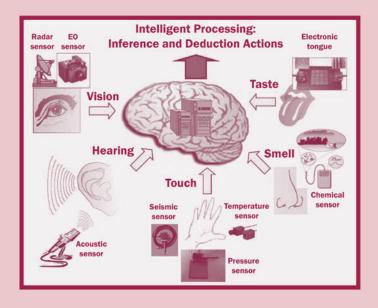
# Intelligent Systems -Fusion, Tracking and Control



G. W. Ng



# **Intelligent Systems - Fusion, Tracking and Control**

### CSI: CONTROL AND SIGNAL/IMAGE PROCESSING SERIES

Series Editor: **Dr Peter A. Cook** 

Control Systems Centre, UMIST, UK

- 1. Stabilisation of Nonlinear Systems: the piecewise linear approach **Christos A. Yfoulis**
- 2. Intelligent Systems Fusion, Tracking and Control **GeeWah Ng**
- \*3. Neural Network Control: Theory and Applications Sunan Huang, Kok Kiong Tan, Kok Zuea Tang
- \*4. Decision Support Systems for Tele-medicine Applications George-Peter K. Economou

\* Forthcoming

# **Intelligent Systems - Fusion, Tracking and Control**

## GeeWah Ng

DSO National Laboratories and National University of Singapore



### RESEARCH STUDIES PRESS LTD.

16 Coach House Cloisters, 10 Hitchin Street, Baldock, Hertfordshire, SG7 6AE, England www.research-studies-press.co.uk

and

Institute of **Physics** Publishing, Suite 929, The Public Ledger Building, 150 South Independence Mall West, Philadelphia, PA 19106, USA

Copyright © 2003, by Research Studies Press Ltd.

### Research Studies Press Ltd. is a partner imprint with the Institute of Physics Publishing

All rights reserved.

No part of this book may be reproduced by any means, nor transmitted, nor translated into a machine language without the written permission of the publisher.

### Marketing:

Institute of **Physics** PUBLISHING, Dirac House, Temple Back, Bristol, BS1 6BE, England www.bookmarkphysics.iop.org

#### Distribution:

#### NORTH AMERICA

AIDC, 50 Winter Sport Lane, PO Box 20, Williston, VT 05495-0020, USA Tel: 1-800 632 0880 or outside USA 1-802 862 0095, Fax: 802 864 7626, E-mail: orders@aidcvt.com

#### UK AND THE REST OF WORLD

Marston Book Services Ltd, P.O. Box 269, Abingdon, Oxfordshire, OX14 4YN, England Tel: +44 (0)1235 465500 Fax: +44 (0)1235 465555 E-mail: direct.order@marston.co.uk

### Library of Congress Cataloging-in-Publication Data

Ng, G. W. (Gee Wah), 1964-

Intelligent systems: fusion, tracking and control / GeeWah Ng. p. cm. -- (CSI, control and signal/image processing series)

ISBN 0-86380-277-X

1. Intelligent control systems. I. Title. II. Series.

TJ217.5.N4 2003

629.8--dc21

2003008304

### **British Library Cataloguing in Publication Data**

A catalogue record for this book is available from the British Library.

ISBN 0 86380 277 X

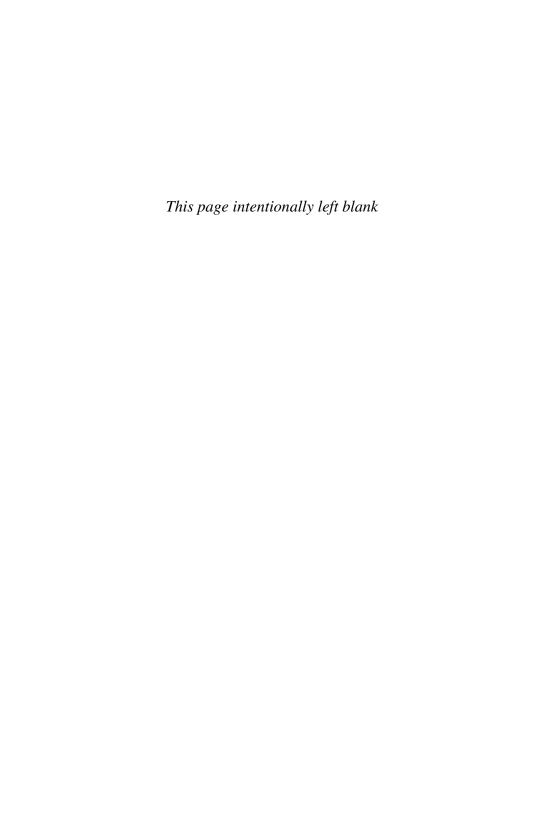
Printed in Great Britain by SRP Ltd., Exeter

# **Editorial Foreword**

This new series of books (CSI: Control and Signal/Image Processing Series) has its origin in the UMIST Control Systems Centre Series, previously edited by my colleagues Professor Peter Wellstead and Dr Martin Zarrop, which concentrated on making widely and rapidly available the results of research undertaken in the Centre. The aim of the new series, while continuing to concentrate on the areas of Control and Signal/Image Processing, is to provide a wider channel of publication for any authors who have novel and important research results, or critical surveys of particular topics, to communicate. My hope and intention is that the series will offer an interesting and useful resource to research workers, students and academics in the fields covered, as well as appealing to the engineering community in general.

The present volume, which is the second in the series, is concerned principally with the application of artificial intelligence to the combination of data arising from several different sources. Data fusion problems have arisen particularly in military applications, such as target identification and tracking, but are by no means confined to this area, being of increasing significance also for industry and commerce, especially with regard to the life sciences. Moreover, the methods used, based for example on neural networks, fuzzy logic and genetic algorithms, are being ever more widely used in data, signal and image processing. I therefore hope and expect that the book will prove to be of wide interest, and appeal to anyone interested in gaining an informed insight into the uses of intelligent systems.

Dr Peter A. Cook Series Editor



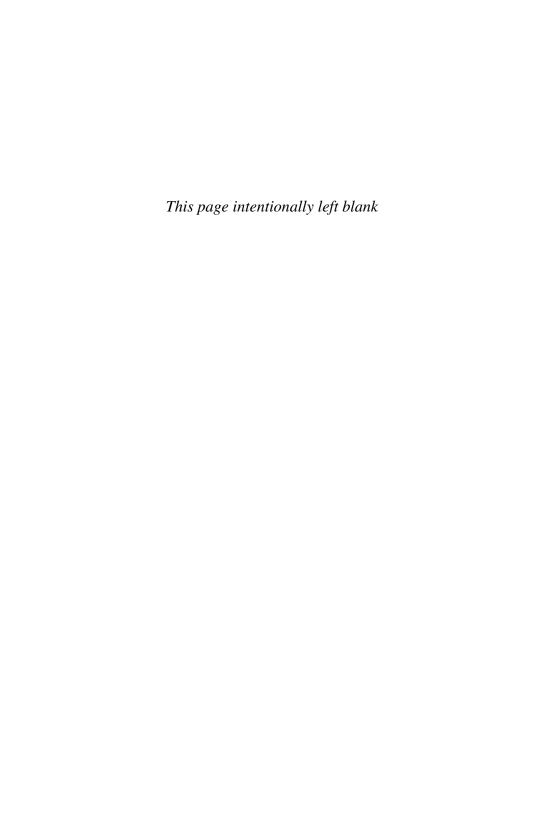
# **Preface**

There are growing interests in developing intelligent systems for civilian and military uses as well as for many other disciplines. The aim of this book is to discuss the design and application for building intelligent systems. Artificial intelligence (AI) algorithms have gone through more than 2 decades of research and experimentation. Researchers and engineers are seeing the results of applying AI to real applications. Today the quest continues, with researchers also looking into life sciences while attempting to understand natural intelligence, and adapting it to intelligent systems' design.

The book is divided into two parts. The first part focuses on the general discussion of what are the essential components of building intelligent systems. These include: the discussion on what is an intelligent system, how to build intelligence in software design and software agent perspective, what are some of the biologically inspired algorithms and man-made sensor systems.

The second part focuses on information processing capabilities. These include: discussion on fusion processes, cognitive intelligence, target tracking methods, data association techniques and controlling of sensor systems. The growing interest in research in cooperative intelligence is also discussed, particularly in the areas of cooperative sensing and tracking. With the technological advances in communication systems, processor design and intelligent software algorithms, systems working interactively with one another, in a coordinated manner, to fulfil desired global goals is becoming achievable. Some experimental research results on cooperative tracking are also shown.

G. W. Ng DSO National Laboratories National University of Singapore



# Acknowledgements

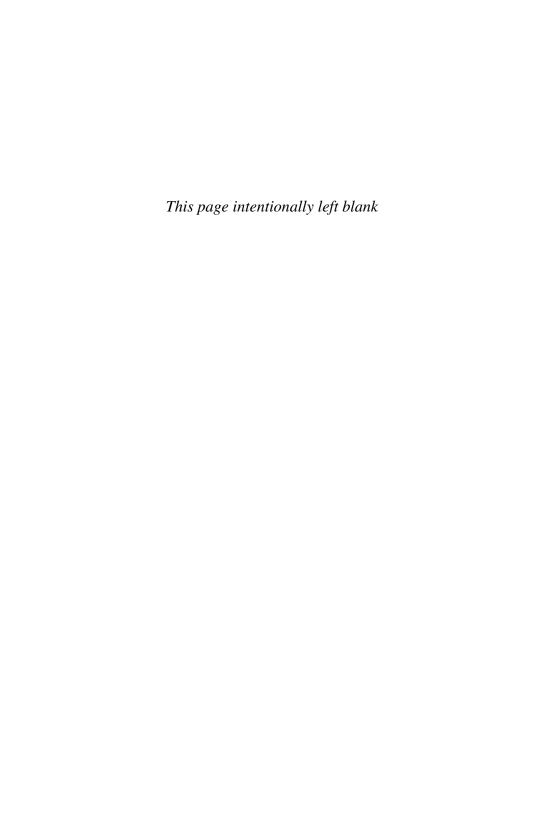
I am most grateful to the series editor (Dr Peter Cook) and publishing editor of Research Studies Press (Mr Giorgio Martinelli) for their patience and understanding in the course of writing this book. My thanks to DSO National Laboratories and National University of Singapore (NUS), where parts of these works were carried out. My thanks also to my colleagues in the Information Exploitation Programme and my students at NUS who have carried out some of the experiments and verification of the algorithms mentioned in the book. In particular, I would like to thank Yang Rong, Geraldine Chia, Khin Hua, Huey Ting, Kim Choo, Eva Lim, Ngai Hung, Wing Tai and Tran Francoise Hong Loan.

Special thanks to: Tuck Hong, Kok Meng and Kiam Tiam for their contribution in Chapter 2 of the book; Dr Tomas Eklov for sharing his work on chemical sensors; Josiah Tan for assistance in designing Figure 5.2; Chew Hung for sharing his work on ontology; Professor Martin G. Helander for sharing his work on naturalistic decision making.

Many thanks to all my colleagues and friends in DSO National Laboratories who make my work interesting; and colleagues in National University of Singapore, including Associate Professor Dennis Creamer and Dr Prasad Patnaik.

Dr John Leis has been of great help in formating the LaTex file of this book. Last but not least, I am truly indebted to my boss, Dr How Khee Yin (Head of Decision Support Centre, DSO National Laboratories) for the opportunity to research and discussion on data and information fusion issues.

This book is dedicated to my wife Jocelyn Tho and daughter Joy Ng. *It is a good thing to give thanks unto the LORD* ... (Psalm 92:1)



# **Contents**

| Ec | litori  | Foreword   | V     |
|----|---------|--|-------|
| Pr | eface   |  | vii   |
| Ac | knov    | edgements  | ix    |
| Li | st of l | igures   | xix   |
| Li | st of ' | ables  | xxiii |
| Ι  |         |  | 3     |
| 1  | Que     | t to Build Intelligent Systems                                 | 5     |
|    | 1.1     | Desire for an Intelligent System                               | 5     |
|    |         | 1.1.1 Will an intelligent system match the human intelligence? | 5     |
|    | 1.2     | What is an Intelligent System?                                 | 6     |
|    |         | 1.2.1 Examples of existing intelligent systems                 | 6     |
|    |         | 1.2.2 Learning from nature                                     | 7     |
|    |         | 1.2.3 How do we qualify a system as intelligent?               | 8     |
|    | 1.3     | What Makes an Intelligent System                               | 9     |
|    |         | 1.3.1 How an intelligent system sees the world?                | 10    |
|    | 1.4     | Challenges of Building Intelligent Systems                     | 11    |
| 2  | Soft    | vare Design for Intelligent Systems                            | 13    |
|    | 2.1     | Introduction   | 13    |

|   | 2.2 | Analys | and Des     | sign Using UML  | 14 |
|---|-----|--------|-------------|---|----|
|   |     | 2.2.1  | Diagrams    | s   | 14 |
|   |     | 2.2.2  | Relations   | ships   | 14 |
|   |     |        | 2.2.2.1     | Dependency  | 15 |
|   |     |        | 2.2.2.2     | Aggregation   | 15 |
|   |     |        | 2.2.2.3     | Association   | 16 |
|   |     |        | 2.2.2.4     | Generalization  | 16 |
|   | 2.3 | Data S | tructure an | nd Storage  | 17 |
|   |     | 2.3.1  | Data stru   | cture   | 18 |
|   | 2.4 | What i | s a Softwa  | re Agent-based System?                                  | 19 |
|   |     | 2.4.1  | What is a   | nn agent?   | 20 |
|   |     | 2.4.2  | What is a   | n multi-agent system?                                   | 21 |
|   |     |        | 2.4.2.1     | Multi-agent approach to distributed problem solving     | 21 |
|   |     |        | 2.4.2.2     | Multi-agent negotiation                                 | 21 |
|   | 2.5 | An Ag  | ent-Orient  | ed Software Development Methodology                     | 22 |
|   |     | 2.5.1  | Analysis    |   | 24 |
|   |     | 2.5.2  | Design      |   | 25 |
|   |     | 2.5.3  | Coding      |   | 25 |
|   |     | 2.5.4  | Testing     |   | 25 |
|   | 2.6 | Object | - and Ager  | nt-Oriented Software Engineering Issues                 | 25 |
| • |     |        |             |   | •  |
| 3 |     |        | -           | Algorithms  | 29 |
|   | 3.1 |        |             |   | 29 |
|   | 3.2 |        |             | mputing   | 30 |
|   |     | 3.2.1  |             | al inspiration  | 30 |
|   |     | 3.2.2  |             | l perspectives  | 31 |
|   |     | 3.2.3  |             | algorithms  | 31 |
|   |     |        |             | Introduction to GAs                                     | 31 |
|   |     |        | 3.2.3.2     | Operator  | 33 |
|   |     |        | 3.2.3.3     | Parameters of GAs                                       | 34 |
|   |     |        | 3.2.3.4     | Selection   | 35 |
|   |     |        | 3.2.3.5     | Further information on crossover and mutation operators | 35 |

|   |      |          |   | XIII |
|---|------|----------|---|------|
|   | 3.3  | Neural   | l Networks                                  | 36   |
|   |      | 3.3.1    | Biological inspiration                      | 36   |
|   |      | 3.3.2    | Historical perspective                      | 37   |
|   |      | 3.3.3    | Neural network structures                   | 39   |
|   |      |          | 3.3.3.1 Basic units                         | 39   |
|   |      |          | 3.3.3.2 Network topology                    | 40   |
|   |      | 3.3.4    | Learning algorithms                         | 43   |
|   |      |          | 3.3.4.1 Supervized learning                 | 43   |
|   |      |          | 3.3.4.2 First order gradient methods        | 45   |
|   |      |          | 3.3.4.3 Reinforcement learning              | 50   |
|   |      |          | 3.3.4.4 Unsupervized learning               | 52   |
|   |      | 3.3.5    | Training of neural networks                 | 54   |
|   | 3.4  | Fuzzy    | Logic                                       | 55   |
|   |      | 3.4.1    | Biological inspiration                      | 55   |
|   |      | 3.4.2    | Historical perspective                      | 55   |
|   |      | 3.4.3    | Basic elements of fuzzy systems             | 55   |
|   |      |          | 3.4.3.1 Fuzzification                       | 56   |
|   |      |          | 3.4.3.2 Fuzzy lingusitic inference          | 57   |
|   |      |          | 3.4.3.3 Defuzzification                     | 58   |
|   | 3.5  | Expert   | t Systems                                   | 59   |
|   |      | 3.5.1    | Knowledge engineering                       | 61   |
|   | 3.6  | Fusion   | of Biologically Inspired Algorithms         | 61   |
|   | 3.7  | Conclu   | usions                                      | 62   |
| 4 | Sens | sing Sys | stems for Intelligent Processing            | 63   |
|   | 4.1  | Introdu  | uction                                      | 63   |
|   | 4.2  | Classif  | fying Types of Sensor Systems               | 64   |
|   | 4.3  |          | ng Technology in Perspective                | 65   |
|   |      | 4.3.1    | Radar systems                               | 68   |
|   |      |          | 4.3.1.1 Imagery radar                       | 68   |
|   |      |          | 4.3.1.2 Moving target indicator (MTI) radar | 69   |
|   |      |          | 4.3.1.3 Doppler radar                       | 69   |
|   |      |          | 4.3.1.4 Millimeter-wave (MMW) radar         | 69   |
|   |      |          |   |      |

|    |            |   | 4.3.1.5 Laser radar   | 09                                     |
|----|------------|---|---|--|
|    |            | 4.3.2   | Electro-optic systems   | 70                                     |
|    |            |   | 4.3.2.1 Quantum detector  | 70                                     |
|    |            |   | 4.3.2.2 Thermal detector  | 70                                     |
|    |            | 4.3.3   | Acoustic and sonar sensor systems   | 70                                     |
|    |            | 4.3.4   | Magnetic sensor systems   | 71                                     |
|    |            |   | 4.3.4.1 Direct measurement of the magnetic field  | 71                                     |
|    |            |   | 4.3.4.2 Indirect measurement  | 71                                     |
|    |            | 4.3.5   | Chemical sensors  | 72                                     |
|    |            |   | 4.3.5.1 Gas sensors   | 72                                     |
|    |            | 4.3.6   | Microsensors  | 73                                     |
|    | 4.4        | Sensor  | System Design Issues  | 73                                     |
|    |            | 4.4.1   | Multiple sensor integration   | 74                                     |
|    |            | 4.4.2   | Sensing platforms   | 74                                     |
|    |            |   |   |  |
| II |            |   |   | 77                                     |
| 5  | Fusi       | on Syst   | ems for Intelligent Processing  | 79                                     |
|    | 5.1        | Data a  | nd Information Fusion System - What and Why   | 79                                     |
|    | 5.2        | What a  | 1 D'CC . I 1 CE ! D 0   | 82                                     |
|    |            |   | are the Different Levels of Fusion Process?   | 02                                     |
|    |            | 5.2.1   | Fusion complexity   | 86                                     |
|    | 5.3        | 5.2.1   |   |  |
|    | 5.3<br>5.4 | 5.2.1<br>Issues   | Fusion complexity   | 86                                     |
|    |            | 5.2.1<br>Issues   | Fusion complexity   | 86<br>86                               |
|    |            | 5.2.1<br>Issues<br>Cognit   | Fusion complexity   | 86<br>86<br>90                         |
|    |            | 5.2.1<br>Issues<br>Cognit<br>5.4.1<br>5.4.2                             | Fusion complexity   | 86<br>86<br>90<br>90                   |
|    | 5.4        | 5.2.1<br>Issues<br>Cognit<br>5.4.1<br>5.4.2                             | Fusion complexity   | 86<br>86<br>90<br>90<br>92             |
|    | 5.4        | 5.2.1<br>Issues<br>Cognit<br>5.4.1<br>5.4.2<br>Fusion                   | Fusion complexity   | 86<br>86<br>90<br>90<br>92<br>93       |
|    | 5.4        | 5.2.1<br>Issues<br>Cognit<br>5.4.1<br>5.4.2<br>Fusion<br>5.5.1          | Fusion complexity  on Fusing Multiple Sources  tive Intelligence  | 86<br>86<br>90<br>90<br>92<br>93       |
|    | 5.4        | 5.2.1<br>Issues<br>Cognit<br>5.4.1<br>5.4.2<br>Fusion<br>5.5.1<br>5.5.2 | Fusion complexity on Fusing Multiple Sources tive Intelligence How to make sense of information? Memory structure for cognitive intelligence processing System Architecture Centralized fusion architecture Decentralized fusion architecture                                 | 86<br>86<br>90<br>90<br>92<br>93<br>93 |
|    | 5.4        | 5.2.1 Issues Cognit 5.4.1 5.4.2 Fusion 5.5.1 5.5.2 5.5.3                | Fusion complexity on Fusing Multiple Sources tive Intelligence How to make sense of information? Memory structure for cognitive intelligence processing System Architecture Centralized fusion architecture Decentralized fusion architecture Distributed fusion architecture | 86<br>86<br>90<br>92<br>93<br>93<br>94 |

|   |     |         |  | XV  |
|---|-----|---------|--|-----|
|   | 5.6 | Techni  | ques for Data and Information Fusion   | 97  |
|   |     | 5.6.1   | Level 1 - Object assessment  | 97  |
|   |     |         | 5.6.1.1 Data association and correlation                                     | 97  |
|   |     |         | 5.6.1.2 State estimation   | 99  |
|   |     |         | 5.6.1.3 Classification, identification and identity                          | 100 |
|   |     | 5.6.2   | Level 2 and 3 - Situation and impact assessment                              | 100 |
|   |     |         | 5.6.2.1 Clustering and aggregation   | 101 |
|   |     |         | 5.6.2.2 Learning, reasoning and knowledge embedding and retrieval strategies | 102 |
|   |     | 5.6.3   | Remark on blending and fusing  | 104 |
|   | 5.7 | Compl   | ementary and Competitive Data  | 104 |
|   |     | 5.7.1   | Complementary data   | 105 |
|   |     | 5.7.2   | Competitive data   | 105 |
|   | 5.8 | Summ    | ary  | 105 |
| 6 | How | to Asso | ociate Data?   | 109 |
|   | 6.1 | Why D   | Oata Association   | 109 |
|   | 6.2 | Metho   | ds of Data Association   | 110 |
|   |     | 6.2.1   | Two-stage data association process   | 111 |
|   | 6.3 | Neares  | st Neighbour   | 112 |
|   | 6.4 | Multip  | le Hypotheses  | 113 |
|   |     | 6.4.1   | Multiple hypothesis tracking   | 113 |
|   |     |         | 6.4.1.1 Measurement-oriented approach  | 114 |
|   |     |         | 6.4.1.2 Track-oriented approach  | 115 |
|   |     | 6.4.2   | PMHT   | 115 |
|   | 6.5 | Probab  | oilistic Data Association  | 115 |
|   |     | 6.5.1   | Probabilistic data association filter (PDAF)                                 | 116 |
|   |     | 6.5.2   | Joint probabilistic data association filter (JPDAF)                          | 119 |
|   | 6.6 | Graph-  | -Based Data Association  | 121 |
|   |     | 6.6.1   | Association graph  | 122 |
|   |     |         | 6.6.1.1 Association strength   | 122 |
|   |     | 6.6.2   | Competing edges  | 123 |
|   |     |         | 6.6.2.1 The reduction algorithms   | 123 |

|   | 6.7 | Data A    | Association using Bio-inspired Algorithms                      | 125 |
|---|-----|-----------|--|-----|
|   |     | 6.7.1     | Neural data association methods                                | 125 |
|   |     | 6.7.2     | Fuzzy logic and knowledge-based data association methods       | 127 |
| 7 | Asp | ects of T | Target Tracking  | 129 |
|   | 7.1 | Introdu   | uction   | 129 |
|   | 7.2 | Basics    | of a Tracking Process  | 131 |
|   | 7.3 | Proble    | m Formulation  | 132 |
|   | 7.4 | Fixed-    | gain Filter  | 133 |
|   |     | 7.4.1     | What are the criteria for selecting the fixed gain?            | 135 |
|   | 7.5 | Kalma     | ın Filter  | 136 |
|   | 7.6 | Multip    | ole Model Approach   | 138 |
|   |     | 7.6.1     | IMM  | 139 |
|   |     |           | 7.6.1.1 Input mixer (Interaction)                              | 139 |
|   |     |           | 7.6.1.2 Filtering updates                                      | 140 |
|   |     |           | 7.6.1.3 Model probability evaluator                            | 141 |
|   |     |           | 7.6.1.4 Output mixer   | 141 |
|   |     | 7.6.2     | Comparison between IMM and Kalman filters                      | 145 |
|   |     |           | 7.6.2.1 Scenario   | 145 |
|   |     |           | 7.6.2.2 Filters  | 145 |
|   |     |           | 7.6.2.3 Results and analysis                                   | 146 |
|   |     |           | 7.6.2.4 Summary  | 147 |
|   |     | 7.6.3     | Comparison between IMM and the generic particle filter $$ . $$ | 148 |
|   | 7.7 | Issues    | Affecting Tracking Systems                                     | 149 |
|   |     | 7.7.1     | Coordinates  | 149 |
|   |     | 7.7.2     | Gating issues  | 150 |
|   | 7.8 | Conclu    | usions   | 151 |
| 8 | Coo | perativ   | e Intelligence   | 153 |
|   | 8.1 | What i    | is Cooperative Intelligence                                    | 153 |
|   |     | 8.1.1     | Characteristics of cooperative intelligence                    | 154 |
|   | 8.2 | Coope     | rative Sensing and Tracking                                    | 155 |
|   |     | 8.2.1     | Network of nodes   | 155 |
|   |     | 8.2.2     | Decentralized tracking   | 156 |

|   |      |         |   | xvii |
|---|------|---------|---|------|
|   | 8.3  | Measu   | rement and Information Fusion   | 157  |
|   |      | 8.3.1   | Decentralized Kalman filter   | 158  |
|   |      |         | 8.3.1.1 Formulation of a global state estimate                          | 159  |
|   |      | 8.3.2   | Information filter  | 161  |
|   |      |         | 8.3.2.1 Implementation of the decentralized information filter equation | 163  |
|   |      | 8.3.3   | Measurement Fusion  | 164  |
|   | 8.4  | Track   | or State Vector Fusion  | 165  |
|   |      | 8.4.1   | Convex combination  | 166  |
|   |      | 8.4.2   | Covariance intersection   | 167  |
|   |      | 8.4.3   | Best linear unbiased estimate   | 168  |
|   |      | 8.4.4   | Bar-Shalom and Campo combination  | 168  |
|   |      | 8.4.5   | Information de-correlation  | 169  |
|   | 8.5  | Experi  | ments on Decentralized Tracking   | 170  |
|   |      | 8.5.1   | Target and sensor modelling   | 170  |
|   |      | 8.5.2   | Scenario and simulation test  | 171  |
|   |      | 8.5.3   | Experiment results  | 171  |
|   | 8.6  | Summ    | ary   | 175  |
| 9 | Sens | or Mar  | nagement and Control  | 177  |
|   | 9.1  | Introdu | uction  | 177  |
|   | 9.2  | Multi-  | level Sensor Management Based on Functionality                          | 179  |
|   | 9.3  | Sensor  | Management Techniques   | 181  |
|   |      | 9.3.1   | Scheduling techniques   | 182  |
|   |      | 9.3.2   | Decision-making techniques  | 184  |
|   |      |         | 9.3.2.1 Normative technique   | 184  |
|   |      |         | 9.3.2.2 Descriptive technique   | 184  |
|   |      |         | 9.3.2.3 Hybrids of normative and descriptive techniques .               | 185  |
|   |      |         | 9.3.2.4 Other decision-making tools                                     | 185  |
|   | 9.4  | Sensor  | Management - Controller and Cueing                                      | 186  |
|   | 9.5  | Simula  | ation Results   | 187  |
|   |      | 9.5.1   | Fuzzy controller  | 187  |
|   |      | 9.5.2   | Example 1   | 189  |

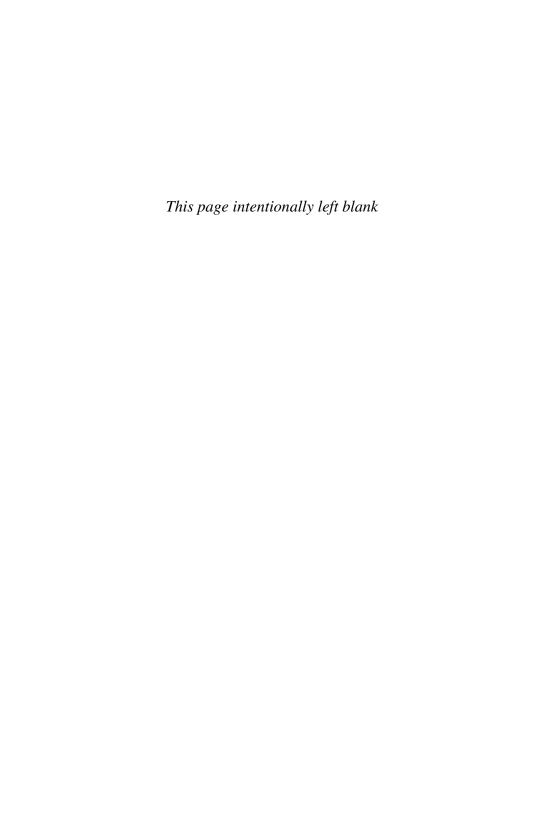
|     |         | 9.5.3    | Example 2   | 189 |
|-----|---------|----------|---|-----|
|     | 9.6     | Softwa   | are Agents for Decentralized Sensor Management    | 190 |
|     | 9.7     | Conclu   | isions  | 190 |
|     |         |          |   |     |
| A   |         |          |   | 197 |
|     | A.1     | Filter N | Models  | 197 |
|     |         | A.1.1    | Stationary model                                  | 197 |
|     |         | A.1.2    | Constant velocity model                           | 197 |
|     |         | A.1.3    | Acceleration models                               | 198 |
|     |         | A.1.4    | Exponentially correlated velocity (ECV) model     | 199 |
|     |         | A.1.5    | Exponentially correlated acceleration (ECA) model | 200 |
|     |         | A.1.6    | The coordinated turning model                     | 201 |
|     |         | A.1.7    | The measurement equation                          | 201 |
|     | A.2     | Generi   | c Particle Filter                                 | 201 |
|     |         | A.2.1    | Generic particle filter algorithm                 | 202 |
|     | A.3     | Bayesi   | an Classifier                                     | 205 |
|     | A.4     | Demps    | ster-Shafer Classifier                            | 207 |
|     | A.5     | Artific  | ial Immunology System                             | 210 |
|     |         | A.5.1    | B-cells or B- lymphocytes                         | 210 |
|     |         | A.5.2    | T-cell or T-lymphocytes                           | 211 |
|     |         | A.5.3    | What can we learn from the immune system?         | 212 |
|     | A.6     | Introdu  | action To Ontology                                | 213 |
|     |         | A.6.1    | Introduction                                      | 213 |
|     |         | A.6.2    | Components of ontologies                          | 213 |
|     |         | A.6.3    | Classification of ontologies                      | 215 |
|     |         | A.6.4    | Principles of ontology design                     | 215 |
|     |         | A.6.5    | Conclusion  | 216 |
| Bil | bliogra | aphy     |   | 217 |
| Inc | dex     |          |   | 235 |

# **List of Figures**

| 1.1  | A child can think and learn. The multiple sensory systems (such as eyes and ears) will enable her to know the environment | ç  |
|------|---|----|
| 2.1  | Dependency relationship   | 16 |
| 2.2  | Aggregation relationship  | 17 |
| 2.3  | Composition relationship  | 18 |
| 2.4  | Association between 2 classes   | 19 |
| 2.5  | Generalisation relationship   | 20 |
| 2.6  | Example of an iterative approach to agent software development  | 23 |
| 2.7  | Spiral development process [193]  | 24 |
| 3.1  | The nerve cell  | 37 |
| 3.2  | The basic elements of neural networks   | 39 |
| 3.3  | Difference between feedforward and recurrent neural networks  | 41 |
| 3.4  | Multilayer feedforward neural network modelling a nonlinear dy-   |    |
|      | namic plant   | 45 |
| 3.5  | Block diagram of a fuzzy system   | 56 |
| 3.6  | Membership function for one (X1) of the three inputs  | 56 |
| 3.7  | Rule-based matrix defining rules of the fuzzy system  | 57 |
| 3.8  | Fuzzy inference using fuzzy approximate reasoning   | 58 |
| 3.9  | Membership function of aggregated output y  | 59 |
| 3.10 | Graphical representation of First-maxima defuzzification method   | 60 |
| 5.1  | Data, information and knowledge.  | 80 |
| 5.2  | Intelligent fusion process  | 81 |
| 5.3  | 5 levels fusion process   | 83 |

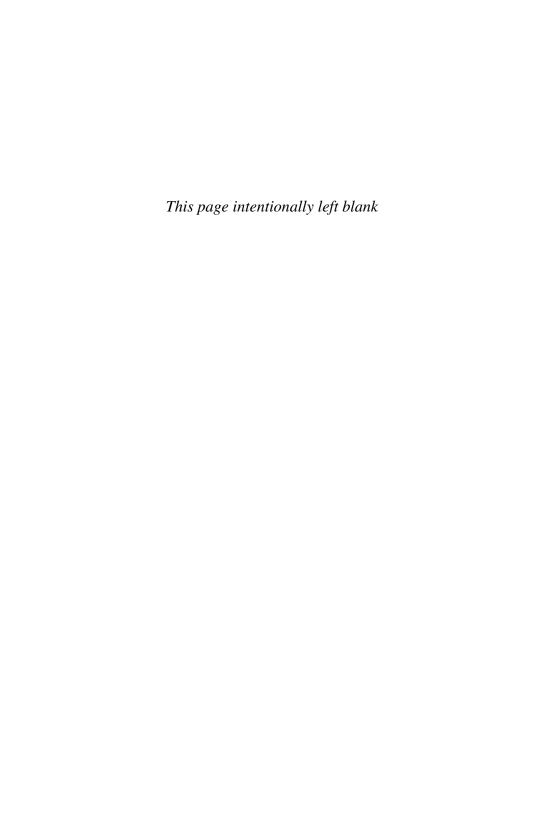
| 5.4  | Possible logical flow of the 5 levels fusion process   | 85  |
|------|--|-----|
| 5.5  | Example of data incest. Node 1 communicates its detection to node 2. Node 2 passes the message to node 3. Node 3 broadcasts the message and node 1 receives. Node 1 assumes that node 3 has the same detection and hence increases its confidence of the detected target | 88  |
| 5.6  | Possible engineering flow of cognitive and high-level fusion process.  | 91  |
| 5.7  | Possible evolving process of the awareness state   | 92  |
| 5.8  | The memory structure.  | 93  |
| 5.9  | Centralized architecture with a central processor.   | 94  |
| 5.10 | Decentralized fusion architecture  | 95  |
|      | Distributed fusion architecture  | 96  |
| 5.12 | Hierarchical architecture  | 97  |
| 5.13 | Hybrid architecture  | 98  |
| 5.14 | Data and information fusion system - multi-disciplinary subject  | 99  |
| 5.15 | Data association techniques  | 100 |
| 5.16 | Filters that are used for state estimation methods   | 101 |
| 5.17 | Classification and identification techniques   | 102 |
| 5.18 | An example of force aggregation  | 103 |
| 6.1  | A simple scenario and its association graph  | 122 |
| 6.2  | Competing edges  | 123 |
| 6.3  | Association graph with local association strength  | 124 |
| 6.4  | Association graph with global association strength   | 125 |
| 7.1  | Concepts of sensor-to-tracking system  | 131 |
| 7.2  | Basic tracking stages in a typical tracking system   | 132 |
| 7.3  | Relationship between $\alpha$ and $\beta$ values. Solid line is plotted from equation (7.7) and '*' is plotted from equation (7.6)   | 136 |
| 7.4  | The IMM filter   | 140 |
| 7.5  | The IMBM algorithm.  | 143 |
| 7.6  | Scenario for testing   | 145 |
| 7.7  | Mode probability of two models   | 147 |
| 7.8  | Position error obtained from Kalman $(Q=1m/s^{2})$ and IMM filters   |     |
|      | mode   | 148 |

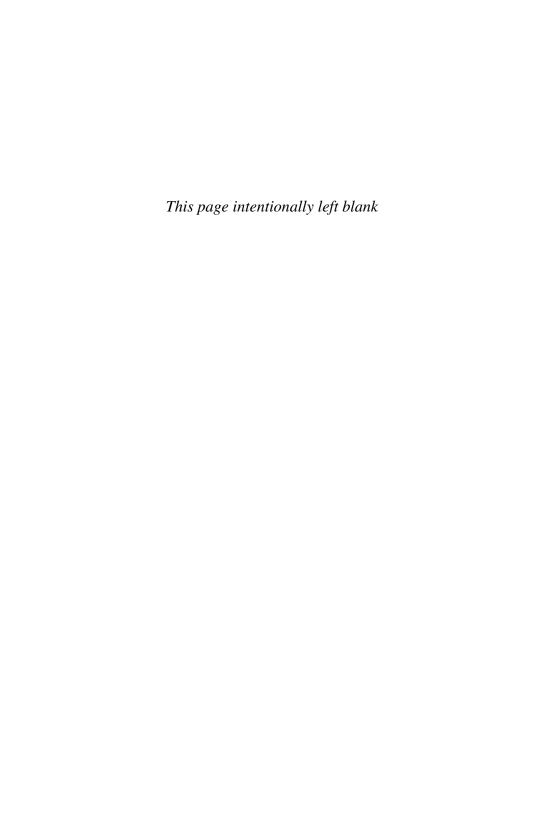
| 7.9  | RMS error obtained from Kalman ( $Q=1m/s^2$ ) and IMM filters Mode   | 149 |
|------|--|-----|
| 7.10 | RMS error obtained from generic particle filters and IMM filter  | 150 |
| 8.1  | Cooperative robots searching for the light source  | 156 |
| 8.2  | Target flight profile and the nodes placement for the experiment. $\ \ .$  | 171 |
| 8.3  | Position error vs time in two nodes  | 173 |
| 8.4  | RMS error vs time in 4 nodes   | 173 |
| 8.5  | RMS error vs time in 6 nodes   | 174 |
| 8.6  | RMS error vs time in 8 nodes   | 174 |
| 9.1  | A closed-loop sensing system framework   | 178 |
| 9.2  | Sensor management levels based on functionality  | 180 |
| 9.3  | Sensor management functional modules   | 181 |
| 9.4  | Sensor management techniques   | 182 |
| 9.5  | Partition of the input universe of discourse for error (k)   | 192 |
| 9.6  | Partition of the input universe of discourse for error (k-1)   | 192 |
| 9.7  | Partition of the output universe of discourse  | 193 |
| 9.8  | The complete closed-loop diagram   | 193 |
| 9.9  | Target moving profile and the sensor field of view (FOV)   | 194 |
| 9.10 | Error results of one sensor  | 194 |
| 9.11 | Position and coverage of the two sensors   | 195 |
| 9.12 | The error results of the two sensors with SMS  | 195 |
| A.1  | Resampling process   | 204 |
| A.2  | Sequential updating process of Bayesian classifier.  | 206 |
| A.3  | Joint classification   | 207 |
| A.4  | Belief and plausibility functions  | 207 |
| A.5  | DS classification process. Note that the update equation is given by $m(A) = (m(A) E_k) \oplus (m(A) E_{k-1}) \dots \dots \dots \dots \dots$ | 209 |
| A.6  | Different interpretations of the 'ABOVE' relation  | 214 |
| A.7  | Simple Ontology  | 214 |

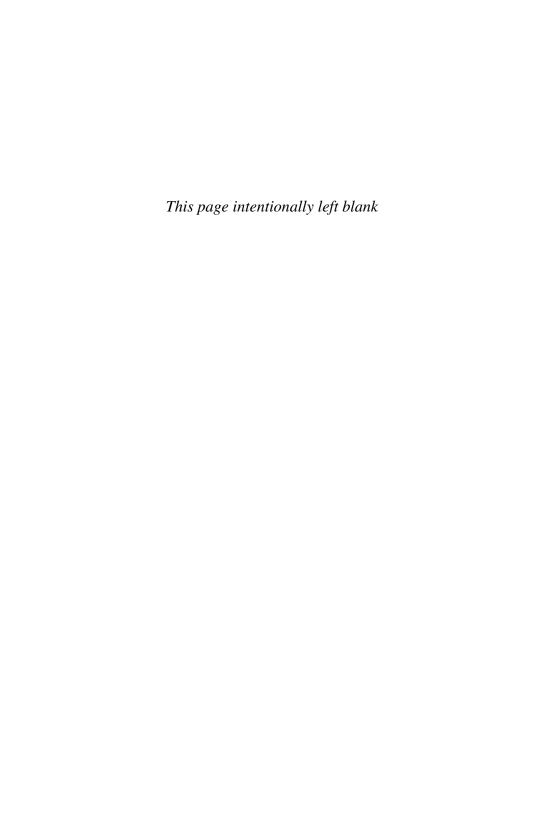


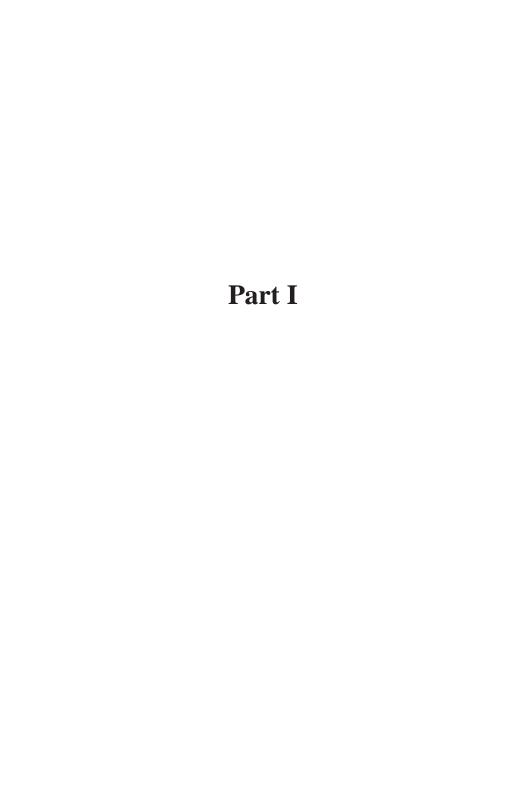
# **List of Tables**

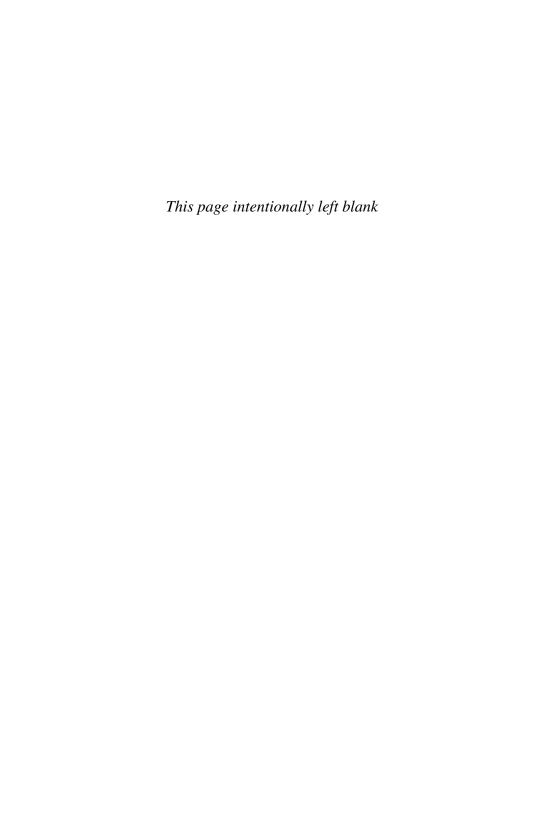
| 2.1 | Diagrams for software design   | 15  |
|-----|--|-----|
| 2.2 | Object versus agent software engineering issues                                    | 26  |
| 4.1 | Classification based on intelligence sources                                       | 65  |
| 4.2 | Human sensory systems  | 66  |
| 4.3 | Man-made sensor systems  | 67  |
| 7.1 | RMS errors with different process noise values for the Kalman filter.              | 146 |
| 7.2 | RMS errors with different process noise values for the IMM filter with two models. | 146 |
| 7.3 | Results of the generic particle filters and the IMM filter                         | 148 |
| 8.1 | Comparison of decentralized tracking algorithms                                    | 172 |
| 8.2 | Track selection method test result   | 175 |
| 9.1 | The 25 fuzzy rules   | 188 |











### CHAPTER 1

# **Quest to Build Intelligent Systems**

If a machine is expected to be infallible, it cannot also be intelligent.

Alan Turing (1912-1954).

### 1.1 Desire for an Intelligent System

An intelligent system is what everyone desires to have. Intelligent systems will be the goal of all our next generation systems development. No one will deliver an unsophisticated system as all would want an intelligent system.

The quest for building intelligent systems is well illustrated in many articles and papers. A quick search from IEEE/IEE journals and conference papers reveals that more than 3000 papers contain the words 'intelligent systems'.

### 1.1.1 Will an intelligent system match the human intelligence?

Are we able to build an intelligent system that is equivalent to ourselves? An intelligent system that can read a book on physics or accounting and answer the questions at the end of the chapter? Can such system look over the shoulder of a person and learn to assemble a motor? Or make new discoveries? Exhibit emotion, tell a joke, or display consciousness? Herbert Simon, a renowned scientist in championing artificial intelligence, believed that the answer to all these questions was 'yes' [7].

Simon's view was that if intelligence can be demonstrated by large collections of carbon-based molecules, then there is no reason, in principle, why the same would not be the case for any other collection of molecules capable of manipulating physical symbols. However, that philosophy led to unending controversy and debates.

There are many different views with regards to how far human beings could achieve in building intelligent systems. These differences in views ultimately rest on individual beliefs. For example, many people believe that the unique capabilities of human beings such as creativity, emotion, and consciousness, cannot be reproduced by another system.

### 1.2 What is an Intelligent System?

What is considered an intelligent system? Does a machine that can interact with a human be classified as intelligent? Toys such as Furbys (or Shelby the next generation Furbys), Sony Aibo (robot dog) and Paro (pet robot - baby harp seal) are interactive to human response. Are these toys intelligent systems? Indeed, the definition of the words 'intelligent system' has often been quite unclear.

### 1.2.1 Examples of existing intelligent systems

Listed below are some examples of systems that claim to be intelligent:

- Transportation systems. Intelligent transportation systems are widely discussed and presented, some examples cited are [40], [130], [115] and [46]. These cover intelligent highways, intelligent traffic controls and intelligent vehicles. Autonomous, collision-avoidance, energy saving and smart sensor technology used in vehicles are some of the areas presented in intelligent vehicle research activities. Under an intelligent transportation system, the focus includes an automatic traffic guiding system for detecting and monitoring of traffic to achieve optimum traffic flow. Such intelligent systems also consist of a network of sensors on-board the car with a centralized and hierarchical fusion architecture. The fusion processor on-board can also be fitted with sensors, such as radar and camera.
- Robotic systems. Research in robotics most of the time tries to mimic living
  things. For example: intelligent hand [135], cockroach-like robot [171], multiarm robot [223] and walking robots [166]. Sensors such as touch and slip
  sensors, and microphone for vibration are often used. Fusion algorithms are
  introduced in multi-sensor robots for the control and coordination of actions
  among independent processors within the robot system.
- Home appliances. Driven by business and market forces, we also see an increase in intelligent home appliances, such as intelligent washing machines using a hybrid of fuzzy rules and reasoning. A smart home appliance is one that incorporates novel sensing and control algorithms so as to either automate some manual functions, such as estimating the clothes load in a washing machine to economize on energy use, or to introduce something completely

new, such as detecting boiling water on an electric cooktop [9]. The positioning of an intelligent ventilator via acoustic feedback [162] for an intelligent home system could reduce the noise from external sources, such as approaching aircraft or noisy vehicles. Algorithms considered for such intelligent home appliances are: fuzzy logic, rule-based and knowledge-based controller.

• Diagnostic systems. These are intelligent online machine fault diagnostic systems. These systems typically use algorithms such as data mining technique, neural network and rule-based reasoning or a combination of them. Data are normally in textual form. An example of an intelligent diagnostic system can be found in [80].

Beside the above listed examples, there are many more that have claimed to be intelligent systems in one way or another. And yet a lot more that do not use the word intelligent may also have a lot more intelligent capabilities than those claimed.

### 1.2.2 Learning from nature

There are groups of researchers that look to nature for inspiration on how to build intelligent systems. Their research includes understanding and learning to engineer the following:

- Swarm intelligence. Animals and insects are able to coordinate and group themselves for protection and to hunt for food. For example, ants are able to coordinate with one another to carry heavy objects, a school of small fish is able to move as a large group for self-protection, bees are able to build a beehive in a cooperative way.
- Artificial immunology system. Our complex immune system is able to defend our body in a very intelligent way. Immune cells such as T-cells, are able to remember the pathogen (foreign objects such as viruses, bacteria, parasites and other microbes) they have combated before, and are able to multiply the 'warrior' cells very rapidly when they encounter the same pathogen again. Hence, learning from our immune system may provide some key answers to a better way of building smart systems for particular applications, such as computer security systems. See Appendix A.5 for a short description of artificial immunology systems.
- Intelligent sensory systems. Animals and insects have, in general, good sensory systems, which are effective and efficient. For examples, a dog's nose is able to detect an object present even when the object is giving out a very low odour signature, whereas the viper snake has a highly sensitive thermal sensing capability [118].

 Advanced locomotion. Animals and insects have highly advanced locomotion and control bio-mechanisms that enable the whole body or platform to move in a very stable and efficient manner within a specific environment.

### 1.2.3 How do we qualify a system as intelligent?

Dictionary definitions of intelligent include:

- Having or showing (usually a high degree of) understanding, clever, quickness
  of mind.
- Endowed with the faculty of reason; having or showing highly developed mental faculties; well informed; knowing, aware (of).
- Having or showing powers of learning, reasoning, or understanding, especially to a high degree.

From the definitions and the examples given, the important attributes of an intelligent system would consist of words such as: learning, intuition, creativity, quickness of mind, clever, reasoning, understanding, autonomous behaviour or self governing, adaptive, ability to self-organize, coordinate and possess knowledge. Other than human beings, no other living things or man-made machines could currently fulfil all the above requirements. However, if a system could fulfil a subset of the attributed words, the system would have to possess some form of intelligence (or sub-intelligence). This will include a system that could perform quickly with a certain logical reasoning, and would be classified as 'clever' or 'smart'.

One school of thought is that all intelligent systems must have the capability to learn. In fact, the ability to learn dominated most of the studies in the late '80s and '90s. Learning capabilities are often related to the study of artificial neural networks. Researchers unanimously agree that learning is an important attribute of intelligent systems. However, that is not all what an intelligent system should have. Based on our human growth and development, we know that some of our functions are more than just learnt. For example, the ability of a new born baby to suck the mother's breast milk is not taught but is innate. Hence, intelligent systems need to include a designed element where knowledge already exist and not always need to be trained or learnt.

The other school of thought is that biologically inspired algorithms, such as fuzzy logic, neural network and genetic algorithms, are non-optimal algorithms and do not provide good solutions to a problem. Hence, it may not be the way to go in building intelligent systems. Mathematical modelling techniques or other statistical and probabilistic algorithms would be better than the so-called computational intelligence algorithms.

However, it is clear that currently no single technique can solve all the attributes we desire in building intelligent systems. Hence, a combination of various tech-

niques is the way to go.

Here we define an intelligent system as any system that could receive sensory information and has the ability to process this information with a computationally efficient and effective software, combined with one or more smart or intelligent algorithms (where the algorithm could be biologically inspired, mathematical modelled, statistical algorithms, etc.) for performing functions, such as control, managing resources, diagnostic and/or decision-making, to achieve multiple or single task/s or goal/s.

The ultimate goal is for an intelligent system to be modelled after human intelligence. Figure 1.1 shows how a child in her early development would already possess the ability to think and learn. Scientists are looking to the brain to enhance our current way of information processing. The ability to engineer such a system, that could match the total functions of the brain in processing and controlling all the sensory systems, plus managing the whole state of the body or platform in adapting and interacting with the dynamics of the environment, is a non-trivial task. In chapters 3 and 5 the discussion on neural networks and cognitive intelligence are respectively related to the modelling and understanding of the brain.



**Figure 1.1:** A child can think and learn. The multiple sensory systems (such as eyes and ears) will enable her to know the environment.

## 1.3 What Makes an Intelligent System

With the above definitions, the basic components of an intelligent system would be, but not limited to:

sensor technology.

- computing technology.
- intelligent/smart algorithm.
- communication technology.
- platform.

This book, in particular, will concentrate on the intelligent or smart algorithms used for developing intelligent systems and also focuses on fusion and control of multiple sensory system algorithms working in a network of fusion nodes.

The book does not present any work on communication technology and the platforms that physically hold these technologies. However, it acknowledges that the development in both the communication and platform technology will certainly play an important role in building intelligent systems. Communication technology has been expanding rapidly. For example, optical transmission rates are hitting the 100GHz and wireless communication, such as bluetooth <sup>1</sup>, 802.11 <sup>2</sup>, ultrawideband<sup>3</sup> and 3G/4G is continuously enhancing distributed networking structures. Hence, improvements in communication technology will have a significant impact on building intelligent systems particularly in the arena of decentralized and distributed networking systems. Likewise, the platforms that carry these technologies have been improving with the discovery of new material and new engineering designs. Our body is an example of the most beautifully designed platform where the human sensory systems and the processing of sensory inputs from our brain enable us to function as an intelligent being.

