition.* John Wiley and Sons, New York, 1997.

[20] T. Burke. *Design and Control of a Modular Wheeled Mobile Robot.* Ph.D Thesis, The University of Oxford, 1994.

[21] T. Burke and H. Durrant-Whyte. Design and control of a modular wheeled mobile robot. In *IEEE Conference on Intelligent Robotics*, 1993.

[22] N. Carlson. Federated square root filter for decentralized parallel processes. *IEEE Transactions on Aerospace and Electronic Systems*, 26(3):517–525, 1990.

[23] J. Castellanos, J. Montiel, J. Neira, and J. Tardos. Sensor influence in the performance of simultaneous mobile robot localization and map building. In *6th International Symposium on Experimental Robotics*, pages 203–212, Sydney, 1999.

[24] D. Catlin. *Estimation, Control and the Discrete Kalman Filter.* Springer Verlag, 1984.

[25] S. Challa, R. Evans, and X. Wang. A fixed lag smoothing framework for target tracking in clutter using out of sequence measurements. In *Proceedings of Defence Applications of Signal Processing (DASP)*, 2002.

[26] S. Challa, M. Palaniswami, and A. Shilton. Distributed data fusion using support vector machines. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 2, pages 881–885, Sunnyvale, 2002.

[27] K. Chang, R. Saha, and Y. Bar-Shalom. On optimal track-to-track fusion. *IEEE Transactions on Aerospace and Electronic Systems*, 33(4):1271–1276, 1997.

[28] K. Chang, Z. Tian, and R. Saha. Performance evaluation of track fusion with information matrix filter. *IEEE Transactions on Aerospace and Electronic Systems*, 38(2):455–466, 2002.

[29] C. Chong. Distributed architectures for data fusion. In *Proceedings First International Conference on Multisource-Multisensor Information Fusion*, pages 84–91, 1998.

[30] M. Chu, H. Haussecker, and F. Zhao. Scalable information driven sensor querying and routing for ad-hoc heterogeneous sensor networks. *International Journal of High Performance Computing Applications*, 16(3):293–314, 2002.

[31] L. Clare, G. Pottie, and J. Agre. Self-organizing distributed sensor networks. In *Unattended Ground Sensor Technologies and Applications*, volume 3713, pages 229–237, 1999.

[32] R. Deaves. Covariance bounds for augmented state kalman filter application. *Electronic Letters, 11th November 1999*, 35, No 23, 1999.

[33] R. Deaves. *The Management of Communications in Decentralised Bayesian Data Fusion Systems.* Ph.D Thesis, Bristol University, Dept of Electrical and Electronic Engineering, 1999.

[34] R. Deaves, D. Nicholson, D. Gough, L. Binns, P. Vangasse, and P. Greenway. Multiple robot system for decentralised SLAM investigations. In G.T. McKee and P.S. Schenker, editors, *Sensor Fusion and Decentralised Control in Robotic Systems III*, volume 4196, pages 360–369, Bellingham, 2000.

[35] M. Dissanayake, P. Newman, H. Durrant-Whyte, S. Clark, and M. Csorba. An experimental and theoretical investigation into simultaneous localisation and map building. In *6th International Symposium on Experimental Robotics*, pages 171–180, Sydney, 1999.

[36] H. Durrant-Whyte, R. Deaves, and P. Greenway. Decentralised multi-platform data fusion. In *Proc. SPIE*, volume 3393, pages 63–71, Bellingham, 1998.

[37] M. Fernandez. *Failure Detection and Isolation in Decentralised Multisensor Systems.* Ph.D Thesis, The University of Oxford, 1993.

[38] A. Gelb. *Applied Optimal Estimation.* MIT Press, Cambridge, 1974.

[39] P. Gibbens, M. Dissanayake, and H. Durrant-Whyte. A closed form solution to the single degree of freedom simultaneous localisation and map building (SLAM) problem. In *39th IEEE Conference on Decision and Control*, pages 191–196, Sydney, 2000.

[40] A. Göktoğan, E. Nettleton, M. Ridley, and S. Sukkarieh. Real time multi-UAV simulator. In *IEEE Conference on Robotics and Automation*, 2003.

[41] M. Grewal and A. Andrews. *Kalman Filtering Theory and Practice.* Prentice Hall, New Jersey, 1993.

[42] S. Grime. *Communication in Decentralised Sensing Architectures.* Ph.D Thesis, The University of Oxford, 1992.

[43] S. Grime and H. Durrant-Whyte. Data fusion in decentralized sensor networks. *Control Engineering Practice*, 2(5):849–863, 1994.

[44] B. Grocholsky. *Information-Theoretic Control of Multiple Sensor Platforms.* Ph.D Thesis, The University of Sydney, 2002.