### 1.3.1 How an intelligent system sees the world?

How does an intelligent system see the world or is aware of its environment? Sensor technology plays an important role in helping an intelligent system see the world. This is well illustrated in our own sensory system. Without the eye, ear, skin, nose and tongue our brain will be totally useless even if it has the most powerful biological algorithms.

Hence the growing importance of intelligent systems is in line with the increase usage and improvement in sensory technology. Smart sensory technology that will make sense of the information detected, and combines with intelligent algorithms and fast processors will make the system better.

<sup>&</sup>lt;sup>1</sup>Delivers data at speed of 700Kbps and range of about 10m.

<sup>&</sup>lt;sup>2</sup>802.11a delivers data at maximum speed of 54Mbps and range of about 30m.

<sup>&</sup>lt;sup>3</sup>Potential in delivering data up to 100Mbps and range of about 60m.

### 1.4 Challenges of Building Intelligent Systems

We will see more claims of intelligent systems research and development activities and even products with different applications. Each of these is indeed a part of the building blocks in making intelligent systems. They each have a synergy in contributing toward our quest of building the ultimate intelligent system. However, whether an intelligent system will ultimately match human creativity, flexibility, and have consciousness is debatable.

This book discusses the challenges of building intelligent system into two parts. The first part will present the essential components that make an intelligent system. These are:

- Software design. Since the current building of intelligent systems relies heavily on software, a robust software design will play an important part. Software programming has moved from assembler and procedural style to object- and agent-oriented programming. The challenge of building good software to support an intelligent system will be discussed. Agent-based software will also be discussed and compared with object-oriented design.
- Biologically inspired algorithms. Intelligent systems will require smart software algorithms to provide all the support needed for control and decision-making in a dynamic and ever-changing environment. Many researchers have been involved in finding better and smarter algorithms for many centuries. To-day the challenge in this area continues, particularly in biologically inspired algorithms. These algorithms have a relationship to the way humans solve problems and are seen as the right path in our quest to achieve the ultimate intelligent algorithms. Algorithms such as neural networks, genetic algorithm and fuzzy logic will be presented.
- Sensor systems. As discussed earlier, how the intelligent system sees the world
  will depend on the sensor systems built on it. Hence, knowing the types of sensor systems available and applying them appropriately will contribute toward
  building robust intelligent systems.

The second part of the book will focus on data and information fusion, tracking and sensor management. Fusion is one of the important aspects of an intelligent system. Fusion brings data from multiple sensors or multiple sources together to form useful information. High-level fusion processes also try to derive, predict and make sense of information. This includes the process of understanding the situation, its impact and ultimately for implementing the decision-making process. Part of the high-level fusion, which is a larger part of the sensemaking process, resides in understanding how our brain interprets information received from the sensory systems <sup>4</sup>. The understanding of information will form knowledge. How does an intelligent system understand the information to form a collection of knowledge and using

<sup>&</sup>lt;sup>4</sup>Studies are on going in these areas at different research institutions. For example, the study on how

this knowledge with good judgment (or wisdom) for short-term and long-term need remains a challenging area of research.

the brain interprets the ears' input has led to the discovery of virtual surrounding, in which two speakers sound like many more [129].

## **CHAPTER 2**

# Software Design for Intelligent Systems

Software is an essential engineering tool for intelligent system design. G. W. Ng

## 2.1 Introduction

In any software project, system analysis and design is an important process - more so in the building of an intelligent system, which currently relies heavily on software programs.

In this chapter, we aim to discuss the general software analysis and design issues for building intelligent software systems using UML (Unified Modelling Language). UML, in our opinion, is expressive enough to capture the design decisions required to design the software program used in intelligent systems. For example, there are a lot of CASE (Computer Aided Software Engineering) tools developed for UML that could be used for building intelligent systems, such as expert systems. Furthermore, UML is extensible and very flexible, hence would be suitable for newer types of smart software, such as intelligent software agents. Software design that includes the capability of agent technology may play an important role in future intelligent systems. Hence, this chapter will also discuss software agent-based systems, propose an agent-oriented software development methodology and discuss the object-oriented versus agent-oriented software engineering issues.

This chapter is organized as follows: Section 2 presents the analysis and design using UML. Section 3 presents the data structure and storage. Section 4 describes what is a software agent-based system. Section 5 proposes an agent-oriented software development methodology. Section 6 discusses on object versus agent-oriented

software engineering issues.

# 2.2 Analysis and Design Using UML

In any software project, software analysis and design phases are meant for defining the architecture and help meet the needs of customers. They are achieved through building models. Models help us visualize a complex system under a controlled environment; it is a simplification of reality. There are a number of methodologies pushing software analysis and design. Of late, Object-Oriented Analysis and Design (OOAD) is particularly popular. Of these, Objectory (by Ivar Jacobson), the Object Modelling Technique (OMT by James Rambaugh) and the Booch Methodology (by Grady Booch) are widely used and successful ones. More recently, a global effort pioneered by these 3 veterans, crystallized their expertize and experiences into a standard known as the Unified Modelling Language (UML) [37].

The UML is a standard language for visualizing, specifying, constructing and documenting a software product. It is methodology neutral and can be used to document a wide variety of software intelligent systems. A methodology will use this expressive language in producing a software blueprint. One example of such use is the CommonKADS Methodology (a complete methodological framework for developing knowledge-based systems). It uses UML notations.

## 2.2.1 Diagrams

Central to UML is a wide choice of diagrams, which are graphical presentations of connected elements and their inter-relationships. They are particularly useful for the visualization and construction of software components. Table 2.1 shows the choice of diagrams.

Note that the component and deployment diagrams exist in UML to help document the organization and dependencies of software components, and how the system is physically deployed. We find that they are not necessary in documenting agent-based systems *per se* but will be required to document the software implementation of the system.

## 2.2.2 Relationships

There are basically 4 types of relationships in object-oriented modelling. These are:

- dependencies, which offer a loose coupling link within one class, using the services of another.
- **aggregation** shows part-whole relationships.

| Number | Diagrams               | Descriptions  |
|--------|------------------------|---|
| 1.     | Activity diagrams      | Activity diagrams show the flow of control within   |
|        |                        | a system or sub-system. They are particularly       |
|        |                        | effective in showing concurrent activity. Using     |
|        |                        | swimlanes, ownership of an object can be            |
|        |                        | clearly shown. Activity diagrams can be potentially |
|        |                        | used to show how various software agents            |
|        |                        | interact concurrently with one another.             |
| 2.     | Sequence diagrams      | Sequence diagrams display how object interacts      |
|        |                        | in a time sequence.                                 |
| 3.     | Collaboration diagrams | Collaboration diagrams display object               |
|        |                        | interactions organized around objects and           |
|        |                        | their links to one another.                         |
| 4.     | Class diagrams         | A class diagram is a collection of objects          |
|        |                        | with common structure, behaviour, relationship      |
|        |                        | and semantics. The class diagram is static          |
|        |                        | by nature. In agent-based systems, classes          |
|        |                        | can be used to represent agents as well.            |
|        |                        | Stereotypes can be used to mark                     |
|        |                        | the distinction for agent classes.                  |
| 5.     | State transition       | A state transition diagram (STD) shows              |
|        | diagrams               | the life history of a given class. It shows the     |
|        |                        | events that cause a class to transit from           |
|        |                        | one state to another. State transition diagrams     |
|        |                        | can be nested; this allow us to effectively         |
|        |                        | manage the complexity. In agent-based               |
|        |                        | systems, the states of each agent could             |
|        |                        | be represented by STD.                              |

Table 2.1: Diagrams for software design.

- associations represent structural links among objects.
- **generalization** for the derivation of inheritance relationships.

## 2.2.2.1 Dependency

This is the relationship of one class depending on another class [37]. For example, when the object of a class appears in the argument list of a method of another class, the latter is using the former to do something. If the used class changes, there is a high chance that the user class is affected. So the user class depends on the used class. The example in Figure 2.1 shows that class A is dependent on class B to perform some services.

## 2.2.2.2 Aggregation

This relationship is useful to show that an object contains other objects, in what is widely called the part-whole relationship. The container class, which is also called the owner class, typically creates and destroys the owned object. In Figure 2.2, the

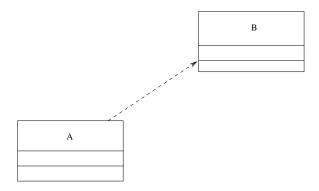


Figure 2.1: Dependency relationship.

IAinNet class really is made up of an array of layers; each of these layers is itself an IAinObj. In UML notation, a diamond shape at the container end pointing towards the contained-object end shows an aggregation through reference. The multiplicity shown indicates that '1' IAinNet object could contain 0 or more IAinObj objects.

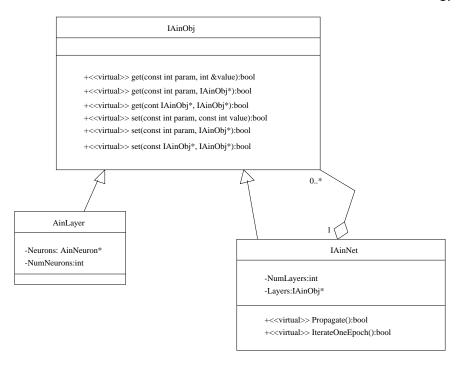
The strong form of aggregation is known as composition. This is in cases when the owned object cannot be shared across different owners. Furthermore, the owned object forms an important component within the owner class. For example, a neuron class is composed of an identification tag (could be an integer or string) and weights (could be floats or doubles) among other things. Figure 2.3 shows an example of composition relationship.

#### 2.2.2.3 Association

This is a structural relationship between 2 objects permitting the exchange of predefined messages between them (see Figure 2.4). The message flow can be unidirectional or bidirectional. For example, the Layer class could maintain a bidirectional relationship with a neuron class. This relationship offers a general relationship between 2 classes, and is used whenever a class uses the services of another class. An aggregation can be seen as an association.

### 2.2.2.4 Generalization

This is an inheritance relationship, where the derived class has all the characteristics exposed by the parent classes. They are more widely called 'is-a' relationship. For example, an InputNeuron class may inherit from a Neuron class because the former 'is-a' neuron with more specific roles serving within the input layer. In Figure 2.5, the class 'Vehicle' is a root class; derived from it are the car, bus or van classes. This relationship is particularly useful for extending software capabilities. A class hier-

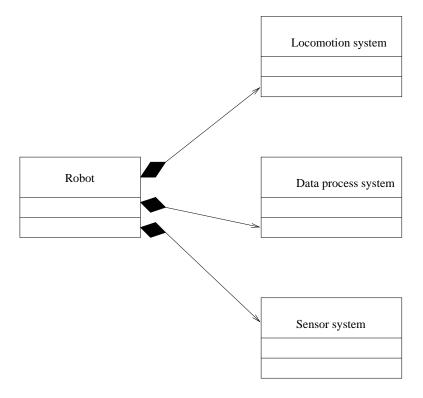


**Figure 2.2:** Aggregation relationship.

archy almost always leverages on polymorphism to extend the behaviour of derived classes. In the example, the vehicle may provide a default drive method, which is polymorphic (using keyword virtual). But in reality, driving a car may be slightly different from driving a bus. So the drive method in the bus is defined so as to override the default operation defined in the vehicle class. The power of a class hierarchy and polymorphism are further explored in a number of design pattern books.

# 2.3 Data Structure and Storage

Quite often, in building intelligent systems, the data structure and storage could be complex. Traditionally, arrays and lists have been used for quick random or sequential access, however, other data structures can be better explored. UML can be effectively used in describing the software design of data structure and storage.



**Figure 2.3:** Composition relationship.

## 2.3.1 Data structure

There are many data structures. A good software design generally has a good data structure serving as the backbone. They are important because they ensure data are organized properly. When the correct structure is used, the storage becomes scalable. Common structures are: lists, queues, stacks, trees, arrays, sets and maps.

In deciding the correct data structure to adopt in an intelligent application, one has to study the following factors:

- How are data represented in the application? Certain data are, by nature, hierarchical. For example, in representing a graphical model in a 3D application, an obvious choice would be using a tree to organize the model.
- How should data be retrieved from the structure? In a neural network back-propagator application, trees seem a good choice. For example, we could have a layer as a branch, and under that branch we could have a number of nodes, each representing a neuron. The propagation is from one branch to the next,

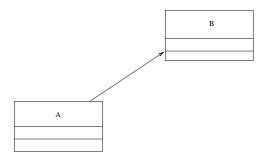


Figure 2.4: Association between 2 classes.

in a lateral traversing manner. An alternative is to use a map data structure to organize a neural network. A map relies on a key to look up some content. A number can identify the layer, and this is used as a key to obtain information on a particular layer.

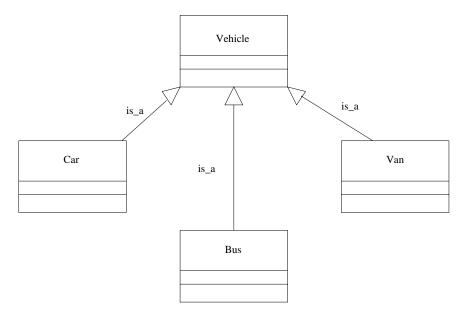
Usually, the choice of a data structure will depend on three issues:

- Ease of programming. Does the choice of a data structure make implementing the system easier?
- Time efficiency. Various data structures offer different performance in terms of insertion, deletion and retrieval.
- Space efficiency. How much physical space a data structure occupies is also important. This is especially so in memory constrained systems.

# 2.4 What is a Software Agent-based System?

Rapid advances in information and communication technologies, notably Internet networking and wireless communications, are providing a new infrastructural and communications means to achieve higher levels of automation, flexibility and integration in the development of distributed software applications. To mutually refine and exploit these technologies, multiagent systems (i.e. systems of multiple agents), according to Jennings *et al.* [112], are being advocated as a next generation models for engineering complex, distributed systems [242, 113].

Agent-based systems could potentially represent a new way of analyzing, designing, and implementing complex software systems in a network-centric infrastructure. It is claimed as a possible basis framework to bring together and extend the vast body of knowledge from artificial intelligence, computer and control sciences, operations research and related fields, such as economics and the social sciences



**Figure 2.5:** Generalisation relationship.

(reference *Proceedings of the ICMAS' 2000* <sup>1</sup>). Examples of potential agent-based systems are: personalized email filters, air-traffic control systems and command and control systems. The potential applications of agents in many small and complex systems are on hand. However, there are also concerns that certain aspects of agents are being dangerously over-hyped and if not controlled, may suffer the same backlashes as artificial intelligence (AI) in the 1980s.

## 2.4.1 What is an agent?

There is no universally agreed definition of the term agent. According to [243], an agent is defined as 'a computer system that is situated in some environment and is capable of autonomous action in this environment in order to meet its design objectives'. In this chapter, we will mainly focus the discussion on software agents. Software agents can occupy many different environments, such as in the Internet or within a core local area network. The autonomous action of an agent is its ability to control its own state and also can interact with humans, other systems or other agents. Agents are flexible in the sense that they are [242]:

• responsive: agents should perceive their environment and respond in a timely fashion to changes that occur in it;

<sup>&</sup>lt;sup>1</sup>ICMAS: International Conference on MultiAgent Systems.

- proactive: agents should not simply act in response to their environment, they should be able to exhibit opportunistic, goal-directed behaviour and take the initiative where appropriate;
- social: agents should be able to interact, when appropriate, with other artificial
  agents and humans in order to complete their own problem solving and to help
  others with their activities.

Hence, an agent that has all the attributes of these flexibility parameters, is also known as an intelligent agent <sup>2</sup>.

## 2.4.2 What is a multi-agent system?

Multi-agent systems refer to systems with multiple agents. In general, the study of multi-agent systems [236] involves three mutually dependent areas, namely: agent model and architecture, communication languages and interaction protocols. An agent model refers to its *internal* architecture of integrated data structures and functionalities that should ideally support planning, reacting and learning, while a multiagent architecture, in general, specifies an infrastructure for coordination or collaboration among the agents. Communication languages enable agents to describe their messages meaningfully for knowledge exchange, while interaction protocols enable the agents to carry out structured exchanges of messages.

## 2.4.2.1 Multi-agent approach to distributed problem solving

When adopting an agent-based approach to distributed problem solving (DPS), it is clear that most problems will require multiple agents to represent the decentralized (or distributed) nature of the problem. Moreover, these agents will need to interact with one another, to achieve their common or individual goals and to manage, so as to ensure a proper temporal order <sup>3</sup> among dependencies that may arise from being situated in a common environment [236, Chapter 2].

## 2.4.2.2 Multi-agent negotiation

One means of interaction is *negotiation*. By negotiation, we mean that agents work out, communicatively, an agreement that is acceptable by all agents. In distributed problem solving, negotiation is one way that agents use in cooperation to solve the problem. Generally, an automated means (or reasoning mechanism) of negotiation

<sup>&</sup>lt;sup>2</sup>An intelligent agent is a software entity capable of *flexible autonomous* actions to meet their design objectives (goals); *autonomous* in that it requires minimal or no human intervention and *flexible* (or intelligent) in that each agent can react to perceived changes in a common environment, interact with other agents when necessary, and take proactive steps towards its design goals agent [236, Chapter 1]

<sup>&</sup>lt;sup>3</sup>In our view, what is *proper* is domain dependent, and is decidedly a subjective opinion of the agent designer/analyst.

for an agent can be implemented as a *negotiation protocol* within which a *decision-making model* (or logic) over the objects of negotiation resides.

In the literature on general negotiation frameworks, agents that can negotiate with exact knowledge of each other's cost and utility functions, or such knowledge learnt in the initial step of interaction, have been proposed [131]. There are agents that negotiate using the unified negotiation protocol in worth-, state-, and task-driven domains, where agents look for mutually beneficial deals to perform task distribution [198, 251]. In negotiation via argumentation (NVA), the agents negotiate by sending each other proposals and counter-proposals. In one NVA approach [132], these proposals are accompanied by supporting arguments (explicit justifications) formulated as logical models. In another NVA approach [116], the distributed constraint satisfaction problem (DCSP) algorithm [247] provides the computational model, extended with the supporting arguments (accompanying the proposals) formulated as local constraints. In yet another approach [208], agents can conduct NVA in which an agent sends over its inference rules to its neighbour to demonstrate the soundness of its arguments. Finally, there are also negotiating agents that incorporate AI techniques (e.g. auction mechanisms and metaphor-based mechanisms that attempt to mimic human practical reasoning in some specific ways).

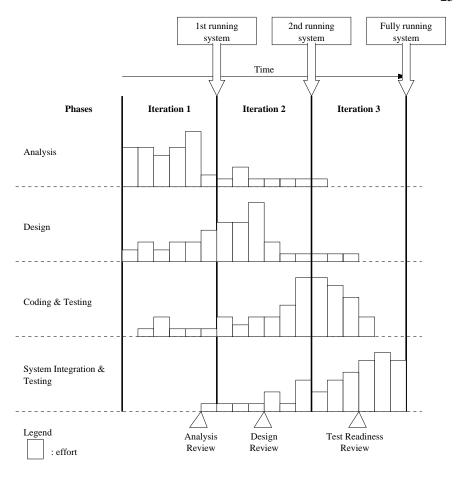
For a complete review of agent negotiation, readers may refer to Jennings *et al.* [112].

# 2.5 An Agent-Oriented Software Development Methodology

Like object-oriented software development, we believe that an iterative approach to developing an agent-based software systems holds the best promise of successfully developing such systems. An iterative approach allows important requirements of the system to be validated early and also serve to clarify unclear requirements. The concepts covered here do not represent a fully functional agent-based development methodology but rather provide several suggestions on what to focus on during analysis, design, coding and testing of multi-agent systems.

Typically, an iterative development has the following characteristics, as illustrated in Figure 2.6.

In an iterative development approach, several phases of the system are created to provide early feedback on the usability and performance of the system. Also, analysis, design, coding and testing can take place almost simultaneously. Note, however, that the effort expanded in analysis, design, coding or testing is different in each time quantum. For example, during the initial development to the system, most of the effort would focus on analysis. Towards the end of the development, most of the effort would be spent on testing. Various formal reviews are also planned during the development in order to track and monitor the progress of the project (i.e. we

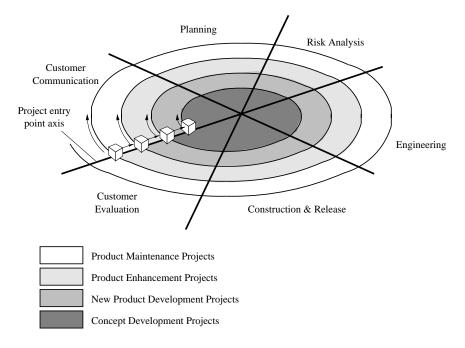


**Figure 2.6:** Example of an iterative approach to agent software development.

develop iteratively but manage it sequentially). These reviews also serve as 'exit points' to the project. For example, if the project cannot meet its expectation, the project management team can choose to terminate the project.

Note that for more exploratory systems, a spiral development methodology can be followed. Figure 2.7 shows a typical spiral development process. However, it will not be discussed in this section.

In the discussion that follows, we will assume that the development of the agentbased system will be built on top of an agent-based infrastructure. Also, we will suggest appropriate UML diagrams suitable for capturing analysis and design decisions.



**Figure 2.7:** Spiral development process [193].

## 2.5.1 Analysis

During the analysis phase, the system requirements of the system are analysed and translated to unambiguous specifications. Use-cases and interactions diagrams can be used to capture these requirements. From the use-case description, agents can be identified.

Having defined the use-case model, the roles of the system need to be found. Four types of roles need to be determined. These are:

- Responsibilities. This represents the dos and don'ts of that role.
- Permissions. This represents the type of information the role can access.
- Activities. This represents the tasks that a role performs without interacting with other roles.
- Protocol. This refers to the specific pattern of interaction between roles.

After the roles have been identified and documented, the other important aspect is the interactions between roles. Note that this is done when analysing the system requirements after the agents are identified. This step examines how the agents collaborate to accomplish certain roles.

## 2.5.2 Design

In the design phase, the roles are mapped into agent types. Agent instances are created from each type. This is analogous to the classes and objects in the object-oriented world. The class diagram can be extended to be an agent diagram by stereo-typing the class as << agent >>. The same idea can be applied to agent instances by extending the object representation. The next step is to develop the service model. This model captures how each agent fulfil its role. The collaboration diagram can be extended to serve as a service model. Finally, the acquaintance model is created. This model captures the communication between agents. The interaction diagram can be extended to represent the acquaintance model.

In detail design, we then decide what type of agents will best implement the system requirements. Here, the software designer will need to decide if an agent will be a BDI (belief, desire, and intention), Reactive, Planning, Knowledge-based or some other new user-defined agent architecture. In addition, in situations where non-agent based entities are required to be incorporated into the system, they should be wrapped up as agents so that the entire design view of the system can be/have an agent-based view. Concepts on agent-based design pattern can also be applied during detail design.

## **2.5.3** Coding

In coding, the facilities provided by the agent-based infrastructure will be used to code the system. For example, in the JACK agent infrastructure, the system is essentially Java codes. However, extensions incorporated into the Java language to provide agent-related facilities are used to implement agent behaviours and functions.

# **2.5.4** Testing

Testing of agent systems follows testing of traditional software system. The test cases should be derived from the use-case model. If certain agent behaviours need to be verified, specific scenarios should be created to make verification possible. The system can then be tested.

# **2.6** Object- and Agent-Oriented Software Engineering Issues

Table 2.2 shows the functionality and feature differences between an object and an agent.

In most of the cases, agent-oriented designs are adapted from an existing objectoriented design methodology. Some even treat an agent as an active object. Agent-

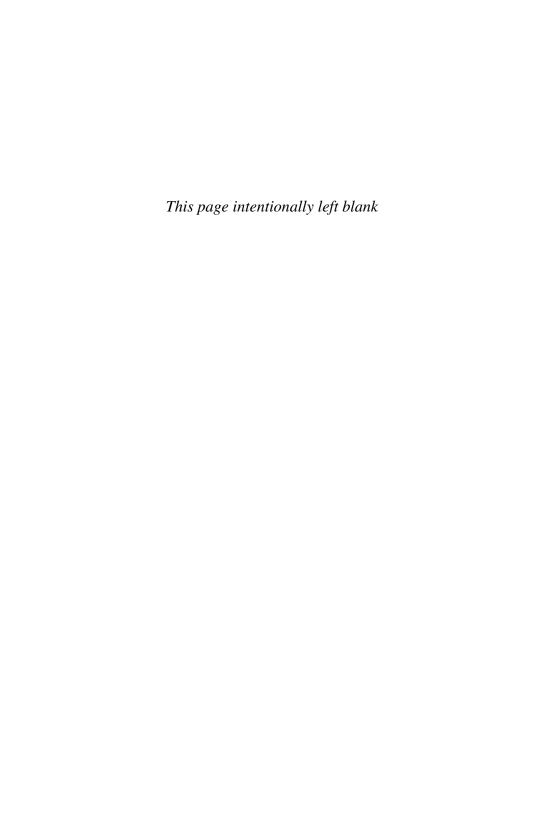
| Function/<br>Feature | Object   | Agent  | Comments   |
|----------------------|--|--|--|
| Method of execution  | By invoking other objects.   | By requesting other agents, it does not invoke other agents. In other words, a request may not be carried out as all agents have autonomy to decide.                               | Agents do it because they decide to do so.   |
| System goal          | Objects in a system may share a common goal.   | In a multi-agent system,<br>we cannot assume that<br>all agent will share<br>a common goal.  |  |
| Communication        | Communicate by message passing.  | Communication means is<br>more complicated than just<br>message passing. Needs<br>complex protocols that<br>involves coordination and<br>interaction with each other.              | Object-oriented communication structure is well structured and studied. Agent-based system protocols are still a research topic.   |
| Encapsulation        | Objects encapsulate some state and may be accessible to other objects (unless it is declared as private). It does not make decisions but simply executes after the direct intervention of human or other agents. | Agents encapsulate some state and are not accessible to other agents It makes decisions about what to do based on this state without direct intervention of human or other agents. |  |
| Behaviour<br>factor  | Object-oriented design<br>does not consider<br>object flexible<br>behaviour.   | Agent-oriented designs<br>have to take into account<br>flexible behaviour such as<br>reactive and pro-active<br>behaviour of an agent.   |  |
| Thread of control    | Objects are quiescent<br>most of the time,<br>becoming active only<br>when another object<br>requires their services.  | Agents are continually active and typically are engaged in an infinite loop of observing their environment, updating their internal state and selecting and executing an action.   | Note that concurrency<br>in object-oriented<br>approach does<br>attempt to fufil some<br>active role of an object<br>but it does not capture<br>the full autonomous<br>characteristics of an<br>agent. |
| UML                  | UML is defacto<br>standard for<br>object-oriented<br>approaches.   | Some agent researchers<br>are attempting to<br>adapt the UML notation<br>for agent-oriented<br>approaches.   | Note that adapting<br>from UML may be one<br>of the best approaches<br>for agent-oriented design.  |

Table 2.2: Object versus agent software engineering issues

oriented software engineering is at an infancy stage of research and development. There are still many issues with regard to multi-agent system software engineering

## design. Issues such as:

- Software design methodology of agents. These are currently under intensive research. Adapting from UML may be one of the best approaches for agentoriented design. However, it is also noted that the extension of UML for agentoriented design would still need significant work particularly the problem of the relationship between agents and objects.
- Relationship of agents with other software paradigms. It is not clear how the
  development of agents systems will coexist with other software paradigms.
  This point also makes adapting the UML for agent-orientated software engineering important. Currently, most of the object-oriented software systems are
  analysed and designed using UML.
- System robustness. With the massive number of agents dynamically interacting with one another in order to achieve their goals, it is not completely clear what happens to system stability and behaviour (chaotic behaviour). Some form of agent structure or organization may be necessary depending on the application domain.



## CHAPTER 3

# Biologically Inspired Algorithms

Nature provides the main inspiration in designing intelligent systems.

G. W. Ng

# 3.1 Introduction

Biologically inspired algorithms provide one of the essential tools in building intelligent systems. As the name implies, these algorithms are developed or invented through modelling and understanding of biological systems or nature's own intelligence, such as the immune system, the human brain, the evolutionary of gene and the collective intelligence of the insects or animals. The biologically inspired algorithms presented here are genetic algorithms, neural networks and fuzzy logic. This chapter will give an overview of these algorithms and also discuss some of the benefits and drawbacks of each algorithm. Included in this chapter is a brief introduction on the traditional expert system <sup>1</sup>. Lastly, we also present the notion of fusing these algorithms to leverage on their strength and offset their weakness. The chapter provides sufficient details to those engineers interested in utilizing these biologically inspired algorithms for various applications.

The chapter is arranged as follows: Section 2 presents the genetic algorithms. Section 3 presents the neural network. Section 4 presents the fuzzy logic algorithm. Section 5 presents the expert system. Section 6 briefly discusses the fusion of the various biologically inspired algorithms. Note that a brief description of the artificial immunological system is presented in the appendix and the cooperative intelligence is covered in chapter 8.

<sup>&</sup>lt;sup>1</sup>Fuzzy logic, neural networks and genetic algorithms could also be viewed as part of the modern expert system or knowledge-based information management system.

# 3.2 Evolutionary Computing

Evolutionary computing refers to algorithms that are based on the evolution of a population toward a solution of a certain problem. The population of possible solutions evolves from one generation to another, ultimately arriving at a satisfactory solution of the problem. The collection of evolutionary computing algorithms can be categorized into 4 methodologies, namely:

- Genetic algorithms.
- Genetic programming.
- Evolutionary strategies.
- Evolutionary programming.

These algorithms have been used successfully in many applications requiring the optimization of multi-dimensional functions. The differences among the various evolutionary computing algorithms are in their:

- Method of generating the evolving population; and
- Method of representing the population within the algorithm.

## 3.2.1 Biological inspiration

All living organisms consist of cells. In each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serves as a model for the whole organism. DNA is shaped as a double spiral with linking bars, like a twisted rope ladder.

A chromosome consists of genes, blocks of DNA. Genes are the body's chemical instructions. Each gene is an instruction to make a particular protein. Genes make us unique, such as colour of eyes, hair, etc. Each gene has its own position in the chromosome. This position is called locus.

A complete set of genetic material (all the chromosomes) is called a genome. A particular set of genes in a genome is called a genotype. The genotype is, with later development after birth, the base for the organism's phenotype, its physical and mental characteristics, such as eye colour, intelligence, etc.

There are 46 chromosomes in each of our body cells, divided into 23 pairs. One half of each chromosome pair came from the mother and the father.

During reproduction, genes from parents form, in some way, a whole new chromosome (also known as recombination or crossover). The newly created offspring can then be mutated. Mutation means, that the elements of the DNA are slightly changed. These changes are mainly caused by errors in copying genes from parents.

The fitness of an organism is measured by the success of the organism during its life.

## 3.2.2 Historical perspectives

Evolutionary computing was inspired by Darwin's theory of evolution. The idea of evolutionary computing was introduced in the mid '60s by Ingo Rechenberg in his work 'Evolution Strategies' [204]. Then the evolution strategies were evolution programs with the objective of finding floating point solutions in optimization problems. Hence, floating point number representation was used.

In the late '60s, Larry J. Fogel [79] and his colleagues developed what they called evolutionary programming. Evolutionary programming uses the selection of the fittest where the only structure modification allowed is the mutation process. There is no crossover and each point in the population represents the entire species.

Genetic algorithms (GAs) were developed by John Holland in 1975 [101]. In 1992 John Koza of Stanford University used genetic algorithms to develop programs to perform certain tasks. He called his method 'genetic programming' (GP). Hence GP is a special implementation of GAs. LISP (LIst PRocessing) programs were then used, because programs in this language can be expressed in the form of a 'parse tree', which is the concept the GA works on.

Of all the 4 methodologies, more work has been done with genetic algorithms and their various applications. Hence, in the following subsection, we will present these genetic algorithms in more detail.

# 3.2.3 Genetic algorithms

GA is among the most popular and rapidly growing area in evolutionary computation. GAs consider many points in the search space simultaneously, and have been found to provide rapid convergence to a near-optimum solution in many types of problems. GA will generally include the four fundamental genetic operations of: selection, crossover, mutation, and competition. GAs show much promise, but suffer from the problem of excessive complexity if used on problems that are too large. GAs are most appropriate for optimization-type problems and have been applied successfully in a number of applications such as control [88], weapon management and scheduling problems.

## 3.2.3.1 Introduction to GAs

GA starts with a set of solutions (represented by chromosomes) called a population. Solutions from one population are taken and used to form a new population. This is motivated by the hope that the new population will be better than the old one. Solutions which are selected to form new solutions (offspring) are selected according

to their fitness - the more suitable they are the more chances they have to reproduce. This reproduction circle (or evolving process) will be repeated continuously until the stop criteria are satisfied. The stop criteria could be that the number of evolving generations or the expected performances are met.

The formation of GAs can be summarized by the following steps:

- Step 1 Generate a random population of n chromosomes (suitable solutions for the problem).
- Step 2 Evaluate the fitness f(x) of each chromosome x in the population.
- Step 3 New population. Create a new population by repeating the following steps until the new population is completed.
- Step 4 Selection. Select two parent chromosomes from a population according to their fitness (the better the fitness, the bigger the chance to be selected).
- Step 5 Crossover. With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, the offspring is an exact copy of parents.
- Step 6 Mutation. With a mutation probability mutate new offspring at each locus (position in chromosome).
- Step 7 Accepting. Place new offspring in a new population.
- Step 8 Replace. Use new generated population for a further run of the algorithm.
- Step 9 Test. If the stop criteria is met, stop, and return the best solution in current population, else go to step 2.

The basic GA steps are very general. There are many variations or modification to these general GA steps in solving different problems.

The fundamental questions are:

- What types of encoding are suitable to create chromosomes? Or how to represent the population?
- What operators are effective for crossover and mutation?
- How to select the parents for crossover?
- How to formulate the fitness evaluation function?

All the above questions could be answered in many different ways. For example, there are many possible ways of encoding the chromosomes, typically used are binary strings (this will be illustrated in the next subsection of this chapter). However, depending on the problem to be solved, integers or real numbers could also be used.

## **3.2.3.2 Operator**

As you can see from the genetic algorithm outline, the crossover and mutation are the most important parts of the genetic algorithm. The performance is influenced mainly by these two operators. Before we explain more about crossover and mutation, some information about encoding the chromosomes is necessary.

How do we encode the chromosome? The chromosome should in some way contain information about the solution that it represents. The chromosomes encoded by a binary string (commonly used) could then look like this:

| Chromosome 1 | 1001100100110110 |
|--------------|------------------|
| Chromosome 2 | 1001111000011110 |

Each chromosome has one binary string. Each bit in this string can represent some characteristic of the solution. Or the whole string can represent a number.

#### Crossover

After we have decided what encoding we will use, we can implement the crossover. Crossover selects genes from parent chromosomes and creates a new offspring. The simplest way is to choose randomly some crossover point where everything before this point is copied from a first parent, and then everything after the crossover point is copied from the second parent.

Crossover can then look like this (where '—' is the crossover point):

| Chromosome 1 | 10011 — 00100110110 |
|--------------|---------------------|
| Chromosome 2 | 10011 — 11000011110 |
| Offspring 1  | 10011 — 11000011110 |
| Offspring 2  | 10011 — 00100110110 |

There are other ways on how to make a crossover. For example, we can choose more crossover points. Specific crossovers made for a specific problem can improve performance of the genetic algorithm.

#### Mutation

After a crossover is performed, mutation takes place. The mutation process helps prevent solutions in the population from falling into a local optimum. Mutation changes randomly the new offspring. For binary encoding we can switch a few randomly - chosen bits from 1 to 0 or from 0 to 1. Mutation can then be as follow:

| Original offspring 1 | 1001111000011110 |
|----------------------|------------------|
| Original offspring 2 | 1001100100110110 |
| Mutated offspring 1  | 1000111000011110 |
| Mutated offspring 2  | 1001101100110110 |

Mutation means different things for different data types. In the above example, which uses a binary string, mutation involves flipping the bits in the string with a given probability. A typical mutator for a tree, on the other hand, would swap subtrees with a given probability.

The mutation depends on the encoding as well as the crossover. For example, when we are encoding permutations, mutation could be exchanging two genes.

#### 3.2.3.3 Parameters of GAs

Here, we cover three GA parameters namely: the crossover probability, mutation probability and the population size.

### **Crossover and Mutation Probability**

There are two basic parameters of GAs - crossover probability and mutation probability. Crossover probability determines how often crossover will be performed. If there is no crossover, the offspring is an exact copy of the parents. If there is a crossover, the offspring is made from parts of the parents' chromosome. If the crossover probability is 100%, then all offsprings are made by the crossover. If it is 0%, the whole new generation is made from exact copies of chromosomes from the old population (but this does not mean that the new generation is the same!). Crossover is made in the hope that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better. However, it is good to leave some part of the population to survive into the next generation.

Mutation probability says how often the gene in the chromosome will mutate. If there is no mutation, the offspring is taken (or copied) after the crossover without any change. If mutation is performed, part of the chromosome is changed. If the mutation probability is 100%, the whole chromosome is changed - if it is 0%, nothing is changed. Mutation is made to prevent GAs from falling into local extreme, but this should not occur very often because then the GAs will in fact change to a random search.

## **Population size**

Population size determines how many chromosomes are in a population (in one generation). If there are too few chromosomes, GAs have fewer possibilities to perform a crossover and only a small part of the search space is explored. On the other hand,

if there are too many chromosomes, the GA slows down. Research shows that after some limit values (which depend mainly on encoding and the problem), it is not useful to increase the population size, because it does not solve the problem any faster.

#### **3.2.3.4** Selection

As you already know from the GA outline, chromosomes are selected from the population to be parents for crossover and mutation. The problem is how to select these chromosomes. According to Darwin's evolution theory the best ones should survive and create new offsprings. There are many methods of selecting the best chromosomes, for example, roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection, elitism and others. Two of them will be described in this section.

#### **Roulette Wheel Selection**

The Roulette wheel selection method provides a mechanism where the parents are selected according to their fitness. The better the chromosomes are, the more chances they will be selected. Imagine a roulette wheel where all the chromosomes in the population are placed. The space the chromosomes occupy in the wheel determine how good are the chromosomes. The 'goodness' of the chromosome is measured by the probability of the reproduction. The probability of the reproduction is computed by  $P(X_i) = \frac{f(X_i)}{\sum_{j=1}^p f(X_j)}$  where  $X_i$  is the i chromosomes.

#### Elitism

When creating a new population by crossover and mutation, we run the risk that we will lose the best chromosome. The elitism method overcomes this by keeping the best-fit chromosome. The method transfers the best chromosome (or a few best chromosomes) to the new population. The rest of the steps are carried out as discussed. Elitism can very rapidly increase the performance of the GA, because it prevents losing the best found solution.

#### 3.2.3.5 Further information on crossover and mutation operators

Crossover and mutation are two basic operators of GAs. The performance of GAs is affected by how the operators are implemented. Listed below are some possible ways of how a crossover and mutation operator could be performed:

### Crossover

- 1. Single point crossover one crossover point is selected. Example: 10010111 + 11010010 = 10010010
- 2. Two point crossover where two crossover points are selected, the binary string from the beginning of the chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent.
- 3. Uniform crossover bits are randomly copied from the first or from the second parent. Example: 11001011 + 11011101 = 11011111
- 4. Arithmetic crossover some arithmetic operation is performed to make a new offspring. Example: 11001011 + 11011111 = 11001001 (AND)

#### Mutation

- 1. Bit inversion selected bits are inverted. Example 11001001 = 10001001
- 2. Add a small number (for real value encoding) the selected values are added (or subtracted) with a small number. Example: (1.29 5.68 2.86 4.11 5.55) = (1.29 5.68 2.73 4.22 5.55).

## Some drawbacks of genetic algorithms

The first drawback of a GA is that it has no concept of an 'optimal solution'. It cannot be proven that the solution is optimal. The second drawback of the GA is that it does not address when to stop evolving in terms of the length of time, number of iterations and the population size you would like to explore during the search for the best solution.

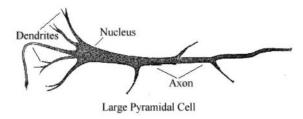
## 3.3 Neural Networks

# 3.3.1 Biological inspiration

The human brain is made up of more than 100 billion nerve cells. Each of these nerve cells is known as a neuron. Neurons are spider-shaped cells (Figure 3.1) and have three basic elements namely: the dendrites, the soma and the axon.

The dendrites receive signals from other neurons. These signals are electric impulses that are transmitted across a synaptic gap by means of a chemical process.

The soma, or cell body, sums the incoming signals. When a sufficient input is received, the cell fires; that is, it transmits a signal over its axon to other neurons.



**Figure 3.1:** The nerve cell.

An axon is the winding tail of the neuron. Its length could be up to 1 meter long. Fatty covering called a myelin sheath often surrounds axons. Myelin acts like insulation around an electric wire and helps speed nerve messages. Axons of one neuron are linked to the dendrites of other neurons. Using this link, each neuron is connected to as many as 25,000 other neurons. The brain has trillions of these interconnections and different pathways for nerve signals to flow through.

The study of artificial neural networks is trying to model or mimic the biological interconnections of the neurons.

## 3.3.2 Historical perspective

The concept of using neural networks (NNs) came into existence in the mid 1940s. In 1943, McCulloch and Pitts designed boolean logic by combining many simple units. The weights on a McCulloch-Pitts network are set so that each unit performs a particular logic function [155]. These logic functions are combined to produce the desired network output. The McCulloch-Pitts network was generally regarded as the first NN. Then came Hebb [96], a psychologist at McGill University, who published a biologically plausible learning rule. Hebb's work formed some important theoretical aspects of the learning algorithms for NNs. In around 1954, Minsky introduced the reinforcement learning (RL), a term borrowed from animal learning literature. RL was also independently introduced and applied in control theory by Waltz [228] and Michie [159]. In 1957, Bellman created dynamic programming (DP). DP provides an efficient mechanism for sequential-decision making which was found to be useful in reinforcement learning in the 90s. In 1958, Frank Rosenblatt coined the term perceptron, which became a popular word used in the NNs for the 80s and 90s.

Perceptron is a single layer NN, hence it has no hidden layer.

Then came the least mean square (LMS) algorithm, also known as the Widrow-Hoff rule or delta rule, developed by Bernard Widrow and Marcian Hoff of Stanford University. This algorithm was applied to a simple adaptive linear neuron (Adaline).

In the 70s, much attention was directed towards the field of artificial intelligence (AI). This shift in focus was partly due to the Minsky and Papert analysis which challenged the incipient neural theory by establishing criteria for what a particular network could and could not do [52]. However, during these silent years, three pioneering works on NNs were done. These are listed below:

- Werbos [237] proposed the concept of backpropagation (BP) for forecasting analysis.
- Grossberg [89] proposed a learning algorithm for the instar and outstar network configuration.
- Willshaw and Ver Der Malsburg [241] proposed a self-organizing feature map on biological grounds to explain the problem of retinotopic mapping from the retina to the visual cortex.

It is interesting to observe that what happened in the 70s is repeated in the early 80s by independent researchers. To begin with, we have Hopfield [102], who introduced the network architecture that has come to be known as the Hopfield network. The Hopfield network has great similarity to the 'Brain-State-in-a-Box' (BSB). Both the networks are fully interconnected. Therefore, each unit is connected to every other unit. The only difference is that BSB network has self-connections while the Hopfield network does not have. At about the same time, Kohonen independently introduced self-organizing feature mapping to capture the essential features of computational maps in the brain [123]. The Kohonen network is quite similar to the Willshaw and Ver Der Malsburg self-organizing feature map. In the same year, Parker [182] independently discussed the BP algorithm. The BP algorithm, sometimes also known as the generalized delta learning rule, was later reinvented and made popular by Rumelhart and McClelland in 1986 [202].

In the mid 80s, there was the development of thermodynamic models. Two examples of such models are harmony theory [211] and the Boltzmann machine. Ackley *et al.* [1] developed a learning rule using the Boltzmann machine. These approaches have much in common. Both employ binary units whose values are determined probabilistically according to the Boltzmann equation.

A sudden upsurge of interest in NNs occurred after the publication of the book by Rumelhart and McClelland [202]. The following year, the IEEE conducted the first conference on neural networks. The International Neural Network Society (INNS) was also formed in the same year and the first issue of its journal, *Neural Networks*, appeared early in 1988. Thereafter the IEEE Control Systems Magazine devoted at least three special issues to the use of NNs for control (April 1988, 1990 and 1992).

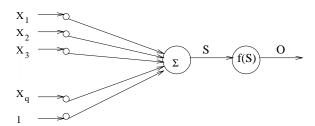
In the late 80s to early 90s, we have seen many developments in NNs. New NN structures such as: the radial basis function network [41], bidirectional associative memories [126], B-splines network [164], digital neurocomputer [97], functional-link networks [181], knapsack packing networks [98] and wavelet networks [248] were proposed. Different combinations of network topologies to improve the representational structure, such as recurrent backpropagation which combines feedback and feedforward topologies, were explored [3, 188, 240].

During this period a number of papers were published seeking to improve the BP algorithm. For examples, backpropagation with momentum, delta-bar-delta [110], Quickprop [76], decoupled momentum [160] and adaptive backpropagation [183]. The number of books published on NNs have also increased manyfold [22, 36, 27, 45, 52, 65, 69, 84, 99, 77, 125, 95, 158, 172, 214, 219, 224, 232, 231, 244].

#### 3.3.3 Neural network structures

#### **3.3.3.1** Basic units

The basic processing element of NNs is often called a neuron (analogy with neurophysiology), unit, node, or sometimes cell. In this book, we will simply call this basic element a unit. Each unit has multiple inputs and a single output, as shown in Figure 3.2. Many of these basic processing elements may be considered to have two basic components, namely the summer and the activation function.



**Figure 3.2:** The basic elements of neural networks.

#### Summer

The summer (or also known as adder) sums up the input signals. The respective input signals, before being summed up by the summer, are weighted by the respective link or synapses of the neuron. In other ways, the summer is doing a linear combination of the weighted input signals.

#### **Activation function**

The activation function transforms the summer output to neuron output through a non-linear function. This transformation squashes (or limits) the amplitude of the output neuron to some finite values. There are a number of different activation functions. These activation functions could be classified as:

- 1. Differentiable/non-differentiable.
- 2. Pulse-like/step-like.
- 3. Positive/zero-mean.

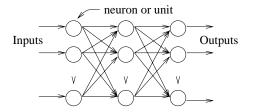
Each of the above classifications are briefly described and examples given:

- Classification 1 distinguishes smooth from sharp functions. Smooth functions
  are needed for some adaptation algorithms, such as backpropagation, whereas
  discontinuous functions are needed to give a true binary output. Examples
  of smooth functions are: sigmoid function, hyperbolic tangent function and
  radial basis function (RBF). An example of a sharp function is the hard-limiter
  or threshold transfer function.
- Classification 2 distinguishes functions which only have a significant output value for inputs near to zero from functions which only change significantly around zero. For examples, the RBF can be pulse-like and the hard-limiter step-like.
- Classification 3 refers to step-like functions. Positive functions change from 0 at  $-\infty$  to 1 at  $\infty$ ; zero-mean changes from -1 at  $-\infty$  to 1 at  $\infty$ . The sigmoid function is a positive function and the hyperbolic tangent function is zero-mean. A hard-limiter function can either be positive or zero-mean function depending on how the limit is defined.

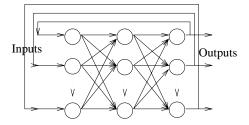
There are cases where the network topology consists of rule units; each of the units corresponds to a specific fuzzy rule. Therefore, all the units form a general fuzzy rule [218].

## 3.3.3.2 Network topology

There are basically two network topologies, namely feedforward neural network (FNN) and recurrent neural network (RNN). Figure 3.3 shows the basic structures of FNN and RNN. Both of these networks can have either fully connected or partially connected structures; and either single layer or multilayer. A single layer neural network (SNN) has no hidden layer, hence it has only an input layer and an output layer (sometimes also known as a two layers network). A multilayer neural network (MNN) has at least one hidden layer.



#### Feedforward Neural Network



Recurrent Neural Network

Figure 3.3: Difference between feedforward and recurrent neural networks.

#### Feedforward neural networks

A FNN is completely feedforward with no past state of the network feeding back to any of its units. FNN is classified as fully connected, if every unit in the layer of the network is connected to every other unit in the adjacent forward layer. However, if some of the communication links (or sometimes called synaptic connections) are missing from the network, we say that the network is partially connected.

FNN is static mapping, so theoretically it is not feasible to control or identify dynamic systems. To extend this essentially steady state mapping to the dynamic domain would mean to adopt an approach similar to the linear theory of ARMA (autoregression moving average) modelling. Here, a time series of past real plant inputs and outputs values are used as inputs to the FNN with the help of tapped delay lines (TDL). The nonlinear dynamic plant in discrete time is

$$y_p(k) = f(y_p(k-1), \dots, y_p(k-n_y), u(k-2), \dots, u(k-n_u))$$
 (3.1)

where f(.) is the unknown nonlinear function,  $y_p$  is the plant output, u is the control signal,  $n_y$  and  $n_u$  are the number of past outputs and inputs of the plant depending on the plant order. To represent this plant, the NN model is fed with the past output and input values of the plant as follows (see Figure 3.4):

$$y_m(k) = NN(y_p(k-1), \dots, y_p(k-n_y), u(k-2), \dots, u(k-n_u))$$
 (3.2)

This assumes that the plant order is known and its states are measurable.

According to Levin [140], to represent the dynamics of the system sufficiently, at least l past measurements of the real plant output and input feeding back to the FNN input are required, where  $l \ge n_u + n_u + 1$ .

The assumptions that all the plant outputs and inputs are measurable and available for feedback is sometimes unrealistic; for instance some plant outputs may not be accessible for sensors (such as rotor currents in a squirrel cage induction motor) or the required sensors may be too expensive or unreliable. Hence for such a system, the FNN may not be suitable and a RNN may be preferred. This will be discussed in the next subsection.

#### Recurrent neural networks

A RNN distinguishes itself from a FNN in that it has at least one feedback loop. It is claimed that the presence of feedback loops has a profound impact on the learning capability of the network and on its performance [188, 185, 240, 133]. In the sense of control theory, the feedback loop makes a RNN a nonlinear dynamic system. The feedback loops commonly involve unit delays if dealing with discrete-time systems, or integrators in the continuous-time case. RNNs may be preferred to FNNs when:

- The measured plant outputs are highly corrupted by noise.
- The dynamics of the nonlinear process are complex and unknown.
- Direct state feedback is impossible and only partial measurement is available from the plant.

There are apparently many structures of RNN using different combinations of feeding back the states to the units in each layer. In RNN, the more general is the structure, therefore more feedback interconnections (or fully-connected recurrent neural network (FRNN)), the 'richer' is the dynamic representation. Hence, FRNNs are sometimes said to have global dynamic representation. The more general types are the Jordan, Elman and Williams-Zipser networks. In a Jordan network, the past output values of the network are fed back into the hidden units or the input layer. In the Elman network, the past values of the hidden units are fed back into themselves [75, 134].

One of the problems in using FRNNs is the stability issue [127, 222]. Here, the topic is addressed by looking at the recurrent weights. FRNNs are normally trained by using the BP algorithm. To improve the stability of FRNNs, it is proposed here that the total dynamic feeding back has to be less than 1. Hence, by considering the FRNN of an Elman network trained by a BP algorithm,

$$0 < \sum_{i=1}^{p} w_{rij}(k) f'_{hj}(S_{hj}(k-1)) < 1$$
(3.3)

where  $w_{rij}(k)$  is the recurrent weight at time step k, p is the number of hidden units and  $f_{hj}$  is the activation function of the jth hidden unit, where usually the sigmoid or hyperbolic tangent function is used. Since  $|f_{hj,\max}^{'}| = 1$ ,  $w_{rij}$  has to be bounded as follows to ensure FRNN stability:

$$0 < w_{rij} < \frac{1}{p} \tag{3.4}$$

## Single layer and multilayer neural networks

NNs in feedforward or recurrent structures can be connected in a single layer neural networks (SNNs) or multilayer neural networks (MNNs). The radial basis function network (RBFN), B-spline network, functional-link network (FLN), CMAC, lattice associative memory network, Adaline and perceptron are feedforward SNNs. Multilayer perceptron (MLP) is a feedforward MNN. Boltzmann machine and bidirectional associative memories (BAM) could be classified as recurrent MNNs. Adaptive resonance theory (ART), Kohonen and Hopfield networks could be classified as recurrent SNNs.

Both the MNN and SNN are universal approximators. A MNN with one hidden layer and sufficient hidden units can approximate any arbitrary nonlinear function [86, 62, 103, 190, 104, 87, 114]. Although a MNN with two hidden layers may give a better approximation for some specific problems [213], DeVilliers and Barnard [68] have demonstrated that a MNN with 2 hidden layers are more prone to fall into local minima.

A SNN is sometimes categorized as locally generalizing [6]. The network is considered local since only a small subset of adaptable parameters can potentially affect the network output in a local region of the input space. The MNN is sometimes categorized as globally generalizing. This is because one or more adaptable parameters in the network can potentially affect the network output at every point in the input space.

# 3.3.4 Learning algorithms

Basically, there are three types of neural network learning algorithms. These are: supervized learning, reinforcement learning and unsupervized learning.

## 3.3.4.1 Supervized learning

## Early learning algorithms

The early learning algorithms are designed for single layer neural networks. They are generally more limited in their applicability. Some of the early algorithms are perceptron learning, LMS learning and Grossberg learning.