[45] B. Grocholsky, H. Durrant-Whyte, and P. Gibbens. Information-theoretic approach to decentralised control of multiple autonomous flight vehicles. In G.T. McKee and P.S. Schenker, editors, *Sensor Fusion and Decentralised Control in Robotic Stystems III*, volume 4196, pages 348–359, Bellingham, 2000.

[46] L. Guibas and F. Zhao, editors. *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks.* Springer-Verlag, 2003.

[47] H. Hashemipour, S. Roy, and A. Laub. Decentralized structures for parallel kalman filtering. *IEEE Transactions on Automation and Control*, 33(1):1054–1061, 1988.

[48] O. Heikkilä, J. & Silvén. A four-step camera calibration procedure with implicit image correction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 1106–1112, 1997.

[49] P. Ho. *Organisation in Decentralised Sensing.* Ph.D Thesis, The University of Oxford, 1995.

[50] L. Hong, S. Cong, and D. Wicker. Distributed multirate interacting multiple model (DMRIMM) filtering with out-of-sequence gmti data. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 2, pages 1054–1061, Sunnyvale, 2002.

[51] B. Horling, R. Vincent, R. Mailler, J. Shen, R. Becker, K. Rawlins, and V. Lesser. Distributed sensor network for real time tracking. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 417–424. ACM Press, June 2001.

[52] Inertial Science Inc. *Technical Manual: Inertial Measurement Unit, ISIS Rev.C.* 1999.

[53] S. Julier and J. Uhlmann. A consistent, debiased method for converting between polar and cartesian coordinate systems. In *Proceedings of SPIE, Aerosense*, Orlando, 1997.

[54] S. Julier and J. Uhlmann. Building a million beacon map. In G. McKee and P. Schenker, editors, *Sensor Fusion and Decentralised Control in Robotic Stystems IV*, volume 4571, pages 1–9, Bellingham, 2001.

[55] S. Julier and J. Uhlmann. General decentralised data fusion with covariance intersection (CI). In D. Hall and J. Llinas, editors, *Handbook of Data Fusion.* CRC Press, 2001.

[56] R. Kalman. New methods in Weiner filtering theory. In *Transactions of the ASME*, volume 82, pages 34–45, 1960.

[57] R. Kalman. A new approach to linear filtering and prediction problems. In J. Bogdanoff and F. Kozin, editors, *Proceedings of the Symposium on the Engineering Applications of Random Function Theory and Probability*, 1963.

[58] J. Kim and S. Sukkarieh. Flight test results of GPS/INS navigation loop for an autonomous unmanned aerial vehicle (UAV). In *Proceedings of ION GPS*, pages 510–517, 2002.

[59] J. Kim and S. Sukkarieh. Airborne simultaneous localisation and map building. In *IEEE Conference on Robotic and Automation*, 2003.

[60] J. Leonard, R. Carpenter, and H. Feder. Stochastic mapping using forward looking sonar. In *International Conference on Field and Service Robotics*, pages 69–74, Pittsburg, 1999.

[61] J. Leonard and H. Durrant-Whyte. Simultaneous map building and localisation for an autonomous mobile robot. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 1442–1447, New York, 1991.

[62] J. Leonard and H. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In *Proceedings of the Ninth International Symposium on Robotics Research*, pages 169–176, 2000.

[63] J. Leonard, J.D. Tardós, S. Thrun, and H. Choset, editors. *Workshop Notes of the ICRA Workshop on Concurrent Mapping and Localization for Autonomous Mobile Robots (W4)*. IEEE Conference on Robotic and Automation, Washington, DC, 2002.

[64] J. Liu, J. Reich, and F. Zhao. Collaborative in-network processing for target tracking. *Journal on Applied Signal Processing*, 2002.

[65] Measurement Devices Ltd. *Technical Manual: ILM300HR OEM Industrial Laser Module.* 2001.

[66] S. Lu, A. Poore, and B. Suchomel. Network-centric mfa tracking architectures. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 1, pages 520–526, Sunnyvale, 2002.

[67] M. Mallick, J. Krant, and Y. Bar-Shalom. Multi-sensor multi-target tracking using out-of-sequence measurements. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 1, pages 135–142, 2002.

[68] J. Manyika. *An Information Theoretic Approach to Data Fusion and Sensor Management.* PhD Thesis, The University of Oxford, 1993.

[69] J. Manyika and H. Durrant-Whyte. Information as a basis for management and control in decentralised fusion architectures. In *IEEE Conference on Decision and Control*, 1992.

[70] J. Manyika and H. Durrant-Whyte. *Data Fusion and Sensor Management: An Information-Theoretic Approach.* Prentice Hall, 1994.

[71] P. Maybeck. *Stochastic Models, Estimation and Control, Volume 1.* Academic Press Inc, New York, 1979.

[72] S. Mitchell. *Two of a Kind Chat in the Blue.* The Australian, 20/8/2002.

[73] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localisation and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002.

[74] A. Mutambara. *Decentralised Control.* PhD Thesis, University of Oxford, 1995.