## **Perceptron learning**

Rosenblatt's single layer perceptron is trained as follows:

- 1. Randomly initialize all the networks weights.
- 2. Apply the inputs x and calculate the sum of each unit  $S_j(k) = \sum_{i=1}^q x_i(k) w_{ij}(k)$ .
- 3. The outputs from each unit are

$$O_j(k) = \begin{cases} 1 & S_j(k) > \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

- 4. Compute the errors  $e_j(k) = O_{dj}(k) O_j(k)$  where  $O_{dj}(k)$  is the known desired output value.
- 5. Update each weight as  $w_{ij}(k+1) = w_{ij}(k) + \eta x_i(k)e_i(k)$ .
- 6. Repeat steps 2 to 4 until the errors reach the satisfactory level.

### LMS or Widrow-Hoff learning

The least mean square (LMS) learning algorithm, sometimes known as Widrow-Hoff learning or delta rule, is quite similar to the perceptron learning algorithm. The differences are:

1. The error is based on the sum of inputs to the unit S rather than the output binary value from the unit O. Therefore

$$e_j(k) = O_{dj}(k) - S_j(k)$$

2. The linear sum of the inputs S are passed through bipolar sigmoidal functions, which produces the output +1 or -1, depending on the polarity of the sum.

## **Grossberg learning**

The Grossberg learning algorithm, sometimes known as instar and outstar training, is updated as follows [231]:

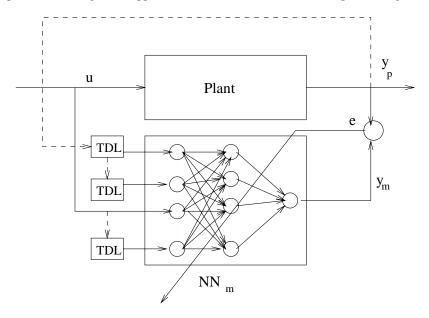
$$w_i(k+1) = w_i(k) + \eta[x_i(k) - w_i(k)]$$

where  $x_i$  could be the desired input values (instar training) or the desired output values (outstar training) depending on the network structure.

#### 3.3.4.2 First order gradient methods

## **Backpropagation**

The backpropagation (BP) algorithm is the most well known and widely used among other learning algorithms described in the literature. It is based on steepest-descent techniques extended to each of the layer in the network by the chain rule. Hence, the algorithm computes the partial derivative  $\frac{\partial E(k)}{\partial W(k-1)}$ , of the error function with respect to the weights. Suppose we want to model a nonlinear plant using a NN



**Figure 3.4:** Multilayer feedforward neural network modelling a nonlinear dynamic plant.

(see Figure 3.4). The NN model output is  $y_m$ . The error function is defined as  $E(k)=\frac{1}{2}[y_p(k)-y_m(k)]^2$  where  $y_p$  is the plant output and  $e_m(k)=y_p(k)-y_m(k)$ . The objective is to minimize the error function E(k) by taking the error gradient with respect to the parameters or weight vector, say W, that is to be adapted. The weights are then updated by using

$$W(k) = W(k-1) + \eta\left(-\frac{\partial E(k)}{\partial W(k-1)}\right)$$
(3.5)

where  $\eta$  is the learning rate and

$$\frac{\partial E(k)}{\partial W(k-1)} = -e_m(k) \frac{\partial y_m(k)}{\partial W(k-1)}$$

$$= -e_m(k) \frac{\partial O_o(k)}{\partial W(k-1)}$$

The network output is  $O_o(k) = y_m(k)$ , in this example.

This standard BP algorithm is simple to implement and computationally less complex than other modified forms of BP algorithms. Nevertheless, it has some disadvantages such as:

- 1. Slow convergence speed.
- 2. Sensitivity to initial conditions [124].
- 3. Trapping in local minima.
- 4. Instability if learning rate is too large.

Despite the above disadvantages, it is popularly used. There are numerous extensions to improve the BP algorithm. Some of the more common techniques are as follows.

### **Backpropagation with momentum**

The basic improvement to a BP algorithm is to introduce a momentum term in the weights updating equation [201]. Hence, the weights updated by backpropagation with momentum (BPM) are

$$W(k) = W(k-1) + \left(-\eta \frac{\partial E(k)}{\partial W(k-1)}\right) + \alpha \Delta W(k-1)$$
(3.6)

where the momentum factor  $\alpha$  is commonly selected between  $0 < \alpha < 1$ . Adding the momentum term improves the convergence speed and helps the network from being trapped in a local minimum.

A modification to equation (3.6), such as

$$W(k) = W(k-1) + \left(-\eta \frac{\partial E(k)}{\partial W(k-1)}\right) + \alpha \Delta W(k-1) + \beta \Delta W(k-2) \quad (3.7)$$

was proposed by Nagata *et al.* [168], where  $\beta$  is a constant value decided by the user. Nagata *et al.* [168] claimed that the  $\beta$  term reduces the possibility of the network being trapped in the local minimum. However, this seems to be repeating the role of the  $\alpha$  term. Hence, the advantage of adding the  $\beta$  term is not clear at the moment.

#### Delta-bar-delta

The delta-bar-delta (DBD) learning rules use an adaptive learning rate to speed up the convergence. The adaptive learning rate adopted is based on a local optimization method. It is an improvement from the delta-delta learning rule. This technique uses gradient descent for the search direction, and then applies individual step sizes for each weight. This means that the actual direction taken in weight space is not necessarily along the line of the steepest gradient [110].

If the weight updates between consecutive iterations are in opposite directions, the step size is decreased, otherwise it is increased. This is prompted by the idea that if the weight changes are oscillating, the minimum is between the oscillations, and a smaller step size might find that minimum. The step size may be increased again once the error has stopped oscillating.

Let  $\eta_{ij}(k)$  denote the learning rate for the weight  $w_{ij}(k)$ , then

$$\eta_{ij}(k+1) = \eta_{ij}(k) + \Delta \eta_{ij}(k) \tag{3.8}$$

and  $\Delta \eta_{ij}(k)$  is as follows:

$$\Delta \eta_{ij}(k) = \begin{cases} \kappa & \bar{\psi}_{ij}(k-1)\psi_{ij}(k) > 0\\ -\beta \eta_{ij}(k) & \bar{\psi}_{ij}(k-1)\psi_{ij}(k) < 0\\ 0 & \text{otherwise} \end{cases}$$
(3.9)

where  $\psi_{ij}(k) = \frac{\partial E(k)}{\partial w_{ij}}$  and  $\bar{\psi}_{ij}(k) = (1-\varepsilon)\psi_{ij}(k-1) + \varepsilon\bar{\psi}_{ij}(k-1)$ . The  $\varepsilon$  is a positive constant.  $\kappa, \beta$  and  $\varepsilon$  are parameters specified by the user. The quantity  $\bar{\psi}_{ij}(k)$  is basically an exponentially decaying trace of gradient values. When the  $\kappa$  and  $\beta$  are set to zero, the learning rates assume a constant value as in a standard BP algorithm.

Results cited by Jacobs [110] indicate that using momentum along with the DBD algorithm can enhance performance considerably. However, it can also make the search diverge wildly; especially if  $\kappa$  is moderately large. The reason is that momentum 'magnifies' learning rate increments and quickly leads to inordinately large learning steps. One possible solution is to keep the  $\kappa$  factor very small, but this can easily lead to slow increase in  $\eta$  and little speedup.

Simulations have also shown that the DBD algorithm is very sensitive to small variations in the value of its parameters, especially  $\kappa$ . To overcome these problems the decoupled momentum is suggested.

## Second order gradient methods

Second order gradient methods make use of second derivatives of the error in the weight space. The key to second order gradient methods is the Hessian matrix H. The Hessian matrix is a matrix of second derivatives of E with respect to the weights W.

$$H(k) = \frac{\partial^2 E(k)}{\partial W(k-1)^2} \tag{3.10}$$

The Hessian matrix contains information about how the gradient changes in different directions in weight space. It answers the question: 'How does the gradient change if I move off from this point in that direction?'.

#### Newton

The Newton method weights update are [58]

$$W(k) = W(k-1) + \eta [H(k)]^{-1} \frac{\partial E(k)}{\partial W(k-1)}$$
(3.11)

However, the Newton method is not commonly used because calculating the Hessian matrix is computationally expensive. Furthermore an Hessian matrix may not be positive definite at every point in the error surface. To overcome the problem, several methods have being proposed to approximate the Hessian matrix.

#### Gauss-Newton

The Gauss-Newton method produces an  $n \times n$  matrix which is an approximation to the Hessian matrix, having elements represented by

$$H(k) = \psi^{T}(k)\psi(k) \tag{3.12}$$

where  $\psi(k) = \frac{\partial e(k)}{\partial W(k-1)}$  [145]. However, the Gauss-Newton method may still have ill-conditioning if H(k) is close to or is singular.

## Levenberg-Marquardt

The Levenberg-Marquardt (LM) method overcomes this difficulty by including an additional term  $\sigma$  which is added to the Gauss-Newton approximation of the Hessian matrix giving

$$H(k) = \psi^{T}(k)\psi(k) + \sigma I \tag{3.13}$$

where  $\sigma$  is a small positive value and I is the identity matrix.  $\sigma$  could also be made adaptive by having

$$\sigma = \begin{cases} \sigma \beta & E_c(k) > E_c(k-1) \\ \frac{\sigma}{\beta} & E_c(k) < E_c(k-1) \end{cases}$$

where  $\beta$  is a value decided by the user. Notice that when  $\sigma$  is large, the algorithm becomes a backpropagation algorithm with learning rate  $\frac{1}{\sigma}$ , and when  $\sigma$  is small, the algorithm becomes Gauss-Newton.

### Quickprop

Fahlman [76] proposed the Quickprop method using heuristics based loosely on the Newton method. Fahlman highlighted that the Quickprop algorithm is based on two risky assumptions:

- The error vs weight curve for each weight can be approximated by a parabola.
- The change in the shape of the error curve, as seen by each weight, is not affected by all the other weights that are changing at the same time.

Hence, Quickprop implicitly assumes that the Hessian matrix is diagonal. The weights update of the Quickprop algorithm,  $\Delta W(k)$ , is as follows:

$$\Delta W(k) = \frac{\Psi(k)}{\Psi(k-1) - \Psi(k)} \Delta W(k-1)$$

where  $\Psi(k) = \frac{\partial E(k)}{\partial W(k-1)}$ . Fahlman claimed that when applied iteratively Quickprop is very efficient.

### Conjugate gradient decent

The idea behind conjugate gradient decent (CGD) is to choose search directions that complement one another. This is to avoid the possibility of 'undoing' the minimization of previous iterations, by choosing appropriate search directions.

Assume that the error surface is quadratic in weight space.

$$E(k) = C^{T}W(k-1) + \frac{1}{2}W^{T}(k-1)H(k)W(k-1)$$
(3.14)

where C is a constant matrix and H is a Hessian matrix, constant throughout the weight space. The search direction at iteration k is denoted by v(k).

Set the first direction, v(1), to minus the gradient  $-\frac{\partial E(1)}{\partial W(0)}$ . Set subsequent directions using:

$$v(k) = -\frac{\partial E(k)}{\partial W(k-1)} + \beta(k)v(k-1)$$
(3.15)

where  $\beta(k)$  is a scalar value like an adaptive momentum term.  $\beta$  may be defined in a number of ways, each of which produces conjugate directions. The Polak-Ribiere rule for  $\beta$  is [99]

$$\beta(k) = \frac{[\Psi(k) - \Psi(k-1)]^T [\Psi(k)]}{[\Psi(k-1)]^T [\Psi(k-1)]}$$
(3.16)

The Hestenes-Stiefel rule for  $\beta$  is [163]

$$\beta(k) = \frac{[\Psi(k-1) - \Psi(k)]^T [\Psi(k)]}{v(k)^T [\Psi(k-1)]}$$
(3.17)

CGD will reach the minimum of a quadratic error surface in, at most, as many steps as there are dimensions of the weight space. For non-quadratic error surfaces, the minimum may not have been reached after this number of steps. Instead, the algorithm is restarted by setting  $\beta$  to zero for a step, and the procedure continues as before.

Conjugate gradients give only a search direction, not a step size. Furthermore, the power of conjugate gradients is only apparent if the error is minimized along the current search direction.

#### 3.3.4.3 Reinforcement learning

Thus far, we have assumed that the correct 'target' or the desired output values are known for each time step or each input pattern. But, in some situations, there is less detailed information available. In the extreme case there is only a single bit of information, saying whether the output is right or wrong. Reinforcement learning (RL) procedures are applicable to this extreme case. Hence, RL is sometimes called 'learning with a critic'. One advantage of RL is that it does not need to compute derivatives. This makes it suitable for some complex systems where derivatives are difficult to obtain. The major disadvantage of RL is its slow (inefficient) learning process [85, 141]. Some of the classifications of RL are: delay and immediate RL [119]; associative and nonassociative RL [95].

In delay RL, the environment only gives a single scalar reinforcement evaluation after a sequence of input state vectors during the system operation. For immediate RL, the environment immediately returns a scalar reinforcement evaluation of the input state vectors.

In nonassociative RL, the reinforcement is the only input that the learning system receives from its environment. In associative RL, the environment provides additional forms of information other than the reinforcement. The combination of adaptive critic element (ACE) and associative search element (ASE) controller described in Barto et al. [21] is an example of learning by delay and associative RL.

Three basic RL algorithms are presented here. Other developments in RL algorithms not discussed here are Temporal Difference (TD) [217], Q-learning [233], Heuristic dynamic programming [238] and TD-Gammon [220].

### Linear reward-penalty learning

Linear reward-penalty learning was proposed by Narendra and Thathachar [170]. When the reinforcement signal is positive (+1), the learning rule is

$$P_i(k+1) = P_i(k) + a[1 - P_i(k)]$$
  
 $P_j(k+1) = (1-a)P_j(k)$  for  $j \neq i$ 

If the reinforcement signal is negative (-1), the learning rule is

$$P_i(k+1) = (1-b)P_j(k)P_i(k)$$
  
 $P_j(k+1) = \frac{b}{r-1} + (1-b)P_j(k)$  for  $j \neq i$ 

where a and b are learning rates,  $P_i(k)$  denotes the probability at iteration k and r is the number of actions taken. For positive reinforcement, the probability of the current action is increased with relative decrease in the probabilities of the other actions. The adjustment is reversed in the case of negative reinforcement.

#### Associative search learning

In the associative search learning the weights are updated as follows [21]:

$$w_i(k+1) = w_i(k) + \eta r(k)e_i(k)$$
(3.18)

where r(k) is the reinforcement signal and e is eligibility. Positive r indicates the occurrence of a rewarding event and negative r indicates the occurrence of a punishing event. It can be regarded as a measure of the change in the value of a performance criterion. Eligibility reflects the extent of activity in the pathway or connection link. Barto et al. [21] generated exponentially decaying eligibility traces  $e_i$  using the following linear differential equation

$$e_i(k+1) = \delta e_i(k) + (1-\delta)y(k)x_i(k)$$
 (3.19)

where  $\delta, 0 \leq \delta < 1$  determines the trace decay rate, x is the input and y is the output.

#### Adaptive critic learning

The weights update in a critic network is as follows [21]:

$$w_i(k+1) = w_i(k) + \eta[r(k) + \gamma \rho(k) - \rho(k-1)]\bar{x}_i(k)$$
(3.20)

where  $\gamma$ ,  $0 \leq \gamma < 1$ , is a constant discount factor, r(k) is the reinforcement signal from the environment and  $\bar{x_i}$  is the trace of the input variable  $x_i$ .  $\rho(k)$  is the prediction at time k of eventual reinforcement which can be described as a linear function of  $\rho(k) = \sum_i w_i(k) x_i(k)$ . The adaptive critic network output  $(r_o)$ , the improved or internal reinforcement signal, is computed from the predictions as follows:

$$r_o(k) = r(k) + \gamma \rho(k) - \rho(k-1)$$
 (3.21)

## 3.3.4.4 Unsupervized learning

In this section, three important unsupervized learning algorithms are discussed, namely: Hebbian learning (and its derivatives), Boltzmann machines learning and Kohonen self-organizing learning.

### **Hebbian learning**

Hebbian learning is sometimes called correlation learning. The weights are updated as follows:

$$w_{ij}(k+1) = w_{ij}(k) + S_i(k)S_j(k)$$
  
 $S_j(k) = \sum_i (O_i(k)w_{ij}(k))$ 

where  $w_{ij}(k)$  is the weight from *i*th unit to *j*th unit at time step k,  $S_j$  is the excitation level of the source unit or *j*th unit and  $O_i$  is the excitation level of the destination unit or the *i*th output unit.

Hebb's concept involves no 'teacher'. In the Hebbian system, learning is a purely local phenomenon, involving only two units and a synapse; no global feedback system is required for the neural pattern to develop.

Since Hebb introduced his learning rule, the rule has been modified and improved by several researchers.

## Signal Hebbian learning

A network using the sigmoidal activation function with Hebbian learning is said to employ signal Hebbian learning. In this case, Hebbian equations are modified to the following form:

$$O_i(k) = f(S_i(k)) = \frac{1}{1 + e^{-S_i(k)}}$$

$$w_{ij}(k+1) = w_{ij}(k) + O_i(k)O_j(k)$$

## Differential Hebbian learning

A variant of signal Hebbian learning is to calculate the product of the previous changes of the outputs to determine the weight change. This method is called differential Hebbian learning, and the equation is as follows:

$$w_{ij}(k+1) = w_{ij}(k) + [O_i(k)O_i(k-1)][O_j(k)O_j(k-1)]$$

#### Modified Hebbian learning or Oja learning

The Oja learning rule is [77]

$$w_{ij}(k+1) = w_{ij}(k) + \eta O_j(k) [O_i(k) - O_j(k)w_{ij}(k)]$$

Hertz et al. [99] proved that the weights have the desired properties and pointed out that the weight updates depend on the difference between the input  $O_i$  and the backpropagated output  $O_j w_{ij}$ . Note that Oja's rule maximizes the average squared output.

### **Boltzmann machines learning**

The Boltzmann machine training algorithm uses a kind of stochastic technique known as simulated annealing to avoid being trapped in the local minima of the network energy function. The Boltzmann machines architecture has interconnections among the input units, the hidden units and the output units. The Boltzmann machine algorithm is as follows:

- 1. Initialize weights.
- 2. Calculate the activation as follows:
  - Select an initial temperature.
  - Until thermal equilibrium, repeatedly calculate the probability that j is active by

$$P_j = \frac{1}{1 + e^{-\Delta E_j/T}}$$

where T is the temperature,  $\Delta E_j$  is the total input received by jth unit and the activation level of unit j is set according to this probability.

- Exit when the lowest temperature is reached. The pattern of activations upon equilibrium represents the optimized solution. Otherwise, reduce the temperature by a certain annealing schedule and repeat step 2.
- 3. Update weights by  $\Delta w_{ij} = c(P_{ij}^+ P_{ij}^-)$ , where c is constant.

## Kohonen self-organizing learning

The Kohonen self-organizing learning algorithm is based on unsupervized learning techniques. The network is trained according to the following algorithms:

1. Apply an input vector X.

2. Calculate the distance  $D_j$  (in n dimensional space) between X and the weight vectors  $W_j$  of each unit. In Euclidean space, this is calculated as follows:

$$D_{ij}(k) = \|(x_i(k) - w_{ij}(k))\|$$
(3.22)

- 3. The unit that has the weight vector closest to x is declared the winner. This weight vector, called  $W_w$ , becomes the centre of a group of weight vectors that lie within a distance D from  $W_w$ .
- 4. Train this group of nearby weight vectors according to the formula that follows:

$$w_{ij}(k+1) = w_{ij}(k) + \eta[x(k) - w_{ij}(k)]$$

for all weight vectors within a distance D of  $W_w$ .

5. Perform steps 1 through 4, cycling through each input vector until convergence.

## 3.3.5 Training of neural networks

To train the neural networks, first we must have a collected set of data with known input and output (or result). Partition these data into two groups, one for training and the other for validation. Note that the validation set could be a small subset of the collected data. The validation set is essential in order to test that the neural networks are correctly trained. The neural network training process could then be implemented as follows:

- 1. Set the number of the hidden layer and the number of neurons for the hidden layer, input layer and output layer. Initiate the weights and bias. Decide the learning rate value.
- 2. Train the networks by performing the following steps:
  - Feed the input data to the input layer of the neural network.
  - Compute the neural network output values.
  - Obtain the error  $e_m(k)$  and calculate  $e_o$ .
  - Update the weights and the bias using the updating rule.
  - Check if the stopping criteria are met, else repeat the training step.
- 3. Validate the trained neural network using the validation data.

Note that the stopping criteria could be:

- The error is less than the desired value.
- The number of epoch is reached.

#### Recommendation

It is recommended that the weights and the learning rate be set between 0 and 1. Learning rate that is very small may take a long time to converge, and a learning rate that is too big may result in system stability problem. If backpropagation with momentum is used, the momentum value could be set at about 0.5. For the activation function, use the tanh(x) as it provides more flexibility than the sigmoid function. This is because tanh(x) provides wider values ranges between 1 and -1, whereas the sigmoid function is between 0 and 1.

In general, one hidden layer is sufficient to model or map all non-linear dynamic systems with sufficient hidden neurons. The number of hidden neurons could be approximated based on the order of the dynamic systems, if the order of the dynamic system could be estimated.

## 3.4 Fuzzy Logic

The fuzzy logic technique has emerged as one of the most successful techniques in the growing field of intelligent systems as far as applications is concerned. It was successfully implemented in both civilian and defence industry for real applications.

## 3.4.1 Biological inspiration

In general, human beings are not able to describe certain actions in a precise and/or quantitative manner. All human controls are descriptive such as in the form of high, low or medium. For example, a driving instructor would teach his student on how much pressure he needs to apply to the acceleration and brake pedals by using word such as *press hard*, *press harder* or *press less*.

## 3.4.2 Historical perspective

The concept of fuzzy logic was introduced by L.A. Zadeh in 1965. In the 60s and 70s, fuzzy logic was not well received as a scientific approach of problem solving. However, in the early 1980s, thanks to the number of successful applications and the increase in publicity of fuzzy logic as an alternative way of solving mathematically difficult model problems, the concept of fuzzy logic became an interest to many researchers and applied engineers.

## 3.4.3 Basic elements of fuzzy systems

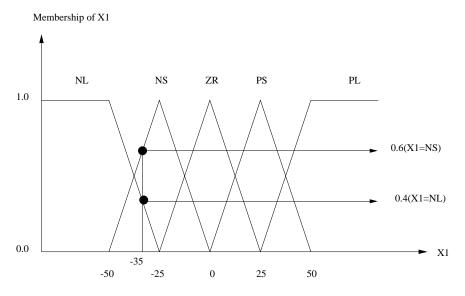
The fuzzy logic system involves three primary processes: (1) Fuzzification, (2) Fuzzy linguistic inference and (3) Defuzzification, as depicted in Figure 3.5.



**Figure 3.5:** Block diagram of a fuzzy system.

#### 3.4.3.1 Fuzzification

Fuzzification is the process of decomposing the system inputs (X1, X2, ...) and outputs (Y1, Y2, ...) into one or more fuzzy sets. The universe of discourse is partitioned into a number of fuzzy subsets and each subset is assigned a linguistic label, for example, Positive large, Positive small. A membership function  $\mu_{X1}, \mu_{X2}$ ... etc. is then determined for each fuzzy subset.



**Figure 3.6:** Membership function for one (X1) of the three inputs.

Figure 3.6 depicts a system of fuzzy sets for input X1 with triangular member-

ship functions. The linguistic variables used, as shown in the above figure, include: NL-Negatively Large, NS-Negatively Small, ZR-Zero, PS-Positively Small and PL-Positively Large.

The degree to which the numerical input values belong to each of the appropriate fuzzy sets is determined via the membership functions. From the example shown in Figure 3.6, input X1 has a membership of 0.6 in NS and 0.4 in NL.

## 3.4.3.2 Fuzzy lingusitic inference

The fuzzy inference engine consists of the rule base, which is formed by assigning fuzzy relationships of the input's fuzzy subsets to the output's fuzzy subsets. The rules are generated such that the constructed inference system is complete, i.e. all combinations of the inputs are formed in the rule base of the fuzzy system. Figure 3.7 shows the rule-based matrix for a three-input fuzzy system, for the case of X1=NL. The four other matrices for X1=NS, ZR, PS and PL are not shown here.

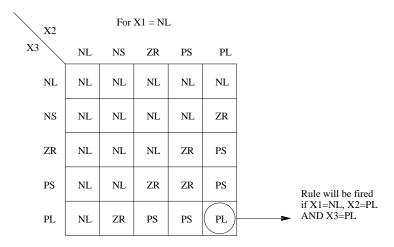


Figure 3.7: Rule-based matrix defining rules of the fuzzy system.

Figure 3.8 shows how the output membership function contributed by each fuzzy rule can be determined using fuzzy approximate reasoning. From the rule-based matrix, one of the rules that would have been fired given such an input combination, is:

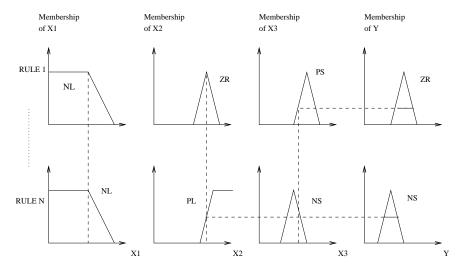


Figure 3.8: Fuzzy inference using fuzzy approximate reasoning.

The individual output (Y1, Y2, ..., Yn) recommended by each rule (1 to n) is then aggregated (using the union operator) as shown in Figure 3.9 to form the combined fuzzy output.

#### 3.4.3.3 Defuzzification

The output of the fuzzy inference process is an aggregated membership function for each output variable. Defuzzification is thus required to convert such aggregates into scalar/crisp values. There are various techniques for defuzzification, namely:

## • Center-of-gravity.

This method determines the center of the area below the combined membership function. The area of overlap (as shown in Figure 3.9) is only considered once. The defuzzified output for the discrete case is calculated by:

$$y* = \frac{\sum_{i=1}^{l} y_i \mu_Y(y_i)}{\sum_{i=1}^{l} \mu_Y(y_i)}$$
(3.23)

• Center-of-sums.

The center-of-sums method is a faster method as compared to the center-of-

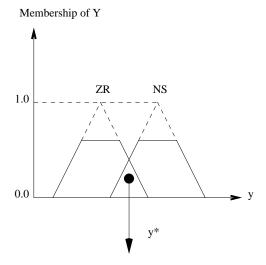


Figure 3.9: Membership function of aggregated output y.

gravity, and the area of overlap is taken twice. The defuzzified output is:

$$y* = \frac{\sum_{i=1}^{l} y_i \sum_{k=1}^{n} \mu_{CLU(k)}(y_i)}{\sum_{i=1}^{l} \sum_{k=1}^{n} \mu_{CLU(k)}(y_i)}$$
(3.24)

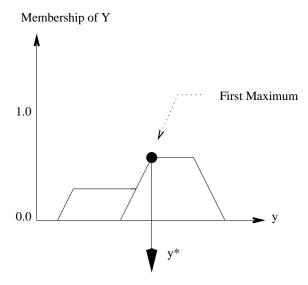
where m is the number of clipped output fuzzy sets from each rule.

• First-of-maxima.

This method takes the smallest value of the fuzzy domain with maximal membership degree in the combined fuzzy output set. It is as depicted in Figure 3.10.

## 3.5 Expert Systems

The view here is that an expert system has a role to play in modelling the human intellect. Although such a system has been viewed as a conventional artificial intelligence approach, it still holds some key functions in modelling our brain. The brain is not just able to learn. But there are some functions that could be hardcoded. For example, the instinctive and reactive behaviour such as the instantaneous withdrawn of our hand when suddenly touching a hot object or the natural behaviour such as the baby's ability to take the first breath, suck milk and cry for attention. These functions or knowledge are normally not taught but are embedded in our gene such that



**Figure 3.10:** Graphical representation of First-maxima defuzzification method.

the brain will activate the process naturally. Hence, an expert system that is based on predefined knowledge may have some analogy to these functions.

Expert systems are software code that contains the knowledge and inference or reasoning capability, which imitate human reasoning in problem solving in a particular domain.

Most expert systems are developed via specialized software tools called shells. Shells are normally equipped with inference or reasoning mechanisms, such as backward chaining or forward chaining (or both), tools for writing hypertext and constructing user friendly interfaces, for manipulating lists, strings and objects and for interfacing with external programs and databases.

Some of the expert systems' inference or reasoning mechanisms are:

- Backward chaining mechanism. This mechanism starts with a statement and
  a set of rules leading to the statement and then works backward, matching
  the rules with information from a database of facts until the statement can be
  either verified or proven wrong.
- Forward chaining mechanism. This mechanism starts with a set rules and a database of facts and works to a conclusion based on facts that match all the premises set forth in the rules.
- Case-based reasoning. This reasoning mechanism relies on stored representations of previously solved problems and their solutions. When a new problem

(or a new case) is presented, the reasoning mechanism will compare with all cases stored in the database. In most situations, there is no exact match to the stored cases. Hence, case-based systems usually need to include an adapting mechanism to match the new cases.

 Rule-based reasoning. This reasoning mechanism is based on a set of 'If-then' rules. These set of rules encode specific knowledge relevant to the domain of operation.

Note that in knowledge-based systems, the reasoning mechanism uses the information collected from specific human knowledge and uses it to reason and solve problems within the knowledge domain. Knowledge-based systems use representation, such as rules and relationships between entities. In this discussion, we consider the knowledge-based system as part of the expert system.

## 3.5.1 Knowledge engineering

For expert systems and knowledge-based systems to work, knowledge needs to be stored and represented in a computer readable form. This involves knowledge engineering. Knowledge engineering is the process of eliciting expertise, organizing it into a computational structure, and building knowledge bases [47]. Hence, knowledge engineering is a technique that is needed to build intelligent systems based on knowledge representation. In the areas of fusion systems with intelligent processing, ontological engineering<sup>2</sup> could potentially emerge as a successor to traditional knowledge engineering.

## 3.6 Fusion of Biologically Inspired Algorithms

In this section, we will introduce another aspect of improving the biologically inspired algorithms by fusing or working coherently together. This feature will influence the strength of each algorithm and offset their weakness. For example, NNs that use gradient-based techniques often have problems in finding global optimum values in the weight updating process. On the other hand, GAs are particularly effective at searching large complex spaces efficiently and are able to achieve near-global optima. Hence, the strength of GAs can be used first to find the right 'hill' or 'valley' (maximum or minimum problems), while those NNs gradient-based techniques can then take over to reach the 'peak' or the 'valley'.

Biologically we observed that neuron networks have some form of evolving process. While it is not clear how this biologically evolutionary learning process takes place, combining artificial NNs and GAs provide some form of evolutionary learning

<sup>&</sup>lt;sup>2</sup>Ontological engineering is a process that facilitates construction of the knowledge base of a system [47].

mechanism. Further, combining of the evolutionary learning mechanism with expert and fuzzy systems may lead to a blueprint of the way the human brain works.

Some of the fusing areas are:

- Neural networks and genetic algorithms. For example, updating of NN weights using genetic algorithms.
- Fuzzy logic and neural networks. For example, using the neural network modelling capability to mimic part of the fuzzy logic process.
- Genetic algorithms and fuzzy logic. For example, using genetic algorithms to alter the membership functions of the fuzzy logic.
- Genetic algorithm, fuzzy logic and neural networks. For example, the fuzzification, inference and defuzzification processes can be parallelized using the
  neural networks structure. Each neuron represents a fuzzy membership function and each link between the layers represents the weight of a fuzzy rule. The
  weights of the fuzzy rules could be optimized by genetic algorithms [120].

## 3.7 Conclusions

Improvement will continue to make biologically inspired algorithms more attractive for use in intelligent systems. Fusing of the biologically inspired algorithms with the traditional algorithms, such as mathematical modelling techniques, statistical and probabilistic techniques, could be considered to improve the intelligence of softcomputing.

More biologically related algorithms may be introduced in the near future as we learn more from the life sciences. One of the increasing research trends is to understand how our brain makes sense of information. Fusion of data and information is one of the important issues in these areas of research. In Chapter 5 onwards, we will discuss this in greater details.

## **CHAPTER 4**

# **Sensing Systems for Intelligent Processing**

Nothing is in the understanding, which is not first in the senses.

John Locke

## 4.1 Introduction

Sensor research and development is one of the important technologies that would affect the design and building of intelligent systems. Without sensor/s, the intelligent system will be handicapped, just like a human loosing its sensory system, for example. The ideal type of sensor should be able to see, hear and feel all targets at an infinite range and altitude, with perfect resolution and accuracy; should penetrate all obstacles such as foliage, smoke, building and terrain. Sensors are physically lightweight, cheap, and can operate 24 hours a day without power constraint and under all weather conditions. Furthermore sensors are stealth, undetectable by target of interest; that is able to sense but not being sensed by what it intended to sense. However, the fact is that such perfect sensor does not exist. These desired characteristics of a sensor are often in conflict. For example:

- Lightweight cheap sensors, in general would have limited processing capability and operating endurance.
- Wide and long range coverage sensors, must often trade off against resolution and accuracy.

New material discovery, processor advancement, and the studying of the life sciences, such as the human and the animal sensory systems, have all helped to improve sensor technology. For example, chemical sensors based on liquid state and solid state materials, have always been used in the estimation of the composition of natural waters. However, they are only reliable under laboratory conditions. Hence, a new type of chemical sensor material based on chalcogenide glass have been developed for this purpose [226]. Sensors produced from such materials are more useful compared to the conventional types due to a higher sensitivity and potential stability, better selectivity and lifetime. Another example will be the study of the animal nose to help improve the design of the odour sensor [203].

Note that in this discussion sensors will also be described as sensor systems; this is because most modern day sensors not only consist of:

- The hardware, but also the software component to perform tasks, such as signal processing, to achieve the detection of an object; and in some cases include identification and tracking software. For example, the radar system does not just consist of the antenna, but also the signal processing unit to achieve trackwhile-scan and beam steering functions.
- One sensor, but could have multiple sensors integrated together for a specific task. For examples, the acoustic sensor system may consist of a number of sensors (or microphones) working together for better acoustic signal detection, and the proximity sensor system may consist of optical sensor and ultrasonic sensors working together.

The chapter consists of the following sections. Section 2 classifies the types of sensor systems. Section 3 summaries and discusses both the human sensory systems and man-made sensor systems. Section 4 discusses the sensor systems design issues.

## 4.2 Classifying Types of Sensor Systems

Sensor systems can be classified according to the following:

- Frequency range. For examples, acoustic sensors at 30kHz or less, radar operating at frequencies ranging from 30Mhz to 10GHz [35].
- Medium, such as air, water, solid, space. For example, air defence radars, underwater sonar sensors, seismic sensors and spaceborne radars.
- Method of measurement. For example, parameters such as pressure, force and temperature can be detected by sensors.
- Emission/non-emission or passive/active sensor systems. Active sensor systems radiate their own energy for target detection. Active sensors include microwave radar, millimeter-wave radar and imaging radars such as the synthetic array radar (SAR and inverse SAR).

Passive sensor systems rely on natural or human-made radiation from the target for detection. Passive sensors cover acoustic sensors, thermal sensors (electro-optical systems, such as infrared search and track or forward-looking infrared sensors) and electronic support measure (ESM).

Methods of intelligence information sources. For example, intelligence information sources collected from radios are known as communication intelligence (COMINT) and information from human is known as human intelligence (HUMINT). The various intelligence <sup>1</sup> sources are classified and shown in table 4.1.

| Classification | Source                | Remark                                   |  |
|----------------|-----------------------|--|--|
| HUMINT         | Human                 | Personal observation and reports.        |  |
| (Human         |                       | Visual observation by pilots of targets. |  |
| Intelligence)  |                       | Human deduction or inferences from       |  |
|                |                       | conversation, etc.                       |  |
| COMINT         | Communication sources | Internal content of messages.            |  |
| (Communication | such as UHF, VHF, HF  | External characteristics of the          |  |
| Intelligence)  | radio                 | transmission such as rate and            |  |
|                |                       | number of transmission and interaction   |  |
|                |                       | between communication nodes              |  |
| ELINT          | Electronic passive    | Parameters such as pulse width,          |  |
| (Electronic    | receivers, such as    | pulse repetitive interval,               |  |
| Intelligence)  | electronic support    | frequency and                            |  |
|                | measure (ESM) system  | scan interval                            |  |
| IMINT          | Imagery sources       | Observation and                          |  |
| (Image         | such as EO (e.g.      | discrimination of objects                |  |
| Intelligence)  | CCD camera and IRST)  | Example: Two tanks observed              |  |
|                | SAR and FOPEN         | at location X                            |  |
| OSINT          | Internet, newspaper   | Key feature of interest                  |  |
| (Open Source   | TV, radio, etc.       | collection of information                |  |
| Intelligence)  |                       |  |  |

Table 4.1: Classification based on intelligence sources

Note: COMINT and ELINT belong to the category of signal intelligence (SIG-INT) sensors.

## 4.3 Sensing Technology in Perspective

Table 4.2 shows the human sensory systems which are by nature passive, i.e. they do not emit signals but receive them. The main sensory systems are: eyes for vision, ears for hearing, tongue for tasting, nose for smelling and the skin for the sense of pressure and temperature. Of all the sensory systems, the brain processes a large percentage of the information coming from the eyes.

<sup>&</sup>lt;sup>1</sup>Note that the word *intelligence* in this paragraph refers to information or news collected by organisations or agencies. Such information is normally classified data.

| Sense       | Sense organs  | Physical interaction | Sensed results           |  |
|-------------|---------------|----------------------|--------------------------|--|
|             |               | sensed               |                          |  |
| Vision      | eyes          | light                | colors                   |  |
|             |               | (electromagnetic     |                          |  |
|             |               | waves)               | high spectral resolution |  |
| Hearing     | ears          | sound waves          | tones                    |  |
|             |               |                      |                          |  |
|             |               |                      | high temporal resolution |  |
| Touch,      |               |                      |                          |  |
| example of  | receptors     | pressure             | pressure, contact        |  |
| temperature | mainly        | (direct contact,     | vibration,               |  |
| and         | in the        | motion, etc.)        | temperature -            |  |
| pressure    | skin          |                      | hot and cold,            |  |
| receptors   |               | thermal flux         | varying spatial          |  |
|             |               |                      | resolution throughout    |  |
|             |               |                      | the body - highest       |  |
|             |               |                      | in the finger tips       |  |
|             |               |                      |                          |  |
| Smell       | receptors in  | chemical             | various scents           |  |
|             | the nasal     | substances           |                          |  |
|             | passages      |                      |                          |  |
| Taste       | taste buds on | chemical             | sweet, sour,             |  |
|             | the tongue    | substances           | bitter and salty         |  |
| Balance     | semicircular  | gravity              | orientation with regards |  |
|             | canals of the |                      | to forces                |  |
|             | inner ear     |                      |                          |  |
| General     | throughout    | chemical             | hunger,                  |  |
| sensations  | body          | balances             | thirst and control       |  |

**Table 4.2:** Human sensory systems

Table 4.3 shows an example of man-made sensor systems. Note that this list is by no means exhaustive. Man-made sensor systems are both passive and active types. The passive sensor systems are like the ears, the imagery active sensor systems are like the eyes and the chemical sensor systems are like the nose to the information processing unit of an intelligent system.

In the following subsection, we will briefly describe the man-made sensor systems that are commonly used in civilian and military environments. These are:

- Radar systems;
- Electro-optic systems;
- Acoustic and sonar systems;
- Magnetic sensor systems;
- Chemical sensors:
- · Microsensors.

| Sense            | Sensor systems   | Physical   | Sensed results   | Applications   | Remarks            |
|------------------|--|--|--|--|--------------------|
|                  |  | interaction  |  |  |                    |
| Vision           | Radar systems:  Imagery radar (SAR, and FOPEN).  MTI radar.  Laser radar.  Doppler radar.  MMW radar.  Electro-optical | electro-<br>magnetic<br>waves                      | image, range,<br>range rate<br>and bearing<br>range,<br>range rate<br>and<br>bearing | air-defence<br>radar, air<br>traffic<br>control,<br>airborne and<br>ground<br>tracking radar | active             |
|                  | sensor systems:  • Quantum detector  • Thermal detector  | light waves  | images   | video camera<br>Infrared scanner   | passive            |
| Hearing          | Acoustic and<br>sonar sensor<br>systems  | sound waves  | range and<br>bearing   | detect air,<br>surface and<br>subsurface<br>objects  | passive/<br>active |
| Touch            | Seismic,<br>temperature and<br>pressure sensor<br>systems  | pressure,<br>temperature<br>and vibration          | range and<br>bearing   | nuclear<br>detonation,<br>object movement<br>on ground                                       | passive            |
| Smell            | Chemical and<br>biological sensor<br>systems   | chemical<br>substances<br>and<br>biological agents |  | detection of<br>gas compounds,<br>elements and<br>viruses                                    | passive            |
| Other<br>sensory | Magnetic sensor<br>systems   | magnetic signal                                    | target location  | altitude control<br>of staellites  | passive            |
| systems          | Shear and stress<br>sensor systems   | fluid  | shear/stress<br>sensitivity  | fluid dynamic<br>monitoring  | passive            |

 Table 4.3: Man-made sensor systems

## 4.3.1 Radar systems

The radar system is one of the most popular sensor used in military and civilian applications. It was introduced in World War Two and since then the technology has grown and expanded rapidly. Radar is used in air (all commercial and military aircrafts are normally installed with a radar), on land (ground radar or radar on ground vehicles for air surveillance or detecting ground targets) and at sea (both surface (e.g. ship) and submerged (e.g. submarine)). Radar can detect both static and moving targets. The probability of detection of a target is often a function of the radar cross section (RCS) of the target and the signal-to-noise ratio due to the background noise. There are many different types of radar systems. The following subsections will briefly describe some of these radar systems.

### 4.3.1.1 Imagery radar

There are two broad categories of imaging radar namely: the real array and synthetic array. Both the real and synthetic arrays obtain the x-direction (or range resolution) by controlling the transmitted pulsewidth. The real array radars obtain the y-direction (or azimuth resolution) from the antenna beamwidth. Since the antenna beamwidth is inversely proportional to antenna length, the y-direction of the real array radars could be improved by using long antennas (narrow beamwidth).

#### Synthetic aperture

The synthetic array radar is also known as the synthetic aperture radar (SAR). The development of the SAR was motivated by the desire to use smaller airborne antennas while increasing the y-direction resolution for ground mapping. SAR use the incremental Doppler shift between adjacent points on the ground, rather than the antenna beamwidth, for the y-direction resolution. The SAR radar provides both fine and coarse resolution imagery in spot and strip modes.

## Foliage penetrating (FOPEN)

The foliage penetrating radar operates on the principle that different radar frequencies passes through foliage with different degrees of easiness. The longer wavelengths in the UHF section of the microwave spectrum penetrate the foliage more easily than the shorter wavelengths, or conversely are reflected less. It is thus possible to create a true ground surface DEM, hence permitting the mapping of the earth's surface which is covered under a forest canopy.

#### 4.3.1.2 Moving target indicator (MTI) radar

MTI radar is used to detect moving targets. Moving target's echo pips (or returns) have a phase shift caused by the motion of the target. This echo pips can be obscured by the other fixed target echoes. Hence, the moving target indicator (MTI) circuitry in a radar system can eliminate the fixed target clutter, and show the moving objects. The returns also enable the computation of the target speed and the target orientation.

## 4.3.1.3 Doppler radar

Doppler radars can be categorized as:

- Continuous wave (CW) Doppler radar.
- Pulse Doppler radar.

The Doppler radar typically operates at less than 15 GHz. The CW Doppler radar transmits a constant frequency signal that detects relative motion between targets, clutter and the radar through the Doppler principle. It is capable of detecting targets that are masked by ground clutter, and are relatively easy to build and require minimal transmitted signal bandwidth. The target range-rate is measured using specialized techniques, e.g frequency modulation of the carrier frequency or multiple pulse repetition frequencies. However, a disadvantage of the Doppler radar lies in the frequency stability of the transmitted signal and the subsequent signal-processing mechanization to maintain the frequency shift achieved between the transmit and receive frequencies.

### 4.3.1.4 Millimeter-wave (MMW) radar

The MMW radar has a wide bandwidth which is necessary for fine range resolution used in target imaging or reduction of clutter. For the detection of targets within short range (10Km), tactical MMW are produced. Such MMW radars are able to detect stationary targets. MMW radars are produced at higher frequencies, from 35 to 230 GHz, due to the low atmospheric attenuation at such frequencies. As such, the angular position resolution of targets can be increased.

#### 4.3.1.5 Laser radar

Laser radars<sup>2</sup> use narrow beamwidths. Laser radars are not suitable for searching capability. However, they are able to detect range fairly accurately and are often used in conjunction with passive infrared search and tracking (IRST) systems and thermal imaging systems. In these applications IRST or thermal imagers locate the

<sup>&</sup>lt;sup>2</sup>Laser radar is also categorized under the electro-optical sensor system.

potential targets in large volume and the laser radar system is pointed in the direction of the target to obtain the range. The laser radar could also be used to confirm the existence of a target.

## 4.3.2 Electro-optic systems

Electro-optic systems typically interact with optical fields. They provide still images and videos. Most of the electro-optic systems are passive devices. These systems normally consist of the fonts (mirrors, lenses, filters), detectors, scanning circuitry, signal processing and display. Electro-optical systems can be classified into two types, namely, the quantum detector and thermal detector [189].

#### 4.3.2.1 Quantum detector

Quantum detectors use semiconducting materials to collect the energy of photons. As the electrical properties of the semiconductor are changed by the absorbed photons, an optical signal is detected. Products, such as digital camera, video camera, which use CCD array as image sensors, belong to this type.

#### 4.3.2.2 Thermal detector

Thermal detectors absorb photons over a relatively broad band of optical wavelengths. This absorbed energy causes a rise in the temperature of the detector material. The optical signal is detected by the change in temperature.

#### Infrared Search-Track sensors - IRST

Infrared (IR) sensors are passive sensors which depend on emitted energy, such as sunlight, to distinguish targets from the background, by the IR radiation they emit. IRST sensors operate by scanning a single or an array of detectors over a surveillence volume for infrared point sources intersected by the targets. Target tracks are then formed as the detections are measured. IR sensors that are mounted on aircraft for detecting ground targets are also know as forward looking infrared (FLIR).

## 4.3.3 Acoustic and sonar sensor systems

Acoustic sensors detect, track and classify targets using acoustic signals which are propagated through water, solids or the air. Examples of such sensors are: sonar sensors for ships and submarines, microphonic sensors for aircraft and vehicles on the ground, and seismometers for nuclear detonation and movement of troops. In particular, the sonar system consists of various stages: detection of the planar wavefront from a point source using an hydrophones array, beam forming to provide an

integrated signal for further processing, null steering for the forming of multiple target-observing beams to track and search for targets, noise cancellation of additive noise, and removal of noise and detection of targets through spectral analysis.

## 4.3.4 Magnetic sensor systems

The application of such sensors is very wide and can be classified into two levels. The first level is the direct determination of the magnetic field itself. The second level is the indirect method. Each one of these applications can be used as a component in a system or equipment, such as a process controller in a submarine or satellite navigation system.

## 4.3.4.1 Direct measurement of the magnetic field

The purpose of direct measurements is to gather information about the direction and magnitude of the magnetic vector itself. It includes:

- 1. Altitude control of satellites, submarines, balloons and aircrafts;
- 2. Geophysical and geological investigations;
- 3. Classical and electronic compasses;
- 4. Prediction of volcanic eruptions and earthquakes;
- 5. Retrieval of data stored on magnetic tapes or disks;
- 6. Acceleration systems used in nuclear physics.

#### 4.3.4.2 Indirect measurement

The indirect measurement is further classified into two areas. The first area is in the measurement of electrical quantities, such as current and power. The second area is the recording of non-electrical and non-magnetic quantities, whereby the information is magnetically encoded and the output sensor signal becomes the carrier of the original quantities, such as linear and angular position, stress, force and weight. The application of indirect measurements includes:

- 1. Contactless keys and DC motors, remote-control values and door locks;
- 2. Transducers for linear detection, such as proximity keys and position sensors;
- 3. Traffic detection, such as determination of submarine location and determination of the spatial positions of ferromagnetic objects.

#### 4.3.5 Chemical sensors

Chemical sensors are for detection of gas, odor, concentration, pH and ions. They are devices that convert a chemical state into an electrical signal. A chemical state is determined by the different concentrations of elements such as atoms or molecules to be detected. Thus, a chemical sensor gives a signal that in some way is related to the chemical state it is exposed to.

The NASA Lewis Chemical Species Gas Sensors Team are developing hydrogen, hydrocarbon and nitrogen oxide sensors for aeronautics and aerospace applications [225]. For example, a hydrogen sensor is used to detect the presence of hydrogen. Presence of hydrogen may potentially indicate that the environments are explosive or dangerous.

#### **4.3.5.1** Gas sensors

A gas sensor is a chemical sensor operated in the gas phase. The response, x, from a gas sensor to a single gas can be described as:

$$x = f_{gas}(C_{gas}) \tag{4.1}$$

where  $f_{gas}$  is a (usually non-linear) function and  $C_{gas}$  is the concentration of the gas. The response is in most cases the difference or ratio between the steady-state value when the sensor is exposed to the test gas and the value from the sensor, such as derivatives or integrals over certain time intervals. Two important parameters when describing the response from a sensor are the sensitivity and selectivity. These parameters are easy to define and interpret for a sensor sensitive mainly to one gas. The sensitivity,  $Y_{gas}$ , for the sensor towards this gas is then defined as:

$$Y_{gas} = \frac{\partial x}{\partial C_{gas}} \tag{4.2}$$

In general, the sensitivity is a non-linear function of concentration. The selectivity, E, of a single sensor is usually defined as the ratio of the sensitivity related to the gas to be monitored and the maximal sensitivity to all other interfering components.

$$E = \frac{Y_{gas}}{\sum_{allothergases}(Y)} \tag{4.3}$$

One main target of chemical sensor research is to enhance the sensitivity and selectivity of the sensor for a specific compound.

A gas sensor usually consists of two parts: the sensing material and the transducer. The sensing material interacts with the analyte, e.g. by absorption/desorption or chemical reactions on the surface and/or in the bulk of the sensor material. The interaction changes some physical properties of the sensing material, for instance the mass or the electrical conductivity, which can be detected. The transducer converts

the chemical information into an electrical signal. A number of possible transducer principles can be used in chemical sensors, e.g. changes in conductivity as detected by voltage drop over a series resistor.

#### 4.3.6 Microsensors

Most chemically related sensors are relatively small. A number of temperature and corrosion detection sensors are also available in miniature size and could be built on top of silicon chips. An example of micro sensors includes a single photodetector. Photodetectors, sometimes also known as photosensors, are simple sensors that detect a single source of energy and produce an output (normally the output is an electrical signal to indicate the presence of an energy source).

Oak Ridge National Laboratory has developed a micro wireless sensor containing an IR LED for signal transmission of up to 1 km. The sensor is sufficiently light and small to be carried on top of a bee [149].

## 4.4 Sensor System Design Issues

While the eye cannot hear, so is the ear that cannot see. Every designer of sensor technology would want to achieve a single all-mighty and intelligent sensor system. But it is difficult, if not impossible to achieve. The physical law of nature always limits our design of sensor capabilities.

Sensor system design issues include:

- Power management. Management of power is particularly critical for standalone sensor systems. The ability to design low power consumption technology and the ability to turn on the sensor only when it is necessary, will help power management. The quest to design a better standalone power source is a continuous research area.
- Physical constraint. The size of the sensor system and the material it needs to be used will be affected by the kind of physical applications.
- Operational environment. Where the sensor systems will be operated will affect their physical design. There are always great differences in design specification for a sensor system built for indoor compared with one for outdoor.
- Specification. This affects the resolution, accuracy and coverage of the sensor systems during design phase.
- Cost constraint. The uses of sensors are frequently affected by cost. Cost will drive the market on how a sensor will be used.

• Integration with other sensors. Integration of sensors will be an increasingly important design issue. This will be further discussed in the next subsection.

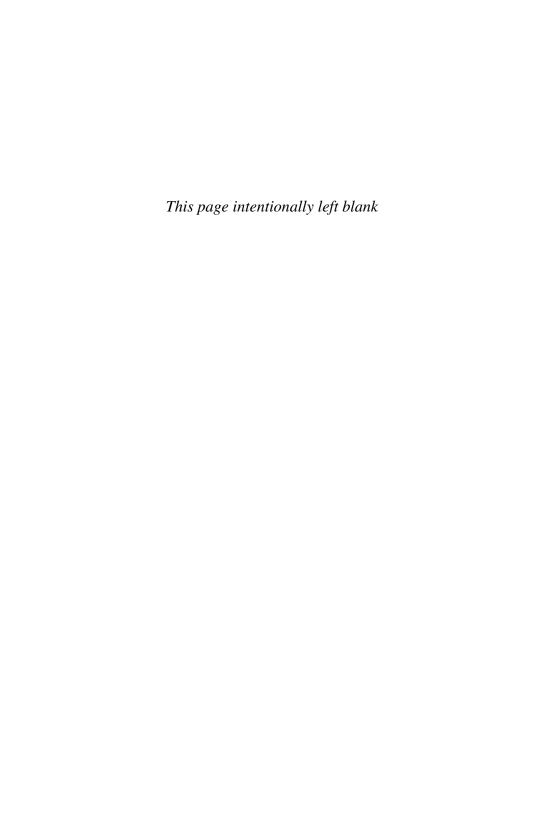
## 4.4.1 Multiple sensor integration

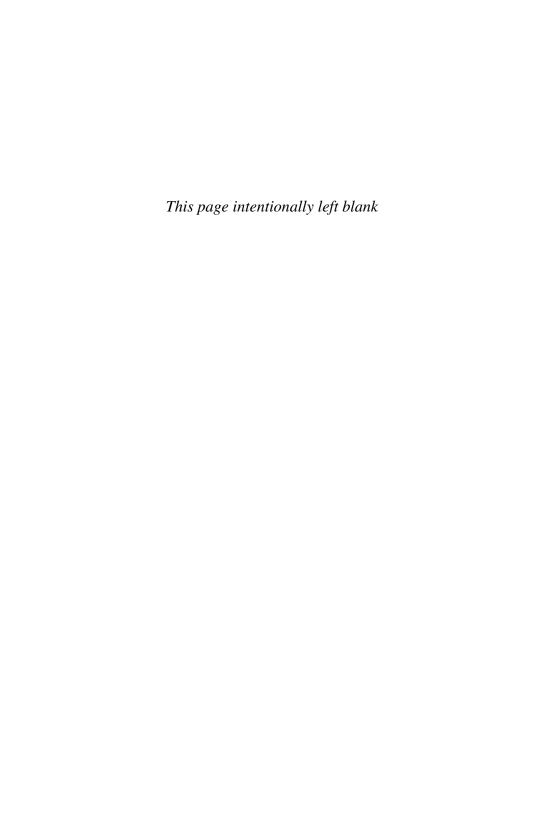
Every sensor system has a specific ability to detect, identify and track specific object/s in a certain environment. The integration of every sensor or sensor systems, by harnessing and controlling their individual strengths and capabilities, could lead to the development of very powerful and/or intelligent sensor systems. Such an integration process would involve multiple sensor data fusion technology. This will be discussed in part 2 of this book.

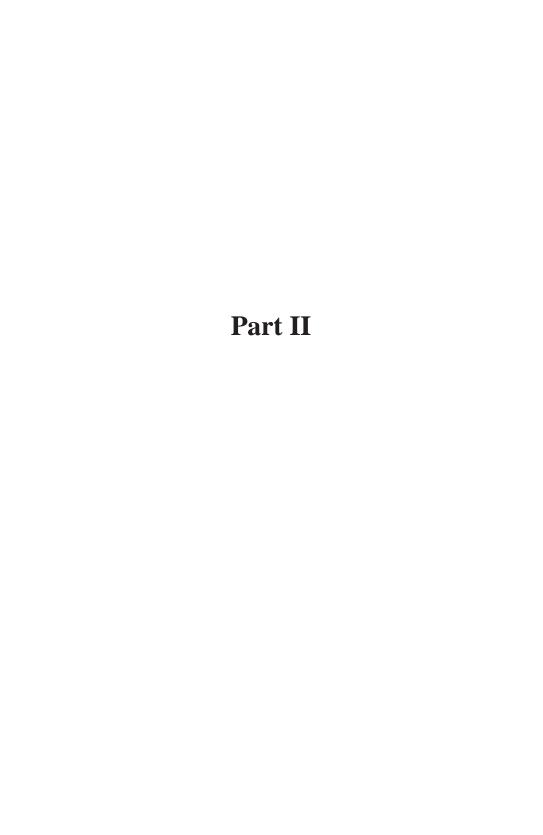
## 4.4.2 Sensing platforms

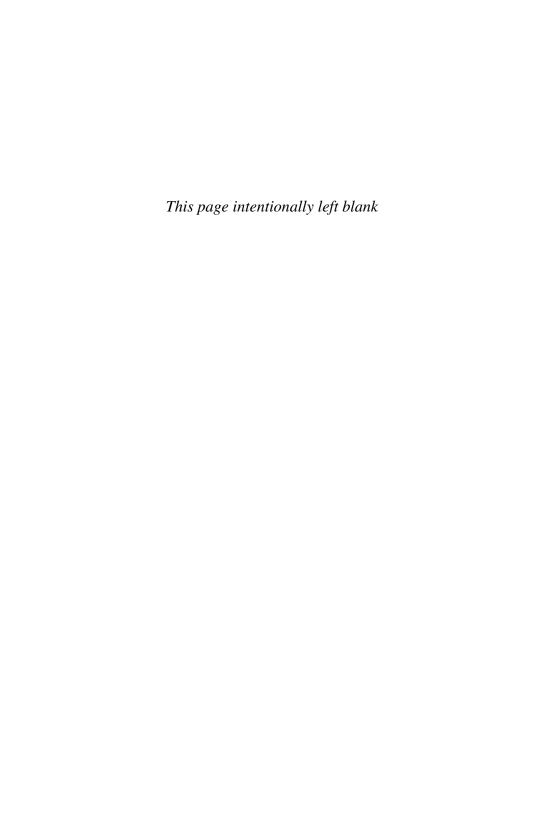
Sensors can be placed on different platforms that could be mobile, static and diversified structures. Here, we will consider sensors located on a single or multiple platforms.

- Single platform with multiple sensing devices. The major reasons to employ multiple sensors on a single platform are that the various sensors provide different, often synergistic, information; search different locations in space; provide redundancy in a target-rich environment and/or under high reliability requirements; and provide increasingly accurate information (e.g., warning, general location, specific location) [43]. A typical example of such system is a fighter plane that carries radar systems and infrared search-and-track sensors. An example of a single platform with multiple sensing devices is the head of an animal, where the various sensors (eyes, nose, ears) are positioned in such a way that a movement of the head will effect one of the sensors, but will also help the other sensors to provide a more comprehensive picture of the surrounding environment.
- Multiple platforms with multiple sensing devices. In most modern military systems, several different platforms are usually involved to aid the data fusion process to produce a more conclusive situational picture. This type of system may also be thought of as a combination of several single platforms that are similar if not the same. Example, the DARPA's Unmanned Ground Vehicle program, see [61], with a unit of autonomous vehicles equipped with cameras. A major problem associated with such multiple platforms, as pointed out in [43], is the transformation of sensor estimates from the coordinate system of one platform to that of another. If the transformation is sufficiently inaccurate, targets common to both sensors cannot be recognized (i.e. associated) and false targets are generated. Hence accurate coordinate transformation is important. This is part of the data fusion process. The data and information fusion will be discussed in Chapter 5.









## CHAPTER 5

# **Fusion Systems for Intelligent Processing**

Data fusion is deceptively simple in concept but enormously complex in implementation.

US Department of Defense, 1990.

## 5.1 Data and Information Fusion System - What and Why

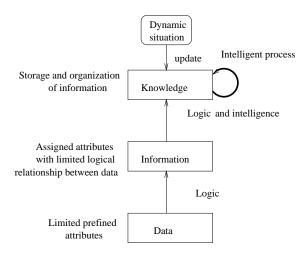
Fusion involves the combination of data and information from more than one source. Data and information fusion is becoming an important field of study due to an increase in data and information flow, and the continuous improvement in communication, computing and sensor technology. This increase is shown in the following areas:

- research and publication of data fusion work.
- military and civilian interest in integrated/networked systems that need data and information fusion, such as network centric warfare and e-commerce.

Fusion systems are applied to the medical field, earth resources monitoring, weather forecasting, vehicular traffic control, robotic navigation, financial analysis, and military command and control (C2) systems. The sources of data could be from sensors or textual<sup>1</sup> reports. Textual data are sometime known as information, whereas the sensor observations are data. Hence, the term sensor fusion or information fusion is used interchangeably, depending on the data sources. In this book, the general term

<sup>&</sup>lt;sup>1</sup>Textual refers to data with assigned attributes that is represented in text form.

data and information fusion are used. Figure 5.1 illustrates the difference between data, information and knowledge. Knowledge is the ability to use information. It involves understanding the information and is derived from the information process. Knowledge is situation dependent and requires intelligent process of information. Information and knowledge are an integral part of cognition <sup>2</sup>. The ability to fuse data and information for meaningful use as a form of knowledge in machine is an essential fusion process.



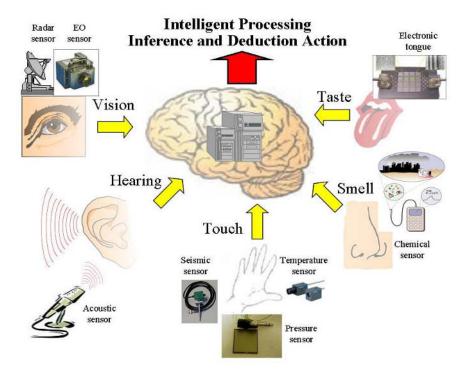
**Figure 5.1:** Data, information and knowledge.

Why data and information fusion systems? Fused data/information are much more enriched in information than data from a single source, else the data/information would not be fused. Hence, fusing data from multiple sources is an intelligent process. This intelligent fusion process is well illustrated in how our brain fuses data from our body's sensory systems such as eye, ear, nose, tongue and skin. Our body's sensory systems provide sense data from our environment via the nervous system to the brain. The brain fuses this information to achieve the awareness of our surrounding environment and attempts to derive knowledge, draw conclusions or inferences from the fused information. Figure 5.2 shows an analog view of an intelligent fusion system as that of a brain using multiple sensory information sources.

Other reasons on why we need data and information fusion systems are:

Robust System. A system that depends on a single source of input is not robust
in the sense that if the single source fails to function properly, the whole system
in operation will be affected. However, the system fusing several data sources
is more robust while in operation or has a higher fault-tolerance rate since

<sup>&</sup>lt;sup>2</sup>Cognition is defined as the process of knowing, perceiving and conceiving. Cognition involves the intellect and the mental process in obtaining and processing of knowledge and information.



**Figure 5.2:** Intelligent fusion process.

other sensors in working order are able to provide similar, if not better data for processing.

- Better situation awareness and inference, hence faster reaction time. By taking advantage of different sources or sensors' accuracy, the combined and fused results would give a better picture of the situation and hence improve inference, which in turn, would lead to better decision making. This is particularly true in military scenarios where increases in the accuracy of weapons and targets mobility require quicker responsive detection and countermeasure reactions. More sensors are able to provide additional information on the target and hence better tracking and faster reaction time for countermeasures.
- Improve data accuracy and reduce data uncertainty and ambiguity. Multiple independent sources of data collectors can help improve data accuracy and reduce uncertainty after intelligently fusing and removing the ambiguity of the data.
- Extended coverage. More data sources will provide extended coverage of information on an observed object or event. Extended coverage can be applied

to both spatial and temporal environments.

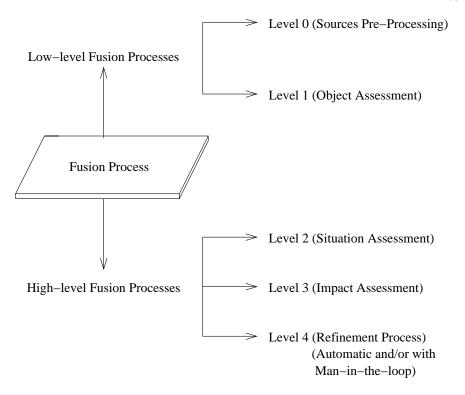
- Cost effectiveness. With the reduced cost of computing, communication and network resources, it is, in general, more cost effective to depend on multiple sources of data input, than relying on a single source to provide all the necessary information. For example, to build a single sensor that can perform multiple functions is much more expensive than to integrate several simple and cheap sensors with specific functions. This is also related to the next point on no single perfect source.
- No single perfect source. Most sources of data collector, such as sensors, are only useful in certain environments. Information gathered by a single source can be very limited and may not be fully reliable and complete. For examples, contact sensors can only detect an object on contact, acoustic sensors can only detect objects that emit acoustic signals, vision sensors are critically dependent on the ambient lighting conditions and electronic surveillance sensor can only detect objects that emit electro-magnetic waves. Hence, it is difficult for one source to meet all the needs.

This chapter is arranged as follows: Section 2 will discuss the multiple levels of data and information fusion process. Section 3 will present the issues in fusing data. Section 4 will discuss the cognitive intelligence for intelligent fusion systems. Section 5 will present the fusion system architecture. Section 6 will briefly discuss the possible techniques and algorithms for fusing data and information. Section 7 will discuss the complementary and competitive data. We conclude with section 8.

## 5.2 What are the Different Levels of Fusion Process?

The Joint Directors of Laboratories (JDL) Data Fusion Subpanel of the US Department of Defense (DOD), defined data fusion as a multilevel, multifaceted process dealing with the association, correlation, estimation, and combination of data and information from multiple sources to achieve refined entity position and identity estimates, and complete and timely assessments of resulting situations and threats and their significance [121]. This section expands on this definition. The multilevel data fusion process could be broadly divided into 5 levels. We categorized these 5 levels into the two stages of processes, namely low-level fusion process and high-level fusion process, as shown in Figure 5.3.

- Low-level fusion process.
  - Level 0 Source pre-processing. This is the lowest level of the data fusion process. The kinds of fusion work include signal processing fusion (fusion at signal level after a signal conditioning process) and imagery fusion at pixel level (also known as pixel level fusion). Hence, most of



**Figure 5.3:** 5 levels fusion process.

these sources involve individual sensor's multiple detection inputs to the signal-processing engine. If the sources are textual, such as from press or Internet, this stage also includes information extraction. This level of fusion also looks at reducing the quantity of data and retaining useful information for the higher-level fusion processes.

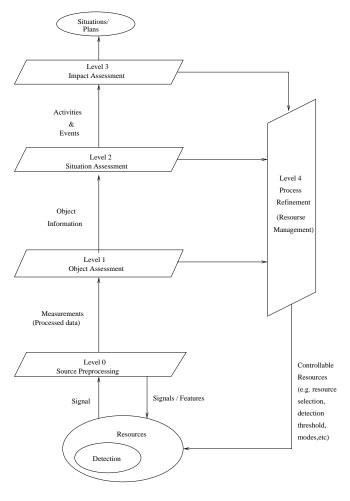
- Level 1 Object assessment. This level of fusion uses the processed data obtained at level 0. The common processing needed at this stage are data and time alignment, association and correlation, grouping or clustering techniques, state estimation and state vector fusion, deghosting of false target (for passive sensors), identity fusion and combining of feature extracted from images (also known as feature-level fusion). The end results of this fusion process are object discrimination (object classification and identification) and object tracking (object state and orientation).
- High-level fusion process. High-level fusion normally starts at the point where
  the objects are known i.e. the type, location and movement (orientation and
  history points), and quantity of these objects. Decision-level fusion, com-

monly discussed in the fusion community, could be said as belonging to the high-level fusion process. In most cases, fusion at levels 2, 3, 4 will ultimately contribute to decision making.

- Level 2 Situation assessment. Level 2 processing seeks for a higher level of inference above level 1 processing. According to the JDL (US, DOD), situation assessment identifies the probable situation using the observed data and events. Basically, it establishes a relationship among objects. Relationships (such as proximity, temporal and communication from content and non-content sources) are assessed to determine the meaning of entities or objects in a specific environment (such as terrain, surrounding media, weather, vegetation and area of interest). The goal at this level includes: deriving higher-order inference and identifying meaningful events and activities (or pattern, in general). An example of higher-order inference for classification of organization structure is the clustering of platoons to infer a company. And examples of activities are coastal surveillance operations, military training exercises, defensive operation and illegal animal hunting. An example of an event is an aircraft crossing the border between two countries. Hence the output of situation assessment is a collection of higher-order inferences in chronological order, which provides a view of what is happening in the world.
- Level 3 Impact assessment. This level is classified as impact assessment. Note that the revised JDL model (1998) has also used impact assessment in place of the threat assessment. As the name implies, this level is to assess the impact of the activities and events derived at level 2 to own perspectives or in military strategic planning perspectives. This includes:
  - \* Assessing the level of threat or danger. For example:
    - · The activity of coastal surveillance operations is this a high threat due to our important assets near the surveillance area, or
    - Assessing the capabilities of the adversary's forces with respect to own forces' strength. What are the adversary's defensive and offensive capabilities?
  - \* Predicting the possible outcome. For example, predicting the possible adversary's intentions and predicting the impact of illegal animal hunting on the region's ecology. If unable to predict the intentions, what are the different possibilities?
  - \* Assessing the vulnerabilities of our assets. What are the risks to own assets and forces?
  - \* Possible own forces' cause of actions. What are the actions to take to ensure own survivability or minimise the risk?
- Level 4 Refinement of fusion process. The last fusion process is the refinement process. The refinement process is to improve the data fusion

from level 0 to level 3. This level is performing resource management, in particular, of the sensor resources (as the sensor is the main focus, this level is also known as sensor management). The common aim is to achieve an effective resource management by performing: task priority, cueing, scheduling, allocation and control of resources. These will form part of the sensor management process that serves as a feedback loop for the complete sensor fusion process.

Figure 5.4 shows the idealized flow of the 5 levels data fusion process.



**Figure 5.4:** Possible logical flow of the 5 levels fusion process.

### **5.2.1** Fusion complexity

The complete JDL data fusion model encompassed the complex integral of multiple sensory systems and the interpretation of the complete chain of data and information for situation and impact awareness. At the low-level fusion process, it is an immediate or short-term demand for situation awareness. At the high-level fusion process, it is toward the projection or long-term demand for situation and impact awareness. The refinement process will then assist to manage the multiple sensory systems (or sources) to meet the short-term and long-term demand.

The complete fusion process is a complex task. The low-level fusion will need to address the complexity of association, short-term history process and structural issues. While high-level fusion needs to handle the intelligent integration of current information with long-term memory information and information recall issues.

In military perspective, how close are data and information fusion processes in fulfilling the principle found in Sun Tzu's Art of War [235], is for one to find out. A partial quote from Sun Tzu's works regarding this principle is as follows:

He who has a thorough knowledge of himself and the enemy is bound to win in all battles. He who knows himself but not the enemy has only an even chance of winning. He who knows not himself and the enemy is bound to perish in all battles.

Know your enemy, know yourself, and your victory will not be threatened. Know the terrain, know the weather, and your victory will be complete. [235]

# **5.3** Issues on Fusing Multiple Sources

Data and information fusion are a multi-disciplinary subject. Data could come from diverse sources where information of the data could be ambiguous, noisy and the quantity could be substantial. Hence, it is not a trivial matter to solve data and information fusion problems.

Listed below are some of the common issues one will face in fusing multiple sources of data:

- Data dimensionality and alignment. Different sensors have different measurement data and hence different dimensionalities and features. Data alignment and transformation would be needed to achieve common format and standards. Note that the registration process is considered as part of data alignment. For example, different sensor systems may have different:
  - Coordinate systems such as Cartesian, spherical and polar coordinates, latitude-longitude-altitude (or LGZ), North-East-Down (or North-East-Up) and Military Grid Reference System (MGRS) representation.
  - Measurement units such as mile, meter, feet and inch.

- Size of data (object state observable) such as different dimensional space.
- Features such as frequency, amplitude, imagery and non imagery.
- Data resolution and accuracy.
- Common reference. Sensors in same or different platforms need to be aligned into a common spatial reference.

Hence techniques are required to perform data transformation and normalization to a common format and standard that are appropriate for data fusion processes.

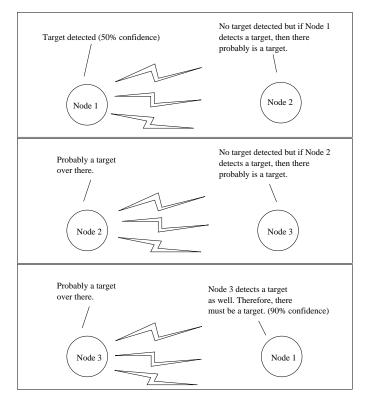
Data alignment is required to transform observed data from many different sources to a common format. Data alignment is part of the registration process.

- Data incest and reliability. Distributed and decentralized data fusion need to handle the problem of data incest and reliability issues. Figure 5.5 illustrates the problem of data incest. Ideally each fused data should contain information on whom is contributing it (this is related to indicating the data dependency and independency factors) and its source's reliability. However, tagging such information across many nodes is a complicated task and may not be easy to implement in practice.
- Time alignments. The synchronization of the data time is important. This is because only data that are 'close' in time can be integrated, if meaningful results are desired. This is particularly important for competitive data. The closeness of time or the window period where the data can be fused depends on the types of fusion system. Time-critical fusion systems such as fusion system on-board air platform, the time window is typically in micro or mini second; whereas fusion systems for non-real time analysis that depend on report sources that may take minutes or hours to arrive, the time window of closeness could be in the order of minutes or hours.

However, for complementary data, the closeness of time may be relaxed. This is because complementary data may still provide useful information to the fusion system even if the data sources are not sufficiently close in time. For more discussion on competitive and complementary data, please see section 5.6.

Time alignments are complicated by:

- Sensors located at geographically different areas and at different platforms.
- Time difference of arrival due to different signal propagation speeds e.g. acoustic signal vs electromagnetic signal vs seismic signal.
- Data source update rate.
- Sampling rate. Sampling rate of the sensor measurement data for most integration algorithms need to be the same.



**Figure 5.5:** Example of data incest. Node 1 communicates its detection to node 2. Node 2 passes the message to node 3. Node 3 broadcasts the message and node 1 receives. Node 1 assumes that node 3 has the same detection and hence increases its confidence of the detected target.

Time stamping and a common clock for synchronization will have to be established. Techniques are required to resolve the time delay due to signal propagation and sensor detection. Alignment to ensure common time frame may be needed.

- Sensor operation platforms.
  - Single or multiple platforms. Sensors in a single or multiple platform have their own unique time synchronization problems.
  - Static or moving platforms. Sensors are in moving or stationary platforms. It is certainly not a trivial problem when the sensors are in different moving platforms e.g. it may not be feasible to achieve any meaningful results having acoustic sensor on a continuous fast moving platform.

Speed of the moving platform is a factor that needs to be considered.

- Quantities of sensors used. The increase in the number of sensors available may result in an exponential increase of:
  - Fusion engine design complexity.
  - Data storage structural complexity.
  - Communication costs [109].

It is common belief that with more sensors, hence more data, is never theoretically bad for decision making. However, as pointed out by Treece [221], more data or information is damaging to the final answer under certain conditions. Some of these conditions are related to data reliability and the environment the sources are operating on. More studies and investigations are needed to identify the conditions that could be damaging for adding more data.

- Sensor selection strategies. Luo and Kay [146] pointed out that the sensor selection is one of the integration functions that can enable a multi-sensor system to select the most appropriate configuration of sensors from among the sensors available. Therefore, proper sensor selection strategies need to be studied. Factors affecting the sensor selection include:
  - Target dynamics.
  - Target densities (e.g. closely-spaced targets).
  - Background noise sources (e.g. high clutter background due to multipath reflection).
  - Sensor performance (e.g. probability of detection).
- Concepts of operation. The high-level fusion process would need to take
  into consideration the concepts of operation, particularly in a military context. With increasing complexity of the battlefield, meaningful integration and
  fusion can only be established if the operational doctrine is well understood
  and the goals are clear.
- Organizational structure and workflow. In order to provide meaningful fusion results, some understanding of the organizational structure and workflow would be useful. Different groups of people in the organization look for different information, hence fusion system would need to fuse the correct set of data and information that the people are looking for.
- Physical and operational constraints. The choice of fusion architecture and techniques are sometime affected by the physical and operational constraints.

The solutions to the above issues are non-trivial. The degrees of difficulty also depend on the level of the fusion process and fusion architecture used. To derive

higher intelligence inference, such as the human brain intelligence, the cognitive intelligence process will need to be studied. The next section will cover a brief description of the cognitive intelligence. And since the fusion architecture will also affect the fusion system design (system engineering design perspective) and the fusion algorithms (in term of efficiency and effectiveness), the section after cognitive intelligence will discuss the fusion architecture.

# 5.4 Cognitive Intelligence

The fusion issue at high level involves understanding of information to perceive and conceive for situation and impact assessment and ultimately awareness. Today, there is no known machine that could do that as well as the human brain can. Hence to engineer this fusion process, the understanding of how our brain processes information and knowledge will be important - specifically the aspect of intelligence pertaining to cognitive processes, i.e. perception, learning, information retrieval from memory and inference, more commonly known as cognitive intelligence. The study of cognitive intelligence includes: understanding the complex human brain in coping with life situations, such as its ability to adapt to new conditions, learn new things, recall information and apply knowledge to solve problems.

#### **5.4.1** How to make sense of information?

What is the cognitive process that enables one to decide which pieces of information are more relevant than others? And how to obtain meaning from data that is otherwise difficult to draw any useful conclusions from? How to recall information at the right time for a useful process? How to understand what others think?

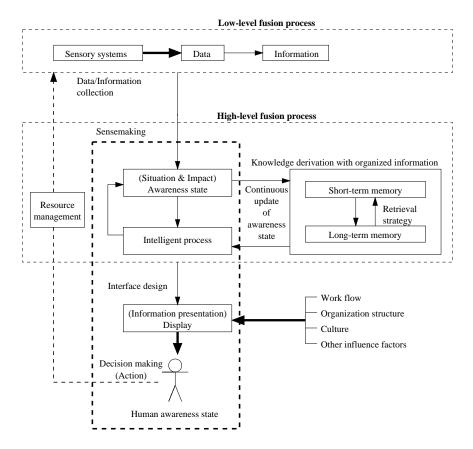
The attempt to engineer the cognitive process has led to cognitive psychology <sup>3</sup>, knowledge engineering, ontological engineering, memory engineering, computational intelligence (e.g. biologically-inspired algorithms), human factor study and many more. More recently, the term cognitive informatics was coined for the study of natural intelligence. Cognitive informatics study the internal information processing mechanisms and natural intelligence of the brain [229] and has originated from information theory and contemporary informatics <sup>4</sup>. Cognitive informatics look into how human beings acquire, process, interpret, and express information by using the brain and how we understand the mind of different individuals. This new interest has led to the First IEEE International Conference on Cognitive Infomatics (ICCI'02) held in 2002.

<sup>&</sup>lt;sup>3</sup>Cognitive psychology includes studies of perception, memory concept formation and problem solving.

<sup>&</sup>lt;sup>4</sup>Informatics refers to the science of information. It involves the studies of the nature of information, it's processes, and ways of transformation between information, matter and energy.

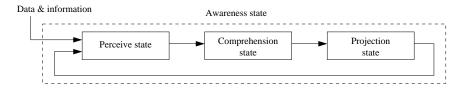
The other aspect of the cognitive process is in attempting to make sense of information or also known as sensemaking. In the general sense, the word 'sensemaking' refers to the derivation of an understanding of a problem given a large set of data that is complex and sometimes even self-conflicting. Sensemaking is a complicated process and is described as a cognitive task model [53, 184]. As such, it is subjective and is dependent on many factors, such as the point-of-view of the people performing it and diversity of the group. Sensemaking is used in situations where information is often vague and ambiguous. For more information on sensemaking, reader may refer to the publication by Leedom [136, 137].

The transformation from information to knowledge is situation dependent [54] and involves smart algorithms. Figure 5.6 shows a possible engineering flow of an attempt to model cognition and high-level fusion to achieve situation and impact awareness.



**Figure 5.6:** Possible engineering flow of cognitive and high-level fusion process.

The awareness state is continuously updated within the closed-loop process of memory and the intelligent algorithms. The memory engineering process will need to handle the storage and organization of useful information and with the ability to recall or retrieve information to enhance the awareness state. Memory engineering process and intelligent process work hand-in-hand to derive knowledge and update the awareness state. The awareness state will continue to evolve from a perceive state to a comprehension state and finally to a projection state (see Figure 5.7). The awareness state will be updated with more data and information input.



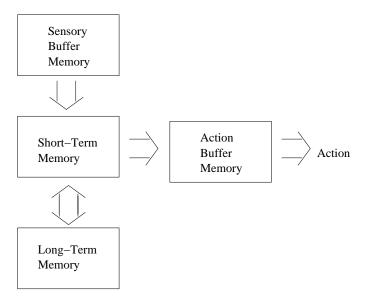
**Figure 5.7:** Possible evolving process of the awareness state.

The intelligent process will involve decision support tools such as the biologically inspired algorithms (or the descriptive techniques), information-theoretic approach, statistical and probabilistic techniques, mathematical modelling and cognitive informatics approaches. The iteration between the intelligent process, awareness state, memory engineering process and the resource management form the high-level fusion process.

Sensemaking will encompass the awareness state, the interface design and understanding of the way human make decision. The interface design include how information can be best presented so that a human can interpret it correctly and comfortably. Hence, the interface design will consider human factors such as stress, workload, expectation, cognition, expertise, personality and culture.

### 5.4.2 Memory structure for cognitive intelligence processing

Memory is an important component in the study of cognitive intelligence. Memory is the foundation for maintaining a stable state of an animate system. It is also the foundation of any natural intelligence [230]. The cognitive memory map plays a role in understanding how the brain processes information. The human brain typically can be mapped into the following 4 memory types namely; the sensory memory, short-term memory, long-term memory and action buffer memory [230]. Figure 5.8 shows the memory structure. Cognitive engineering views the learning as a process in which information is obtained, stored in the long-term memory and able to successfully recall it when needed [94].



**Figure 5.8:** The memory structure.

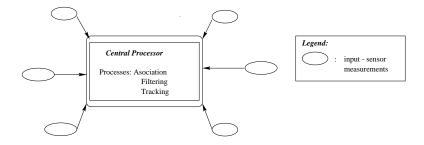
# 5.5 Fusion System Architecture

There are several ways of classifying data and information fusion system architecture. From literature survey and studies, we could classify them into the following 4 types of fusion architecture, namely:

- Centralized fusion architecture.
- Decentralized fusion architecture.
- Distributed fusion architecture.
- Hierarchical and hybrid fusion architecture.

#### **5.5.1** Centralized fusion architecture

In a centralized fusion architecture, the fusion unit is located at a central processor that collects all information from the different sources (Figure 5.9). All decisions are made at the central processor.



**Figure 5.9:** Centralized architecture with a central processor.

#### 5.5.2 Decentralized fusion architecture

A decentralized fusion architecture consists of a network of nodes, where each node has its own processing facility. There is no need for a central fusion or central communication facility, as a common place does not exist where fusion or global decisions are made as fusion occurs at each node, on the basis of local information and information from neighbouring nodes [74].

Three characteristics of a decentralized fusion architecture are:

- 1. No single central fusion centre.
- 2. No common communication facility.
- 3. Nodes have no global knowledge of the network topology.

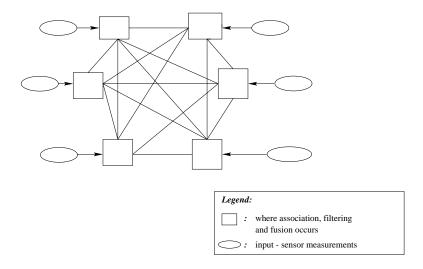
Decentralized fusion architecture could be further categorized as fully connected decentralized fusion architecture or partially connected decentralized fusion architecture. Figure 5.10 shows a fully connected decentralized fusion architecture.

#### 5.5.3 Distributed fusion architecture

Distributed fusion architecture is shown in Figure 5.11. Distributed fusion architecture is where each sensor's measurements are processed independently (e.g. perform data association and state estimation) before sending the object (e.g tracks) to a central processor for further fusion with other distributed sources input. Distributed fusion architecture could be viewed as an extension of the centralized fusion architecture.

### 5.5.4 Hierarchical fusion and hybrid fusion architecture

The other possible architectures in the literature are a combination of centralized, distributed and decentralized architectures. Here, we classify two types of mixed



**Figure 5.10:** Decentralized fusion architecture.

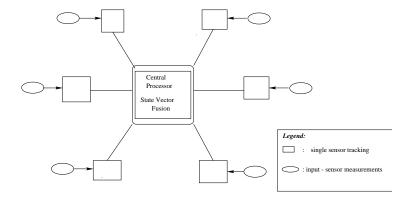
architectures, namely hierarchical fusion architecture (see Figure 5.12) and hybrid fusion architecture (see Figure 5.13).

# 5.5.5 Is there a difference between decentralized and distributed fusion?

Although the terms decentralized and distributed fusions are used interchangeably in most of the literature, there is a slight difference between the two terms.

In a decentralized case, the various decision makers (or estimators or controllers) do not all share the same information at a given time and there are no central facilities. In a distributed case, there still exists the notion of central processing. Distributed networks reduce computation and communication by dispersing the work among a number of hierarchical processors which 'report' to a central processor [23].

The difference between decentralized and distributed fusion architecture is shown in Figures 5.10 and 5.11 respectively. However, one common similarity is that both the distributed and decentralized fusion are able to work in a multiple sensors environment.



**Figure 5.11:** Distributed fusion architecture.

#### 5.5.6 Is there a single best architecture?

The answer to the above question is an obvious *NO*. This is because architecture selection is a function of need, demand, existing connection, data exploitation level, data availability, strategic planning scheme, and the organisational structure using the fusion system.

However, centralized architecture is, in general, better discussed, implemented and used than decentralized architecture. Most of the existing fusion systems, in one way or another adopt centralized architecture. Decentralized architecture is difficult to implement partly due to communication and computing constraints, cost, spatial and temporal issues. Robust decentralized algorithms that could solve these issues are currently being investigated. For example, using an information filter (IF), rather than decentralized Kalman filter (KF), in a decentralized architecture's state estimation process. This is because the information filter is computationally less complex and claims to occupy less bandwidth than the Kalman filter. This will help reduce the computational load and communication costs.

As systems become more interconnected, mixed architectures will prove increasingly more popular. For example, building a fusion system where the current sensor systems have already performed fusion at level 1 process.

Most of the stand-alone platforms (e.g. car, aircraft, ship, tank and single lift systems) will remain working in a centralized architecture. We refer to a stand-alone platform as a system with only one central processor. Transforming multiple stand-alone platforms to decentralized fusion architecture will depend very much on how the constraints (as stated earlier) could be resolved.

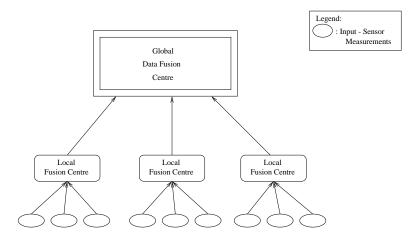


Figure 5.12: Hierarchical architecture.

# 5.6 Techniques for Data and Information Fusion

Data and Information fusion techniques are drawn from many disciplines. Figure 5.14 shows that the building of data and information fusion systems often involves multi-discipline subjects and techniques. These multiple techniques are often applied jointly or sequentially [92].

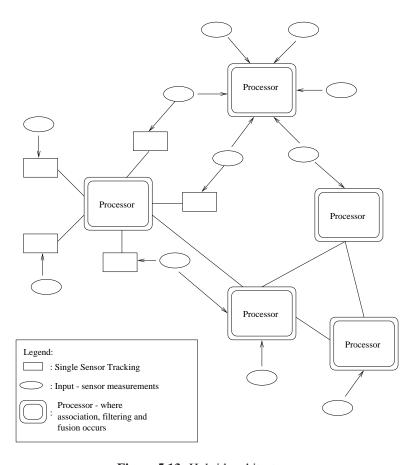
### 5.6.1 Level 1 - Object assessment

At the level 1 fusion process (object assessment), the analysis is more at individual object. Each object is studied and characterized with regard to its class, position and orientation. The techniques required include data association, state estimation and classification.

#### 5.6.1.1 Data association and correlation

Data association is the problem of attempting to determine the set of data that referred to the same entity (where entity could be the desired object or event). Is there a difference between association and correlation? Hall [92] defined data association and correlation as follows:

Association - The process of assigning and computing weight that relate observations or tracks from one set of data to observations or tracks to another set of data. For example, frame to frame association, observation to observation association, observation to track association, and track to track association.

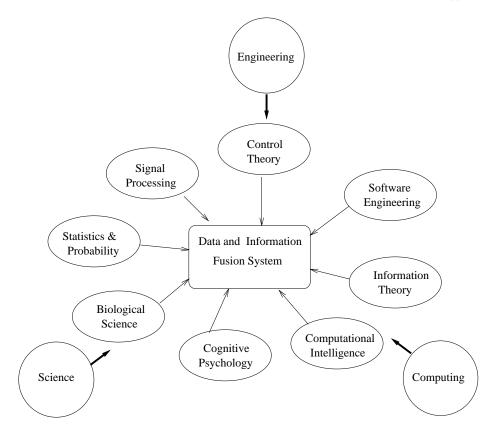


**Figure 5.13:** Hybrid architecture.

• Correlation - The structural, functional or qualitative correspondence between comparable entities; a decision making process which employs an association metric as a basis for allocating or assigning sensor measurement and/or report to the hypothesis entities of interest.

Another definition of the term correlation is: *Correlation is a bivariate measure of association (strength) of the relationship between two variables.* 

In simple terms, association involves an action of finding the joint relationship among a set of data. The findings of this relationship could be used for clustering purposes or identifying data or information that are referring to the same object or subject matter. On the other hand, correlation involves understanding the shared relationship, or connection of cause and effect between two data or two set of data. Hence, correlation is naturally part of the association process.

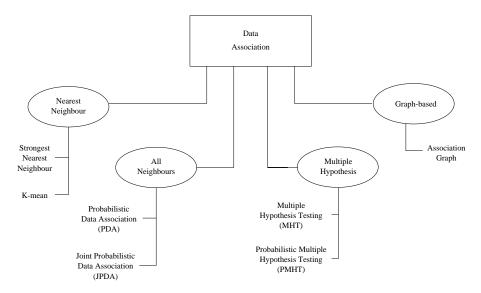


**Figure 5.14:** Data and information fusion system - multi-disciplinary subject.

Figure 5.15 shows the possible algorithms for performing data association. In Chapter 6, we will further discuss the issues and some of the techniques used for data association.

#### **5.6.1.2** State estimation

State estimation is the problem of attempting to determine the value of an object's state, given the observations or measurements of the object (commonly the term system is used rather than object for state estimation). Figure 5.16 shows the various filters that could be used for state estimation. It is noted that for state estimation, the IMM (Interactive multiple models) algorithm is well accepted as the best method. In Chapter 7, we will further discuss the tracking algorithms.



**Figure 5.15:** Data association techniques.

#### 5.6.1.3 Classification, identification and identity

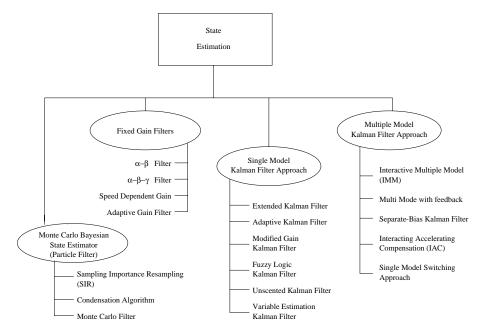
Classification is the problem of attempting to determine the class of objects or general platform types. For example, car, ship, tank, and aircraft are general platform classification. Identification is the next level of classification or the platform specific types. For example, under the class of car, we could have Toyota car, Honda car and Mazda car. The identification of specific platform types may also include identity. The identity of object or target, in military context, refers to friend, hostile, neutral or unknown. The techniques that have been considered and could potentially be used are shown in Figure 5.17. Note that Figure 5.17 is partially adopted from [122]. The probabilistic classifier, namely the Bayesian and Dempster-Shafer classifiers are described in the appendix.

### 5.6.2 Level 2 and 3 - Situation and impact assessment

At the level 2 fusion process (situation assessment), the techniques commonly involve building the relationship between objects and clustering of objects <sup>5</sup>. The objective is to derive higher-level inference, such as activities and events.

Level 3 involves an impact assessment, which includes determining the threat level, estimating the opposing force capabilities, predicting the intent at certain specific times (possible course of action) and estimate the impact on its own forces.

<sup>&</sup>lt;sup>5</sup>This includes activities/events aggregation



**Figure 5.16:** Filters that are used for state estimation methods.

#### 5.6.2.1 Clustering and aggregation

Clustering and aggregation algorithms are needed to group or aggregate related objects and activities together to obtain part of higher-order inference process. For example, in a military organization a group of infantry platoons will form a company and a combination of infantry, logistics and support companies will form a battalion. Figure 5.18 shows a framework for force aggregation commonly applied to military purposes. Clustering and aggregation algorithms normally depend on factors such as:

- time relationship;
- geometrical proximity;
- equipment types;
- quantities (or in military the unit size);
- communication means:
- environment (terrain, weather and foliage);
- doctrine and governing roles;

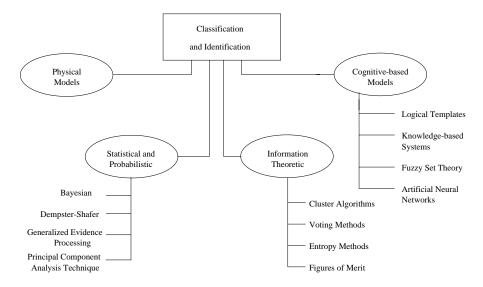


Figure 5.17: Classification and identification techniques.

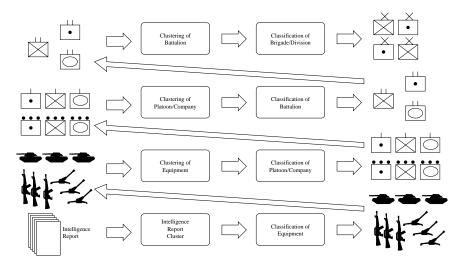
- social and cultural behaviour and
- other functional dependence factors.

# 5.6.2.2 Learning, reasoning and knowledge embedding and retrieval strategies

Various learning, reasoning and knowledge embedding and retrieval strategies will be needed for level 2 and 3 assessments. These include:

- Probabilistic networks. Probabilistic networks, such as Bayesian networks are seen as having a high potential in capturing the complex relationship between various objects, activities and events. This is essential for pattern recognition and trend recognition<sup>6</sup>. Other statistical decision, evidence combination and hypothesis generation methods may also be used to map the network of features and functional dependency between objects or activities.
- Decision trees and graph-based inference. Decision trees and graph-based inference approaches are used to map complex and dynamic environments. Decision trees and graph-based inference have similar characteristics as a probabilistic network. They extract statistically-based information or decision rules unsupervized from a given set of data. An example of decision tree approach is

<sup>&</sup>lt;sup>6</sup>Trend recognition will be essential for projection and prediction of possible outcomes.



**Figure 5.18:** An example of force aggregation.

the fuzzy decision tree and an example of graph-based approach is goal lattice techniques.

- Computational intelligence. Computational intelligence techniques, such as fuzzy logic, neural networks and genetic algorithm, will be combined to derive higher-order inference for intelligent systems to assess the situation and impact. These include enhancing existing knowledge-based and case-based systems with real-time updating and adaptive correction of the system knowledge.
- Cognitive intelligence. The understanding of the human brain and how it
  works may lead to better engineering of the natural intelligence-thinking machine. Classical information theory will remain useful in engineering cognitive
  intelligence for fusion processes. Others include: the study of the natural flow
  process of how a brain will react such as:
  - 1. When there is lack of information, the natural response is to seek more information.
  - 2. When this is impossible (as for example, when the staff tries to understand the enemy commander's intent), the most common tactic is to reason based on assumptions.
  - 3. When neither of these work, the most common response is to find ways of coping with possible adverse outcomes.

(Naturalistic decision making process - courtesy of Professor Martin G. Helander, Nanyang Technological University)

The quest to understand this natural flow of decision making is known as naturalistic decision making.

The techniques used for the level 4 refinement process will be covered in Chapter 9.

...unless I can read your mind I won't know what's going to happen on Wall Street except probabilistically.

Sloan's Jesse Ausubel (The Wall Street Journal Europe, May 2003)

### 5.6.3 Remark on blending and fusing

Fusing is a stronger term than blending when used in combining data. Blending generally means mixing two or more sources together. An example: blend flour and sugar. In data and information fusion, blending could simply mean combining different data sources by weighting these sources of data. However, fusing could mean more than just blending. Fusion involves tighter joining of two or more sources. Examples include melting of two or more metals to form a new metal. However, in this book, we do not seek to differentiate the use of blending and fusing of data. We see both words as contributing towards the combining of data.

# 5.7 Complementary and Competitive Data

Data from multiple sources observing the same object or event generally can contribute to either complementary or competitive information. Data sources are said to provide complementary or competitive information when the observed data fulfil the following criteria:

- Independent data source. The data are collected by independent sources.
- Same object or common event/space. The data collected by the sources are observing the same object or common event.

To qualify that the data collected by the sources are observing the same object or common event, data association is required. In that perspective, sensor systems are either strategically placed to spot certain common areas (minimum or no association process needed) or there are some common features that the sensor systems could use to associate and confirm that they are looking at the same object or event. For example two sensor systems may use the closeness of the geolocation to determine the object being associated. More discussion of the data association will be presented in Chapter 6.

### 5.7.1 Complementary data

Complementary data will lead to a wider awareness of the situation and a more comprehensive picture. Complementary data supplement one another to make the part more than the whole. What one sensor lacks will be provided by another sensor - when considering fusion in complementary data. Hence, complementary data will lead to an increase in information space.

Examples of complementary data are:

- Two video cameras positioned at different angles. The two cameras provide
  complementary information as each camera sees a different perspective of the
  same object due to the angle each camera is viewing. With the two set of
  information put together, it may be possible to construct the 3-D picture of the
  object.
- Laser radar and the forward looking infrared (FLIR). The laser radar can provide the range information and the infrared image can show special features of the same object.
- Low-resolution vision camera and ultrasonic range finder. The low-resolution vision camera locates the direction, such as the direction of an obstacle. The ultrasonic range finder will determine the obstacle's depth and shape.

### 5.7.2 Competitive data

Competitive data will help to enhance the reliability, accuracy and fault tolerance of systems by presenting independent measurements of the same object feature or event. For example, two acoustic sensor systems measuring independently the same object emitting an acoustic frequency. Using these two sources of competitive data, fusion techniques will attempt to reduce the overall uncertainty and increase the accuracy of the data.

# 5.8 Summary

This chapter has presented the motivation for data and information fusion. It high-lighted the various issues and techniques for fusing data and information. Data and information fusion with regards to cognitive intelligence and architectures were discussed. Given the above explanations and discussions, it is hoped that fusion systems will increasingly play an important role in both civilian and military systems.

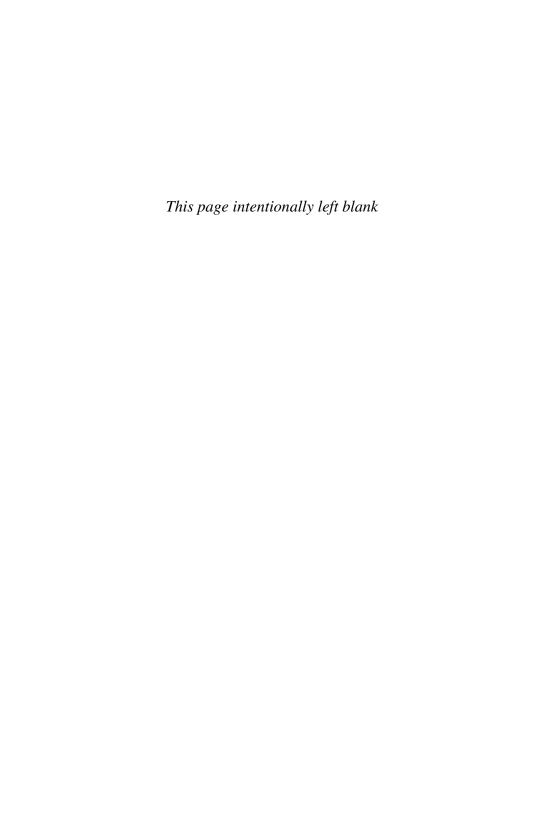
There are many possible research areas in building intelligent data and information fusion systems. Listed below are some possible areas:

- Theory. Proving of particular fusion techniques will lead to optimal results, such as proving that the various state vector fusion techniques or covariance intersection methods are optimal in performance.
- Systems. Systems can be divided into two categories, i.e. those with multiple subsystems or single system with one central processor.
  - A system with multiple subsystems where each subsystem has its own processor. These include large-scale and small-scale systems. Examples of large-scale systems are space stations (e.g. the Russian Mir station), military command and control systems, space shuttle and urban traffic control system. A small-scale system may include medium to miniature modular robotics. Various interesting research and studies are ongoing on how data fusion could enhance such systems where multiple sensors are used in different subsystems. Studies include a network of sensors (for example a network of unattended sensors where the sensors are normally cheap, small and easily transportable), a network of platforms (both static and mobile platforms, trying to fulfill network centric warfare), cooperative capabilities among platforms or subsystems and agent-based negotiation among platforms.
  - A system with a central processor. This includes the so-called platformcentric types. Research in this area includes optimal resource allocation of the multiple sensors, improved tracking and classification, particularly for mission critical platforms such as fighter aircraft.
- Algorithm. There are many areas of research for improving algorithms. Some of which are:
  - State estimation, such as IMM algorithm with feedback, non-linear state estimation techniques and information filter for decentralized state estimation.
  - Association, such as improving probabilistic multiple hypotheses tracking (PMHT) algorithms and combining of various association algorithms with probabilistic techniques such as MHT with Bayesian networks.
  - Deghosting continuous research has been on going to solve some of this NP-complete problem of fusing multiple input data from passive sensors.
  - Low-level fusion, such as image fusion, better hybrid approach for object discrimination.
  - Fusion of different sensors using computational intelligence methods, some of which are attempting to achieve adaptive data and information fusion (this include doing away with some classical fix threshold setting).
- High-level fusion processes. Research into better learning, reasoning and knowledge embedding and retrieval strategies. These include:

- Research into better situation and impact assessment. Improved smart
  or intelligent techniques of inferring activities and events. Research into
  the cognitive informatics area for better pattern recognition, trend recognition, military plan recognition and activities/events aggregation.
- Research into better resource allocation approaches such as using a combination of computational intelligence methods with traditional resource optimization approaches.
- Research into ontological engineering to represent knowledge and capturing the sensor systems capabilities. This will enable fusion systems interoperability and also facilitate working with different sensor systems.
   Readers may refer to the appendix for an introduction to ontology.

As you fuse, remember the source. The fusion of a human egg and sperm does produce intelligent beings.

G.W. Ng



### CHAPTER 6

# **How to Associate Data?**

It is better not to say anything than to say the wrong thing.

Common words of wisdom

# 6.1 Why Data Association

Data association is one of the important steps in the chain of a data fusion process. Data associations are also important in many other fields of studies. Data association addresses the problem of how to differentiate, identify and associate the correct data to the right object, given a set of data. These include clustering of data, data mining and correlating the data. Irrespective of how good the backend algorithm maybe, such as tracking algorithm or classification algorithm, the fusion process will fail without a good association of data from multiple sources.

Data association is needed across the data fusion level, so long as the fusion at each of the levels has different data being sent or considered. At each of the fusion processes, data association problems could be defined and handled differently. This is mainly due to the difference in data features, data density and the different need of the fusion process (such as tracking, identification or situation/impact assessment needs).

For example, the need of tracking in the level 1 fusion process, data association is defined as the process of assigning a new measurement to a target in order to perform the filtering operation. The wrong association of data will result in poor tracking performance. The association algorithm may be different in different environments. By environment, we refer to:

- Target densities. In a multitarget tracking environment many targets are observed simultaneously.
- Clutter densities. The problem in a cluttered environment is too many false

returns within the correlation gate. The difficulty of tracking a manoeuvring target through a cloud of spurious returns generated by clutter, false measurement, or man-made interference, arises from the uncertainty in plot-to-track association. The problem is even more difficult when there are multiple manoeuvring targets.

Hence, it is important to have reliable data association techniques in order to properly track the targets in view. This general problem is discussed in detail in seven books [16, 12, 31, 227, 18, 93, 30]. The content in this chapter draw information from these seven books, particularly in the perspective of tracking system.

#### 6.2 Methods of Data Association

Data association methods could be classified into various groups. One of the common grouping is shown in Figure 5.15 (see chapter 5.5). In the context of tracking targets, data association can be classified into:

- Target-oriented approach. This approach assumes that each measurement has originated from either a known target or clutter.
- Track-oriented approach. This approach hypothesizes that each track is either undetected, terminated, associated with a measurement, or linked to the start of a manoeuvre.
- Measurement-oriented approach. This approach generates a number of candidate hypotheses based on the measurement received and evaluates these hypotheses as more measurement data are received.

Association algorithms specifically used for track formation are:

- Joint likelihood function method.
- Logic-based multitarget track initiation. The Hough transform is the well-known method for initiating multiple target tracks.
- Roecker's two-dimensional passive sensors method [197]. This method monitors the residual sequence produced by the Kalman filter and the differences in the inclination or hinge angles between sensors. Utilizing both monitoring processes, incorrect return-to-track data association, incorrect track-to-track data association (ghosting), and target manoeuvres can be detected. This monitoring method is claimed to be much faster than the multiple hypotheses tracking (MHT) and joint probabilistic data association filter (JPDAF) approaches.

Another method of classifying the data association algorithms is the non-Bayesian and Bayesian association techniques. Examples are:

- Non-Bayesian association techniques:
  - Nearest neighbour standard filter (NNSF)
  - Strongest neighbour standard filter (SNSF)
  - Generalized nearest neighbour filter (GNN) [4]
  - Joint likelihood function method (track formation)
  - Logic-based multitarget track initiation
- Bayesian association techniques:
  - Probabilistic data association filter (PDAF)
  - Multiple model probabilistic data association filter (MMPDAF)
  - Interacting multiple model probabilistic data association filter (IMM-PDAF)
  - Joint probabilistic data association filter (JPDAF)
  - Joint probabilistic data association coupled filter (JPDACF)
  - Multiple manoeuvre model JPDA (M<sup>3</sup>-JPDA) [128]
  - Interacting multiple model with joint probabilistic data association filter (IMMJPDAF)
  - Multiple hypotheses tracking (MHT)

Here, data association will be discussed based on the following 4 groups namely:

- Nearest neighbour.
- Multiple hypotheses.
- Probabilistic data association (also known as All Neighbour Association).
- Graph-based data association

## **6.2.1** Two-stage data association process

In practise, data association is normally implemented in two stages, namely the coarse and fine data association stages. The coarse data association stage serves as a first step<sup>1</sup> in removing data that are clearly not associated to one another. For example, a different classification type or different frequencies range or a sensor detectable angle and range, such that the difference is greater than a certain threshold. Coarse data association could generally use the nearest neighbour algorithms. Gating criteria used in most correlation and association algorithms are basically performing coarse association step. This is the step that will help prune the unnecessary

<sup>&</sup>lt;sup>1</sup>Or first cut iteration by discarding unwanted data and cluster essential data.

data or information to make the next stage of association more effective in term of computational process and accuracy of results.