[75] A. Mutambara. *Decentralised Estimation and Control for Multisensor Systems.* CRC Press, 1998.

[76] A. Mutambara and H. Durrant-Whyte. Nonlinear information space: A practical basis for decentralisation. In *Sensor Fusion VII*, volume 2355, 1994.

[77] E. Nettleton and H. Durrant-Whyte. Delayed and asequent data in decentralised sensing networks. In G.T. McKee and P.S. Schenker, editors, *Sensor Fusion and Decentralised Control in Robotic Stystems IV*, volume 4571, pages 1–9, Bellingham, 2001.

[78] E. Nettleton, H. Durrant-Whyte, P. Gibbens, and A. Goktoğan. Multiple platform localisation and map building. In G.T. McKee and P.S. Schenker, editors, *Sensor Fusion and Decentralised Control in Robotic Stystems III*, volume 4196, pages 337–347, Bellingham, 2000.

[79] E. Nettleton, P. Gibbens, and H. Durrant-Whyte. Closed form solutions to the multiple platform simultaneous localisation and map building (SLAM) problem. In Bulur V. Dasarathy, editor, *Sensor Fusion: Architectures, Algorithms, and Applications IV*, volume 4051, pages 428–437, Bellingham, 2000.

[80] P. Newman. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem.* PhD Thesis, University of Sydney, 1999.

[81] D. Nicholson and R. Deaves. Decentralised track fusion in dynamic networks. In O.E. Drummond, editor, *Signal and Data Processing of Small Targets 2000*, volume 4048, Bellingham, 2000.

[82] D. Nicholson, C. Lloyd, S. Julier, and J. Uhlmann. Scalable distributed data fusion. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 1, pages 630–635, Sunnyvale, 2002.

[83] L. Pao and N. Baltz. Control of sensor information in distributed multisensor systems. In *Proceedings of American Control Conference*, pages 2397–2401, San Diego, 1999.

[84] A. Papoulis. *Probability and Statistics.* Prentice-Hall Inc., Englewood Cliffs, 1990.

[85] B. Rao. *Data Fusion Methods in Decentralised Sensing Systems.* PhD Thesis, The University of Oxford, 1991.

[86] B. Rao and H. Durrant-Whyte. A fully decentralized algorithm for multi-sensor Kalman filtering. In *IEE Transactions Schedule D*, volume 138(5), pages 413–420, 1991.

[87] B. Rao and H. Durrant-Whyte. A decentralised bayesian algorithm for identification of tracked targets. In *IEEE Transactions of Systems Man and Cybernetics*, volume 23(6), pages 1683–1698, 1993.

[88] B. Rao, H. Durrant-Whyte, and A. Sheen. A fully decentralised multi-sensor system for tracking and surveillance. In *International Journal of Robotics Research*, volume 12(1), pages 20–45, 1991.

[89] W. Reid. *Riccati Differential Equations.* Academic Press Inc., New York, 1972.

[90] W. Rencken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2192–7, 1993.

[91] M. Ridley, E. Nettleton, S. Sukkarieh, and H. Durrant-Whyte. Tracking in decentralised air-ground sensing networks. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 1, pages 616–623, Sunnyvale, 2002.

[92] R. Smith and P. Cheeseman. On the representation of spatial uncertainty. *Int J. Robotics Research*, 5(4):56–68, 1987.

[93] A. Stevens, M. Stevens, and H. Durrant-Whyte. Oxnav: Reliable autonomous navigation. In *Preceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2607–2612, 1995.

[94] L. Stone, C. Barlow, and T. Corwin. *Bayesian Multiple Target Tracking.* Artech House, Norwood, 1999.

[95] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of IEEE Conference on Robotics and Automation*, 2000.

[96] S. Thrun, D. Koller, Z. Ghahmarani, and H. Durrant-Whyte. SLAM updates require constant time. In *Proceedings of Fifth International Workshop on Algorithmic Foundations of Robotics*, 2002.

[97] B. Triggs. *The Wombat: Common Wombats in Australia.* University of New South Wales Press Ltd, 1988.

[98] J. Uhlmann. *Dynamic Map Building and Localisation, New Theoretical Foundations.* PhD Thesis, The University of Oxford, 1995.

[99] S. Utete. *A Network Manager for a Decentralized Sensing System.* Masters Thesis, The University of Oxford, 1992.

[100] S. Utete. *Network Management in Decentralized Sensing Systems.* PhD Thesis, The University of Oxford, 1994.

[101] S. Utete and H. Durrant-Whyte. Reliability in decentralised data fusion networks. In *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 215–221, 1994.

[102] UVOnline. *World First for Multiple UAV Applications.* www.uvonline.com, 7/8/2002.

[103] S. Williams. *Efficient Solutions to Autonomous Mapping and Navigation Problems.* PhD Thesis, The University of Sydney, 2001.