After the coarse data association, the association process will then proceed to the fine association stage. The fine data association process will further refine the association process and involves a lot more information, such as the error density, the probabilistic of the uncertainty and the relationship between data sources or the correlation <sup>2</sup>. An important point to note is that the fine data association process may also be closely related to the fusing of data. Hence, fine association for target tracking is also linked to the state estimator. It provides a closed loop path for the state estimator, and both the algorithms (namely the state estimator and the data associator) work iteratively in improving the tracking process. For example, the IMMPDAF (interactive multiple model probabilistic data association filter), where the state estimation performed by the IMM is with the PDAF algorithm, jointly working hand in hand, in an iterative manner. The probabilistic approach of implementing the fine data association for target tracking will be discussed in section 6.5. Some of the algorithms used in the fine association process are probabilistic association methods, bio-inspired methods and graph-based competitive methods.

# 6.3 Nearest Neighbour

This is the simplest data association technique and is often used as a coarse data association stage. This method clusters or selects the data that is nearest to one another. How near is near is a factor that is argueable and is most often decided by a threshold value that is implemented by the designer. The threshold is either computed, based on the worst case situation, or a sensible value chosen by the expert. In tracking, the nearest neighbour method selects the measurement nearest to a predicted target position.

The nearest measurement criteria can be based on:

- Absolute distance. Example  $|x_1-x_2|=D$  where  $x_1$  and  $x_2$  are the measureable state.
- Euclidean distance. Example  $||x_1 x_2|| = D$  or
- Statistical distance function. Example of the nearest measurement criteria according to distance measurement is

$$D = [z_i(k+1) - \hat{z}_i(k+1|k)]S(k+1)^{-1}[z_i(k+1) - \hat{z}_i(k+1|k)]^T$$

where S is the covariance matrix of the innovation and z is the true measurement and  $\hat{z}$  is the predicted value.

<sup>&</sup>lt;sup>2</sup>The computed values are also known as correlation strength.

Improvement to the nearest neighbour method, such as the generalized nearest neighbour filter [4], which uses a combination of the theories in nearest neighbour and the probabilistic data association filter (PDAF), is also reported in the literature.

The strongest neighbour standard filter (SNSF) selects the strongest measurement in terms of signal intensity among the validated ones. This assumes that signals intensity information is available. A theoretical analysis and performance prediction of tracking in clutter with strongest neighbour filters is found in [143].

However, the nearest neighbour approach may lead to very poor performance in an environment where spurious measurements occur frequently.

# **6.4** Multiple Hypotheses

The multiple hypotheses method performs both the coarse and fine association stage together. It generally start off by associating all the possible data generated at the same instant of time - thereafter it will start to prune the wrong association. Depending on the pruning process, it will determine how close the association will be. There are generally two types of multiple hypotheses methods. These are:

- MHT (Multiple hypotheses tracking) or sometimes also known as multiple hypotheses testing.
- PMHT (Probabilistic multiple hypothesis tracking).

## 6.4.1 Multiple hypothesis tracking

MHT was developed for tracking multiple targets in a cluttered environment, thus it treats the combined problem of target detection and tracking in an integrated framework. The MHT hypothesis computation normally use Bayes rule or Bayesian network.

Generally, researchers have claimed that MHT is better than JPDAF for lower false target densities. The main limitation of MHT is the increase in computational load when the target or false return density increases. This limitation is normally overcome by means of pruning or fixed length windowing techniques. For a tracking system, the MHT technique can be considered into the following two approaches:

- Measurement-oriented approach and
- Track-oriented approach.

#### 6.4.1.1 Measurement-oriented approach

The measurement-oriented approach was pioneered by Reid [196]. Reid's tracking algorithm is known as the standard multiple hypotheses tracking (MHT) algorithm. MHT is one of the more suitable methods for tracking multiple targets in a clutter environment.

The MHT generates a number of candidate hypotheses based on the measurements received and evaluates these hypotheses as more measurement data are received. If the hypothesis at time k is represented by  $H(k) = [h_l(k), \quad l = 1, ..., n]$ , then the probability of a hypothesis  $h_l(k)$  can be expressed recursively using the Bayes' rule

$$P(h_{l}(k)|Z(k)) = P(h_{g}(k-1), a_{i}(k)|Z(k))$$

$$= \frac{1}{c}P(Z(k)|h_{g}(k-1), a_{i}(k))$$

$$*P(a_{i}(k)|h_{g}(k-1)) *P(h_{g}(k-1))$$
(6.1)

where  $h_g(k-1)$  is the gth hypothesis of the set of hypotheses up to time k-1;  $a_i(k)$  is the ith possible detection-to-target assignment in the current scan; Z(k) is the set of detections in the current frame; c is the normalization constant. Note that the first term on the r.h.s. of equation (6.1) is the likelihood function of the measurement set Z(k), given the joint prior and the current hypotheses. The second term is the probability of the current data association hypothesis, given prior hypothesis  $h_g(k-1)$ . The third term is the probability of the prior hypothesis from which the current hypothesis is derived from.

The MHT algorithm has the ability to detect a new track through maintaining the hypothesis tree structure. The probability of a true track is given through the Bayes decision model as

$$P(\lambda|Z) = \frac{P(Z|\lambda) * P_o(\lambda)}{P(Z)}$$

where  $P(Z|\lambda)$  is the probability of receiving the measurement data set Z given that a true signal source  $\lambda$  is present;  $P_o(\lambda)$  is the a *priori* probability of a true signal source appearing with surveillance volume; and P(Z) is the probability of receiving the set of detections Z. The details of the MHT algorithm can be found in [31].

MHT considers all the possibilities of not only track maintenance but also track initiation and deletion in an integrated framework. It evaluates the probabilities that there is a target from which a sequence of measurements originated. It does not assume a known number of targets. The MHT considers all branching hypotheses (including new tracks), hence it is a good method of insuring proper return-to-track data association. However, as the number of cluster tracks and observations grow as time elapses, the computational complexity will grow exponentially. Hence, its practical implementability could be limited, especially for some processors-limited applications [197, 64].

In summary, the MHT seeks to find the most probable hypothesis through an exhaustive search over all the hypotheses. The application of MHT to the infrared surveillance system problem can be found in [29].

#### 6.4.1.2 Track-oriented approach

The track-oriented approach is proposed in [10] to deal with data association problem in multitarget tracking. This approach hypothesized that each track is undetected, terminated, associated with a measurement, or linked to the start of a manoeuvre. In this approach, the number of data association hypotheses could increase rapidly, with the increase in the number of targets and the number of measurements. Therefore, in a multitarget tracking algorithm, the computational cost for data association hypotheses would be excessive when the number of targets and the number of measurements are large.

However, it is claimed that a track-oriented MHT is computationally more efficient. All associations with a sufficiently high quality, based on the closeness of measurements and tracks, such as those measured by a logarithmic probability quotient, are retained to form new alternative tracks. The probabilities that each alternative track is correct, based on the assumption that there is only one target, are then computed recursively using Bayesian techniques.

#### 6.4.2 PMHT

PMHT was introduced by Streit and Luginbuh in 1994 and subsequently being studied and enhanced by a number of researchers [239, 200]. PMHT addresses the problem of origin uncertainty in multitarget tracking when multiple measurements are received without target labels; that is, the target of origin of a measurement is unknown.

PMHT is derived by formulating a joint meaurement-assignment density, where the measurement-to-track assignments are modeled by a discrete but unknown random variable. The joint density is marginalized over the assignments, and the Expectation-Maximization (EM) method is used to estimate a set of measurement-to-track probabilities for each measurement and each target model. The PMHT method assume the number of target models and the synthetic measurements are known. For a more detailed discussion of this method, reader may refer to [239, 200].

### 6.5 Probabilistic Data Association

Probabilistic data association is also known as the all neighbour method. This method most often handles the data association at the fine stage.

#### 6.5.1 Probabilistic data association filter (PDAF)

The PDAF was first proposed by Bar-Shalom and Tse [20]. The algorithm assigns a probability, called the association probability, to every hypothesis associating a validated measurement to a target. The validated measurements refer to measurements that lie in the validation gate of a target at the current time. A validation gate centred around the predicted measurement of the target set up to select the set of validated measurements is

$$(z(k) - \hat{z}(k|k-1))^T S^{-1}(k)(z(k) - z(k|k-1)) \le \gamma$$
(6.2)

where S(k) is the covariance of the innovation and  $\gamma$  determines the size of the gate. The set of validated measurements at time k is

$$Z(k) = z_i(k), \quad i = 1, ..., m_k$$
 (6.3)

where  $z_i(k)$  is the *i*th measurement in the validation region at time k. This is also known as the 'all neighbour' modified filter.

The standard PDAF equations are as follows [195]:

State prediction

$$\hat{x}(k|k-1) = F\hat{x}(k-1|k-1) \tag{6.4}$$

Measurement prediction

$$\hat{z}(k|k-1) = H\hat{x}(k|k-1) \tag{6.5}$$

Innovation of ith measurement

$$\nu_i(k) = z_i(k) - \hat{z}(k|k-1) \tag{6.6}$$

Covariance prediction

$$P(k|k-1) = FP(k-1|k-1)F^{T} + GQG^{T}$$
(6.7)

Innovation covariance and Kalman gain

$$S(k) = HP(k|k-1)H^{T} + R$$
(6.8)

$$K(k) = P(k|k-1)H^{T}S(k)^{-1}$$
(6.9)

Updated covariance if target originated measurements were known

$$P^{o}(k|k) = P(k|k-1) - K(k)S(k)K(k)^{T}$$
(6.10)

Overall covariance update

$$\nu(k) = \sum_{i=1}^{m_k} \beta_i(k) \nu_i(k)$$
 (6.11)

$$P(k|k) = P^{o}(k|k) + K(k)[\beta_{o}(k)S(k) + \sum_{i=1}^{m_{k}} [\beta_{i}(k)\nu_{i}(k)\nu_{i}(k)^{T}] - \nu(k)\nu(k)^{T}]K^{T}(k)$$
(6.12)

where  $m_k$  is the number of validated returns at kth instant. Updated state estimate

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)\nu(k)$$
(6.13)

The PDAF association probabilities are [205, 50]

$$\beta_i(k) = \frac{p_i(k)}{\sum_{i=0}^{m(k)} p_i(k)}$$

where

$$p_i(k) = \begin{cases} \lambda(1 - P_d P_g) & \text{if} \quad i = 0 \\ \frac{P_d}{(2\pi)^{M/2} |S(k)|^{1/2}} exp[-\frac{1}{2} r_i(k)^2] & \text{if} \quad [\Omega(k)] = 1; \quad i \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

and 
$$\lambda=\frac{m_k}{V(k)}, V(k)=\frac{\pi^{M/2}}{\Gamma(M/2+1)}\gamma^M|S(k)|^{1/2}$$
,[205]

$$\Omega(k) = \left\{ egin{array}{ll} 1 & \mbox{if the return belongs to the validation gate of the target} \\ 0 & \mbox{otherwise} \end{array} \right.$$

M is the dimension of the state vector and  $\lambda$  is the clutter density.  $P_d$  is the probability of detecting the correct return and  $P_g$  is the probability of validating a detected return.

A weighted average of the state, estimated under all the hypotheses associating different measurements (returns) to a particular target serves as the PDAF estimate of the state of that target. The associating of different measurements to a particular target serves as the PDAF estimate of the state of that target. Hence, the association probabilities are used as weights.

The disadvantages of the PDAF are:

- Miss-tracking. Since the PDAF ignores the interference from other targets, it
  may, sometimes, result in miss-tracking of closely-spaced targets [82]. Hence,
  may perform poorly while tracking is crossing targets or when the targets are
  close to each other.
- Suboptimal Bayesian approach. The PDAF is a suboptimal Bayesian approach to the problem of tracking when the source of measurement data is uncertain, due to clutter and missed detections.

- Single target only. The PDAF is designed for the association of a single target
  in a clutter. The number of false alarms is usually modelled by a Poisson
  density and false alarms are assumed to be distributed uniformly in space.
  When there are multiple targets in a surveillance area, the PDAF does not
  perform well, because the false alarm model is no longer valid. It is due to the
  persistent returns generated by interfering targets.
- Need to provide separate track initiation and deletion algorithms. PDAF assumes that a track has been established and, therefore, track initiation and deletion algorithms have to be provided separately.
- Mainly good for tracking non-manoeuvring targets in cluttered environments.
   If the target undertakes a manoeuvre, it is highly likely that the PDAF will lose the target track [153].

Various modifications to combine and improve the PDAF with the state estimation are suggested. Most of these algorithms are discussed in Bar-Shalom and Li [18]. Here, we will just state the algorithms, their assumptions and advantages.

- Multiple model probabilistic data association filter (MMPDAF). This is a combination of the non-switching multiple model and the PDAF. The result is an adaptive estimator that can adjust itself to the 'true model' of the target while in a cluttered environment. The assumptions of the MMPDAF are:
  - The system obeys one out of N models.
  - The models do not switch in time.
  - The models differ only in the process and/or measurement noise levels.

The MMPDAF can track a manoeuvring target in clutter with a number of PDAFs, based on different models, that are running in parallel [18].

- IMM combined with PDAF (IMMPDAF). The IMMPDAF can be used to
  - Initiate tracks.
  - Carry out track maintenance for manoeuvring targets.
  - Terminate tracks.

This is accomplished with the following set of models:

- Model 1: Undetectable target (no target).
- Model 2: True target moving with nearly constant velocity.
- Model 3: Motion with large acceleration increments.
- Model 4: Nearly constant acceleration motion.

This approach assumes that the modelling assumptions are correct.

In [194], it is claimed that the IMMPDAF is incapable of performing track detection when the signal-to-noise ratio (SNR) is low [194]. Hence, this technique may not recover from a track loss situation.

One improvement in a IMMPDAF is to use the amplitude information. Hence, the technique is known as the IMMPDAFAI. IMMPDAFAI uses the statistical information of the detector output for target returns and clutter to improve track maintenance for low SNR targets in a dense cluttered environment [138]. The amplitude likelihood ratio

$$\lambda_i = \frac{p_1^{\tau}(a_i(k))}{p_0^{\tau}(a_i(k))} \tag{6.14}$$

is used to modify the standard PDAF association probabilities to include amplitude discrimination.  $p_o(a)$  and  $p_1(a)$  are probability density functions of the amplitude if it is due to noise only and if it is originated from a target, respectively. The amplitude likelihood ratio term affects the association probabilities by favouring a high amplitude measurement.

- Directional probability data association (DPDA) [50]. The DPDA incorporates the directional information and the association probabilities which are estimated using both the Mahalanobis distance and the track direction.
- Exponentially weighted probabilistic data association filter (EWPDA) [153].
   In [153], it is claimed that the algorithm can track a manoeuvring target in a cluttered environment.

### 6.5.2 Joint probabilistic data association filter (JPDAF)

JPDAF is designed for tracking multiple targets in a clutter environment. The JPDAF is the same as the PDAF, except that the association probabilities are computed using all observations and all tracks [82]. It seeks to determine the probability  $\beta_i^t$  that measurement i originated from target t. Hence, for a known number of targets it evaluates the measurement-to-target association probabilities (for the latest set of measurements) and combines them into the corresponding state estimates.

If the association probability  $\beta_i^t(k)$  is known, the Kalman filter update equation track t can be written as:

$$\hat{x}^{t}(k|k) = \hat{x}^{t}(k|k-1) + K(k)\bar{\nu}^{t}(k)$$

where  $\hat{x}^t(k|k)$  and  $\hat{x}^t(k|k-1)$  are the estimate and prediction of the state of target t respectively. K(k) is the filter gain and a weighted sum of the residuals associated with the m(k) observation to target t is

$$\bar{\nu}^t(k) = \sum_{i=1}^{m(k)} \beta_i^t(k) \nu_i^t(k)$$

where  $\nu_i^t(k) = z_i(k) - Hx^t(k|k-1)$ . Therefore, it incorporates all observations within the neighbourhood of the predicted target position to update the position estimate using an *a posteriori* probability weighted sum of residuals. Targets are divided into clusters.

In the JPDAF the joint probability is calculated over all targets and measurements (or sometimes known as hits). The joint probability is the probability of all the individual events combined as a joint event. The JPDAF includes the possibility of multiple measurements in one-track extension gate coming from other targets. Similar to the PDAF, the JPDAF algorithm performs associations only on the last frame using established tracks and is, therefore, unsuitable for track initiation [186].

The constraints of the JPDAF are:

- No measurement can come from two targets.
- No two measurements can come from the same target.
- $\sum_{i=0}^{m(k)} \beta_i^t(k) = 1$ .
- $\beta_i^t(k)$  should be large if  $\rho_i^t$  is large.
- $\nu_i^t$  is allowed to be large when  $\nu_i^t$  is small for all  $T \neq t$ .

The JPDAF is reported to be better than MHT when the false target densities are high, such as in sonar or radar air-to-ground tracking applications [31].

The disadvantages of the JPDAF are:

- Lack of an explicit mechanism for track initiation. A JPDAF assumes that tracks have been established and the number of targets are known.
- Computationally intensive in multitarget and dense clutter environments. This
  is because the number of possible hypotheses associating different returns to
  targets under consideration increases rapidly with the number of returns the
  presence of clutters increases the complexity further. This may limit its practical application.

Various algorithms are discussed to modify and improve the JPDAF. These are given below:

Nearest neighbour joint probabilistic data association filter (NNJPDAF) approach. This approach differs only slightly from the conventional method of recursive track-to-measurement association. Conventionally, the tracks are extended and the statistical distance from the predicted position to the measurements in the extension gates are calculated. A two-dimensional assignment is made to determine which individual measurement is to be used to update a track. The simplest two-dimensional assignment is the nearest neighbour algorithm.

- Joint probabilistic data association fixed-lag smoother (JPDAS) is proposed in [147] to improve the tracking accuracy. This algorithm combine the JPDAF and the augmentation approach in [28]. This method shows that a significant improvement in the accuracy of track estimation of both non-manoeuvring and manoeuvring targets can be achieved by introducing a time lag of one or two sampling periods between the instants of estimation and latest measurement.
- IMMJPDAF [14, 11]. IMMJPDAF refers to IMM combined with a JPDAF to form a new algorithm that claims to be able to track a target that 'splits' into two targets in a clutter environment. This algorithm evaluates the split-track probability. It is a generalization of the IMMPDAF algorithm.
  - To incorporate the splitting into two targets, two new models are added on top of those discussed in the IMMPDAF approach, namely the 'Just split' target and the 'Split target' models.
- Nearest neigbour probabilistic data association (NNPDA) [78]. This is a modification of the JPDAF in which the probabilities of association are used to associate observations with track files directly on a one-to-one basis.

The JPDAF method has been extensively used in sonar and other surveillance systems. The reported applications include: observations having random arrival times [19], adaptive thresholding [81], unresolved measurements [49], and manoeuvre detection and tracking in a clutter [26].

# 6.6 Graph-Based Data Association

Graph theory is an old subject and has been used for many applications. One of the applications is the task assignment problem, which is closely related to data association we discussed here. The association process for graph-based technique will involve ploting the association graph<sup>3</sup>. Association graph consists a set of vertices or points and a set of edges or links between vertices.

We will illustrate the construction of the association graph for removing false target or deghosting <sup>4</sup>. This application is particularly useful for fusion systems that need to consider passive sensor system information. Most passive sensors, such as ESM and acoustic sensor can only detect and process the direction of arrival (DOA) or bearing of the target. The target location is then obtained by the triangulation of the multiple bearings that are associated to the same target. However, associating the correct bearings from multiple passive sensor systems to the right targets is not a trivial problem.

<sup>&</sup>lt;sup>3</sup>Note the term association graph was also used by Yen and Chen [246] for data mining algorithms.

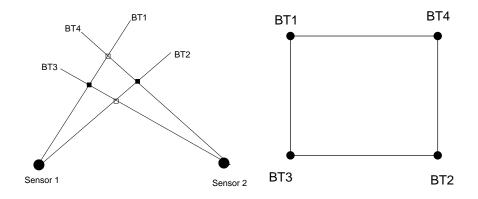
<sup>&</sup>lt;sup>4</sup>The false targets are also known as ghost targets. Hence, the word deghosting means removing of the false target.

## 6.6.1 Association graph

Each vertex of the association graph represents a bearing, and the edge between two vertices represents the association strength between the two bearings. Only bearings from different sensors have edges. The coarse data association will remove or cut the edge that does not meet the criteria. For example, the criteria could include:

- Triangulated location is within the sensor system detection range.
- Bearings are not in parallel or more than 90 degrees apart.
- The bearings are referring to the same target type.

Figure 6.1 shows the association graph of a simple scenario.



**Figure 6.1:** A simple scenario and its association graph.

The vertices are  $\{BT1, BT2, BT3, BT4\}$ , and the edges are  $\{BT1-BT3, BT3-BT2, BT2-BT4, BT4-BT1\}$ .

## 6.6.1.1 Association strength

The association strength <sup>5</sup> on each edge is a critical part to decide whether two linked vertices should be associated together. The computation method of association strength may be different on different types of sensor systems. Here we assume that sensor systems have three measurements that can be used to compute the association strength. For example, in the electronic support measure (ESM) the three

<sup>&</sup>lt;sup>5</sup>Sometime also known as correlation strength.

measurements could be the frequency, pulse repetitive interval and the pulse width. The association strength s is computed by:

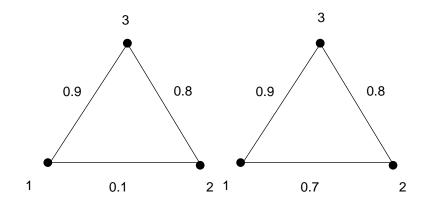
$$s = \frac{w_1 s_1 + w_2 s_2 + w_3 s_3}{w_1 + w_2 + w_3} \tag{6.15}$$

where  $s_1, s_2, s_3$  are three measurements, and  $w_1, w_2, w_3$  are the three weights of  $s_1, s_2$  and  $s_3$  respectively.

# 6.6.2 Competing edges

The term competing edge in an association graph comes from the concept of competition. We view two vertices, 1 and 2, as competing to associate with another vertex 3, if the two vertices 1 and 2 are not compatible, but each of them could be compatible with vertex 3. The edge between vertex 1 and 3 and the edge between vertex 2 and 3 are defined as competing edges.

In Figure 6.2, on the left graph, vertex 1 and vertex 2 are not compatible due to the weak association strength between them (0.1), but both of them are compatible with vertex 3. Therefore, we define edge 1-3 and edge 2-3 as competing edges. However, on the right graph, edge 1-3 and edge 2-3 are not the competing edges. This is because vertex 1 and vertex 2 has strong association strength of 0.7.



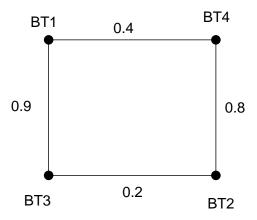
**Figure 6.2:** Competing edges

#### 6.6.2.1 The reduction algorithms

After computing the association strength on all the edges, the edges on the association graph can be further cut. There are two important values to be assessed: The first is the local strength of a edge between two vertices, which is simply the raw association strength, as computed in equation 6.15. The second is the global strength

of an edge. The global association strength is computed by taking the local strength and subtracting the maximum of all the local strengths of the competing edges. This indicates the relative strength of the edge under competition from other edges.

There are two steps to cut edges. We use the example in Figure 6.1 to illustrate the steps. Its association graph with local association strengths is shown in Figure 6.3.



**Figure 6.3:** Association graph with local association strength.

Step 1: Cut edges by the local association strength. The edges will be cut if their local association strengths are less than a predefined threshold. Assume the threshold is defined as 0.3, then the edge BT2-BT3 is cut.

Step 2: Cut edges by the global association strength. The global association strength will be computed for the remaining edges. If the global association strength is greater than a predefined threshold, then the edge is not to be cut, but all its competing edges are cut. If the global association strength is not greater than the threshold, the edge is to be cut. For example, in Figure 6.4 (left graph) we assume the threshold is defined as 0.2. The global association strength of edge BT1-BT3 is 0.5, which is greater than the threshold. The edge BT1-BT3 is not to be cut, but its competing edge BT1-BT4 is cut. The process iterates till all remaining edges are checked. The final result is shown on the right graph of Figure 6.4.

The association graph algorithms can be summarized as follows:

- 1. Construct the association graph.
- 2. Prune the edges by coarse association.
- 3. Calculate the local association strength for the remaining edges.
- 4. Cut the edges by thresholding the local association strength.

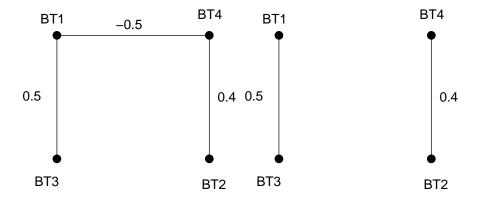


Figure 6.4: Association graph with global association strength

- 5. Compute the global association strength.
- 6. Cut the edges by thresholding the global association strength.

# 6.7 Data Association using Bio-inspired Algorithms

There may be potential new ways we could learn and engineer by understanding how nature associates entities. One of these key areas is understanding how our brain could seamlessly associate large volumes of data and information. This could be one of the most interesting research areas.

Although we are far from achieving the way nature associates the multitude of data and information coming from our sensory systems, some researchers have tried using biologically inspired (bio-inspired) techniques, such as neural network and fuzzy logic, to enhance existing data association algorithms. These methods will be briefly discussed in the next subsection.

#### 6.7.1 Neural data association methods

The central feature of the PDAF and JPDAF is the determination of association probabilities  $\beta_i^t$ . Since the computation of association probabilities is complex, especially with the presence of clutter, the neural network approach is suggested to improve the computational efficiency of  $\beta_i^t$ .

The main motivation for using a neural network is the claim that the complexity of computing the association probabilities is reduced. As far as we know, literature survey reviewed that there are 2 types of neural networks used for data association, namely:

- Boltzmann machine networks [107, 108].
- Hopfield neural networks [205, 139].

In [107], Iltis and Ting show that the Boltzmann machine network, with simple binary computing elements, can estimate the association probabilities. The results are similar to the JPDAF, however with less computational load. Iltis and Ting have also shown in [108] that by using sufficient parallel Boltzmann machines the association probabilities can be computed with arbitrarily small errors.

The Hopfield's neural network for computing the association probabilities is viewed as a constrained optimization problem. The constraints are obtained by a careful evaluation of the properties of the JPDA association rule. The problem is formulated in the same manner as the Hopfield's travelling salesman problem (TSP). That is, the energy function of the data association problem is similar to the energy function of the TSP. In the data association, by minimizing the energy function, it is hoped that the association probabilities  $\beta_i^t$  from the likelihoods could be computed accurately with minimum computational load.

The use of the Hopfield neural network to compute  $\beta_i^t(k)$  was first demonstrated in [205]. The distance is measured from track to measurement. The energy function of the data association problem (DAP), using the Hopfield neural network, is given as follows [205]:

$$\begin{split} E_{DAP} &= \frac{A}{2} \sum_{i=0}^{m(k)} \sum_{t=1}^{T} \sum_{r \neq t}^{T} V_i^t V_i^r + \frac{B}{2} \sum_{l=0}^{m(k)} \sum_{t=1}^{T} \sum_{j \neq l}^{T} V_i^t V_j^t \\ &+ \frac{C}{2} \sum_{t=1}^{T} \{ \sum_{i=0}^{m(k)} V_i^t - 1 \}^2 + \frac{D}{2} \sum_{i=0}^{m(k)} \sum_{t=1}^{T} (V_i^t - \rho_i^t)^2 \\ &+ \frac{E}{2} \sum_{i=0}^{m(k)} \sum_{t=1}^{T} \sum_{r \neq t}^{T} \left( V_i^t - \sum_{j \neq l}^{m(k)} \rho_j^r \right)^2 \end{split}$$

where  $\rho_i^t = \frac{p_i^t(k)}{\sum_{j=0}^{m(k)} p_j^t(k)}$  is the normalized version of the likelihood function  $p_i^t(k)$ . Taking the derivative of  $E_{DAP}$  with respect to  $V_i^t$  and obtaining the neuron dynamics:

$$\frac{du_i^t}{dt} = -au_i^t - A \sum_{r \neq t}^T V_i^r - B \sum_{j \neq i}^{m(k)} V_j^t - C \left( \sum_{j=0}^{m(k)} V_j^t - 1 \right) - [D + E(T-1)]V_i^t + (D+E)\rho_i^t + E \left( T - 1 - \sum_{r=1}^T \rho_i^r \right).$$
(6.16)

The solution from equation (6.16) is calculated using the neuron input-output transfer function to obtain the neuron output voltage (association probability) [55]:

$$\beta_i^t(k) = V_i^t = f(u_i^t) = \frac{1}{2}(1 + \tanh(u_i^t))$$
(6.17)

 $\beta_i^t$  is approximated from the output voltage of  $V_i^t$  of a neuron in an  $(m+1) \times n$  array of neurons, where m is the number of measurements and n is the number of targets.

The connection weights are given as:

$$T_{xi,yi} = -A\delta_{xy}(1 - \delta_{ij}) - B\delta_{ij}(1 - \delta_{xy})$$
$$-C - Dd_{xy}(\delta_{jj+1} + \delta_{jj+1})$$
$$I_{xi} = Cn$$

In [139], a modified Hopfield network is used to approximately compute  $\beta_i^t$ . This modified Hopfield network uses the Runge-Kutta method and the Aiyer network's structure [2]. Using the Aiyer network structure, the new connection weights are given as:

$$T_{xi,yi} = -A\delta_{xy}(1 - \delta_{ij}) - B\delta_{ij}(1 - \delta_{xy}) - 2A_1\delta_{xy}\delta_{ij} -C + 2(An - A + A_i)/n^2 - D(1 - \delta_{ij})(1 - \delta_{xy})(d_{xi} + d_{yj}) I_{xi} = Cn$$

and with the following recommended choice of choosing the parameters  $(A, B, C, D, A_1)$ : B = A and  $A_1 = \frac{31}{32A}$  and  $C = \frac{A}{10}$ .

The performance of the neural data association depends mainly upon the selection of these five parameters. The modified Hopfield neural network is shown to have a much better performance in the sense that it follows the correct path closer, especially in the manoeuvre region.

It is claimed that the advantage of using the Hopfield neural network is that it reduces the complexity of computing the association probabilities. For a large number of targets, it only requires a larger array of neurons as opposed to the digital approach, which requires an exponential increase of computer resources.

The Hopfield neural network is also used to reduce a likelihood matrix, properly defined, to the assignment matrix [209]. The neural network processes the elements of the likelihood matrix to produce an assignment matrix.

## 6.7.2 Fuzzy logic and knowledge-based data association methods

Fuzzy logic and knowledge-based techniques are potentially possible techniques for data association. Fuzzy logic has been applied to data fusion in a number of ways.

Stover *et al.* [215] have developed a general purpose fuzzy logic architecture. However, using fuzzy membership functions for data association is a new untested area.

Using a knowledge-based approach is proposed by [249]. The function of the filter is to classify the radar measurements within the gate into two groups, namely the true returns and false returns. When the returns (measurements) are received, the respective aligned radar attributes to these returns are sent to the expert system. According to the rules stored *a priori*, a probability mass  $m(a_i)$  is assigned to each return, where i=1,2 indicate whether the return is true or false, respectively.

Fusion systems should make as little mistakes as possible. Hence, the fused results should be of high certainty, if not, it is better to return the unprocessed data to the human operator for interpretation.

## CHAPTER 7

# **Aspects of Target Tracking**

## 7.1 Introduction

This chapter will cover the essential components needed for understanding tracking systems, and particularly the algorithms used for target tracking. It does not intend to cover all the fundamental theory, derivation and proof of the tracking algorithms and the various equations used. Readers who would like to know such details can refer to the references quoted.

Tracking is defined as the estimation of the state of a moving target [18]. Examples of the state of a moving target are:

- bearing and rate of change of bearing (for one-dimensional measurement data).
- position, velocity, acceleration (for two or three-dimensional measurement data).

The moving targets described here refer to any man-made moving objects such as trucks, missiles, planes, submarines, ships and other engine or non-engine driven vehicles.

The heart of the tracking system consists of two main components, namely: the data association module and the kinematics state estimation module. The focus of this chapter is the kinematics state estimation module. The core part of the kinematics state estimation is the filtering and prediction algorithm. There are many different types of filters, such as the fixed-gain filter (also called the fixed-coefficient filter, or more commonly, it is known as the  $\alpha$ - $\beta$  and  $\alpha$ - $\beta$ - $\gamma$  trackers), the Kalman filter and, more recently, the adaption of the Kalman filter in the multiple-model form such as interactive multiple model (IMM). Other filters, such as particle filter and condensation algorithm, could also be used for state estimation. However, the Kalman filter and its various adaptations have been the main workhorse and more popularly used

for the estimation of states. Note that early tracking algorithms, such as the fixed-gain filter (e.g.  $\alpha$ - $\beta$  and  $\alpha$ - $\beta$ - $\gamma$  filters), can also be viewed as a form of Kalman filter only that its 'Kalman' gain is fixed.

Though the Kalman filter provides good performance and is efficient in tracking, it has limitations in handling highly non-linear and manoeuvring targets. To solve the problem, some Kalman filter-related approaches were developed. They are:

- Extended Kalman filter: It uses the first-order approximation to solve the non-linear problem.
- Varies adaptive Kalman filters: It turns the Kalman filter parameters on real time to reduce manoeuvring effects. The parameter turning can be based on fuzzy logic, neural networks or other parametric algorithms.
- Separate-bias Kalman estimator: Uses two parallel filters. The first filter, the bias-free filter, is based on the assumption that bias is non-existent. The second filter, the bias filter, produces an estimate of the bias vector. The final output is from the first filter corrected by the second filter [106].
- Single model switching approach: A manoeuvring detector and several models exist in the tracking system. Only one model is active at a given time, and the manoeuvring detector decides the model selection.
- Interactive multiple model (IMM) filter: Multiple models run in a parallel manner, where the final target state is the combined state based on mode probability. The mode probability is computed using the residual of measurement [11].
- Interacting acceleration compensation (IAC) filter: It is a hybrid approach combining the IMM with the separate-bias Kalman estimator. However, the performance is not better than the IMM [234].
- Multiple model with feedback: It also uses parallel multiple models, as the IMM, but the state combination is based on the covariance of state in each model [5].

In recent years, many researchers introduced the sequential Monte Carlo methods in tracking systems, and proposed particle filters to solve non-linear non-Gaussian problems. There are several methods in particle filters, such as, bootstrap filter, Monte Calro filter, and conditional density propagation (CONDENSATION) from a field of computer vision. Some hybrid algorithms, combining Kalman filters with particle filters, also appeared recently.

# 7.2 Basics of a Tracking Process

Target tracking is part of the data processing and can be implemented on a general-purpose processor in a software form or burnt-in a processor card. Figure 7.1 shows how the tracking system will receive data from a typical sensor system.

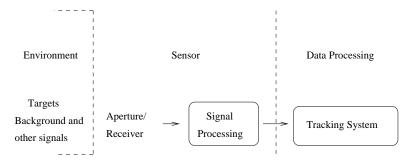


Figure 7.1: Concepts of sensor-to-tracking system.

Figure 7.2 shows the basic block diagram and the flow of the sensor data in a tracking system.

A track formation stage involves two important parts. These are:

- Association Clustering data that refer to the same target.
- Initiation of track This involves starting up a tentative track and confirming
  the track. A tentative track is created for any new data detected that is not
  associated to any existing tracks. A detection ratio is then normally used to
  confirm the tentative track and is computed based on a number of detections
  over the number of scans; where scan refers to the update rate of the sensor.

Track maintenance is the process of maintaining a confirmed track. Filtering of the sensor measurement data play the key role in this process. The steps of a typical filter at time k are:

- 1. Receive input measurements at kth scan.
- 2. Compute the estimated state using the computed gain, the predicted state and the measured state (smoothing process).
- 3. Output the estimated state.
- 4. Compute the next predicted state (k+1) (prediction process).

This filtering process is also known as state estimation. In this context, the fixed-gain filter and the Kalman filter will be introduced at an elementary level. The concepts

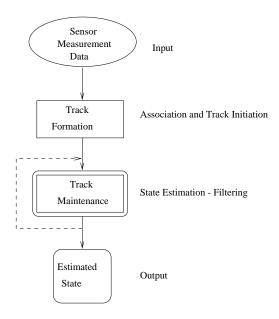


Figure 7.2: Basic tracking stages in a typical tracking system.

behind the multiple model, which is the current state-of-the-art technology for target tracking, will also be presented.

Part of the track maintenance process is to check for deleting track that does not fulfil the updating criteria. The updating criteria depend on the number of scans or update of measurement data on the same target. A simple form of updating criteria is: if no measurement is received for over N scans, then the track will be deleted.

## 7.3 Problem Formulation

The dynamics of a system model of a manoeuvring target in a tracking system is given by

$$\dot{x} = f(x, u, w) \tag{7.1}$$

$$z(k) = h(x(k), v(k)) \tag{7.2}$$

where x is the state vector, u is the control vector, w is the process noise vector representing possible deviations in f(.). z(k) is the discrete-time measurement vector at time k, and v(k) is the measurement noise vector.

The dynamics of the target is a continuous-time process as indicated by equation (A.17). f(.) defines the motion for the target in a form of a differential equation.

The measurement process is in discrete-time because most sensors used for target tracking record the position and velocity at a given instance of time. Hence, the measurement equation (A.18) in discrete form is

$$z(k) = H(k)x(k) + v(k)$$
 (7.3)

The function f(.) is usually unknown to the tracking system. Hence, the target dynamics are commonly modelled by a deterministic linear dynamic system in discrete time, as follows:

$$x(k+1) = F(k)x(k) + G(k)u(k) + w(k). (7.4)$$

At time k, x(k) (assumed to be a vector of dimension  $n_x$ ) is the state of the system normally containing the position and velocity of the target. F(k) is the transition matrix of dimension  $(n_x \times n_x)$  and it defines a linear constraint on the target dynamics, u(k) is the unknown input control, a vector of dimension  $n_u$ , G(k) is the input gain (or noise matrix)  $(n_x \times n_u$  matrix) and  $w(k) \sim N(0, Q(k))$  is the process noise which is usually assumed to be white Gaussian noise with variance Q.

In tracking manoeuvring targets, the control vector is not directly observable by the tracking system. Since the input control is unknown, the dynamics model that is assumed for a target in a tracking system is further simplified as follows:

$$x(k+1) = F(k)x(k) + w(k)$$
(7.5)

The process noise w(k) can be used to model the unknown target acceleration. The unknown acceleration term can also be included in x to form a third-order model. However, the acceleration most often varies with time in such a manner that a model cannot be clearly identified during tracking.

Furthermore, target manoeuvres are generally neither a stochastic process nor have a predictable motion. For example, modern weapon systems equipped with advance technology, such as a missile guidance program, may produce target of uncertain and highly manoeuvring behaviour. This results in a bias or lag in the target state estimation.

The following section describes the various tracking algorithms.

# 7.4 Fixed-gain Filter

Fixed-gain filters, such as the  $\alpha$ - $\beta$  and  $\alpha$ - $\beta$ - $\gamma$  filters, are among the early tracking algorithms. The gains of such filters are precalculated and remain fixed throughout the entire tracking process.

The standard  $\alpha$ - $\beta$  filter can be used to track non-manoeuvring targets. The extension of the  $\alpha$ - $\beta$  filter is the  $\alpha$ - $\beta$ - $\gamma$  filter. The  $\alpha$ - $\beta$ - $\gamma$  filter is designed for manoeuvring

targets with constant acceleration. The  $\gamma$  relationship can sometime be expressed as  $\gamma = \frac{\beta^2}{2\alpha}$  [117]. This filter will be biased if the acceleration varies with time throughout the manoeuvre.

The  $\alpha - \beta$  filter is defined by the following equations:

$$\begin{array}{rcl} x_s(k) & = & x_p(k) + \alpha [x_o(k) - x_p(k)] \\ \\ v(k) & = & v(k-1) + \frac{\beta}{T} [x_o(k) - x_p(k)] \\ \\ a(k) & = & a(k-1) + \frac{\gamma}{T^2} [x_o(k) - x_p(k)] \end{array}$$

and prediction process is described by

$$x_p(k+1) = x_s(k) + Tv(k)$$

where  $x_0(k)$  is the target's observed position (or measured position) at the kth scan,  $x_p(k)$  is the target's predicted position using information at the (k-1)th scan,  $x_s(k)$  is the smoothed target position, v(k) is the target velocity, a(k) is the target acceleration, T is the radar scan time and  $\alpha$ ,  $\beta$  and  $\gamma$  are the fixed-gain filter parameters.

The  $\alpha$ - $\beta$  filter and  $\alpha$ - $\beta$ - $\gamma$  filter are explained in a number of books and literature such as [17, 31, 117].

The advantages of fixed-gain filters are its simplicity and low computational cost. The disadvantages include:

- 1. Gains selected are fixed, hence it can represent only one form of target motion.
- 2. Limited ability in clutter environment.
- 3. May have stability problem under extreme circumstances due to decoupling (independent of  $\alpha$ ,  $\beta$  and  $\gamma$  values).

We can overcome this filtering dilemma by initially starting with a small  $\alpha$  until the manoeuvre is detected, then increase  $\alpha$  for tracking manoeuvring targets. But the problems are:

- the response of the filter is significantly delayed because the gains are not increased until a manoeuvre is detected.
- the α gain must be set artificially high for α β tracker in order to account for the absence of acceleration from the motion model.
- after the target goes back to constant velocity following a manoeuvre, the decision to reduce  $\alpha$  is often delayed.

Despite such disadvantages, fixed-gain filters are still in use for tracking applications, particularly for system using a burnt-in tracking processor. This is partly because the filters have low computational costs, simple to implement and a number of modifications have been done to improve it. Some of these modifications are:

- Variable update time  $\alpha$ - $\beta$  filter [59]. This filter has the update time chosen according to the magnitude of the difference between the measured and the predicted positions.
- Speed-dependent gain

$$\alpha = k_1 * speed + k_2$$
  
$$\beta = k_3 * speed + k_4$$

 $\alpha$  and  $\beta$  are dependent on the speed.  $k_1,k_2,k_3$  and  $k_4$  are fixed values and are decided by the designer or user.

• Gain adjustments based on residuals [44]. This approach works by reducing α and β if the measurement errors are unbiased (speed and headings are accurate); as soon as a bias appears, α and β begin to increase. To compute the α and β values, the filter needs to recursively estimate the variances of the measurement noise (x, y, z for Cartesian and range and azimuth in polar coordinates) and the covariance of the successive measurement errors (a measure of how the variance is changing).

The measurement error here is the residual. This has some similarity to the Kalman filter in that both are using the residual to adjust its gain.

• Fuzzy rule to tune the  $\alpha$ - $\beta$ - $\gamma$  values [48].

# 7.4.1 What are the criteria for selecting the fixed gain?

The first problem is to find a means to optimally select a value for  $\alpha$  and  $\beta$ . An optimal relationship of  $\alpha$  and  $\beta$  is

$$\beta = \frac{\alpha^2}{2 - \alpha} \tag{7.6}$$

The relationship of  $\alpha$  and  $\beta$  is plotted as '\*', as shown in Figure 7.3. The important point to note in this relationship is that good noise reduction requires a small  $\alpha$  (hence, small  $\beta$ ), and good tracking through manoeuvres requires a large  $\beta$ .

A means for selecting  $\alpha$  was introduced by Kalata [117], who defined a variable known as the tracking index  $\Gamma$ , where

$$\Gamma(k) = \frac{\sigma_v T^2}{\sigma_w} = \frac{\beta}{\sqrt{1-\alpha}}.$$

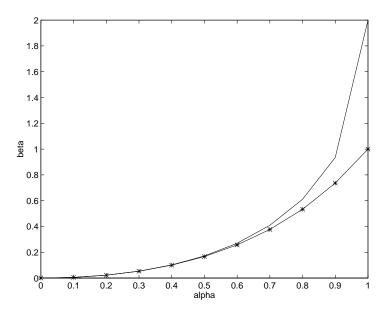
He also suggested the relationship:

$$\beta = 2(2 - \alpha) - 4\sqrt{1 - \alpha} \tag{7.7}$$

This relationship is more useful in steady state conditions. The  $\alpha$  and  $\beta$  relationship is as shown in Figure 7.3.

In general, the recommended fixed values for  $\alpha$  and  $\beta$  at the start are:

$$\alpha = 0.5 
\beta = 0.167$$



**Figure 7.3:** Relationship between  $\alpha$  and  $\beta$  values. Solid line is plotted from equation (7.7) and '\*' is plotted from equation (7.6).

# 7.5 Kalman Filter

Kalman filtering is an optimal state estimation process applied to a dynamic system that involves random perturbations. More precisely, the Kalman filter gives a linear, unbiased, and minimum error variance recursive algorithm to optimally estimate the unknown state of a dynamic system from noisy data taken at discrete real-time intervals [57].

The study of Kalman filtering alone has resulted in the publication of a number of books and many journal articles. The advantages of using the Kalman filter include [30]:

- The gain sequence is chosen automatically, based on the assumed target manoeuvre and measurement noise models. This means that the same filter can be used for varying targets and measurement environments by changing a few key parameters. For example, as a target closes in range, its intensity increases, so angular measurement accuracy usually improves.
- 2. The Kalman gain sequence automatically adapts to changing detection histories. This includes a varying sampling interval as well as missed detection.
- The Kalman filter provides a convenient measurement of the estimation accuracy through the covariance matrix. This measure is required to perform the data association functions.
- Having a measure of the expected prediction error variance is useful for manoeuvre detection.
- 5. Upon detection of a manoeuvre, the Kalman filter model provides a convenient way to adjust for varying target dynamics.
- 6. Through use of the Kalman filter it is possible to, at least, partially compensate for the effects of misassociation in the dense multiple target tracking environment. For example, an increase in the Kalman covariance matrix elements to reflect the expected error associated with uncertain data association.

However, there are a few disadvantages. These include:

- 1. Target probability density function (pdf) is assumed Gaussian. But in many applications this assumption is inappropriate.
- 2. Higher computational load than the fixed-gain filter.

The Kalman filter equations are expressed as

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)[\tilde{z}(k)]$$
 (7.8)

$$P(k|k) = [I - K(k)H]P(k|k-1)$$
(7.9)

where

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

$$S(k) = HP(k|k-1)H^{T} + R$$
  

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

and z(k) is the measurement value,  $\tilde{z}(k)$  is the residual (or innovations) vector (difference between the observed and predicted values), P is the Kalman filter covariance matrix. S(k) is the residual covariance matrix and R is the measurement error covariance matrix. R elements are normally fixed. For a one-dimensional measurement, R becomes a scalar measurement, the noise variance  $\sigma^2$ . H is the measurement transition matrix

Prediction of the next stage:

$$\hat{x}(k+1|k) = F\hat{x}(k|k) + G(k)u(k)$$

$$P(k+1|k) = FP(k|k)F^T + Q$$

where G(k)u(k) = f(k+1|k) is assumed to be a known deterministic input, such as the relative position change associated with ownship motion. Q is the process noise covariance matrix, is assumed known and Gaussian. P(k+1|k) is the prediction error covariance matrix.

The error covariance matrix is defined in terms of the zero-mean Gaussian estimation error vector:

$$P(k) = E[x(k) - \hat{x}(k)][x(k) - \hat{x}(k)]^{T}$$

The Kalman filter can also be explained as consisting of a low gain filter (for the nearly uniform motion) and a high gain filter (for the manoeuvring situation) [142]. Increasing Q will make the Kalman filter more responsive to future target dynamics. Hence, the first method to overcome the manoeuvring problem is to make the Kalman filter more responsive by varying the process noise.

If a target is manoeuvring, then one motion model will be inadequate to describe the full range of possibilities. Hence, multiple model approach are used for tracking manoeuvring targets.

# 7.6 Multiple Model Approach

The basic idea of all multiple model approaches, as applied to tracking manoeuvring targets, is that manoeuvres are typically abrupt deviations from basically a straight-line target motion. As this process is very difficult to represent with a single manoeuvre model, multiple models, representing different potential target manoeuvre states, run in parallel and are continuously evaluated by using the history of the filters' residuals.

The multiple model approach, particularly the IMM method, is currently the most widely accepted method for difficult manoeuvring target conditions.

Bayes's rule and the residuals are used to determine the relative probabilities of validity of the models. The output is typically a probability-weighted composite

of the individual filters, otherwise, it may prove to be more accurate to output the estimates from the filter with the highest probability.

Referring to equation (A.21), for the multiple model approach, the ith dynamics model is denoted by

$$x_i(k+1) = F_i(k)x_i(k) + w_i(k)$$
(7.10)

#### 7.6.1 IMM

The design of an IMM estimator consists of the following [18]:

- selection of the models for the various modes of behaviour of the system.
- selection of the Markov chain transition probabilities among the modes.
- selection of the parameters of the various models typically the process noise levels.

The IMM algorithm can be divided into the following four parts, namely:

- An input mixer (Interaction).
- A filter for each model (Updates).
- A model probability evaluator.
- An output mixer.

The flow diagram of an IMM filter with N models is given in Figure 7.4.

#### 7.6.1.1 Input mixer (Interaction)

The input state estimate mixer merges the previous cycles mode-conditioned state estimates and covariance, using the mixing probabilities, to initialize the current cycle of each mode-conditioned filter.

The filtering process starts with the *a priori* state estimates  $x_j^0(k-1|k-1)$ , state error covariance  $P_j(k-1|k-1)$ , and the associated probabilities  $\mu_j(k-1)$  for each model. The initial state estimate and covariance for model j at time k,  $M_j(k)$ , is computed as:

$$\hat{x}_{j}^{o}(k-1|k-1) = \sum_{i=1}^{N} \hat{x}_{i}(k-1|k-1)\mu_{i|j}(k-1|k-1)$$

$$P_{j}^{o}(k-1|k-1) = \sum_{i=1}^{N} \mu_{i|j}(k-1|k-1)\{P_{i}(k-1|k-1) + P_{i}(k-1|k-1)\}$$

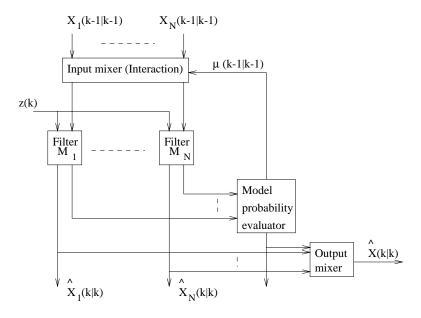


Figure 7.4: The IMM filter.

$$\begin{split} & [\hat{x}_i(k-1|k-1) - \hat{x}_j^o(k-1|k-1)] [\hat{x}_i(k-1|k-1) \\ & - \hat{x}_j^o(k-1|k-1)]^T \} \end{split}$$

where

$$\mu_{i|j}(k-1|k-1) = \frac{1}{\bar{c_j}} p_{ij} \mu_j(k-1)$$
$$\bar{c}_j = \sum_{i=1}^{N} p_{ij} \mu_i(k-1)$$

and  $p_{ij}$  is the assumed transition probability for switching from model i to model j, and  $\bar{c}_j$  is a normalization constant.

## 7.6.1.2 Filtering updates

The updates for each subfilter or model are performed using the Kalman filter or extended Kalman filter equations. Different models are used and usually there is a second-order model with a few third order models. The second-order model is dominating when the target is in non-manoeuvring state and the third-order model for the manoeuvring state with different process noise levels.

An example of a set of Kalman filtering equations providing the model updates

for  $M_i(k)$  is as follows:

$$\begin{array}{rcl} x_j(k|k-1) & = & F_j(k-1)x_j^0(k-1|k-1) + G(k-1)U(k-1) \\ P_j(k|k-1) & = & F_j(k-1)P_j^0(k-1|k-1)(F_j(k-1))^T + Q_j(k) \\ S_j(k) & = & H_j(k)P_j(k|k-1)(H_j(k))^T + R(k) \\ K_j(k) & = & P_j(k|k-1)(H_j(k))^T(S_j(k))^{-1} \\ \tilde{z}_j(k) & = & z_j(k) - H_j(k)x_j(k|k-1) \\ x_j(k|k) & = & x_j(k|k-1) + K_j(k)[\tilde{z}_j(k)] \\ P_j(k|k) & = & [I - K_j(k)H_j(k)]P_j(k|k-1) \end{array}$$

It can also be explained as consisting of a low gain filter (for the nearly uniform motion) and a high gain filter (for the manoeuvring situation) [142].

#### 7.6.1.3 Model probability evaluator

The likelihood of  $M_j(k)$  is computed with the filter residuals  $\tilde{z}_j(k)$ , the covariance of the filter residuals  $S(k)_j$ , and the assumption of a Gaussian distribution. The likelihood of  $M_j(k)$  is given by

$$\Lambda_{j}(k) = \frac{1}{\sqrt{2\pi|S(k)|}} exp[-0.5(\tilde{z}_{j}(k)^{T}(S_{j}(k))^{-1}\tilde{z}_{j}(k)]$$

The model probabilities update is

$$\mu_j(k) = \frac{1}{c} \Lambda_j(k) \bar{c}_j \tag{7.11}$$

$$c = \sum_{j=1}^{r} \Lambda_j(k) \bar{c}_j \tag{7.12}$$

## 7.6.1.4 Output mixer

The output mixer combines all the state estimates and covariances from individual filter outputs as follows:

$$\begin{split} \hat{x}(k|k) &= \sum_{j=1}^{N} \hat{x}_{j}(k|k) \mu_{j}(k) \\ P(k|k) &= \sum_{j=1}^{N} \mu_{j}(k) \{ P_{j}(k|k) + [\hat{x}_{j}(k|k) - \hat{x}(k|k)] [\hat{x}_{j}(k|k) - \hat{x}(k|k)]^{T} \} \end{split}$$

The IMM filter is built from a number of different dynamic motion models that describe different aspects of target motion. For a particular target manoeuvre, the filter

will automatically choose the ideal mix of models. An IMM filter with constant velocity and mean-jerk models is reported to have the ability to track manoeuvring targets in spherical coordinates [33]. The mean-jerk model used is a standard constant acceleration model with time-correlated acceleration errors instead of the standard 'white' acceleration errors. Time-correlated acceleration errors means that the acceleration modelling error at time k is influenced by the acceleration modelling error at time k-1.

It is reported that the performance of the IMM algorithm is not very sensitive to the choice of the transition probabilities [17]. The choice of transition probabilities provides, to a certain degree, the trade-off between the peak estimation errors at the onset of the manoeuvre and the maximum reduction of the estimation errors during the uniform motion. The IMM algorithm could also be readily combined with the probabilistic data association (PDA) technique.

However, there are several shortcomings in the IMM approach to multiple targets tracking. These shortcomings are:

- existence of severe bias if the models do not match the dynamic target manoeuvres, such as unpredicted non-linear manoeuvres.
- heavy computational load which may be critical in real-time applications [144, 8]. This is because in order to operate well, the IMM requires a large number of filters to cover the different possible manoeuvres and number of targets.
- mainly suitable for constant velocity and constant acceleration.
- the accuracy of IMM is governed by the transition probabilities of the Markov chain. The tuning of the Markov chain transition matrix, that is, the *a priori* information, is then crucial to obtaining the correct ordering of the *a posteriori* regime probabilities [12].
- if the observation noise is non-Gaussian, the IMM method degrades. This degradation is due to the non-optimal Kalman filter used in the IMM and the miscalculation of the model probabilities [245].
- the weight probabilities perform badly when the measurement noise is high and the targets move with high manoeuvring acceleration [180].

There are various modifications to improve the IMM algorithm. Some of these are as follows:

- Auto-tuning interactive multiple models [173]. This method enhances the IMM algorithm by automatically tuning the process noise covariance value using the fuzzy and exponential decaying techniques.
- The IMM filter contains correlated acceleration models. In this approach, the IMM filter has been formulated using a constant velocity and two correlated

- acceleration models. It is claimed that this method provides improved tracking performance over an IMM filter containing constant velocity and constant acceleration models [34].
- The interacting multiple bias model (IMBM) [32]. The IMBM algorithm utilizes the IMM algorithm and the two-stage state estimation. It consists of a filter for the bias-free portion of the state model, a filter for each bias model  $^1$ , a model probability evaluator for each bias model, an estimate mixer at the input of the bias filters and an estimate combiner at the output of the bias filters. The flow diagram of an IMBM algorithm with two bias models is given in Figure 7.5, where x(k|k) is the state estimate based on both models,  $b_j(k|k)$  is the bias estimate for the time k based on model j,  $\Lambda_j(k)$  is the likelihood of model j at time k. The IMBM algorithm is claimed to be able to track manoeuvring targets.

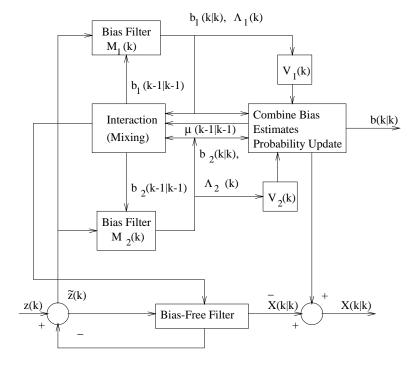


Figure 7.5: The IMBM algorithm.

Selected filter interacting multiple model (SFIMM) [144]. This algorithm is a
modified form of the IMM algorithm. Two additional procedures, namely data
selection and decision procedures, are added to the standard IMM procedure.

<sup>&</sup>lt;sup>1</sup>The target acceleration is modelled as a system bias.

The decision rules are used to choose a specific subset of filters close to the target acceleration.

At each scan the decision procedure checks the likelihood function or the weighting factor for each model, and makes a decision as to whether a subset shift is necessary or not. This information is then passed to the data selection block, which then activates the required filters. By doing so, only limited filters are computed in each cycle. Hence, a saving in the computation load compared with the IMM algorithm.

• Adaptive interacting multiple model (AIMM) [165]. The AIMM algorithm introduces a two-stage Kalman filter to the standard IIMM algorithm. The two-stage Kalman filter consists of a bias-free filter and a biased filter to estimate the acceleration value, â. Then using a limited number of filters, accelerations near to the estimated acceleration are formed. For example, if five filters are used, then the acceleration for the five filters is as follows:

$$\hat{a}, \hat{a}(1 \pm \epsilon_1), \hat{a}(1 \pm \epsilon_2)$$

where  $\epsilon$  is a preselected value. The AIMM algorithm is proposed mainly to improve the computational efficiency of the standard IMM algorithm.

• Non-linear interactive multiple model (NIMM) [245]. The NIMM algorithm uses a non-linear filter known as the Masreliez filter [152] in place of the Kalman filter. The Masreliez filter employs a non-linear score functions as the correction term in the state estimate equation, while retaining the computationally appealing structure of the Kalman filter. Using the Masreliez filter in the IMM, it is claimed that the calculation for the model probabilities is improved. This results in a non-linear IMM (NIMM) which claims to significantly improve the tracking performance, particularly in the glint environment.

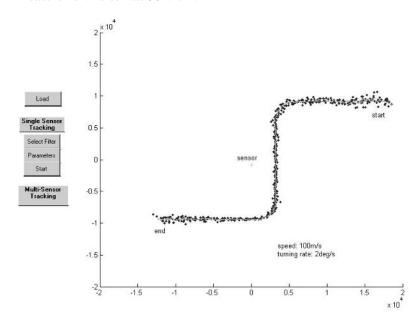
However, Wu and Cheng's method has avoided the convolution operation and simplified the evaluation of the non-linear score function by applying a normal expansion for the distribution of the measurement prediction [245]. Daeipour and Bar-Shalom [63] have shown that the use of the IMM with the extended Kalman filter has a better performance than the non-linear score function method for target tracking with glint noise.

- Innovation filter IMM adaptive algorithm (IF-IMMA) [180]. This algorithm
  proposes to reduce the effect on the weight probabilities of the measurement
  noise. The IF-IMMA is a modification of the IMM algorithm. Instead of using
  the normal constant velocity and constant acceleration model, it replaces the
  model with a Zhou model for tracking manoeuvring targets.
- Parallelization of the IMM algorithm [192]. This approach suggests that, in a situation with multiple targets, a coarse grained parallelization across IMMs is a good strategy to reduce the computational burden of the IMM algorithm.

# 7.6.2 Comparison between IMM and Kalman filters

#### 7.6.2.1 Scenario

Figure 7.6 depicts a target moving from east to west, then turns south, followed by a turn to the west again. The target speed was set to be 100m/s, and the two turning rates to be both 2 deg/s. The true path of the target is shown as a solid line. The sensor measures the target's range and the bearing every second. The measurement noise in range has a standard deviation of 100m, while the bearing noise has a standard deviation of 2 deg. The measurements are shown as dots. The RMS error of the measurement noise was 384.27m.



**Figure 7.6:** Scenario for testing.

#### 7.6.2.2 Filters

The setting of the filters are as follows:

- Kalman Filter A constant velocity model with acceleration process noise is used.
- IMM Filter Two models are used.

| Process noise standard deviation | RMS error (m) |
|----------------------------------|---------------|
| 1.0                              | 374.35        |
| 2.0                              | 234.33        |
| 3.0                              | 217.92        |
| 4.0                              | 210.51        |
| 5.0                              | 207.33        |
| 6.0                              | 206.21        |
| 7.0                              | 210.62        |
| 8.0                              | 212.03        |

**Table 7.1:** RMS errors with different process noise values for the Kalman filter.

| Model 1<br>process noise<br>standard deviation | Model 2<br>process noise<br>standard deviation | RMS error |
|--|--|-----------|
| 0.1  | 2.0  | 204.96    |
| 0.1  | 3.0  | 221.35    |
| 0.1  | 4.0  | 211.52    |
| 0.1  | 5.0  | 206.52    |
| 0.1  | 6.0  | 203.51    |
| 0.1  | 7.0  | 202.39    |
| 0.1  | 8.0  | 206.04    |

**Table 7.2:** RMS errors with different process noise values for the IMM filter with two models.

- Model 1: Constant velocity with a small acceleration process noise. It is used to model straight movements.
- Model 2: Constant velocity with a large acceleration process noise. It is used to model target turning.

### 7.6.2.3 Results and analysis

The result of the RMS error of the Kalman and IMM filters are shown in table 7.1 and table 7.2 respectively.

From the test results, we see that IMM filter produces a better result than the Kalman filter in terms of RMS error. This is because when the target is moving with a constant velocity, the IMM low process noise model becomes dominant, as its mode probability is higher. Once the target manoeuvres, the IMM high process noise model takes over. The mode probability of the high process noise model thus becomes higher. This is shown in Figure 7.7 between time steps 200 and 300.

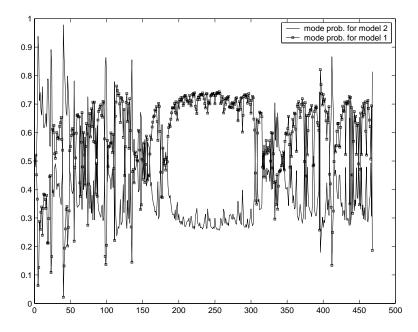
The IMM filter is also less sensitive to the process noise changes than the Kalman

filter. This is mainly because of the multiple models used in the IMM filter.

Figures 7.7, 7.8 and 7.9 are generated using IMM filter models with the following process noise:

- Model 1 with standard deviation of process noise at  $0.1m/s^2$ ;
- Model 2 with standard deviation of process noise set at  $7.0m/s^2$ .

The Kalman filter used in figures 7.8 and 7.9 is set with  $Q = 1m/s^2$ .

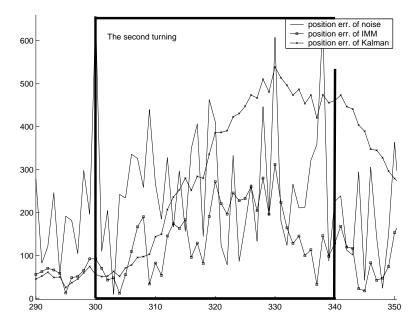


**Figure 7.7:** Mode probability of two models.

Figure 7.8 depicts the position error vs. time when the Kalman filter and IMM filter are used. From the graph, it can be seen that the position error of the Kalman filter is much higher than the IMM filter when target is turning.

#### 7.6.2.4 **Summary**

In summary, the overall performance of the IMM filter is observed to be better than that of the Kalman filter. However, the performances of both filters can be improved with careful tuning of their parameters. In general, the IMM filter with multiple models have an edge over the Kalman filter, which only depends on a single model.



**Figure 7.8:** Position error obtained from Kalman  $(Q = 1m/s^2)$  and IMM filters mode.

# 7.6.3 Comparison between IMM and the generic particle filter

Using the same scenario as in Figure 7.6, the IMM filter and the generic particle filters are compared. Figure 7.10 and table 7.3 show the results of the RMS error vs time. Details of the generic particle filter algorithm are in the appendix.

| Type of Filter | Tuned parameters     | RMS error |
|----------------|----------------------|-----------|
| IMM            | Model 1: $0.1m/s^2$  | 193.94    |
|                | Model 2: $7.0m/s^2$  |           |
| PF-Gaussian    | Noise Deviation: 139 | 197.48    |
| PF-Cauchy      | Noise Deviation: 139 | 209.90    |

Table 7.3: Results of the generic particle filters and the IMM filter

In this scenario - where the target does not perform highly non-linear behaviour - the IMM filter is comparatively better than the generic particle filter. Particle filters also have a much higher computational load due to the need to perform multiple sampling processes. This may constraint its application for real time processes.

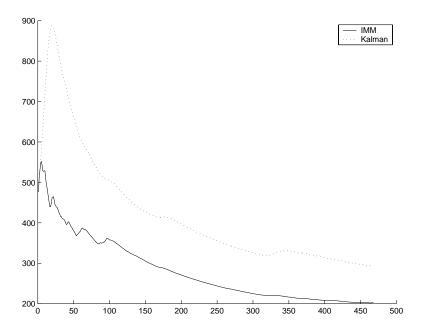


Figure 7.9: RMS error obtained from Kalman  $(Q=1\,m/s^2)$  and IMM filters Mode

# 7.7 Issues Affecting Tracking Systems

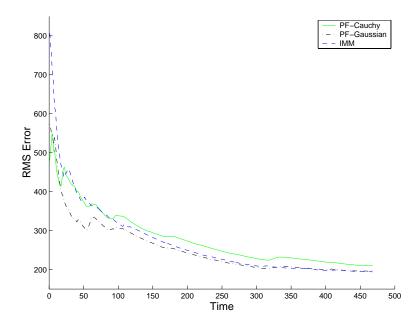
This section will highlight some of the issues faced in implementing tracking systems. The issues covered are:

- · Coordinates.
- Gating issues.

#### 7.7.1 Coordinates

Tracking is normally done in Cartesian coordinates. Cartesian coordinates have the advantage of allowing the use of linear target dynamic models for extrapolation - meaning the predicting and updating of states.

However, sensors normally obtain measurements in azimuth coordinates or a combination of range, azimuths and elevation - for example, most radar sensors will give range and azimuth. A non-linear transformation is required to relate the measurements to the Cartesian state variables. For unbiased non-linear transformations, reader may refer to [161]. However, we may also leave the measurements in the



**Figure 7.10:** RMS error obtained from generic particle filters and IMM filter.

original form and input them through the use of the non-linear measurement form of the extended Kalman filter.

# 7.7.2 Gating issues

Gating is a technique used for eliminating unlikely observation-to-track pairings. A gate is formed about the predicted measurement and all observations that satisfy the gating relationship (falling within the gate) are considered for track updates. The manner in which the observations are actually chosen to update the track depends on the method of data association. For example the nearest neighbour method, global nearest neighbour, probabilistic data association method or multiple hypotheses method. These association methods are discussed in Chapter 6.

The possible gating techniques are:

1. Rectangular gates - Simplest gating technique. It is defined as: an observation (z(k)), with elements  $z_i(k)$  is said to satisfy the gates of a given track if all elements,  $z_i$ , of the residual vector  $\tilde{z}$  satisfy the relationship:

$$|\tilde{z}| \leq K_g \sigma_r$$

where  $\sigma_r = \sqrt{\sigma_o^2 + \sigma_p^2}$ ,  $\sigma_o$  is the measurement variance and  $\sigma_p$  is the prediction variance when  $K_q \geq 3$ .

2. Ellipsoidal gates - A gate (G) is defined such that association is allowed if the following relationship is satisfied by the norm  $(d^2)$  of the residual vector:

$$d^2 = \tilde{z}^T S^{-1} \tilde{z} \le G \tag{7.13}$$

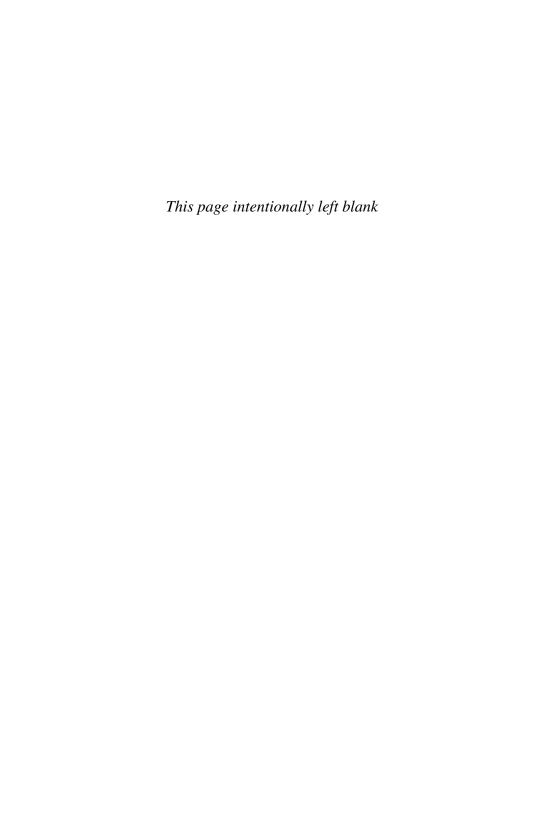
- 3. Multiple gating concepts. Where the starting gate is relatively big and subsequently reduced, such as:
  - first gate large gate, can consider the worst-case target manoeuvre conditions normally 3 times the size of the target maximum speed.
  - second gate small gate, using the maximum speed of the target or the largest eigenvalue of the covariance matrix size.

After gating is the fusion process. Data or measurement plots that fall within the gate will be fused into a single plot and updated during the state estimation process.

## 7.8 Conclusions

The chapter has presented the basic aspects of tracking systems, particularly in the area of kinematics state estimation. The algorithms used for state estimation were also discussed. The advantages and disadvantages of the state estimators (filters) were presented.

A comparison between the IMM filter and Kalman filter was made and showed that the IMM filter is well known for its good performance in estimating the state of a moving target. The IMM filter has been in use in most modern tracking system. However, for fast computational requirements, such as airborne platforms, the adapted fixed gain filters are also used, particularly in burnt-in processors. This type of tracking algorithm is usually embedded or burnt-in in the processor.



## **CHAPTER 8**

# **Cooperative Intelligence**

# 8.1 What is Cooperative Intelligence

Cooperative intelligence refers to a network of nodes <sup>1</sup> working jointly to achieve the desired tasks in the most effective and efficient manner. Each node has the capability to:

- exchange information with other nodes.
- perform information processing.
- manage its resources.
- team with other nodes to achieve its goals.

Cooperative intelligence has been observed in animals and inserts behaviour. For examples, ants are able to work cooperatively in carrying heavy objects and wolves are able to cooperate with each other in hunting for food. Cooperative intelligence could be seen as consisting of two classes, namely:

Lower-order cooperative intelligence or commonly known as swarm intelligence. Studies on swarm intelligence attempt to learn from insects, such as ants and bees, to obtain cooperative intelligent behaviour. These insects may seem to be primitive, however, by working together, they are able to display some kind of evolved intelligent behaviour.

<sup>&</sup>lt;sup>1</sup>A node may exist in a platform, such as ship, aircraft, land vehicle, mobile and static C2 system, robot, personal digital assistant (PDA) or directly within the sensor system. Each node has some form of processing unit to process the data and information. Nodes are sometime also called fusion nodes or sensor nodes.

Swarm intelligence is a shift in mindset: from centralised control to decentralised control and distributed intelligence.

Eric Bonabeau, Icosystem Corp.

• Higher-order cooperative intelligence. Human beings are able to cooperate with one another to achieve multiple desired tasks or goals.

## 8.1.1 Characteristics of cooperative intelligence

Some of the key characteristics of cooperative intelligence are:

- Decentralized control. Nodes are not centrally controlled. Each node has its own 'brain' to process information and makes a decision based on its goals or mission objectives.
- Emergent intelligence. Each node has its own level of intelligence, however, it is often limited. By working together, a network of nodes will have enough emergent intelligence to undertake bigger and more complex tasks that an individual node may not be able to solve. Note that the term collective intelligence has also been used for the same type of behaviour.
- Adaptability. The nodes will have to adapt to the dynamic environment. They do not possess an explicit or complete model of the environment.
- Self-organizing behaviour. Adaptability will also include the characteristic
  of organizing itself and with other nodes. Individual nodes may be detached
  from the group due to a communication breakdown or by being destroyed.
  The remaining nodes can then self-organize to achieve the desired goals. Selforganizing behaviour includes coalition or dynamic teaming among the nodes.
- Reactive behaviour. Each node will exhibit its own reactive behaviour. Although by working together they have a collective intelligence, individual nodes will also have their own local reactive behaviour to counter any immediate threat or time-critical response.

This chapter will focus on the discussion of cooperative intelligence in the areas of cooperative tracking. Section 2 will briefly explain what is cooperative sensing and tracking. Sections 3 and 4 will further explain some of the techniques used for cooperative tracking. Section 5 presents some experiments and findings.

# 8.2 Cooperative Sensing and Tracking

#### 8.2.1 Network of nodes

Here, we would like to engineer a system of cooperative intelligence within a network of nodes where each of the nodes will have:

- a power source.
- a communication link.
- smart/intelligent algorithms.
- direct links to one or more sensor systems (both homogeneous and heterogeneous sensor systems).
- a processor to process the data and information (e.g. handling of data from other nodes and able to perform fusion processes).

With multiple sources of information, each node will perform a fusion process, such as classifying and identifying objects, associating and correlating objects, tracking, monitoring and deriving events or activities (situation assessment), predicting and assessing the threat or intention, and last but not least managing its suite of resources (resource management). Each node will also share its information with other nodes in order to achieve a global picture of the situation under its consideration.

Software agent technology could also be used to improve information processing in the nodes. For example, software agents' negotiation techniques could be used for:

- dynamic allocation of the limited resources in the node.
- coalition of nodes for achieving better global goals.
- managing the data and information to be exchanged about the nodes so as to reduce communication and computing costs.

A network of nodes working cooperatively for sensing and tracking have, but are not limited to, the following advantages:

- Fault tolerance and robustness: failure of one node does not mean the failure
  of the entire network. The network of nodes is still able to sense and track the
  desired target.
- Improved situation awareness: networked nodes (indirectly they can also be seen as a network of sensor systems) can provide better coverage and give different perspectives of information, which results in enhancing situation awareness.

 Scalability: the number of nodes in the network can easily be scaled up for wider and improved coverage. This means that other nodes can join the network seamlessly without loading the individual node's processing capability.

Figure 8.1 shows 4 robots working cooperatively to search and detect the light source. Each of the robots is a node, with PDA as its processing brain, ah-hoc network communication mode and sensors connected to its body. The robots worked cooperatively to perform localization, mapping, tracking and derive awareness of the environment via its sensors and shared information from other robots to achieve the desired tasks in a shorter time.



**Figure 8.1:** Cooperative robots searching for the light source.

Networked nodes naturally are connected in a decentralized architecture. Hence, in tracking, the decentralized tracking algorithms, which enable some form of scalability and methods of improving tracking through sharing of data, will have to be considered.

# 8.2.2 Decentralized tracking

There are 3 approaches for decentralized target tracking. These are:

- Measurement fusion. The state estimation process uses the combination of sensor measurements, or observations directly from sensors, to form target tracks.
- Information fusion. The state estimation process uses the combination of information, where information is data converted from sensor measurements, to form target tracks.
- Track or state vector fusion. The state estimation process uses a combination of individual states, estimated on the same target from each of the sensor systems or nodes, to form target tracks.

We will attempt to discuss in greater detail each of these approaches in the next two sections. However, before we proceed further, we will first standardize some notations and equations that we will be using throughout the chapter.

The standard linear process equation (A.20) describing the target motion at time k is given by

$$x(k+1) = F(k)x(k) + G(k)u(k) + w(k).$$
(8.1)

For simplicity, we shall consider the following process equation that omits the input gain:

$$x(k+1) = F(k)x(k) + w(k)$$
(8.2)

In addition, the measurement equation is given by

$$z(k+1) = H(k+1)x(k+1) + v(k+1)$$
(8.3)

where z(k) is the measurement vector, H(k) is the measurement model or observation matrix and v(k) is the measurement noise.

Note the properties of w(k) and v(k) are:

$$E[w(k)] = 0$$

$$E[v(k)] = 0$$

$$E[w(k)w^{T}(j)] = Q(k)\delta_{kj}$$

$$E[v(k)v^{T}(j)] = R(k)\delta_{kj}$$

$$E[w(k)v^{T}(j)] = 0$$

# 8.3 Measurement and Information Fusion

Theoretically, measurement and information fusion are the same concept. A Kalman filter (KF) is used by both approaches. The Kalman filter is also one of the important

components in tracking algorithms for both centralized and decentralized architectures. Hence, we will start by describing the KF algorithm.

The KF gives the minimum mean square error (MMSE) state estimate if we were to assume a Gaussian distribution for the initial error in the state vector and the noise in the system [18].

At time k, the estimate of the state vector  $\boldsymbol{x}(k)$  based on observations up to time j is

$$\hat{x}(k|j) \equiv E[x(k)|z(1), \dots, z(j)] \quad k \ge j \tag{8.4}$$

with the mean squared error (MSE) of the estimate as

$$P(k|j) \equiv E[(x(k) - \hat{x}(k|j))(x(k) - \hat{x}(k|j))^{T}|z(1), \dots, z(j)]. \tag{8.5}$$

For a system described by equations (8.2) and (8.3), the KF provides a two stage (prediction and update) recursive solution for the estimate  $\hat{x}(k+1|k+1)$  of the state x(k+1), in terms of the estimate  $\hat{x}(k|k)$  and the new observation z(k+1).

Prediction:

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) \tag{8.6}$$

$$P(k+1|k) = F(k)P(k|k)F^{T}(k) + Q(k)$$
(8.7)

Update:

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) 
+ K(k+1)[z(k+1) - H(k+1)\hat{x}(k+1|k)]$$
(8.8)

$$P^{-1}(k+1|k+1) = P^{-1}(k+1|k) + H^{T}(k+1)R^{-1}(k+1)H(k+1)$$
(8.9)

where

$$K(k+1) = P(k+1|k+1)H^{T}(k+1)R^{-1}(k+1)$$
(8.10)

is the Kalman gain matrix.

Note that equation (8.9) gives the information form of the variance. This set of KF equations is similar to the one for the KF used in centralized tracking algorithms.

### 8.3.1 Decentralized Kalman filter

The idea behind the decentralized Kalman Filter is to compute a partial state estimate at each of the nodes in the system, then integrate the various state estimates together to form a global state estimate, via communication with the other nodes. Consider a system with m sensors. The measurement vector

$$z(k) = [z_1^T(k), \dots, z_m^T(k)]^T$$

is partitioned into m subvectors, corresponding to the measurements of each individual sensor, each with dimension  $m_i$ . Similarly, the measurement matrix and

measurement noise vectors are partitioned as follows:

$$H(k) = [H_1^T(k), \dots, H_m^T(k)]^T$$
  
$$v(k) = [v_1^T(k), \dots, v_m^T(k)]^T,$$

with the assumption that the partitions are uncorrelated, that is

$$E[v(k)v^T(k)] = \text{blockdiag}\{R_1, \dots, R_m(k)\}.$$

Decentralization of the KF algorithm is equivalent to a duplicate of the KF algorithm being placed at each of the nodes (or sensor nodes). Thus, the local estimates from the individual nodes in the decentralized KF algorithm are equivalent to the results obtained from a centralized algorithm.

Thus, equations (8.2) and (8.3) become respectively:

$$x_i(k+1) = F_i(k)x_i(k) + w_i(k)$$
 (8.11)

$$z_i(k+1) = H_i(k+1)x_i(k+1) + v_i(k+1),$$
 (8.12)

where  $F_i(k)$  is the local system model and the  $x_i(k)$ 's are states associated with the system.

Then, the prediction and update equations are:

Prediction:

$$\hat{x}_i(k+1|k) = F_i(k)\hat{x}_i(k|k)$$
 (8.13)

$$P_i(k+1|k) = F_i(k)P_i(k|k)F_i^T(k) + Q_i(k)$$
(8.14)

Update:

$$\tilde{x}_{i}(k+1|k+1) = \hat{x}_{i}(k+1|k)$$

$$+ K_{i}(k+1)[z_{i}(k+1) - H_{i}(k+1)\hat{x}_{i}(k+1|k)]$$
 (8.15)
$$\tilde{P}_{i}^{-1}(k+1|k+1) = P_{i}^{-1}(k+1|k) + H_{i}^{T}(k+1)R_{i}^{-1}(k+1)H_{i}(k+1)$$
 where

$$K_i(k+1) = \tilde{P}_i(k+1|k+1)H_i^T(k+1)R_i^{-1}(k+1)$$
(8.17)

with the tilde denoting a partial estimate based only on the observation of the ith sensor node.

Thus, each of the nodes makes an observation and computes a partial (local) state estimate according to equations (8.13 - 8.17). Then, how do we achieve a global state estimate?

### 8.3.1.1 Formulation of a global state estimate

As each node computes the partial estimate  $\tilde{x}_i(k+1|k+1)$ , this information is communicated to the other nodes and it also integrates the information that it receives [73].

There are two parts to the decentralized Kalman filter - the integration of the variance matrix and the state vector.

Through the partition of the observation model and the observation noise, we have

$$H^{T}(k)R^{-1}(k)H(k) = \sum_{i=1}^{m} H_{i}^{T}(k)R_{i}^{-1}(k)H_{i}(k).$$
(8.18)

Rearranging equations (8.9) and (8.16) and substituting the results into (8.18), we obtain the integration equation

$$P^{-1}(k+1|k+1) = P^{-1}(k+1|k) + \sum_{j=1}^{m} [\tilde{P}_{j}^{-1}(k+1|k+1) - P_{j}^{-1}(k+1|k)]$$
(8.19)

which is placed at the individual nodes to obtain

$$P_i^{-1}(k+1|k+1) = P_i^{-1}(k+1|k) + \sum_{j=1}^m [\tilde{P}_j^{-1}(k+1|k+1) - P_j^{-1}(k+1|k)].$$
(8.20)

Similarly, through the partition of the observation vector and model, we have

$$H^{T}(k)R^{-1}(k)z(k) = \sum_{i=1}^{m} H_{i}^{T}(k)R_{i}^{-1}(k)z_{i}(k).$$
(8.21)

Next, we premultiply the rearranged version of (8.9) by P(k+1|k+1) and substitute the result into equation (8.10) to obtain

$$I - K(k+1)H(k+1) = P(k+1|k+1)P^{-1}(k+1|k).$$
(8.22)

We also have

$$I - K_i(k+1)H_i(k+1) = \tilde{P}_i(k+1|k+1)P_i^{-1}(k+1|k), \tag{8.23}$$

which is obtained by performing similar operations, as above, on equations (8.16) and (8.17).

Our next step includes premultiply equation (8.8) (respectively equation (8.15)) by  $P^{-1}(k+1|k+1)$  (respectively  $\tilde{P}_i(k+1|k+1)$ ) and applying equation (8.10) (respectively equation (8.17)) to obtain the following pair of equations:

$$H^{T}(k+1)R^{-1}(k+1)z(k+1) = P^{-1}(k+1|k+1)\hat{x}(k+1|k+1) - P^{-1}(k+1|k)\hat{x}(k+1|k).$$
(8.24)  

$$H_{i}^{T}(k+1)R_{i}^{-1}(k+1)z_{i}(k+1) = \tilde{P}_{i}^{-1}(k+1|k+1)\tilde{x}_{i}(k+1|k+1) - P_{i}^{-1}(k+1|k)\tilde{x}_{i}(k+1|k).$$
(8.25)

These two equations are next substituted into equation (8.21) to obtain the state integration equation

$$\hat{x}(k+1|k+1) = P(k+1|k+1)[P^{-1}(k+1|k)\hat{x}(k+1|k) + \sum_{j=1}^{m} {\{\tilde{P}_{j}^{-1}(k+1|k+1)\tilde{x}_{j}(k+1|k+1) - P_{j}^{-1}(k+1|k)\hat{x}_{j}(k+1|k)\}}]$$

$$(8.26)$$

which is placed at each node in the decentralized network to obtain

$$\hat{x}_{i}(k+1|k+1) = P_{i}(k+1|k+1)[P_{i}^{-1}(k+1|k)\hat{x}_{i}(k+1|k) + \sum_{j=1}^{m} \{\tilde{P}_{j}^{-1}(k+1|k+1)\tilde{x}_{j}(k+1|k+1) - P_{j}^{-1}(k+1|k)\hat{x}_{j}(k+1|k)\}]$$

$$(8.27)$$

Note that the state error information,

$$\tilde{P}_{j}^{-1}(k+1|k+1)\tilde{x}_{j}(k+1|k+1) - P_{j}^{-1}(k+1|k)\hat{x}_{j}(k+1|k)$$

and the variance error information,

$$\tilde{P}_{j}^{-1}(k+1|k+1) - P_{j}^{-1}(k+1|k)$$

are transmitted by each node to the other nodes.

### **8.3.2** Information filter

The information version of the Kalman filter (which is mathematically equivalent to the Kalman filter) allows for the decentralization of the standard continuous estimation and control functions [71] [178].

Given a set of multiple sensor observations,

$$z_i(k) = H_i(k)x(k) + v_i(k), \quad i = 1, ..., N$$
 (8.28)

the global state estimate cannot be obtained by simply summing the weighted innovations that are communicated from other nodes, that is,

$$\hat{x}(k|k) \neq \hat{x}(k|k-1) + \sum_{i=1}^{N} K_i(k)[z_i(k) - H_i(k)\hat{x}(k|k-1)], \tag{8.29}$$

This is because of the correlation of the innovation  $z_i(k) - H_i(k)\hat{x}(k|k-1)$ , generated from each sensor as common information, is shared through the prediction  $\hat{x}(k|k-1)$ . However, expressing in the information form, due to the uncorrelation

of each sensor's information variables  $i_i(k) = H_i^T(k)R^{-1}(k)z(k)$ , estimates can be constructed from linear combinations of the observation information

$$\hat{y}(k|k) = \hat{y}(k|k-1) + \sum_{i=1}^{N} i_i(k).$$
(8.30)

Note that i(k) is the new information about x(k) contained in z(k) with its associated Fisher information matrix I(k), where i(k) and I(k) are defined as follows:

$$i(k) = H^{T}(k)R^{-1}(k)z(k)$$
 (8.31)

$$I(k) = H^{T}(k)R^{-1}(k)H(k)$$
 (8.32)

Rewriting the state estimate and covariance in terms of the new variables, we have

$$\hat{y}(i|j) = P^{-1}(i|j)\hat{x}(i|j)$$
 (8.33)

$$\hat{y}(i|j) = P^{-1}(i|j)\hat{x}(i|j)$$
 (8.33)  
 $Y(i|j) = P^{-1}(i|j).$  (8.34)

Hence, the information filter can be summarized as follows [72]:

Prediction:

$$Y^{-1}(k|k-1) = F(k)Y^{-1}(k-1|k-1)F(k)^{T} + Q(k)$$

$$\hat{y}(k|k-1) = Y(k|k-1)F(k)Y^{-1}(k-1|k-1)\hat{y}(k-1|k-1)(8.36)$$

Estimate:

$$\hat{y}(k|k) = \hat{y}(k|k-1) + i(k) \tag{8.37}$$

$$Y(k|k) = Y(k|k-1) + I(k).$$
 (8.38)

Finally, the estimated state is obtained as

$$\hat{x}(k|k) = Y^{-1}(k|k)\hat{y}(k|k). \tag{8.39}$$

We now make a brief deviation to discuss the advantages and disadvantages of the information filter (IF).

### Advantages

- The equations (8.37) and (8.38) of the IF are computationally simpler as oppposed to the update equations of the KF (Equations (8.8) and (8.9));
- Using the information matrix Y, an estimation process may start with no *a priori* information, that is,  $Y_0$  is a zero matrix. Such a condition will have no bias towards the *a priori* estimate. (Note that this is not possible if the covariance matrix P was initialized as a zero matrix.);
- The observational update of Y is more robust against roundoff errors.

### Disadvantages

- Loss of 'transparency' of the representation as it is more difficult to interpret the physical significance of information;
- Loss of physical significance of the associated state vector components. This is because the coefficients of the linear combinations change with the state of information/uncertainty in the estimates.

In order to decentralize the information filter, as with the Kalman filter, the central IF algorithm needs to be reproduced at each node and the results simplified. Local estimates are first generated at each node by fusing locally available observation information  $i_i(k)$  with locally available prior information  $\hat{y}_i(k|k-1)$ , to yield a local information estimate  $\tilde{y}_i(k|k)$ . The difference between this local estimate and prediction (corresponding to new information gained) is then transmitted to other nodes in the network. If the network is fully connected, each of the other nodes receives the new information. The communicated information is then integrated with the local information. As with the decentralized KF algorithms, the local estimates obtained using the IF algorithms are exactly the same as if the data fusion problem has been solved on a single central processor using the conventional information filter.

### 8.3.2.1 Implementation of the decentralized information filter equation

The derivation of the decentralized information filter equations will not be discussed in this book <sup>2</sup>; instead, we will just provide a summary of the steps required to implement the information filter algorithm.

At each node i, the global information state vector is obtained as

$$\hat{y}_i(k|k) = \hat{y}_i(k|k-1) + \sum_j H_j(k)^T R_j^{-1}(k) z_j'(k), \tag{8.40}$$

where  $z_j^\prime(k)=z_j(k)$  for the linear filter. For the non-linear filter, the expression for z is

$$z_j'(k) = z_j(k) - (h_j[k, \hat{x}_j(k|k-1)] - \nabla_x h_j[k, \hat{x}_j(k|k-1)] \hat{x}_j(k|k-1)).$$
(8.41)

<sup>&</sup>lt;sup>2</sup>Interested readers may refer to [73].

The information matrix is given by

$$P_i^{-1}(k|k) = P_i^{-1}(k|k-1) + \sum_j H_j(k)^T R_j^{-1}(k) H_j(k).$$
 (8.42)

The partial information state vector is given by

$$\tilde{y}_i(k|k) = \hat{y}_i(k|k-1) + H_i(k)^T R_i^{-1}(k) z_i'(k), \tag{8.43}$$

and the partial information matrix by

$$\tilde{P}_i(k|k)^{-1} = P_i^{-1}(k|k-1) + H_i(k)^T R_i^{-1}(k) H_i(k). \tag{8.44}$$

The prediction equations are

$$P_i(k|k-1) = F_i(k)P_i(k-1|k-1)F_i(k)^T + Q_i(k), \tag{8.45}$$

and

$$\hat{y}_i(k|k-1) = P_i^{-1}(k|k-1)F_i(k)P_i(k-1|k-1)\hat{y}_i(k-1|k-1). \tag{8.46}$$

Finally, we have the partial state estimate

$$\tilde{x}_i(k|k) = \tilde{P}_i(k|k)\tilde{y}_i(k|k), \tag{8.47}$$

and the global estimate

$$\hat{x}_i(k|k) = P_i(k|k)\hat{y}_i(k|k).$$
 (8.48)

### **8.3.3** Measurement Fusion

The measurement fusion algorithm is suitable for the centralized architecture. It is also possible to be used in a decentralized architecture when network transmission costs can be reduced by other technologies. In the measurement fusion algorithm, the sensors' measurements are fused to obtain a combined measurement, and then to estimate the target state by the combined measurement. Here, we use Kalman filter to illustrate the measurement fusion algorithm.

The tracking is modeled by the following transition equation and measurement equation:

$$x(k+1) = F(k)x(k) + w(k)$$
(8.49)

$$z_i(k+1) = H_i(k+1)x(k+1) + v_i(k+1)$$
(8.50)

where i is the sensor index, v(k) and  $w_i(k)$  are white Gaussian noise with covariance Q and  $R_i$  respectively.

Assume there are N independent measurements from N sensors associated to the same target. The combined measurement z and its error covariance R are given by:

$$z(k) = R(k) \sum_{i=1}^{N} R_i^{-1}(k) z_i(k)$$
(8.51)

$$R(k) = \left[\sum_{i=1}^{N} R_i^{-1}(k)\right]^{-1}$$
(8.52)

and the combined measurement matrix H is:

$$H(k) = R(k) \sum_{i=1}^{N} R_i^{-1}(k) H_i(k)$$
(8.53)

The Kalman filter prediction and estimation by the combined measurement are shown as follows:

Prediction:

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k)$$
 (8.54)

$$P(k+1|k) = F(k)P(k|k)F^{T}(k) + Q(k)$$
(8.55)

Innovation covariance:

$$S(k+1) = H(k+1)P(k+1|k)H^{T}(k+1) + R(k+1)$$
(8.56)

Kalman gain:

$$W(k+1) = P(k+1|k)H^{T}(k+1)S(k+1)^{-1}$$
(8.57)

**Estimation:** 

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1)(z(k+1) - H(k+1)\hat{x}(k+1|k))$$
(8.58)

$$P(k+1|k+1) = [I - W(k+1)H(k+1)]$$

$$P(k+1|k)$$
(8.59)

# 8.4 Track or State Vector Fusion

The track fusion approach combines tracks, where a track is the state vector of a target. Hence, track fusion is also commonly known as state vector fusion. In the following subsections, we will provide descriptions of the main approaches to track fusion and their underlying assumptions. The primary methods are [56]:

- Convex combination (CC).
- Covariance intersection (CI).
- Bar-Shalom and Campo's combination (BC).
- Information de-correlation.
- Best linear unbiased estimate (BLUE) or also known as linear minimum variance estimate.

We consider the state to be estimated as a finite dimensional random vector x with mean  $\bar{x}$  and covariance  $\bar{P}$ .

### **8.4.1** Convex combination

The convex combination (CC) method assumes that the cross covariance between two state estimates,  $\hat{x}_i$  and  $\hat{x}_j$ , can be ignored. The equations to combine two state vectors and its covariance are as follows:

Error Covariance: 
$$P = (P_i^{-1} + P_j^{-1})^{-1}$$
 (8.60)

State Estimate: 
$$\hat{x} = P(P_i^{-1}\hat{x}_i + P_j^{-1}\hat{x}_j)$$
 (8.61)

The general equations for n state estimates are:

Error Covariance: 
$$P = (\sum_{k=1}^{n} P_k^{-1})^{-1}$$
 (8.62)

State Estimate: 
$$\hat{x} = P(\sum_{k=1}^{n} P_k^{-1} \hat{x}_k)$$
 (8.63)

The convex combination method is simple to implement, hence it is also commonly known as simple convex combination. However, this method is optimal only if the assumption is true, meaning the cross covariance  $P_{ij}=0$  and  $R_{ij}=0$ ; else the method is sub-optimal.

If the same Kalman filter is applied to all the local tracks in track fusion, and all local kalman filters use the same process noise covariance Q and the same measurement noise covariance R, then, this implementation, will result in all the local tracks to generate the same state error covariance P, after the Kalman filter is stable. This means that a set of individual tracks with identical error covariance are going to be fused together in track fusion. The combined state vector becomes the average of all the individual state vectors. This result is that good and bad tracks (based on track quality) make an equal contribution to the combined track. Thus, it is necessary to develop a track selection algorithm to eliminate tracks with poor track quality.

The track quality can be computed based on the normalized distance function [111]:

$$d^{2}(k+1) = v^{T}(k+1)S(k+1)^{-1}v(k+1)$$
(8.64)

where the measurement residual v(k+1) is:

$$v(k+1) = z(k+1) - H(k+1)F(k)\hat{x}(k|k)$$
(8.65)

and the innovation covariance S(k+1) is:

$$S(k+1) = H(k+1)P(k+1|k)H^{T}(k+1) + R(k)$$
(8.66)

Then, the track quality U(k+1) is computed by:

$$U(k+1) = \alpha U(k) + (1-\alpha)d^2(k+1)$$
(8.67)

where  $\alpha$  is the weight on the history.  $\alpha$  can be chosen from 0 to 1. The value of U represents the track quality. The smaller the U, the better the quality.

Next step is to eliminate those tracks whose qualities are relatively poor. The smallest  $U_{min}$  (the best quality) is selected among the individual tracks, and the relative quality  $A_i$  of the track i is computed by:

$$A_i(k+1) = \frac{U_i(k+1)}{U_{min}(k+1)}$$
(8.68)

Only tracks that meet the following condition are selected to fuse:

$$A_i(k+1) < T_a \tag{8.69}$$

where  $T_a$  is the threshold.  $T_a$  is a number greater than 1.

### 8.4.2 Covariance intersection

The covariance intersection (CI) algorithm can be viewed as a weighted form of convex combination in a state estimation. It is suitable for track fusion when the tracks have an unknown correlation. The CI algorithm consists of the following:

Error Covariance: 
$$P = [\omega P_i^{-1} + (1 - \omega) P_j^{-1}]^{-1}$$
 (8.70)

State Estimate: 
$$\hat{x} = P[\omega P_i^{-1} \hat{x}_i + (1 - \omega) P_j^{-1} \hat{x}_j]$$
 (8.71)

where  $0 \le \omega \le 1$ .

The parameter  $\omega$  represents the weight that is assigned to the estimates  $\hat{x}_i$ . It is often chosen such that the determinant or trace of P is minimal. Since the trace of

the covariance matrix P provides a scalar measure of the estimation uncertainty of  $\hat{x}$ ,  $tr(P_j)/tr(P_i) \to 0$  implies  $\omega \to 0$ . When  $P_i = P_j$ , the parameter  $\omega$  is independent of P, the  $\omega$  can be chosen as 0.5 [179].

There is an alternative method to find the  $\omega$ . If we assume that the performance of the sensor improves when the target is nearer to the sensor, then one method for computing the  $\omega$  is the relationship between target and sensor distance. This is shown as follows:

$$\omega_i = d_j/(d_i + d_j) \tag{8.72}$$

$$\omega_j = d_i/(d_i + d_j) \tag{8.73}$$

where  $d_i$  is the distance from the target to sensor i and  $d_j$  is the distance from the target to the sensor j.

In order to extend the CI algorithm to the fusion of an arbitrary n state estimates, where n > 2, we can either recursively apply equations (8.70) and (8.71) to the estimates or apply the following directly:

Error Covariance: 
$$P = \left[\sum_{i=1}^{n} \omega_i P_i^{-1}\right]^{-1}$$
 (8.74)

State Estimate: 
$$\hat{x} = P\left[\sum_{i=1}^{n} \omega_i P_i^{-1} \hat{x}_i\right]$$
 (8.75)

where  $\sum_{i=1}^{n} \omega_i = 1$ . The CI method is simple to implement if the  $\omega$  can be computed.

### 8.4.3 Best linear unbiased estimate

The best linear unbiased estimate (BLUE) is given by:

Error Covariance: 
$$P = \bar{P} - V_{x\hat{z}}V_{z\hat{z}}^{-1}V_{x\hat{z}}^{T}$$
 (8.76)

State Estimate: 
$$\hat{x} = \bar{x} + W_1(\hat{x}_i - \bar{x}) + W_2(\hat{x}_j - \bar{x})$$
 (8.77)

where

$$\hat{z} = \begin{bmatrix} \hat{x}_1^T & \hat{x}_2^T \end{bmatrix}$$

$$\begin{bmatrix} W_1 & W_2 \end{bmatrix} = V_{x\hat{z}}V_{\hat{z}\hat{z}}^{-1}$$

$$V_{x\hat{z}} = \text{covariance betwen } x \text{ and } \hat{z}$$

$$V_{z\hat{z}} = \text{covariance of } \hat{z}.$$

# 8.4.4 Bar-Shalom and Campo combination

Bar-Shalom and Campo combination method is a special case of BLUE. It combines the filtered state vectors (derived from sensors' measurements) from other nodes in order to form a new estimate. This method is based on the track correlation on the same process noise [13]. The multiple tracks can be combined as follows [15]:

Error Covariance: 
$$P = P_i - (P_i - P_{ij})(P_i + P_j - P_{ij} - P_{ji})^{-1}(P_i - P_{ji})$$
 (8.78)

State Estimate: 
$$\hat{x} = \hat{x}_i + (P_i - P_{ij})(P_i + P_j - P_{ij} - P_{ji})^{-1}(\hat{x}_j - \hat{x}_i)$$
 (8.79)

where the cross covariances  $P_{ij}$  is given by

$$P_{ij}(k|k) = [I - K_i(k)H_i(k)][F(k-1)P_{ij}(k-1|k-1)F^T(k-1) (8.80) +Q(k-1)][I - K_j(k)H_j(k)]^T$$

Let us consider fusion of estimates from N sensors [15]. We denote the state estimates and error covariance from sensor i respectively as  $\hat{x}_i$  and  $P_i$  and the cross covariance between sensors i and j as  $P_{ij}$ . The state estimate and error covariance are thus

Error Covariance: 
$$P = (\mathbf{I}^T \mathbf{P}^{-1} \mathbf{I})^{-1}$$
 (8.81)  
State Estimate:  $\hat{x} = (\mathbf{I}^T \mathbf{P}^{-1} \mathbf{I})^{-1} \mathbf{I}^T \mathbf{P}^{-1} \hat{\mathbf{X}}$  (8.82)

State Estimate: 
$$\hat{x} = (\mathbf{I}^T \mathbf{P}^{-1} \mathbf{I})^{-1} \mathbf{I}^T \mathbf{P}^{-1} \hat{\mathbf{X}}$$
 (8.82)

where

8.4.5

:  $n \times n$  identity matrix, with n the length of each state

:  $[I, I, \dots, I]^T (Nn \times n \text{ matrix})$ 

 $\hat{\mathbf{X}}$  :  $[\hat{x}_1, \hat{x}_2, \dots, \hat{x}_N]^T$ .

 $Nn \times Nn$  covariance matrix with blocks

$$\mathbf{P} = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1N} \\ P_{21} & P_{22} & \dots & P_{2N} \\ & & & \ddots & & \\ P_{N1} & P_{N2} & \dots & P_{NN} \end{bmatrix}$$

# **Information de-correlation**

Assuming that  $R_{ij} = 0$  but  $\bar{P}^{-1} \neq 0$ , the error covariance and the state estimate obtained from the fusion of two state estimates are given respectively by

Error Covariance: 
$$P = (P_i^{-1} - \bar{P}^{-1} + P_j^{-1})^{-1}$$
 (8.83)

State Estimate: 
$$\hat{x} = P(P_i^{-1}\hat{x}_i - \bar{P}^{-1}\bar{x} + P_j^{-1}\hat{x}_j)$$
 (8.84)

where  $\bar{x}$  is the common prior information shared by the estimates, if  $\hat{x}_i$  and  $\hat{x}_j$  are any two arbitrary estimates that are to be fused.

Note that this method can be extended to arbitrary fusion architectures if the communication history is known and correlation in the estimation errors is due to prior information.

# 8.5 Experiments on Decentralized Tracking

This section describes the experiment to test the decentralized tracking algorithms in a network of nodes. The experiment is carried out with the objective of comparing the various decentralized tracking algorithms' performance in a network of nodes.

# 8.5.1 Target and sensor modelling

The target is generated using the 2-D constant velocity model, and the measurements in range and azimuth are converted to Cartesian coordinates unbiased using [161]:

$$x_m = \lambda_b^{-1} r_m cos b_m \tag{8.85}$$

$$y_m = \lambda_b^{-1} r_m sinb_m \tag{8.86}$$

$$\lambda_b = e^{-\sigma_b^2/2} \tag{8.87}$$

where  $\lambda_b$  is the bias compensation factor,  $r_m$  is the measured range,  $b_m$  is the measured azimuth or bearing, and  $\sigma_b^2$  is the error variance of the measured azimuth, which is set to 4 in these experiments. .

The state vector x in the simulation is defined as  $[x, y, \dot{x}, \dot{y}]^T$ , and measurement vector z is  $[x, y]^T$ . The state transition equation and the measurement equation are given by:

$$x(k+1) = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{bmatrix} a(k)$$
(8.88)

$$z(k+1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x(k+1) + v(k+1)$$
 (8.89)

where a(k) is  $[\ddot{x}, \ddot{y}]^T$  at time k. It is the process noise used to handle uncertain acceleration in the constant velocity transition model. In our simulation, a(k) is set to  $[10, 10]^T$ . w(k) is the measurement noise, and its covariance R is set to 160000.

### 8.5.2 Scenario and simulation test

The target flight profile and the nodes placement for the experiment are shown on Figure 8.2. The target moves from northwest to the southeast, and then makes a right turn, and moves from north to south, and finally moves from northwest to southeast again after a left turn. The target speed is 100 m/s, and turning rate is  $5^{\,o}/\text{s}$ . The radar measurement noises are simulated by white Gaussian distribution with standard deviation of 100 m and  $2^{\,o}$  for range and azimuth, respectively.

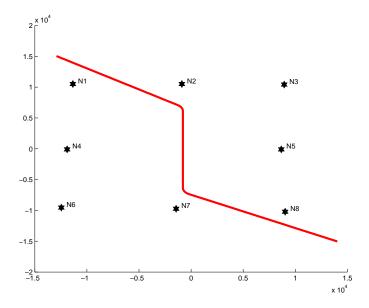


Figure 8.2: Target flight profile and the nodes placement for the experiment.

Four simulation tests were carried out. These tests are:

- 1) Two nodes sharing data, namely node N1 and N8.
- 2) Four nodes sharing data, namely node N1, N3, N6 and N8.
- 3) Six nodes sharing data, namely N1, N2, N3, N6, N7 and N8.
- 4) Eight nodes sharing data, namely N1, N2, N3, N4, N5, N6, N7 and N8.

# 8.5.3 Experiment results

Table 8.1 shows the RMS position errors for five decentralized tracking algorithms for 100 runs. Figures 8.3, 8.4, 8.5 and 8.6 show the RMS position errors versus time. It is quite obvious that the sharing of data among the nodes for the combined tracking gives less RMS errors compared to an individual node tracker.

| Node    | Algorithm | RMS error    | Noise        |  |
|---------|-----------|--------------|--------------|--|
|         |           | ( <b>m</b> ) | reduction(%) |  |
| 2 nodes | -         | 654.78       | Noise        |  |
|         | CC        | 193.92       | 70%          |  |
|         | BC        | 193.92       | 70%          |  |
|         | CI        | 122.95       | 81%          |  |
|         | IF        | 202.55       | 69%          |  |
|         | MF        | 202.30       | 69%          |  |
| 4 nodes | -         | 660.93       | Noise        |  |
|         | CC        | 147.08       | 78%          |  |
|         | BC        | 147.08       | 78%          |  |
|         | CI        | 106.63       | 84%          |  |
|         | IF        | 157.75       | 76%          |  |
|         | MF        | 157.80       | 76%          |  |
| 6 nodes | -         | 623.32       | Noise        |  |
|         | CC        | 119.64       | 81%          |  |
|         | BC        | 119.64       | 81%          |  |
|         | CI        | 87.38        | 86%          |  |
|         | IF        | 128.06       | 79%          |  |
|         | MF        | 128.24       | 79%          |  |
| 8 nodes | -         | 605.45       | Noise        |  |
|         | CC        | 105.57       | 83%          |  |
|         | BC        | 105.57       | 83%          |  |
|         | CI        | 82.79        | 86%          |  |
|         | IF        | 111.51       | 82%          |  |
|         | MF        | 111.52       | 82%          |  |

**Table 8.1:** Comparison of decentralized tracking algorithms

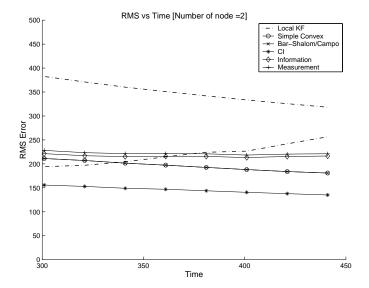


Figure 8.3: Position error vs time in two nodes.

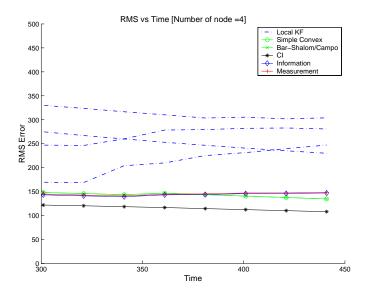


Figure 8.4: RMS error vs time in 4 nodes.

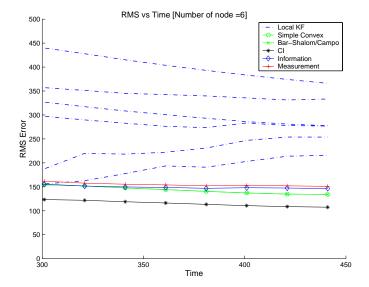


Figure 8.5: RMS error vs time in 6 nodes.

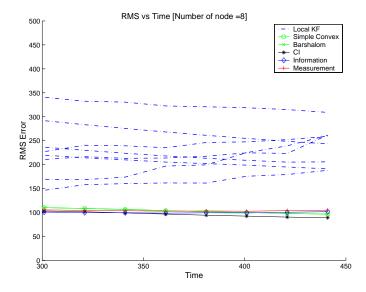


Figure 8.6: RMS error vs time in 8 nodes.

| Node    | Track     | RMS error    | Noise        |  |
|---------|-----------|--------------|--------------|--|
|         | selection | ( <b>m</b> ) | reduction(%) |  |
|         | (Yes/No)  |              |              |  |
| 2 nodes | =         | 654.87       | Noise        |  |
|         | No        | 193.92       | 70%          |  |
|         | Yes       | 141.99       | 78%          |  |
| 4 nodes | -         | 660.93       | Noise        |  |
|         | No        | 147.08       | 78%          |  |
|         | Yes       | 122.86       | 81%          |  |
| 6 nodes | -         | 623.32       | Noise        |  |
|         | No        | 119.64       | 81%          |  |
|         | Yes       | 101.01       | 84%          |  |
| 8 nodes | -         | 605.45       | Noise        |  |
|         | No        | 105.57       | 83%          |  |
|         | Yes       | 97.65        | 84%          |  |

Table 8.2: Track selection method test result

We also test our track selection method on the simple convex combination algorithm for 100 runs. The test results are shown on Table 8.2. We can see that the RMS position errors are further reduced by the track selection algorithm.

The experiment shows that the network of nodes can improve the tracking performance using decentralized tracking algorithms. Use of information from other nodes helps to improve the tracking performance compared with a node that only uses its own local sensor information. However, the RMS error cannot be reduced infinitely by increacing the number of nodes. The saturating state is not investigated in our simulation. The measurement and information fusion performance results are about the same. The covariance intersection algorithm has less RMS error when reasonable  $\omega$  values are chosen. The  $\omega$  values are approximated, based on the distance of the sensor to the target.

The issue of selecting tracks for combination, presents a track selection method based on the track quality. Simulation results show the method could further reduce the RMS error.

# 8.6 Summary

This chapter briefly introduces the concepts of cooperative intelligence. The uses of cooperative intelligence in decentralized tracking were discussed in detail. Various decentralized state estimation fusion algorithms were also presented. A number of these algorithms are currently still undergoing testing and verification with regard to their usefulness in decentralized fusion. The investigations include:

- Bandwidth. How much reduction in bandwidth or amount of communication messages needs to be sent across the node to achieve good track fusion.
- Computational complexity. A node may be a small electronic gadget on board a moving platform and hence has limited processing power due to space and battery constraints.
- Selective fusion. This concern issues such as when to fuse and when not to fuse, to obtain the best track accuracy.

# **CHAPTER 9**

# Sensor Management and Control

### 9.1 Introduction

Sensor management is an important area of technology for modern tactical sensor systems and multisensor fusion systems. Sensor management and controlling of sensors are discussed in [43, 67, 100, 156, 177, 174, 175, 176, 191] and are also considered as part of the multisensor fusion process.

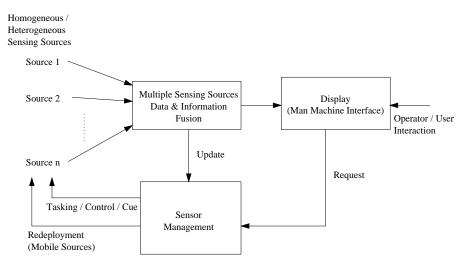
The task of sensor management work is challenging due to [169, 156]:

- Insufficient resources.
- Limited sensor capabilities.
- Highly dynamic environment.
- Varied sensor capabilities and performance.
- Limited processing capabilities.
- Unplanned sensor failures.
- Data and information fusion requirement.

In [216, 187], the authors highlighted the need for the pilot to be occupied with other activities and leave the control of sensors to a machine. Thus, the aircraft must have a sensor manager that is able to select, prioritize, specify, distribute and schedule tasks for the sensor suites mounted in the flying platform. In addition to this scenario, we also see many other examples, such as sensors on land vehicle [61]

or multi-site sensor systems, where there is need for a sensor management system (SMS) to aid the complete operation of the sensors.

We view the SMS working hand in hand with the data and information fusion process and the sensor systems (or the generic term sensing sources). The following diagram shows the complete closed-loop sensing system framework (see Figure 9.1).



**Figure 9.1:** A closed-loop sensing system framework.

The goal of SMS can be defined as to manage, coordinate and integrate the sensor usage to accomplish specific and often dynamic mission objectives. The word 'manage' gives a sense of control over the sensors, 'coordinate' brings out the efficient use of the sensors and 'integrate' brings together all the sensors into one family or combining all the sensor systems. The ultimate goals of the SMS are to optimize the overall performance of the fusion systems. This includes the tracking system [30] and the classification/identification system. Examples of mission objectives are:

- to identify a low flying target in area A.
- to track a target in area B.

The mission objectives could be predefined or entered dynamically in real time. In fulfilling these goals, the sensor management in effect will help to improve the process of data and information fusion.

The basic of the sensor manager's job [60, 167] is to 'select the right sensor to perform the right service on the right object at the right time'. In other words, it seeks to answer the following questions [67]:

- What sensor or what group of sensors?
- Which service or which mode or task?
- Where to point or how to control the sensor?
- When to start?

While it is true that sensor management was borne from the task of performing resource allocation and scheduling of tasks, with the current advances in sensor technology, its role and function have expanded. We can classify its role and function in a more systematic and well-defined structure.

The chapter is organized as follows. Section 2 discusses the roles and function of sensor management in a multi-level approach. Section 3 presents the sensor management techniques. Section 4 discusses the sensor management in the control aspect. Section 5 demonstrates sensor management using a fuzzy controller and a cueing process. Section 6 briefly introduces the use of software agents for decentralized sensor management. We conclude at section 7.

# 9.2 Multi-level Sensor Management Based on Functionality

The sensor management can be classified based on its functionality [191, 30]. Most of the discussions on sensor management have been centred on the architectural design such as centralized, decentralized and hierarchical (micro and macro) design [227]. However, the discussion on this design can be further improved or better understood if we can breakdown the different levels of sensor management based on functionality. As in the data and information fusion process, sensor management can be classified into the following 3 levels:

- Level 1 This is the lowest level of work in sensor management involving individual control of each sensor, such as direction, pointing, change in frequency and power level. For example, in a directed radar where the tracking of the target can be guided by a SMS. The SMS is then a controller to the sensor system. Such SMS can be built closely coupled with the sensor system and the fusion process. The goal of this SMS controller is to reduce the tracking error, based on the fusion input. We will discuss this in greater detail in the subsequent section.
- Level 2 This is the medium level sensor management. At this level the SMS focuses more on the sensor tasks and different modes of the sensor with respect to the operational needs. SMS will work out the method to prioritize the tasks and determine when and how a mode should be activated. Besides these,

the SMS can also look at some basic functions of sensor to sensor switching. Hence, at this stage we look at:

- Sensor task scheduling.
- Sensors cueing (sensor handing over, target acquisition by another sensor aiding).
- Sensor modes/functions (sensor's own modes changing process).
- Level 3 This can be seen as the higher-level of sensor management. Each
  sensor can only detect a limited amount of information. However, the sensor
  manager can direct several sensors in an integrated fashion or sensor mix to
  supply information to the fusion process. The total information will provide
  greater content than the information from a single sensor. Examples of the
  tasks at this level are:
  - Dynamic sensor placement or deployment (e.g. to provide good coverage).
  - Effective/optimal sensor mix (e.g. to minimize the emission).

Sensors can be placed on a moving platform, such as ground vehicle or aircraft. A SMS can coordinate, manage or suggest positions for the sensor and its platform to best view the target of interest, provide better coverage space for detection and avoid being seen. The management of sensors will take into account the operating environment, such as the terrain.

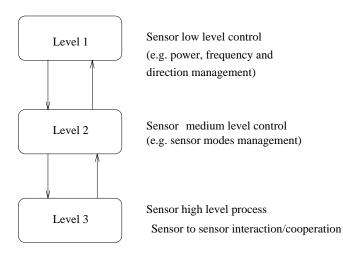


Figure 9.2: Sensor management levels based on functionality.

Figure 9.2 shows the concept of the various sensor management levels. We may not have exhausted all possible tasks within each level. Note that the discussion of

the sensor management levels are intended only for better understanding and convenient categorization of the SMS's functionality. It is not intended as a prescription for designing a system.

Figure 9.3 shows a general breakdown of the various sensor management functional modules.

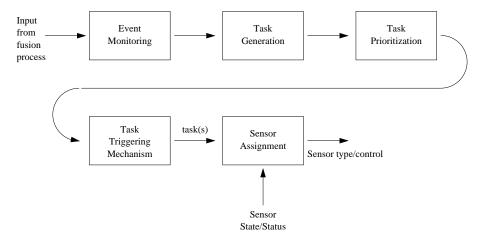


Figure 9.3: Sensor management functional modules.

# 9.3 Sensor Management Techniques

A variety of sensor management techniques have been proposed or applied through the last decade, where researchers and engineers tried to seek better ways of solving the illusive sensor management work. However, the implementation of sensor management techniques is viewed by most researchers as solving either, if not both, of the following problems:

- Decision-making.
- Scheduling.

Decision-making techniques for SMS aim at answering the question: Which sensor tasks are the most important to schedule or carry out, that is, to prioritize the tasks. On the other hand, scheduling techniques are used to generate a list or sequence of tasks to be assigned to the sensors, placing emphasis on the time line of each sensor. Note that the resource allocation techniques such as assignment methods and deployment strategies are also part of sensor management process. However, these techniques are not discussed in this chapter.

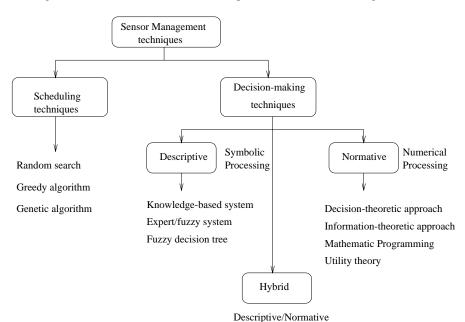


Figure 9.4 classifies the various techniques used for sensor management. In each

Figure 9.4: Sensor management techniques.

of the following two subsections, we will present and discuss the main ideas of the techniques described and proposed in the published literature.

# 9.3.1 Scheduling techniques

In [191], two general approaches for scheduling sensor tasks are discussed. Tasks are described as time critical 'hard deadline tasks' (tasks that are of value only if performed within a small window of opportunity), 'fluid deadline tasks' that are not time critical but have weak constraints, and also tasks with different priority. An example of each of the above tasks is presented below:

- ullet Brick packing approach. The basic idea is to break the scheduling time line, T, into smaller tasking intervals  $\Delta T$ . This time interval is then packed according to some measure of optimality that involves the solving of epoch goals by exhaustive tree searching. It is applied to find the schedule that contains the most tasks or by a sub-optimal approach that uses a heuristically guided depth-first tree search to select specific tasks.
- ullet Best first approach. As the name suggests, this approach produces a queue of priority-ordered fluid tasks in every  $\Delta T$  and a separate list of time-ordered

hard deadline tasks. This approach is preferred for solving macro-level sensor management tasks.

In [25], the authors describe a greedy service algorithm (GSA) that assigns services to sensors. This algorithm assumes that services are greedy in the sense that the highest priority service will always demand to be assigned to the 'best' available sensor to perform the service. It is essentially a depth-first search of a subset of all possible sensor assignments. The GSA approach has two significant advantages over other algorithms for the creation of sensor task schedules. First, only those nodes for sensors and services that can be allocated are expanded. Second, since the search is depth first, no backtracking is ever performed; resulting in a significant reduction in search time and memory requirements.

Rothman and Bier [199] describe two techniques. These are:

- Random allocation. Selections are made from a list randomly, assuming that
  the set of tasks that the sensor manager can perform during a given cycle is
  available in this list, and randomly allocated to sensors which can perform the
  services. This is a very basic algorithm, easily outperformed by any other.
- Shortest job-first allocator. Allocates the jobs that will take the least amount of time on any available sensor first. An important characteristic of this approach is that it can be shown to provide the minimum turnaround time of any resource algorithm.

Musick and Malhotra [167] give an in-depth comparison of two techniques:

- Random search algorithm. An undirected process that attempts to find schedules using the Monte Carlo method. Generates a schedule as a list of jobs. The strength of this approach is its simplicity. The random scheduler has no memory of previous schedules. It is very robust.
- Genetic search algorithm. Similar to the random scheduler, except that the
  genetic algorithm remembers its successes and failures. While random search
  is a blind search, the genetic algorithm is a heuristically guided search. This
  algorithm's strength is that it uses reliable schedules to produce better ones. It
  is also robust.

In [250, 157], a dynamic sensor scheduling algorithm called the on-line, greedy, urgency-driven, pre-emptive scheduling algorithm (OGUPSA), is introduced. This algorithm uses three policies successively to dynamically allocate and schedule and distribute a set of arriving tasks among a set of sensors. These three policies are:

- Most-Urgent-First to pick a task.
- Earliest-Completed-First to select a sensor, and

Least-Versatile-First to resolve ties among sensors, and if a tie still exists, then
a random selection is used to break it.

An important component of OGUPSA is the information contained in an applicable sensor table that is used to assign requested tasks to specific sensors.

# 9.3.2 Decision-making techniques

Decision-making techniques for SMS can be divided into two subclasses, these are:

- Normative technique.
- Descriptive technique.

### 9.3.2.1 Normative technique

As Popoli [191] put it, a normative technique produces decisions based on some axiomatic description of general human decision-making and specific *a priori* data relevant to the decision at hand. In short, a normative approach gives answers that a human **should** give in a similar situation. A major advantage of normative approaches is that they lead to systems for which performance criteria can be analyzed. The performance of a normative system can be guaranteed with respect to a defensible criteria (related to the axiomatic bases used for the decision). This technique includes utility theoretic reasoning and evidential reasoning.

The authors in [91] consider the sensors of a multi-sensor system to be members or agents of a team, able to offer opinions and bargain in group decision. The authors presented their analysis on the coordination and control of this structure using the theory of team decision-making. They introduced the term utility function that combines local and global objectives by considering team member utilities and joint team actions.

The authors in [150] adopt a normative approach which is based on axiomatic formalization of the decision making process and present a complete and consistent probabilistic framework which, starting from a Bayesian information update, leads to fusion algorithms and subsequently sensor management. They propose using information as the expected utility of actions.

### 9.3.2.2 Descriptive technique

A descriptive technique tries to produce decisions that match what a human would have decided based on available information. In short, a descriptive approach gives answers a human **would** give in a similar situation. A major advantage of descriptive approaches is that they are amenable to situations for which either objectively-verifiable information is not available or to situations for which there is a strong

motive to make use of information that clearly is subjective. This technique includes fuzzy reasoning, fuzzy decision tree [191, 216] and knowledge-based approaches.

The essence of a descriptive approach is that the goal generally is to mimic expert human behaviour for a particular problem domain. We simply hope to capture the behaviour of a human faced with the same problem. The approaches differ primarily in their definition of how a problem should be solved rather than in the information used to solve it.

The thrust of normative approaches is to establish a formal decision-making criterion and remain faithful to it; the thrust of descriptive approaches is to provide a system by which any human decision-making behaviour (whether or not it is formally consistent) can be easily codified. Descriptive approaches can be constructed on a weaker problem domain description than is required by a normative approach.

### 9.3.2.3 Hybrids of normative and descriptive techniques

As Denton *et al.* [67] put it, there is no single mathematical formalism that can be effectively used to implement all of the required sensor management functions.

The authors in [25] discuss the issue that the choice of the algorithm used in the determination of the 'best' sensor to perform a given service depends upon the service in question. For example, the selection of the best sensor to perform a kinematics track update employs a utility theory approach in which a numerical utility for each sensor able to perform the update is computed, and the sensor with the highest utility is selected to perform the service. On the other hand, the selection of the best sensor for the identification of services is performed by using a series of 'If ... Then' rules.

### 9.3.2.4 Other decision-making tools

The author in [42] discusses the use of a neural network for sensor management. The task of the neural network is to predict the condition of the plant. This information can then be used to estimate the sensed data tolerance ranges at each operating condition for improved fault detection and isolation, hence help to improve the model estimation of sensor values. The study has shown that a neural network is effective at both recognizing the plant condition and in exploiting data redundancy, an intrinsic challenge in the interdependencies of the sensed plant variables.

Hintz and McIntyre [100] present a methodology for quantifying the relative contribution of specific sensor actions to a set of mission goals. They call the approach goal lattices, since the mission goals and the ordering relationship applied to it can be represented as a lattice.

# 9.4 Sensor Management - Controller and Cueing

As highlighted in section 2, level 1 of the sensor management process involves controlling of the sensor action. Hence, conventional and advance control techniques may play a part in sensor management's work at this level.

However, the process of control in SMS is not trivial [148]. Depending on the system involved, the SMS can be complicated by many factors, such as:

- Moving platforms.
- Geographical differences in sensor locations.
- Complexity in the fusion process.
- Complexity and different types of sensor systems.

The task of the controller can have many different functions or roles, such as controlling the:

- Sensor directions/pointing to best locate/identify the moving target.
- Sensor mode and when to switch mode.
- Sensor power level.

All the control tasks have the same objective, namely to improve the fusion process, such as in tracking and identifying the target. Figure 9.8 shows an example of a possible SMS behaving as a controller in a closed-loop form with the sensor system and the fusion process.

With sensor manager providing command and control to the sensors, the sensor system will provide the fusion system with measurements, such as track data, and in turn the data fusion algorithm will provide the sensor manager with fused data.

However, the SMS is certainly more than just a controller. As discussed in section 2, there are different levels of processes in the SMS. One of this is the sensor cueing process.

Here, we define sensor cueing as taking the following two forms, namely:

- · sensor hand-over and
- target acquisition with another sensor aiding.

The SMS may cue a sensor to hand over its current tracking target to another sensor due to, say,

• another higher priority target entering its trackable region, or

• when the target is about to leave its trackable region.

The issue here is that SMS has to coordinate the handing over to ensure a smooth transition and to reduce tracking or identification errors. On the other hand, a cued target acquisition [30] process involves the SMS selecting which sensor to acquire the target from and disseminating to that sensor the information required for such acquisition. For example, the SMS may cue one of its radars to acquire range information of a target that is being maintained primarily by an EO sensor (bearing only).

Sensors vary greatly in their ability to detect the presence of targets or the onset of a particular target behaviour. Therefore, sensor management may improve the response time of a sensor by a proper cueing process, hence obtaining information from another sensor. Multiple sensors performance modelling or some expert rules can be used for cueing to ensure no lost of target information.

In the next section, we demonstrate the SMS as a complete system based on what we have discussed in this section.

### 9.5 Simulation Results

In this experiment, we considered a 20km range (probability of detection 0.90) ground-based EO sensor with a field of view (FOV) of 15 degrees, and capable of a panning movement of 120 degrees. The tracking process takes place over a period of 90 seconds.

Without the SMS, a normal EO sensor tracking a target will probably move at a fixed panning rate, following the heading of the target.

In this simulation, the target moves at a constant velocity (20m/s) within the first 20 seconds, accelerates  $(5m/s^2)$  between 21 and 50 seconds and decelerates  $(-5m/s^2)$  from 51 and 90 seconds. Mid-way through the deceleration process, the target makes a right turn (time interval 65 to 73 seconds). An IMM filter is used in the fusion process to track the target. The estimated target state is then sent to the sensor management system.

# 9.5.1 Fuzzy controller

The fuzzy controller used in the SMS has two inputs, namely error (k) and error (k-1), where error (k) represents the bearing error, in degrees, of the sensor boresight to the target at the kth time step. Note that this  $\operatorname{error}(k)$  is obtained by taking the estimated bearing of the target minus the sensor boresight bearing.

The choice of error (k) as one of the input variables is intuitive. Nonetheless, most similar fuzzy control systems might use the rate of change of error or  $\Delta$  error (k) as the second input variable, where  $\Delta$  error (k) = error (k) - error (k-1). The

time change in this simulation from k to k-1 is 1 second. From the above equation, it is clear that both pairs of error  $(k)/\Delta$  error(k) and error (k)/error (k-1) contain the same amount of information and thus will provide equivalent inputs to the system. Our choice is based on simplification reasoning.

The crisp value of error (k) and error (k-1) are mapped into five fuzzy sets, labelled in linguistic terms of: VL(very low), L(low), M(medium), H(high), and VH(very high).

The meaning of each linguistic term is determined by the specific membership functions, which are defined by triangular functions on the universe of discourse of error (k) and error (k-1) respectively. Figures 9.5 and 9.6 show the input universe of discourse and the specific membership function.

The output of this fuzzy system is the panning rate (degree/sec) for the sensor, and its universe of discourse is divided into thirteen fuzzy sets labelled in the linguistic terms of: VL (very low), MVL (mid-very low), L (low), ML (mid-low), M (medium), MH (mid-high), H (high), MVH (mid-very high), VH (very high), MVVH (mid-very very high), VVH (very very high), and VVVH (very very very high). The membership functions are also triangular.

Figure 9.7 shows the output universe of discourse and the specific membership functions.

The fuzzy system has a total of 25 IF-THEN rules, as summarized in table 9.1. A sample of these rules is given below.

- IF  $\operatorname{error}(k-1)$  is L AND  $\operatorname{error}(k)$  is M THEN panning rate = VH.
- IF  $\operatorname{error}(k-1)$  is H AND  $\operatorname{error}(k)$  is L THEN panning rate = ML.
- IF  $\operatorname{error}(k-1)$  is M AND  $\operatorname{error}(k)$  is VL THEN panning rate = MVL.

| $\mathbf{e}(k-1)\backslash\mathbf{e}(k)$ | VL  | L  | M    | H     | VH    |
|--|-----|----|------|-------|-------|
| VL                                       | MVL | ML | MVVH | MVVVH | VVVH  |
| L  | MVL | ML | VH   | VVH   | VVVH  |
| M  | MVL | ML | VH   | VVH   | MVVH  |
| Н  | MVL | ML | MVH  | MVVH  | MVVVH |
| VH                                       | MVL | L  | Н    | VH    | VVH   |

**Table 9.1:** The 25 fuzzy rules

where e(k) and e(k-1) are the error(k) and error(k-1) respectively. We use the centre of gravity approach for the defuzzification process.

The complete closed-loop diagram is shown in Figure 9.8. The desired goal or the set point of the SMS is to achieve zero error.

# 9.5.2 Example 1

In the first simulation, we considered one sensor. Figure 9.9 shows the target moving profile, the sensor position and FOV. Figure 9.10 shows the results of the sensor tracking error with respect to the target track with and without the sensor managed by the SMS.

With a fixed panning rate of about 1 degree per second, the sensor lost track of the target by the 11th time step and only managed to pick it up again on the 42nd, before losing it totally after the 51st. On the other hand, with a SMS, the sensor managed to keep track of the target throughout the first 76 time steps. Note that due to the limited panning angles of the EO sensor, regardless of whether the SMS is in use, the sensor will not be able to sense the target beyond the 76th time step. This led us to the next example where 2 EO sensors are used for a better coverage.

# 9.5.3 Example 2

In this example, we introduce an additional EO sensor to the previous example. A cueing process is added to the SMS on top of the fuzzy controller. The position and coverage of the two sensors are as shown in Figure 9.11.

Theoretically the sensor management cueing modules will command the next sensor to take over when the target is within the sensor's FOV. The cueing process is based on a simple rule statement, as follows:

### IF

sensor 1 is within 10 degrees of its panning limits and target is within sensor 2's FOV,

#### THEN

cue sensor 2 to take over the tracking process from sensor 1.

Alternatively,

### IF

another target is entering the trackable region of Sensor 1, and Sensor 1's current target is within Sensor 2's trackable region,

#### THEN

cue Sensor 2 to take over the tracking process from Sensor 1.

Figure 9.12 shows the error output of the two sensors controlled by the SMS. The switch over occurred at the 62nd time step. Note that the errors correspond to the target moving pattern.

# 9.6 Software Agents for Decentralized Sensor Management

More recently, investigations are carried out using software agents to facilitate resource allocation in a decentralized environment such as decentralized sensor network. The studies seek to use intelligent agents for the control of sensors in performing various tasks, such as the tracking of a moving target, improve fusion processes through dynamic team formation and task allocation in a distributed/decentralized environment.

Brännström [39] proposed an agent architecture, with mobile agents for a dynamic sensor network. Agents were divided into different types according to their roles and they communicated and cooperated closely with each other to ensure the proper functioning of the sensor network. In [206], an algorithm was proposed, using the metaphor of Belief-Desire-Intention (BDI), for the allocation of tasks to resources in a decentralized network, such that a task was assigned to a unique resource. Each task has an agent attempting to bidding for the resources using the BDI metaphor. An arbitration agent or arbitrator, was used in the algorithm to determine the agent's intention that should be executed. In [212], the distributed task allocation algorithm using computational geometry techniques were used. Soh et al. also provided important insights into coalition formation for a time-critical and resourceconstrained environment. It was acknowledged that in a multiagent system, where agents have only local knowledge of the environment in which they are situated, optimal coalition formation is difficult. Thus, in such a time-critical system involving the use of agents, it was more appropriate to trade-off sub-optimal solutions that met minimum requirements (satisfying coalitions) than seek to form coalitions that achieved maximum performance.

However, there are also issues in using software agents for decentralized sensor management. For example, Howard *et al.* [105] raised the concern of negotiation amongst agents in a distributed sensor network may take too long for the coalition of sensor agents to be useful in performing tasks. Hence, it may not be suitable for time critical systems. Investigations are currently on-going in addressing some of these issues.

# 9.7 Conclusions

We have presented the various classification and roles of sensor management. The roles of sensor management as a controller were also discussed and demonstrated in the simulation studies.

The sensor movement with the fixed panning rate will not be able to adapt to target changes in direction and neither could they do a proper switching between sensors. However, the sensor controlled by a properly designed SMS, such as the

fuzzy controller and the cue process, can adapt to a change in target movement and detect the target.

There are, in general, more management issues for active sensors, such as the control of the various active modes. Some examples of the active modes are burnthrough modes, very high resolution modes, search mode and track mode. For passive sensors, the management issues could be unique to each of the sensors. Example: sensor management could act as a feedback controller to control the passive sensor direction (e.g. adaptive scan rate), control the detection threshold or the time integration of the sensor's data processing.

Lastly we discussed the use of software agents for decentralized sensor management.

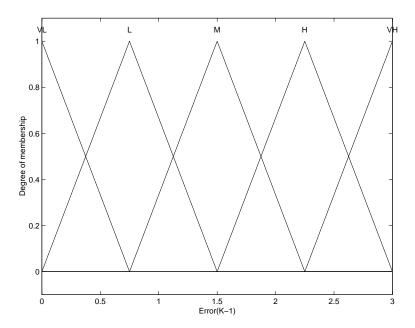


Figure 9.5: Partition of the input universe of discourse for error (k).

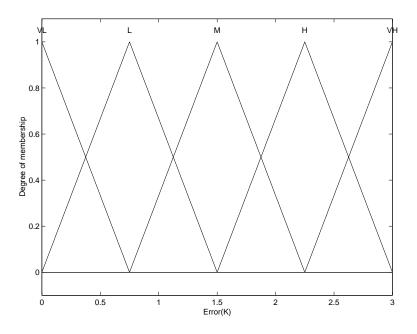


Figure 9.6: Partition of the input universe of discourse for error (k-1).

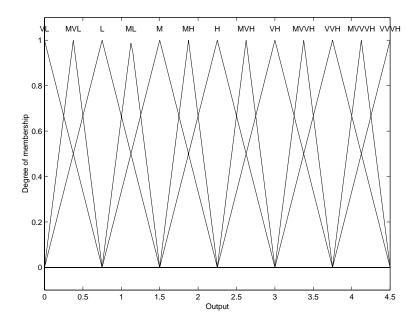


Figure 9.7: Partition of the output universe of discourse.

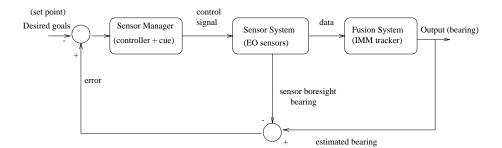
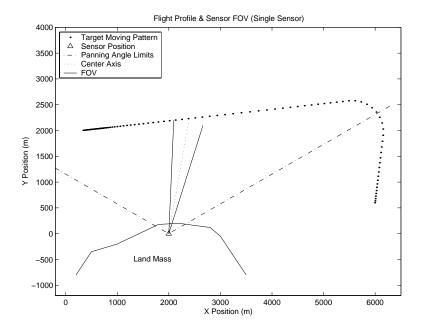


Figure 9.8: The complete closed-loop diagram.



**Figure 9.9:** Target moving profile and the sensor field of view (FOV).

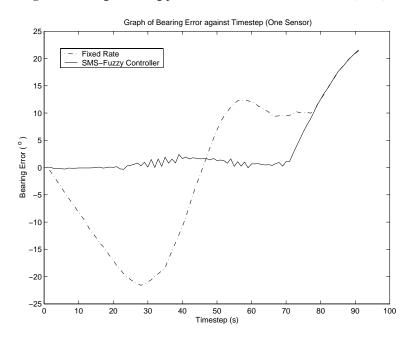


Figure 9.10: Error results of one sensor.

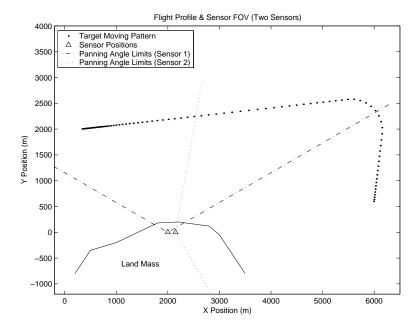


Figure 9.11: Position and coverage of the two sensors.

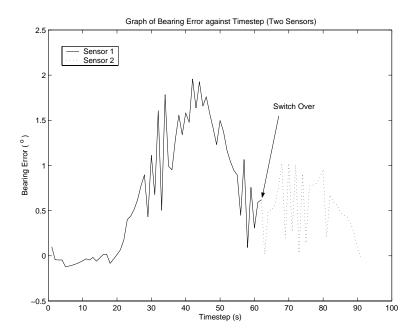
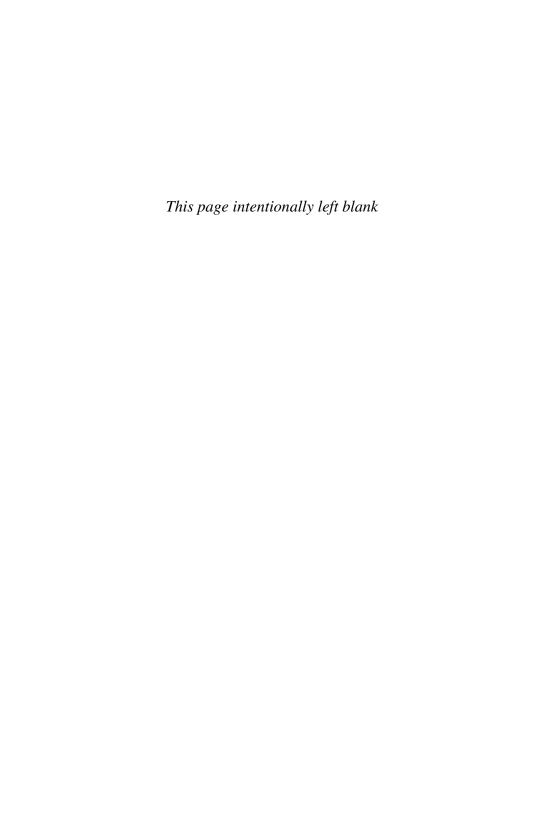


Figure 9.12: The error results of the two sensors with SMS.



## Appendix A

#### A.1 Filter Models

### A.1.1 Stationary model

1-D stationary model. For estimating the bearing of a target hovering or moving at a constant bearing towards a sensor with zero mean random bearing velocity. The following can be applied.

$$X_{k+1} = X_k + Tw_k \tag{A.1}$$

where:  $X_k = [x_k]$ ,  $w_k = [\dot{x_k}]$ , and T is the sampling interval.

2-D stationary model. For estimating the position of a target at a stationary position with zero mean random velocity. The model equation can be described as:

$$X_{k+1} = X_k + Tw_k \tag{A.2}$$

where:  $X_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}$ ,  $w_k = \begin{bmatrix} \dot{x_k} \\ \dot{y_k} \end{bmatrix}$ , and T is the sampling interval.

### A.1.2 Constant velocity model

• 1-D constant velocity model. For estimating the position of a target moving at constant velocity with zero mean process noise.

$$X_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} X_k + \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix} w_k \tag{A.3}$$

where: 
$$X_k = \begin{bmatrix} x_k \\ \dot{x_k} \end{bmatrix}$$

 2-D constant velocity model. For estimating the position of a target moving at constant velocity motion with zero mean random acceleration. The model equation can be described as:

$$X_{k+1} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} X_k + \begin{bmatrix} \frac{T^2}{2} & 0 \\ 0 & \frac{T^2}{2} \\ T & 0 \\ 0 & T \end{bmatrix} w_k$$
 (A.4)

where: 
$$X_k = \begin{bmatrix} x_k \\ y_k \\ \dot{x_k} \\ \dot{y_k} \end{bmatrix}$$
.

#### A.1.3 Acceleration models

1-D acceleration model (Ramachandra's model I). For estimating the position
of a vehicle moving at constant acceleration to a sensor with zero mean process
noise. The model equation is described as:

$$X_{k+1} = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} X_k + \begin{bmatrix} 0 \\ 0 \\ a_k \end{bmatrix}$$
 (A.5)

where: 
$$X_k = \begin{bmatrix} x_k \\ \dot{x_k} \\ \dot{x_k} \end{bmatrix}$$
.

And T is the sampling interval and  $a_k$  is the process noise that disturbs the acceleration of the vehicle. It accounts for both maneuvering and other modeling errors, and is assumed to have zero mean and constant variance  $\sigma_a^2$ .

The covariance matrix Q for the process noise is thus given by:

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_a^2 \end{bmatrix} \tag{A.6}$$

• 1-D acceleration model (Ramanchandra's model II). The model can be defined by:

$$X_{k+1} = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} X_k + \begin{bmatrix} \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{bmatrix} w_k$$
 (A.7)

where: 
$$X_k = \begin{bmatrix} x_k \\ \dot{x_k} \\ \ddot{x_k} \end{bmatrix}$$

• 2-D acceleration model (extension of Ramachandra's model I).

$$X_{k+1} = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0\\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2}\\ 0 & 0 & 1 & 0 & T & 0\\ 0 & 0 & 0 & 1 & 0 & T\\ 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} X_k + \begin{bmatrix} 0 & 0\\ 0 & 0\\ 0 & 0\\ 0 & 0\\ 1 & 0\\ 0 & 1 \end{bmatrix} w_k \tag{A.8}$$

where: 
$$X_k = \begin{bmatrix} x_k \\ y_k \\ \dot{x_k} \\ \dot{y_k} \\ \dot{x_k} \\ \ddot{y_k} \end{bmatrix}$$

### A.1.4 Exponentially correlated velocity (ECV) model

In this model, it was assumed that the target velocity decays exponentially between measurements with no continuous process noise and undergoes an instantaneous change at each sampling time.

$$X_{k+1} = \begin{bmatrix} 1 & \tau(1 - e^{-\theta}) \\ 0 & e^{-\theta} \end{bmatrix} X_k + U_k$$
 (A.9)

where:  $X_k = \begin{bmatrix} x_k \\ \dot{x_k} \end{bmatrix}$ .

Covariance of  $U_k$  is:

$$Q = \begin{bmatrix} \tau^2 \sigma_v^2 [4e - 3 - e^{-2\theta} + 2\theta] & \tau \sigma_v^2 [1 - e^{-\theta}]^2 \\ \tau \sigma_v^2 [1 - e^{-\theta}]^2 & \sigma_v^2 [1 - e^{-2\theta}] \end{bmatrix}$$
(A.10)

where:  $\theta=\frac{T}{\tau},$  T is the sampling time and  $\sigma_v^2$  is variance of the target velocity,  $\tau$  is the target time.

The measurement model is:

$$Z_{k+1} = \begin{bmatrix} 1 & 0 \end{bmatrix} X_{k+1} + v_{k+1}$$
 (A.11)

where variance of  $v_k$  is  $R = \sigma_v^2$ .

### Exponentially correlated acceleration (ECA) model

This model is applied to a vehicle moving with a random exponentially correlated acceleration with a white noise process.

The model equation is:

$$X_{k+1} = \begin{bmatrix} 1 & T & \tau^2(\theta + e^{-\theta} - 1) \\ 0 & 1 & \tau(1 - e^{-\theta}) \\ 0 & 0 & e^{-\theta} \end{bmatrix} X_k + U_k$$
 (A.12)

where: 
$$X_k = \begin{bmatrix} x_k \\ \dot{x_k} \\ \dot{x_k} \end{bmatrix}$$
 and

T is the sampling time,  $\tau$  is the correlation time of the vehicle acceleration and  $\theta = \frac{T}{\tau}$ . The process noise covariance Q is expressed as:

$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix}$$
 (A.13)

where: 
$$\begin{array}{l} q_{11} = \tau^4 \sigma_a^2 [1 - e^{-2\theta} + 2\theta (1 - 2e^{-\theta} - \theta + \frac{\theta^2}{3})] \\ q_{12} = \tau^3 \sigma_a^2 (1 - e^{-\theta} - \theta)^2 \\ q_{13} = \tau^2 \sigma_a^2 (1 - e^{-2\theta} - 2\theta e^{-\theta}) \\ q_{22} = \tau^2 \sigma_a^2 [(1 + 2\theta) - (2 - e^{-\theta})^2] \\ q_{23} = \tau \sigma_a^2 (1 - e^{-\theta})^2 \\ q_{33} = \sigma_a^2 (1 - e^{-2\theta}) \end{array}$$

and  $\sigma_a^2$  is the variance of the target's acceleration.

### A.1.6 The coordinated turning model

Applicable to vehicle motion with a constant turn rate and constant speed.

$$X_{k+1} = \begin{bmatrix} 1 & \frac{\sin(\kappa)T}{\omega(k)} & 0 & -\frac{1-\cos(\kappa)T}{\omega(k)} & 0\\ 0 & \cos(\kappa)T & 0 & -\sin(\kappa)T & 0\\ 0 & \frac{1-\cos(\kappa)T}{\omega(k)} & 1 & \frac{\sin(\kappa)T}{\omega(k)} & 0\\ 0 & \sin(\kappa)T & 0 & \cos(\kappa)T & 0\\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} X_k + \begin{bmatrix} \frac{T^2}{2} & 0 & 0\\ T & 0 & 0\\ 0 & \frac{T^2}{2} & 0\\ 0 & T & 0\\ 0 & 0 & T \end{bmatrix} v_k$$
(A.14)

The measurement equation is thus:

$$Z_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} X_{k+1} + \begin{bmatrix} v_x(k+1) \\ v_y(k+1) \end{bmatrix}$$
(A.15)

where: 
$$X_k = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ \omega \end{bmatrix}$$

### **A.1.7** The measurement equation

The measurement equation converts state vector X to the measurement vector Z. For example, in 2-D constant velocity model, the measurement equation is:

$$Z_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} X_{k+1} + \begin{bmatrix} v_x(k+1) \\ v_y(k+1) \end{bmatrix}$$
 (A.16)

where: 
$$Z_{k+1} = \begin{bmatrix} x_m(k+1) \\ y_m(k+1) \end{bmatrix}$$
.

And:

 $x_m(k+1)$  = measured x coordinate at scan k+1.

 $y_m(k+1)$  = measured y coordinate at scan k+1.

 $v_x(k+1)$  = random noise on x measurement at scan k+1.

 $v_y(k+1)$  = random noise on y measurement at scan k+1.

### A.2 Generic Particle Filter

The dynamics system model of a manoeuvring target in a tracking system is given by

$$\dot{x} = f(x, u, w) \tag{A.17}$$

$$z(k) = h(x(k), v(k)) \tag{A.18}$$

where x is the state vector, u is the control vector, w is the process noise vector representing possible deviations in f(.). z(k) is the discrete-time measurement vector at time k, and v(k) is the measurement noise vector.

The dynamics of the target is a continuous-time process, as indicated by equation A.17. f(.) defines the motion for the target in a form of a differential equation. The measurement process is in discrete time because most sensors used for target tracking record the position and velocity at a given instance of time. Hence, the measurement equation A.18 in discrete form is

$$z(k) = H(k)x(k) + v(k) \tag{A.19}$$

The function f(.) is usually unknown to the tracking system. Hence, the target dynamics are commonly modelled by a deterministic linear dynamic system in discrete time as follows:

$$x(k+1) = F(k)x(k) + G(k)u(k) + w(k).$$
(A.20)

At time k, x(k) (assumed to be a vector of dimension  $n_x$ ) is the state of the system normally containing the position and velocity of the target. F(k) is the transition matrix of dimensions  $(n_x \times n_x)$  and it defines a linear constraint on the target dynamics, u(k) is the unknown input control, a vector of dimension  $n_u$ , G(k) is the input gain (or noise matrix)  $(n_x \times n_u$  matrix) and  $w(k) \sim N(0, Q(k))$  is the process noise which is usually assumed to be white Gaussian noise with variance Q.

In tracking manoeuvring targets, the control vector is not directly observable by the tracking system. Since the input control is unknown, the dynamics model that is assumed for a target in a tracking system is further simplified as follows:

$$x(k+1) = F(k)x(k) + w(k)$$
 (A.21)

The process noise w(k) can be used to model the unknown target acceleration. The unknown acceleration term can also be included in x to form a third-order model. However, the acceleration most often varies with time in such a manner that a model cannot be clearly identified during tracking.

### A.2.1 Generic particle filter algorithm

The following steps show the algorithm of the generic particle filter:

#### 1. Initialization

- N = Number of particles.
- $X^{(i)}(1) = [x(1), y(1), 0, 0]^T$  for i = 1, ..., N.

#### 2. Prediction step

• For each particle  $i=1,\ldots,N$ , evaluate the (k+1|k) state of the process system using the state at time k with the process noise at time k.

$$\hat{X}^{(i)}(k+1|k) = F(k)\hat{X}^{(i)}(k) + \left( \begin{array}{c} Cauchy\ Distributed \\ Process\ Noise \end{array} \right)_{(k)}$$

#### 3. Evaluate importance weights

For each particle  $i = 1, \dots, N$ ,

 Find the predicted observation state of the measurement system using the predicted current state of the system and the measurement noise at time k.

Calculate the likelihood (importance weights) according to the distribution given.

$$likelihood^{(i)} = \mathcal{N}(\hat{z}^{(i)}(k+1|k); z^{(i)}(k+1), var)$$
 (A.22)

• Normalize the importance weights.

$$\tilde{w}^{(i)} = \frac{likelihood^{(i)}}{\sum_{j=1}^{N} likelihood^{(j)}}$$
(A.23)

#### 4. Resampling/Selection

Resample using multinomial sampling techniques by multiplying the particles with higher importance weights and suppressing the particles with lower importance weights. Modify the current state using the new indexed weights calculated.

• Compute cumulative weights,

$$CumWt^{(i)} = \sum_{j=1}^{i} \tilde{w}^{(j)}$$
 (A.24)

• Generate random variables uniformly distributed in  $\mathcal{U}^{(i)} \sim \mathcal{U}(0,1)$  with the number of steps according to the number of particles.

• Determine which particle to multiply and which to suppress (see Figure A.1):

For every particle inIndex i, where  $i = 1, \dots, N$ 

$$Case(1): \quad If \ 0 < \mathcal{U}^{(i)} < CumW \, t^{(1)}$$

or

$$Case(2): \quad If \; CumW \, t^{(j)} < \mathcal{U}^{(i)} < CumW \, t^{(j+1)}$$

for j = 1, ..., N - 1. Replace the old inIndex of that particular particle i with the new outIndex of the CumWt, i.e. 1 for Case(1) or j + 1 for Case(2) and repeat for the next inIndex i = i + 1.

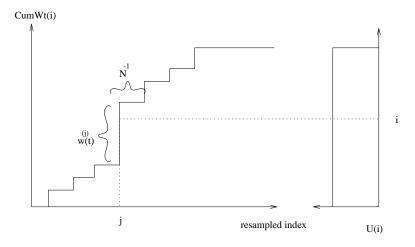


Figure A.1: Resampling process.

#### 5. Propagation step

• Incorporate the new state values after resampling of time step k for the calculation of time step k+1.

$$\hat{x}^{(1:N)}(k+1|k+1) = \hat{x}^{(outIndex)}(k+1|k)$$
 (A.25)

• Compute posterior mean

$$\hat{x}(k+1) = mean[x^{i}(k+1|k+1)], for i = 1,...,N$$
 (A.26)

• Repeat Step 2 to Step 5 for each time step.

### A.3 Bayesian Classifier

The Bayesian classifier uses a probabilistic approach of updating the likelihood of a hypothesis given a previous likelihood estimate and new observation of the target of interest. An example of a classification hypothesis is 'The target is of Type X'. The whole updating process is summarized by the following equation:

$$P(H_i|E) = \frac{P(E|H_i)P(H_i)}{\sum_{j=1}^{n} P(E|H_j)P(H_j)}$$
 for  $i = 1:n$  (A.27)

where

 $P(H_i|E)$  : posterior probability of hypothesis  $H_i$  being true, given the new

observation E, also known as the *a posteriori* probability.

 $P(E|H_i)$  : conditional probability of making observation E given that the

 $H_i$  is true, and it is also called the *transitional* probability; and  $P(H_i)$ : a priori probability of hypothesis  $H_i$  being true.

Note that the n hypotheses  $H_1,H_2,\ldots,H_n$  must be mutually exclusive and exhaustive and  $\sum_{j=1}^n P(H_j)=1$ .

By the end of the updating process given an observation E, we obtain a set of posterior probabilities of all the hypotheses. Commonly used decision logic will be to select the hypothesis with the largest posterior probability as the true hypothesis - this approach is known as the MAP (Maximum A Posteriori) method. In addition, most decision logic also requires this maximum posterior probability to exceed a predefined threshold before a declaration on the target's classification is done.

However, in practical applications, a single observation usually will not lead to the maximum posterior probability to exceed the given threshold, and usually several observational updates are needed to confirm the target classification.

The following equation is used instead of equation (A.27) given a sequential update of observations  $\{E_1, E_2, \dots, E_{k-1}, E_k, \dots\}$  to update the posterior probability of each hypothesis at time step k:

$$P(H_i|E_k) = \frac{P(E_k|H_i)P(H_i|E_{k-1})}{\sum_{j=1}^n P(E_k|H_j)P(H_j|E_{k-1})} \quad \text{for } i = 1:n.$$
 (A.28)

An important assumption here is that the observations are independent conditioned upon the same hypothesis. That is,

$$P(E_k, E_{k-1}|H_i) = P(E_k|H_i)P(E_{k-1}|H_i) \qquad \forall k, i = 1:n,$$
 (A.29)

although the observations are in general not independent, that is,

$$P(E_k, E_{k-1}) \neq P(E_k)P(E_{k-1}).$$
 (A.30)

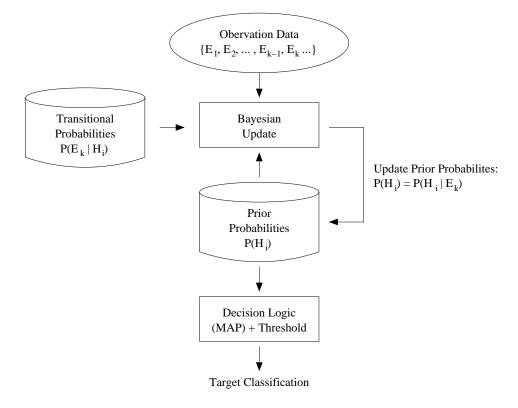


Figure A.2: Sequential updating process of Bayesian classifier.

Figure A.2 depicts the sequential updating process of the Bayesian classifier.

Next, we consider the case of two sensors making a sequences of observations  $\{X_1, X_2, \ldots, X_k, \ldots\}$  and  $\{Y_1, Y_2, \ldots, Y_k, \ldots\}$ , respectively. In this case, the Bayesian equation for the posterior probabilities updating becomes:

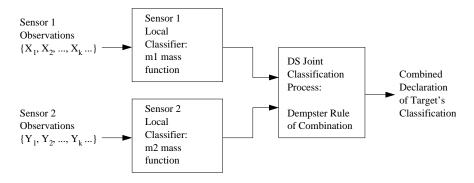
$$P(H_i|X_k, Y_k) = \frac{P(X_k|H_i)P(Y_k|H_i)P(H_i|X_{k-1}, Y_{k-1})}{\sum_{j=1}^n P(X_k|H_j)P(Y_k|H_j)P(H_j|X_{k-1}, Y_{k-1})}$$
 for  $i = 1 : n$ .
(A.31)

As in equation (A.28), the assumption of independent observations being made by Sensor 1 and Sensor 2 is in place. That is,

$$P(X_k, Y_k | H_i) = P(X_k | H_i) P(Y_k | H_i)$$
  $\forall k, i = 1 : n.$  (A.32)

By applying equation (A.29), the classifier is effectively a joint classifier/fuser of two individual local classification processes, as depicted in Figure A.3.

Equation (A.29) can be easily generalized to handle cases of more than two



**Figure A.3:** Joint classification.

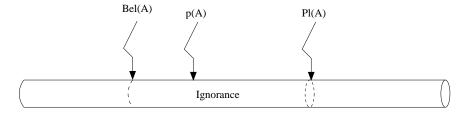
sensors.

### A.4 Dempster-Shafer Classifier

The Dempster-Shafer classifier carries out evidential reasoning process using the Dempster-Shafer theory developed by Shafer [207] and originally initiated by Dempster [207]. The Dempster-Shafer (DS) theory starts with a universal set:

$$\Theta = \{H_1, H_2, \dots, H_n\} \tag{A.33}$$

whose elements  $H_i$  are mutually exclusive and exhaustive. The set  $\Theta$  is called the Frame of Discernment. In the case of classification application, the  $H_i$  defined here could be identical to those defined in the above Bayesian Classifier. However, the difference here is that the attention is drawn onto the power set (the set of all subsets) of  $\Theta$ , denoted by  $2^{\Theta}$ , instead of the individual  $H_i$ .



**Figure A.4:** Belief and plausibility functions.

Let A be an element of  $2^{\Theta}$ . We call A a proposition, where it is possibly a union of some of the hypotheses  $H_i$ .

Examples of  $A \in 2^{\Theta}$ :

$$A = \{H_2, H_5\},\$$

$$A = \{H_3\},\$$

$$A = \{H_2, H_3, H_4, H_6\},\$$

$$A = \{H_4, H_5, H_6\}.$$

The DS reasoning consists of a basic probability mass assignment, which is a function acting on the elements of the power set of  $\Theta$ ,  $m:2^{\Theta} \to [0,1]$ , such that

$$m(\Phi) = 0,$$

$$\sum_{A \subseteq \Theta} m(A) = 1.$$
(A.34)

A 'belief function' and a 'plausibility function' are associated with m, are defined as

$$Bel(A) = \sum_{B\subseteq A} m(b)$$
  
 $Pl(A) = \sum_{A\cap B\neq \Phi} m(B) = a - Bel(\overline{A})$  (A.35)

respectively, for all  $A \subseteq \Theta$  (Figure A.4).

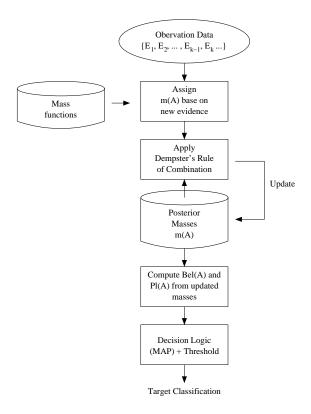
The belief function provides a global measure of the belief that a proposition A is true. On the other hand, the plausibility measure of proposition A is the amount of belief that could possibly be put on A if further evidence that support A arrives. Note that  $Bel(A) \leq Pl(A)$ , and the true probability of A lies between them. The gap between Bel(A) and Pl(A) denotes the classifier's ignorance about A. In summary, the DS method distributes the total mass of uncertainty over all the subsets of  $\Theta$  instead of over the individual  $H_i$ .

The DS theory provides a rule called the Dempster's Rule of Combination (also called the orthogonal sum) to combine two mass functions m1 and m2 and is defined as:

$$m1 \oplus m2(A) = \frac{\sum_{B \cap C = A} m1(B)m2(C)}{1 - K}$$

$$K = \sum_{B \cap C = \Theta} m1(B)m2(C). \tag{A.36}$$

The term K indicates the amount of conflicts between the two mass functions, and 1-K is the re-normalizing term to ensure that the combined mass function satisfies condition (A.34). The orthogonal sum is associative and commutative.



**Figure A.5:** DS classification process. Note that the update equation is given by  $m(A) = (m(A)|E_k) \oplus (m(A)|E_{k-1})$ 

The classification process of the individual sensor DS classifier is shown in Figure A.5.

For a DS classifier, the choice of the decision logic is not as straight forward as in the case of the Bayesian classifier. But the concept is still similar, as in it we wish to select a proposition with the highest probability and check it against the classification declaration threshold. What is tricky in this case is that:

- Firstly, we may want to decide on a single hypothesis and the set of propositions consists of not only the singletons of the individual hypothesis, which is of the classifier's main interest, but also all possible unions of them.
- Secondly, should the belief measure or the plausibility measure be used in the judgement? In the literature, most researchers use the belief measure for making decisions since it directly represents the amount of support given by all evidences available.

Smets [210] proposed an alternative approach as he introduced the Pignistic (from Latin pignus, meaning a bet) probability that transformed the belief function into a probability function for decision-making, and is defined as

$$BetP(H_i) = \sum_{H_i \in A} \frac{m(A)}{|A|} \qquad \forall H_i \in \Theta.$$
 (A.37)

The idea is to distribute the ambiguous mass of a proposition equally over its elements.

The existence of the Dempster's rule of combination also readily puts the DS classifier as a suitable candidate to take on the role of the Joint Classifier, especially when local classifiers output soft decisions that contain probabilistic measures of the confidences of the hypotheses or propositions in question.

### A.5 Artificial Immunology System

Biologists had studied the immune system, particularly the human immune system, for century. Recently, engineering researchers have been looking at the amazing and complex immunological defense mechanisms that make up the immune system. They are aiming at modeling or mimicking these mechanisms for practical applications, such as: intelligent computer and network security systems, smart defense systems, identification and pattern recognition systems, and general optimization problems.

Without the immune system the body would not be able to defend itself against disease and illness. The immune system consists of monocytes, which act as sentinels; helper and suppressor T-cells which sound the alarm; antibodies, killer cells and cytotoxic T-cells which acts like an artillery; and macrophages which clean up the battlefield [51]. The body is like a large battlefield where the immune system has to ensure that it wins the battle 100% when invaded. When the immune system is functioning properly, we hardly notice the constant battles that rage within us.

The components of the human immune system are many and varied. The major constituents include the bone marrow, thymus, lymph nodes, spleen, tonsils, appendix, Peyer's patches and lymphatic vessels. These all protect the body against dangerous toxins and invading microorganisms. Readers who are interested to know more about the artificial immune systems may refer to the book by Leandro N. de Castro and Jonathan Timmis [66].

### A.5.1 B-cells or B- lymphocytes

B-cells or B-lymphocytes provide the antibodies or immunoglobulins to the body. They are precursors to the powerful plasma cells, which provide the body with valuable antibodies or immunoglobulins. Antibodies constitute the immune system's

single most powerful weapons. B-cells use antibodies as ammunition. For example, when a virus appears that is too strong for even the mighty macrophages, plasma cells launch antibodies, which, with the help of other immune cells, break the enemy to pieces.

Specialized training ensures that the B-cells have specific immunity to antigens. Antigens are foreign makers. It helps all immune cells differentiate invaders from body cells. Approximately one billion B-cells in the body are sent for 'training' and divided into groups. Depending on the group a B-cell is placed in, it will learn how to identify and kill one of a million different kinds of antigens that it may come into contact with. This is the cell's specific purpose, or its specificity. The B-cell will only be able to identify and confront that one specific antigen it specialized in. A healthy body produces enough B-cells to make the entire trained force a very effective army against invaders that enter the body.

### A.5.2 T-cell or T-lymphocytes

About 75% of the lymphocytes are trained in the thymus gland and are called T-cells or T-lymphocytes. The T-cells do not use antibodies. Mature activated T-cells will evolve into one of the following kinds of cells: helper cells, suppressor cells, killer cells, cytotoxic cells or memory cells.

Helper cells are responsible for prompting many of the immune system cells into action. It identifies an invader and rush to the spleen and lymph nodes where it sends chemical signals that stimulate other cells. By way of chemical signals called lymphokines, the helper cells transmit messages that rally B-cells and macrophages into battle.

Suppressor cells suppress the activator messages sent out by the helper cells. This keeps the immune system from working too hard and killing many of its own members.

Killer cells destroy cells which have penetrated the system and have been taken over by antigens. For example, when cancer takes over cells, the killer cell's responsibility is to destroy the occupied cell. Normally killer cells reside in the lymph nodes, but when antigens are present the killer cells swarm out to find the antigen of their specificity.

Cytotoxic cells also use poisons to kill the enemy. When cytotoxic cells come in contact with their target antigen, the cytotoxic cells release a cytotoxic substance that either kills or renders the invader helpless.

Memory cells. Each time a B- or T-cell meets an invader, it memorizes its specific qualities or makeup. This information is stored in the immune system, and new lymphocytes recruits are trained to recognize this agent as an enemy. The next time the invader appears in the body, these memory cells identify it and begin the immediate production of antibodies against it.

Lymph organs, such as lymph vessels and lymph nodes, aid the immune function by cleansing the blood and organisms that are floating in the bloodstream.

### A.5.3 What can we learn from the immune system?

Listed below are some possible areas where we can learn and attempt to engineer possible algorithms for intelligent systems:

- Multi-layer defense mechanism. The complete immune system is a multi-layer defense mechanism. The first layer of defense in an immune system is the physical barrier, namely the skin, nasal hairs and mucous tissues. Itching effects that cause coughing and sneezing help to expel the invader. The second layer of defense is the biochemical barrier, such as fluids (saliva, sweat and tears), that contain destructive enzymes. Stomach acids will kill most microorganisms ingested in food and drinks. The third layer of defense are organs such as the lymph node and spleen (also see the adaptive immune system). These contain a system of passages through which lymph and blood circulate constantly. In this way harmful elements can be removed from the blood and lymphs before they have the opportunity to multiply. The complete immune systems are on constant 'seek and destroy' mission.
- Attack mechanism. The immune system's strategic attack may be divided into three phases. These are: recognition of the antigen, activation of lymphocytes and effector phase.
- Pattern recognition mechanism. By direct matching of pattern. New pattern will be learnt and adaptively modeled for future matching.
- Specialized function. The B-cell and the several groups of T-cells have specialized functions. Each plays a role in destroying a particular invader.
- Working cooperatively. The complete immune system works together to battle
  against a multitude of pathogens. For example, one can see helper cells as
  the 'bugler' giving call to arm for battle. Protein emissions from helper cells
  actually tell B-cells to begin transformation into plasma cells. The plasma
  cells, together with other immune cells, break the pathogen. This complex
  collaboration process continues until all the foreign invaders are destroyed.
- Decentralized control. The components of the immune system are decentralized and spread across the entire body. It is not controlled by a central organ. Each component of the immune system act autonomously in identifying and eliminating the pathogens. The lymphoid organs are distributed throughout the body. These are subjected to decentralized 'training' in the 'instruction center', such as the thymus gland and lymphoid tissues.

 Adaptation mechanism. The immune system can adapt itself structurally and in numbers to the invaders' challenges. It has the capabilities to learn and remember the identification of the invaders, and to clone itself in large numbers when challenged.

### A.6 Introduction To Ontology

Lately, there has been a lot of interest in ontologies, partly due to the interest in the semantic web and partly because ontologies promise to be a means of achieving semantic interoperability between different systems. In this section, we examine the background and the components of an ontology and also highlight aspects of their development and reasoning processes.

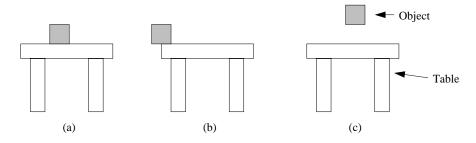
#### A.6.1 Introduction

In May 2001, Tim Berners-Lee (the founder of the Web) shared his vision of the future in a Scientific American article entitled 'The Semantic Web' [24]. His vision was 'an extension of the current web in which information is given well-defined meaning' that computers or software agents can understand and utilize in performing complex tasks. For instance, an agent can arrange on behalf of its user an appointment with a specialist by understanding the meaning of the various information it is presented with; like medical condition, doctor's schedule and location of clinics. The well-defined meaning or semantic is provided by ontologies, leading some researchers to conclude 'ontologies are the basic infrastructure for the Semantic Web' [151].

An ontology is an artifact built using formal languages that allows the intension of the words used by a community, when they communicate, to be defined as precisely as possible. When the community speaks of a certain object or a certain thought, they hold in their mind certain concepts that are not expressed in the utterances they make. For instance the concept of 'ABOVE' encompasses many different configuration of objects, all of which is understood to exhibit the 'ABOVE' relationship as shown in Figure A.6. For an 'ABOVE' relation, definitions can include the notion of contact or lack of contact as well as the extent of area being encompassed. An ontology seeks to make these hidden concepts explicit, necessitating its users and builders to provide definitions that are unambiguous and as close to the intension of the concepts as possibly allowed by the limitations of the formal language.

### A.6.2 Components of ontologies

An ontology has four elements: concept, instance, relation and axiom. Each ontology talks about concepts and their instantiations. Concepts can be seen as a cate-



**Figure A.6:** Different interpretations of the 'ABOVE' relation.

gorization of the domain into different categories that may or may not overlap. An individual or instance is said to belong to a concept if it embodies the characteristics of the concept. In our simple ontology (see Figure A.7), we have four concepts: human beings, male human beings, female human beings and apples. We also have two individuals, John who is a male human being and Mary who is a female human being.

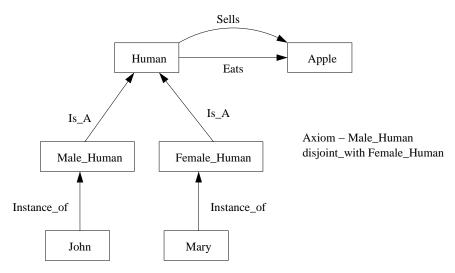


Figure A.7: Simple Ontology.

Relations and axioms differentiate an ontology from a simple lexicon. A relation describes how concepts and instances are associated. In our simple ontology, human beings is related to male human beings and female human beings by the  $Is_A$  or subsumption relation. This means that instances of male human beings share all the characteristics on human beings and, similarly, instances of female human beings share all the characteristics of the parent concept. Human beings are related to apples

via the EAT relationship denoting the idea that apples function as food for human beings as well as via the SELLS relationship. There can be any number of relationships defined between any pair of concepts. An axiom is also included in this ontology. The concept of male human beings is disjoint from the concept of female human beings showing that these two concepts do not share common instances. Every axiom shapes an ontology closer to the intended expression of the conceptualisation. In this particular example, we are excluding from consideration androgynous human beings (i.e. human beings born with either both male and female sex organs).

### A.6.3 Classification of ontologies

An ontology can be classified into one of three roles: top-level, domain and application. These is not static classification, as a top-level ontology for a group of systems may be an application ontology of a different group of systems. In general, a top-level ontology provides broad definitions that can be used across several domains. An application ontology is a specific ontology that may be irrelevant to another application as it describes constraints that applies to the application.

The choice of the language used to encode the ontology, as well as the target level of the ontology, determines its expressiveness. In general, the more expressive is an ontology, the harder and longer it is to reason with the ontology. Borgida and Patel-Schneider illustrated the difficulty with an example [38]. In the application, ontologies are used as terminological services, allowing constraints between terms to be checked during its execution. These ontologies would have limited expressiveness so as to reduce the computational requirements for the reasoning task. Ontologies used as references for a discourse are much more expressive since the requirements for reasoning tasks are fewer, i.e. the ontologies are used to establish consensus about the semantics of the terms used and not to infer new knowledge.

### A.6.4 Principles of ontology design

Gruber articulated five design principles for good ontology design [90] that we can distill into two: clarity and modularity. A good ontology would be clear to its designers, users and also to the machines. The purpose of the ontology would be documented and the ontological entities described in a clear way that it is obvious, i.e. what and how each term is used. Each term is defined concisely and comprehensively. Any ambiguous term should be renamed so that the ambiguity can be resolved. For instance, BANK can be interpreted as a financial institution or as a geographic feature. In this case, we should name the term so that these characteristics are clear in the names (e.g. BANK-FINANCIAL-INSTITUTION, BANK-GEOGRAPHIC-FEATURE). Relationships and formal semantic should also be stated, if possible, in order to provide further clarity to the terms.

Like good software engineering practice, an ontology should also be developed

as smaller, internally coherent components (or modules). A smaller ontology tends to be self-contained and easier to reuse than a larger, more monolithic ontology since it has less ontological commitments. The complexity of building an ontology is also managed by focusing on the development of smaller modules and the integration of the modules into the desired ontology.

A closely-related concept to modularity is extensibility. While developing the ontology, we should bear in mind its future extensions. An ontology should not be too generic nor should it be overly specific. In other words, it should be well tailored to its purpose and domain.

An ontology can be represented in different languages (e.g. frame, first-order logic, etc.) The chosen ontology representation language should be flexible enough that the ontology can be readily translated to a different application implementation language (e.g. C++/Java classes, production rules). This would allow the designed ontology to have minimal encoding bias. The DARPA Agent Markup Language with the Ontology Inference Layer (DAML+OIL) is based on a family of restricted first-order logics known as Description Logics that provides the formal foundation. We are able to perform reasoning through a variety of tools like the Java Theorem Prover (JTP) [83], Jena API [154] and Z/Eves with DAML+OIL semantic library [70]. Using this language also has an advantage - as a migration path from the University of Maryland (http://www.mindswap.org) exists to transform ontologies written in DAML+OIL to the upcoming standard W3C Web Ontology Language.

#### A.6.5 Conclusion

In summary, an ontology encodes knowledge in a formal language that can be used to establish consensus among a community of users, as well as to be processed by a computer or software agent in its reasoning task. An ontology consists of concepts, instances, relations and axioms, whose development are guided by two principles: clarity and modularity.

# **Bibliography**

- [1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.
- [2] S. V. B. Aiyer, M. Niranjan, and F. Fallside. A theoretical investigation into the performance of the Hopfield model. *IEEE Transactions on Neural Networks*, 1(2):204–215, June 1990.
- [3] L. B. Almeidal. A learning rule for asynchronous perceptron with feedback in a combinatorial environment. *Proceedings of the IEEE First International Conference on Neural Networks*, 2:609–618, 1987.
- [4] A. T. Alouani, N. Auger, and T. R. Rice. On the generalized nearest neighbor filter. In *22nd Southeastern Symposium on System Theory*, pages 261–264. IEEE Computer Society Press, 11-13 March 1990.
- [5] A. T. Alouani and T. R. Rice. Multiple model tracking with feedback. *Signal Processing, Sensor Fusion, and Target Recognition X, Proceedings of SPIE*, 4380:13–24, April 2001.
- [6] P. E. An and C. J. Harris. Multi-dimensional locally generalizing neural networks for real time control. In H. B. Verbruggen and M. G. Rodd, editors, *Artificial Intelligence in Real-Time Control 1992 IFAC Symposia Series*, pages 87–91. Pergamon Press, Delft, The Netherlands, 1993.
- [7] S. L. Andresen. Herbert A. Simon: AI Pioneer. *IEEE Intelligent Systems*, 16(4):71–74, July/August 2001.
- [8] A. Averbuch, S. Itzikowitz, and T. Kapon. Radar target tracking Viterbi versus IMM. *IEEE Transactions on Aerospace and Electronic Systems*, 27(3):550–563, May 1991.
- [9] V. V. Badami and N. W. Chbat. Home appliances get smart. *IEEE Spectrum*, pages 36–43, August 1998.
- [10] Y. Bar-Shalom. *Multitarget-Multisensor Tracking: Advanced Applications*. Artech House, Inc, 1990.

- [11] Y. Bar-Shalom. Tracking of splitting targets in clutter using an interacting multiple model joint probabilistic data association filter. In *Proceeding of the 30th Conference on Decision and Control*, pages 2043–2048. IEEE, Brighton, England, December 1991.
- [12] Y. Bar-Shalom. *Multitarget-Multisensor Tracking: Applications and Advances*, volume 2. Artech House, Inc, 1992.
- [13] Y. Bar-Shalom and L. Campo. The effect of common process noise on the two-sensor fused track covariance. *IEEE Transactions on Aerospace and Electronic Systems*, AES-22:803–805, Nov 1986.
- [14] Y. Bar-Shalom, K. C. Chang, and H. A. P. Blom. Tracking of splitting targets in clutter using an interacting multiple model joint probabilistic data association filter. In *Multitarget-Multisensor Tracking: Applications and Advances*, chapter 4, pages 93–110. Artech House, 1992.
- [15] Y. Bar-Shalom, H. Chen, and T. Kirubarajan. Performance limits of track-to-track fusion vs. centralized estimation: Theory and application. *Fusion 2001*, 1, August 2001.
- [16] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*, volume 179 of *Mathematics in Science and Engineering*. Academic Press, Orlando, Florida, 1988.
- [17] Y. Bar-Shalom and X. R. Li. *Estimation and tracking: Principles, Techniques and Software*. Artech House, Inc, 1993.
- [18] Y. Bar-Shalom and X. R. Li. Multitarget-Multisensor Tracking: Principles and Techniques. Lecture Notes. YBS Publishing, Storrs (CT 06269-3157), 1995.
- [19] Y. Bar-Shalom and G. D. Marcus. Tracking with measurements of uncertain origin and random arrival times. *IEEE Transactions on Automatic Control*, 25:802–807, August 1980.
- [20] Y. Bar-Shalom and E. Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11:451–460, September 1975.
- [21] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems Man and Cybernetics*, 13(5):834–846, September/October 1983.
- [22] R. Beale and T. Jackson. *Neural Computing: an introduction*. Institute of Physics Publishing, 1990.
- [23] T.M. Berg and H. Durrant-Whyte. Distributed and decentralized estimation. *Proceedings of the International Conference on Intelligent Control and Instrumentation*, 2:1118–1123, Feb 1992.

- [24] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, pages 34–43, 2001.
- [25] S. G. Bier, P. L. Rothman, and R. A. Manske. Intelligent sensor management for beyond visual range air to air combat. *Proceedings of the IEEE 1988 National Aerospace and Electronics*, 1:264–269, May 1988.
- [26] K. Birmiwal and Y. Bar-Shalom. On tracking a maneuvering target in clutter. *IEEE Transactions on Aerospace and Electronic Systems*, 20:635–644, Sept 1984.
- [27] H. Bischof. Pyramidal Neural Networks. Lawrence Erlbaum Associate Inc., 1995.
- [28] K. K. Biswas and A. K. Mahalanabis. Optimal fixed-lag smoothing for time delayed system with colored noise. *IEEE Transactions on Automatic Control*, 17:387–388, 1972.
- [29] S. Blackman, R. Dempster, and T. Broida. Multiple hypothesis track confirmation for infrared surveillance systems. *IEEE Transactions on Aerospace and Electronic Systems*, 29(3):810–824, July 1993.
- [30] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House, 1999.
- [31] S. S. Blackman. *Multiple-target tracking with radar applications*. Artech House, 1986.
- [32] W. D. Blair and G. A. Watson. Interacting multiple bias model algorithm with application to tracking maneuvering targets. In *Proceedings of the 31st Conference on Decision and Control*, pages 3790–3795. IEEE, 16-18 December 1992.
- [33] W. D. Blair, G. A. Watson, and T. R. Rice. Interacting multiple model filter for tracking maneuvering targets in spherical coordinates. In *IEEE proceedings* of the SOUTHEASTCON'91, volume 2, pages 1055–1059. IEEE, 7-10 April 1991.
- [34] W. D. Blair, G. A. Watson, and T. R. Rice. Tracking maneuvering targets with an interacting multiple model filter containing exponentially-correlated acceleration models. In *System Theory, Proceedings*, pages 224–228. IEEE Computer Society Press, 10-12 March 1991.
- [35] B. Blake. *Jane's Radar and electronic Warfare systems*. An International Thomson Publishing Company, 1995-96.
- [36] A. Blum. *Neural Networks in C plus plus: An object-oriented framework for building connectionist systems.* New York: Wiley, 1992.

- [37] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison Welsey, 1998.
- [38] A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the classic description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.
- [39] M. Brännström and E. Jungert. A scalable agent architecture for a dynamic sensor network. *Fundamenta Informaticae*, XXI:1–13, 2002.
- [40] E. A. Bretz. Transportation. *IEEE Spectrum*, 36(1):98–103, Jan 1999.
- [41] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex System*, 2:321–355, 1988.
- [42] T. A. Brownell. Neural networks for sensor management and diagnostics. *Proceedings of the IEEE 1992 National Aerospace and Electronics Conference*, 3:923–929, May 1992.
- [43] D. M. Buede and E. L. Waltz. Issues in sensor management. In *Proceedings* of the 5th IEEE International Symposium on Intelligent Control, volume 2, pages 839–842. IEEE Computer Society Press, September 1990.
- [44] B. H. Cantrell, G. V. Trunk, F. D. Queen, J. D. Wilson, and J. J. Alter. Automatic detection and integrated tracking. *IEEE International Radar Conference*, pages 391–395, 1975.
- [45] A. Carling. Introducing Neural Networks. John Wiley and Sons, 1992.
- [46] M. Cellario. Human-centered intelligent vehicles: Toward multimodal interface integration. *IEEE Intelligent Systems*, 16(4):78–81, July/August 2001.
- [47] C. W. Chan. Cognitive informatics: a knowledge engineering perspective. *Proceedings of the First International Conference on Cognitive Informatics*, 2002.
- [48] K. C. C. Chan, V. Lee, and H. Leung. Robust target tracking using a fuzzy filter. *IEEE International Conference on System, Man and Cybernetics*, 5:4464–4467, 1995.
- [49] K. C. Chang and Y. Bar-Shalom. Joint probabilistic data association for multi-target tracking with possibly unresolved measurements and maneuvers. *IEEE Transactions on Automatic Control*, 29:585–594, July 1984.
- [50] J. Chen, H. Leung, T. Lo, J. Litva, and M. Blanchette. A modified probabilistic data association filter in real clutter environment. *IEEE Transactions on Aerospace and Electronic Systems*, 32:300–313, 1996.
- [51] J. F. Chen. *Plant Foods and Nutritional Immunology: A Scientific Approach*. Bright Ideas Press, Provo, UTAH, USA, 1996.

- [52] M. Chester. Neural Networks A Tutorial. Prentice Hall International, 1993.
- [53] E. H. Chi and S. K. Card. Sensemaking of evolving web sites using visualization spreadsheets. *IEEE*, pages 18–25, 1999.
- [54] V. Chiew. A software engineering cognitive knowledge discovery framework. Proceedings of the First IEEE International Conference on Cognitive Informatics, 2002.
- [55] L. Chin. Application of neural networks in target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 30(1):281–287, January 1994.
- [56] C. Y. Chong and S. Mori. Convex combination and covariance intersection algorithms in distributed fusion. *Fusion 2001*, 1, August 2001.
- [57] C. K. Chui and G. Chen. *Kalman Filtering*. Springer Series in Information Sciences. Springer-Verlag, 1987.
- [58] A. Cichocki and R. Unbehauen. *Neural Networks for Optimization and Signal Processing*. John Wiley and Sons, 1993.
- [59] S. A. Cohen. Adaptive variable update rate algorithm for tracking targets with a phased array radar. *IEE Proceedings*, *Pt F*, 133(3):277–280, June 1986.
- [60] M. L. Cone and S. Musick. Scheduling for sensor management using genetic search. *Proceedings of the IEEE 1996 National Aerospace and Electronics Conference, NAECON 1996*, 1:150–156, May 1996.
- [61] D. J. Cook, P. Gmytrasiewicz, and L. B. Holder. Decision-theoretic cooperative sensor planning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1013–1023, October 1996.
- [62] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematical Control, Signal and Systems*, 2:303–314, 1989.
- [63] E. Daeipour and Y. Bar-Shalom. An interacting multiple model approach for target tracking with glint noise. *IEEE Transactions on Aerospace and Electronic Systems*, 31(2):706–715, April 1995.
- [64] R. Danchick and G. E. Newnam. A fast method for finding the exact N-best hypotheses for multitarget tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 29(2):555–560, April 1993.
- [65] J. E. Dayhoff. Neural Network Architectures. Van Nostrand Reinhold, 1990.
- [66] L. N. de Castro and Jonathan Timmis. *Artificial Immune Systems: A new computational intelligence approach*. Springer-Verlag, 2002.
- [67] R. V. Denton, E. I. Alcaraz, J. Llinas, and K. J. Hintz. Towards modern sensor management systems. *Science of Command and Control*, Part 3 Coping with Change:119–134, 1994. AFCEA International Press.

- [68] J. DeVilliers and E. Barnard. Backpropagation neural nets with one and two hidden layers. *IEEE Transactions on Neural Networks*, 4(1):136–141, January 1993.
- [69] P. DeWilde. Neural Networks Models: An Analysis. London: Springer, 1996.
- [70] J. S. Dong, J. Sun, and H. Z. Wang. Approach to semantic web. *Proceedings of the International Conference on Formal Engineering Methods (ICFEM'02)*, pages 156–167, 2002.
- [71] H. Durrant-Whyte, B. Grocholsky, and P. Gibbons. An information theoretic approach to decentralized control of multiple autonomous flight vehicles. Sensor Fusion and Decentralised Control in Robotic Systems III, Proceedings of SPIE, 4196:337–347, 2000.
- [72] H. Durrant-Whyte and J. Manyika. *Data Fusion and Sensor Management: a decentralized information theoretic approach*. Ellis Horwood Limited, 1994.
- [73] H. Durrant-Whyte, B.S.Y. Rao, and H. Hu. *Towards a Fully Decentralized Architecture for Multisensor Data Fusion*. University of Oxford Department of Engineering Science, 1991.
- [74] H. Durrant-Whyte and M. Stevens. Data fusion in decentralized sensing networks. *Fusion 2001*, 1, August 2001.
- [75] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
- [76] S. E. Fahlman. Faster-learning variations on back-propagation: An empirical study. In D. S. Touretzky, editor, *Proceedings 1988 Connectionist Models Summer School*, pages 38–51. Morgan-Kaufman, Pittsburgh, 1988.
- [77] L. Fausett. Fundamentals of Neural Networks. Prentice-Hall International, 1994.
- [78] R. Fitzgerald. Development of practical JPDA logic for multitarget tracking by microprocessor. In *Multitarget-Multisensor tracking: Advanced Applications*, pages 1–23. Artech House, 1990.
- [79] L. J. Fogel, A. J. Owen, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, 1966.
- [80] A. C. M. Fong and S. C. Hui. An intelligent online machine fault diagnosis system. *Computing and Control Engineering Journal*, 12(5):217–223, October 2001.
- [81] T. E. Fortmann. Detection thresholds for multi-target tracking in clutter. In *Proceedings of the 1981 IEEE Conference on Decision and Control*, pages 1401–1408, 16-18 December 1981.

- [82] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Transactions on Ocean Environments*, 8:173–184, July 1983.
- [83] G. Frank. A general interface for interaction of special-purpose reasoners within a modular reasoning system. *Proceedings of the AAAI Fall Symposium*, pages 57–62, 1999.
- [84] J. A. Freeman and D. M. Skapura. *Neural Networks Algorithms, Applications, and Programming Techniques*. Addison-Wesley, 1992.
- [85] L. M. Fu. Neural Networks in Computer Intelligence. London:McGraw-Hill, 1994.
- [86] K. Funahashi. On the approximation realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.
- [87] K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:801–806, 1993.
- [88] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, Man and Cybernetics*, 16(1):122–128, 1986.
- [89] S. Grossberg. Classical and instrumental learning by neural networks. In *Progress in theoretical biology*, volume 3, pages 51–141. New York, Academic Press, 1974.
- [90] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.
- [91] G. Hager and H. F. Durrant-Whyte. Information and multi-sensor coordination. *Uncertainty in Artificial Intelligence*, 2(381-394), 1988.
- [92] D. Hall. An introduction to multisensor data fusion. TechReach Inc, 1998.
- [93] D. L. Hall. *Mathematical Techniques in Multisensor Data Fusion*. Artech House, 1992.
- [94] K. M. Harmeyer and K. C. Wetzel. Brain research and implications for engineering education. *IEEE, FIE'96 Proceedings*, pages 466–470, 1996.
- [95] S. Haykin. *Neural Networks A comprehensive foundation*. Macmillan College Publishing Company, Inc., 1994.
- [96] D. O. Hebb. The organisation of behavior. Wiley, New York, 1949.
- [97] R. Hecht-Nielsen. Neurocomputing. Addison-Wesley, 1990.
- [98] B. J. Hellstrom and L. N. Kanal. Knapsack packing networks. *IEEE Transactions on Neural Networks*, 3(2):302–307, March 1992.

- [99] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.
- [100] K. J. Hintz and G. McIntyre. Goal lattices for sensor management. *Proceedings of SPIE, Signal Processing, Sensor Fusion, and Target Recognition VIII*, 3720-25:249–255, April 1999.
- [101] J. H. Holland. *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan press, 1975.
- [102] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings National Academy of Sciences*, 79:2554–2558, 1982.
- [103] K. Hornik. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- [104] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.
- [105] M. Howard, D. Payton, and R. Estkowski. Coalitions for distributed sensor fusion. *Proceedings of 5th Annual Conference on Information Fusion*, 2002.
- [106] M. B. Ignagni. Separate-bias kalman estimator with bias state noise. *IEEE Transactions on Automatic Control*, 35(3):338–341, December 1990.
- [107] R. A. Iltis and P. Y. Ting. Data association in multi-target tracking: A solution using a layered Boltzmann machine. In *Proceeding IJCNN'91*, volume 1, pages 31–36. IEEE, 1991.
- [108] R. A. Iltis and P. Y. Ting. Computing association probabilities using parallel boltzmann machines. *IEEE Transactions on Neural Networks*, 4(2):221–233, March 1993.
- [109] S. S. Iyengar, D. N. Jayasimha, and D. Nadig. A versatile architecture for the distributed sensor integration problem. *IEEE Transaction on Computer*, 43(2):175–185, February 1994.
- [110] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.
- [111] T. W. Jeffrey. Track quality estimation for multiple-target tracking radars. In *Proceedings of the 1989 IEEE Radar Conference*, pages 76–79. IEEE, Mar 1989.
- [112] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: Prospect, methods and challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

- [113] N. R. Jennings, K. Sycara, and M. Wooldridge. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, 2000.
- [114] L. Jin, P. N. Nikiforuk, and M. M. Gupta. Approximation of discrete-time state-space trajectories using dynamical recurrent neural networks. *IEEE Transactions on Automatic Control*, 40(7):1266–1270, July 1995.
- [115] W. D. Jones. Keeping cars from crashing. *IEEE Spectrum*, September 2001.
- [116] H. Jung, M. Tambe, and S. Kulkarni. Argumentation as distributed constraint satisfaction: Applications and results. *Proceedings of the 5th International Conference on Autonomous Agents*, July 2001.
- [117] P. R. Kalata. The tracking index: A generalized parameter for  $\alpha$ - $\beta$  and  $\alpha$ - $\beta$ - $\gamma$  target trackers. *IEEE Transactions on Aerospace and Electronic Systems*, 20(2):174–182, March 1984.
- [118] G. Kaufman. Sensitive snakes offer thermal expertise. *DefenseNews*, page 4, May 27 June 2 2002.
- [119] S. S. Keerthi and B. Ravindran. A tutorial survey of reinforcement learning. *Sadhana Academy Proceedings in Engineering Sciences*, 19(6):851–889, 1994.
- [120] J. Kim, Y. Moon, and B. P. Zeigler. Designing fuzzy net controllers using genetic algorithms. *IEEE Control Systems*, 15(3):66–72, June 1995.
- [121] L. A. Klein. Sensor and Data Fusion Concepts and Applications. SPIE Optical Engineering Press, 1993.
- [122] L. A. Klein. *Sensor and Data Fusion Concepts and Applications*. SPIE Optical Engineering Press, 1999. Second Edition.
- [123] T. Kohonen. Self-organised formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.
- [124] J. F. Kolen and J. B. Pollack. Back propagation is sensitive to initial conditions. In R. P. Lippmann, S. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems Volume 3*, pages 860–866. San Mateo: Morgan Kaufmann, 1990.
- [125] G. A. Korn. Neural Networks and fuzzy-logic control on personal computer and workstation. The MIT Press, London, 1995.
- [126] B. Kosko. Bidirectional associative memories. *IEEE Transactions on Systems Man and Cybernetics*, 18(1):49–60, January/February 1988.
- [127] E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. A. Ioannou. High-order neural network structures for identification of dynamical systems. *IEEE Transactions on Neural Networks*, 6(2):422–431, 1995.

- [128] Y. Kosuge. Maneuvering target tracking using multiple maneuver model joint probabilistic data association. In *Proceedings IECON'91. 1991 International Conference on Industrial Electronics, Control and Instrumentation*, pages 2059–2064. IEEE, 28 October 1 November 1991.
- [129] A. Kraemer. Two speakers are better than 5.1. *IEEE Spectrum*, pages 71–74, May 2001.
- [130] G. Kramer. Envisioning a radar-based automatic road transportation system. *IEEE Intelligent System*, 16(3):75–77, May/June 2001.
- [131] S. Kraus. Beliefs, time and incomplete information in multiple encounter negotiations among autonomous agents. *Annals of Mathematics and Artificial Intelligence*, 20(1-4):111–159, 1997.
- [132] S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence*, 104(1-2):1–69, 1998.
- [133] C. C. Ku and K. Y. Lee. Diagonal recurrent neural networks for dynamical systems control. *IEEE Transactions on Neural Networks*, 6(1):144–155, January 1995.
- [134] C. M. Kuan, K. Hornik, and H. White. A convergence result for learning in recurrent neural networks. *Neural Computation*, 6(3):420–440, 1994.
- [135] P. Kyberd. The intelligent hand. *IEE Review*, pages 31–35, September 2000.
- [136] D. K. Leedom. Final report on sensemaking symposium. *Sensemaking Symposium*, pages 1–31, 23-25 October 2001. Evidence Based Research, Inc.
- [137] D. K. Leedom. Final report sensemaking experts panel meeting. *Command and Control Research Program*, pages 1–61, 25-26 June 2002. Evidence Based Research, Inc.
- [138] D. Lerro and Y. Bar-Shalom. Interacting multiple model tracking with target amplitude feature. *IEEE Transactions on Aerospace and Electronic Systems*, 29(2):494–509, April 1993.
- [139] H. Leung. Neural network data association with application to multiple-target tracking. *SPIE*, 35(3):693–700, March 1996.
- [140] A. U. Levin and K. Narendra. Recursive identification using feedforward neural networks. *International Journal of Control*, 61(3):533–547, 1995.
- [141] C. J. Li, L. Yan, and N. W. Chbat. Powell's method applied to learning neural control of three unknown dynamic systems. *Optimal Control Application and Methods*, 16:251–262, 1995.

- [142] X. R. Li and Y. Bar-Shalom. Design of an interacting multiple model algorithm for air traffic control tracking. *IEEE Transactions on Control Systems Technology*, 1(30):186–194, September 1993.
- [143] X. R. Li and Y. Bar-Shalom. Theoretical analysis and performance prediction of tracking in clutter with strongest neighbor filters. In *Proceedings of the 34th IEEE Conference on Decision and Control*, volume 3, pages 2758–2763. IEEE, 13-15 December 1995.
- [144] H. J. Lin and D. P. Atherton. An investigation of the SFIMM algorithm for tracking manoeuvring targets. In *Proceedings of the 32nd Conference on Decision and Control*, pages 930–935. IEEE, December 1993.
- [145] L. Ljung. System Identification-theory for the user. Prentice-Hall Inc., 1987.
- [146] R. C. Luo and M. G. Kay. Multisensor integration and fusion in intelligent systems. *IEEE Transactions on Systems Man and Cybernetics*, 19(5):901–931, September/October 1989.
- [147] A. K. Mahalanabis, B. Zhou, and N. K. Bose. Improved multi-target tracking in clutter by PDA smoothing. *IEEE Transactions on Aerospace and Electronic Systems*, 26(1):113–121, January 1990.
- [148] R. Malhotra. Temporal considerations in sensor management. *Proceedings of the IEEE, NAECON 1995*, 1:86–93, May 1995.
- [149] W. W. Manges and G. O. Allgood. Wireless sensors buyer beware. *Sensors*, 18(4):18–30, April 2001.
- [150] J. M. Manyika and H. F. Durrant-Whyte. On sensor management in decentralized data fusion. *Proceedings of the 31th Conference on Decision and Control*, FP4:3506–3507, 1992.
- [151] C. Masolo. The wonderweb library of foundational ontologies preliminary report. *WonderWeb Deliverable*, 2002.
- [152] D. J. Masreliez and R. D. Martin. Approximation non-gaussian filtering with linear state and observation relations. *IEEE Transactions on Automatic Control*, 20:107–110, 1975.
- [153] M. A. Mayor. An exponentially weighted probabilistic data association filter. In *First IEEE Conference on Control Applications*, pages 1038–1043. IEEE, 13-16 Sept 1992.
- [154] B. J. McBride. Implementing the RDF model and syntax specification. *Proceedings of the Second International Workshop on the Semantic Web SemWeb 2001*, 2001.
- [155] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

- [156] G. A. McIntyre and K. J. Hintz. An information theoretic approach to sensor scheduling. In *Proceedings of the SPIE, Signal Processing, Sensor Fusion, and Target Recognition V.*, volume 2755, pages 304–312. SPIE, April 1996. Orlando, Florida.
- [157] G. A. McIntyre and K. J. Hintz. Sensor meaurement scheduling: an enhanced dynamic, preemptive algorithm. *Optical Engineering*, 37(2):517–523, February 1998.
- [158] J. M. Mendel. *A Prelude to Neural Networks: Adaptive and Learning Systems*. Prentice Hall International, 1994.
- [159] D. Michie and R. A. Chambers. Boxes: An experiment in adaptive control. In J. T. Tou and R. H. Wilcox, editors, *Machine Intelligence*, chapter 2, pages 137–152. Edinburgh: Oliver and Boyd, 1968.
- [160] A. A. Minai and R. D. Williams. Fast back propagation with adaptive decoupled momentum. In B. Soucek, editor, *Fast Learning and Invariant Object Recognition*, chapter 2, pages 9–26. John Wiley and Sons Inc., 1992.
- [161] L. Mo, X. Song, Y. Zhou, Z. K. Sun, and Y. Bar-Shalom. Unbiased converted measurements for tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3):1023–1027, July 1998.
- [162] R. Mohajeri and F. R. Fricke. Nonlinear estimation with radar observations. *IEEE Control Systems*, 18(3):64–73, June 1998.
- [163] M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.
- [164] J. Moody. Fast-learning in multi-resolution hierarchies. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems Volume 1*, pages 29–39. Morgan Kaufmann, 1989.
- [165] A. Munir and D. P. Atherton. Adaptive interacting multiple model algorithm for tracking a manoeuvring target. *IEE Proceeding-Radar, Sonar Navigation*, 142(1):11–17, February 1995.
- [166] G. Muscato. Soft computing techniques for control of walking robots. *IEE Computing and Control Engineering Journal*, pages 193–200, August 1998.
- [167] S. Musick and R. Malhotra. Chasing the elusive sensor manager. *Proceedings of the IEEE, NAECON 1994*, 1:606–613, May 1994.
- [168] S. Nagata, M. Sekiguchi, and K. Asakawa. Mobile robot control by a structured hierarchical neural network. *IEEE Control Systems Magazine*, 10(3):69–76, April 1990.

- [169] P. G. Nagy and S. G. Bier. Intelligent internetted sensor management systems for tactical aircraft. In *Proceedings of the IEEE 1988 National Aerospace and Electronics*, volume 1, pages 321–327, May 1990.
- [170] K. S. Narendra and M. A. L. Thathachar. Learning automata a survey. *IEEE Transactions on Systems Man and Cybernetics*, 4:323–334, 1974.
- [171] G. M. Nelson and R. D. Quinn. Posture control of a cockroach-like robot. *IEEE Control System*, 19(2):9–14, 1998.
- [172] G. W. Ng. Application of Neural Networks to Adaptive Control of Nonlinear Systems. Research Studies Press Ltd and John Wiley & Sons Inc., 1997.
- [173] G. W. Ng, A. Lau, and K. Y. How. Auto-tuning interactive multiple model. In *Proceedings of SPIE, Acquisition, Tracking, and Pointing XII*, pages 131–138. SPIE, 15-16 April 1998.
- [174] G. W. Ng and K. H. Ng. Sensor management what, why and how. *International Journal on Multi-Sensor, Multi-Source Information Fusion*, 1:67–75, December 2000.
- [175] G. W. Ng, K. H. Ng, and L. T. Wong. Sensor management-control and cue. *Proceeding of the International Conference on Information Fusion*, 2:WEB1–16, 10-13 July 2000. Fusion2000.
- [176] G. W. Ng, K. H. Ng, Peter L. T. Wong, C. H. Tan, and E. L. Chia. Sensor management: Coverage versus survivability a multiple objective optimization problem. In *Proceedings of SPIE, Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2003.*, pages 421–428. SPIE, April 2003. vol 5099.
- [177] K. H. Ng and G. W. Ng. On the use of utility in decision-making for sensor management. *DSO National Laboratories, Informatics Lab, Technical report number: IFL/001178/99*, 1999.
- [178] D. Nicholson, S. J. Julier, and J. K. Uhlmann. DDF: An evaluation of covariance intersection. *Fusion* 2001, 1, August 2001.
- [179] W. Niehsen and R. B. GmbH. Information fusion based on fast covariance intersection filtering. *Fusion* 2002, 2:901–904, August 2002.
- [180] Q. Pan, P. Wang, and H. Zhou. Innovation filter and its application to the IMM algorithm using Zhou model. *International Conference on Circuits and Systems*, pages 801–804, 1991.
- [181] Y. H. Pao, M. P. Stephen, and J. S. Dejan. Neural-net computing and the intelligent control of systems. *International Journal of Control*, 56(2):263–289, 1992.

- [182] D. Parker. Learning logic. *Invention Report*, pages 584–64, October 1982. Stanford University, Department of Electrical Engineering.
- [183] A. G. Parlos, B. Fernandez, A. F. Atiya, J. Muthusami, and W. K. Tsai. An accelerated learning algorithm for multilayer perceptron networks. *IEEE Transactions on Neural Networks*, 5(3):493–497, 1994.
- [184] D. L. Paul. Wicked decision problems in remote health care: Telemedicine as a tool for sensemaking. *Proceedings of the 32nd Hawaii International Conference on System Sciences*, pages 1–10, 1999.
- [185] B. A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1:263–269, 1989.
- [186] L. I. Perlovsky. Fuzzy tracking of multiple objects. In *Proceedings of the 1991 IEEE workshop on neural networks for signal processing*, pages 589–592, 30 Sept 1 October 1991.
- [187] G. Petterson, L. Axelsson, T. Jensen, M. Karlsson, and A. Malmberg. Multi-source integration and temporal situation assessment in air combat. In *Proceedings of Information, Decision and Control, IDC 99*, pages 371–375, 1999.
- [188] F. J. Pineda. Recurrent backpropagation and the dynamical approach to adaptive neural computation. *Neural Computation*, 1(2):161–172, 1989.
- [189] L. J. Pinson. Electo-Optics. John Wiley and Sons, Inc., 1985.
- [190] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceeding IEEE*, 78:1481–1497, 1990.
- [191] R. Popoli. The sensor management imperative. In *Multitarget-Multisensor Tracking: Applications and Advances*, volume 2, pages 325–392. Artech House, 1992.
- [192] R. L. Popp. Parallelization of a multiple model multitarget tracking algorithm with superlinear speedups. *IEEE Transactions on Aerospace and Electronic Systems*, 33(1):281–290, January 1997.
- [193] R. S. Pressman. *Software Engineering: A practitioner's approach*. Software Engineering Series. McGraw-Hill International, 2000.
- [194] T. Quach and M. Farooq. Application of the MHT to track formation of a single maneuvering acoustic source in clutter. In *Proceedings of the 36th Midwest Symposium on Circuits and Systems*, volume 1, pages 684–687, 16-18 August 1993.
- [195] V. Raghavan, K. R. Pattipati, and Y. Bar-Shalom. Efficient L-D factorization algorithms for PDA, IMM, and IMMPDAF filters. *IEEE Transactions on Aerospace and Electronic Systems*, 29(4):1297–1310, October 1993.

- [196] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, December 1979.
- [197] J. A. Roecker. Track monitoring when tracking with multiple 2-d passive sensors. *IEEE Transactions on Aerospace and Electronic Systems*, 27(6):872–876, November 1991.
- [198] J. S. Rosenschein and G. Zlotkin. Designing conventions for automated negotiation. *AI Magazine*, 15(3):29–46, 1994.
- [199] P. L. Rothman and S. G. Bier. Evaluation of sensor management systems. *Proceedings of the IEEE, NAECON 1989*, pages 1747–1752, May 1989.
- [200] Y. Ruan, P. Willet, and R. Streit. Assignment versus PMHT in multi-sensor/multi-target tracker. *Proceedings of the IEE (Part B) conference on Target Tracking: Algorithms and Applications*, November 1999.
- [201] D. E. Rumelhart, G. E. Hinton, and R. J. William. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: explorations in the microstructure of cognition*, chapter 8, pages 318–362. London: MIT Press, Cambridge MA, 1986. Vol. 1.
- [202] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing: explorations in the microstructure of cognition*, volume 1. London: MIT Press, 1986.
- [203] A. Schiffman. Electronic nose. IEEE Spectrum, 1998.
- [204] H. P. Schwefel. *Numerical optimization for computer models*. Chichester: Wiley and Sons, UK, 1981.
- [205] D. Sengupta and R. Iltis. Neural solution to the multitarget tracking data association problem. *IEEE Transactions on Aerospace and Electronic Systems*, 25(1):96–108, January 1989.
- [206] K. T. Seow and K. Y. How. Multi-agent negotiation for collaborative assignment using the metaphor of belief-desire-joint-intention. Proceedings of the First International Conference of Autonomous Agents and Multi-agent Systems, 2002.
- [207] G. Shafer. A Mathematical Theory of Evidence. Princeton University Press, 1976.
- [208] C. Sierra, N. R. Jennings, P. Noriega, and S. Parsons. A framework for argumentation-based negotiation. *Proceedings of The Fourth International Workshop on Agent Theories, Architectures and Languages*, pages 167–182, 1997.

- [209] S. Silven. A neural approach to the assignment algorithm for multiple-target tracking. *IEEE Transactions on Ocean Environments*, 17(4):326–332, October 1992.
- [210] P. Smets and R. Kennes. The transferable belief model. *Artificial Intelligence*, 66:191–243, 1994.
- [211] P. Smolensky. Harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: explorations in the microstructure of cognition*, chapter 6, pages 194–281. London: MIT Press, Cambridge MA, 1986. Vol. 1.
- [212] L. K. Soh and C. Tsatsoulis. Satisficing coalition formation among agents. Proceedings of the First International Conference of Autonomous Agents and Multi-agent Systems, 2002.
- [213] E. D. Sontag. Feedback stabilization using two-hidden-layer nets. Technical Report Report SYCON-90-11, Rutgers University, Department of Mathematics, Rutgers Center for Systems and Control, October 1990.
- [214] B. Soucek. *Fast Learning and Invariant Object Recognition*. John Wiley and Sons, Inc., 1992.
- [215] J. A. Stover, D. L. Hall, and R. E. Gibson. A fuzzy-logic architecture for autonomous multisensor data fusion. *IEEE Transactions on Industrial Electronics*, 43(3):403–410, June 1996.
- [216] D. Stromberg and G. Pettersson. Parse tree evaluation a tool for sensor management. *Fusion'98*, 1998.
- [217] R. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3(1):9–44, August 1988.
- [218] W. Tao and H. Burkhardt. Application of fuzzy logic and neural network to the control of a flame process. In *Intelligent Systems Engineering*, pages 235–240. Institute of Electrical Engineers, London, September 1994. IEE Conference publication No. 395.
- [219] J. G. Taylor. *Neural Networks and their Applications*. John Wiley and Sons, 1996.
- [220] G. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.
- [221] J. B. Treece. Adding measurement dimension can decrease the probability of correct correlation. *SPIE*, 3373:320–330, 1998.
- [222] A. C. Tsoi and A. D. Back. Locally recurrent globally feedforward networks: A critical review of architectures. *IEEE Transactions on Neural Networks*, 5(2):229–239, March 1994.

- [223] T. Tsuji, A. Jazidie, and M. Kaneko. Cooperative multi-arm robots. *IEEE Transactions on Systems Man and Cybernetics*, 27(5):862–867, October 1997.
- [224] L. P. J. Veelenturf. *Analysis and Applications of Artificial Neural Networks*. Prentice Hall International(U.K.), 1995.
- [225] S. vL Henkel. Research and developments. Sensors, 18(4):14, April 2001.
- [226] Yu. G. Vlasov, A. V. Legin, E. A. Bychkov, and M. I. Ozernoy. Chemical sensors based multi-element system for sea environment monitoring and control. *OCEANS 94*′, 2:341–345, September 1994.
- [227] E. Waltz and J. Llinas. Multisensor Data Fusion. Artech House, 1990.
- [228] M. D. Waltz and K. S. Fu. A heuristic approach to reinforcement learning control systems. *IEEE Transactions on Automatic Control*, AC-10(4):390–398, October 1965.
- [229] Y. Wang. On cognitive informatics. *Proceedings of the First IEEE International Conference on Cognitive Informatics*, 2002.
- [230] Y. Wang and Y. Wang. Cognitive models of the brain. *Proceedings of the First IEEE International Conference on Cognitive Informatics*, 2002.
- [231] P. D. Wasserman. *Neural Computing Theory and Practice*. Van Nostrand Reinhold: New York, 1989.
- [232] P. D. Wasserman. *Advanced methods in neural computing*. Van Nostrand Reinhold: New York, 1993.
- [233] C. J. C. H. Watkins. Learning from Delayed Rewards. PhD thesis, Cambridge University, Cambridge, England, 1989.
- [234] G. A. Watson and W. D. Blair. Interacting acceleration compensation algorithm for tracking maneuvering targets. *IEEE Transactions on Aerospace and Electronic Systems*, 31(3):1152–1158, July 1995.
- [235] C. H. Wee, K. S. Lee, and W. H. Bamband. *Sun-Tzu War and Management*. Addison-Wesley Publishing Company, 1996.
- [236] G. Weiss. Multiagent System: A Modern Approach to Distributed Artificial Intelligence. The MIT Press, London, U.K., 1999.
- [237] P. J. Werbos. *Beyond regression: new tools for prediction and analysis in the behaviour sciences.* PhD thesis, Harvard University, 1974.
- [238] P. J. Werbos. Consistency of HDP applied to simple reinforcement learning problem. *Neural Networks*, 3:179–189, 1990.

- [239] P. Willet, Y. Ruan, and R. Streit. The PMHT: Its problems and some solutions. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):738–754, July 2002.
- [240] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.
- [241] D. J. Willshaw and C. Ver Der Malsburg. How patterned neural connections can be set up by self-organisation. *Proceedings of the Royal Society London*, B194:431–435, 1976.
- [242] M. Wooldridge. Agent-based software engineering. *IEE Proceedings on Software Engineering*, 144(1):26–37, 1997.
- [243] M. Wooldridge and N. R. Jennings. Intelligent agents: theory and practice. *The knowledge Engineering Review*, 10(2):115–152, 1995.
- [244] J. K. Wu. Neural Networks and Simulation methods. New York: M. Dekker, 1994.
- [245] W. R. Wu and P. P. Cheng. A nonlinear IMM algorithm for maneuvering target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 30(3):875–886, July 1994.
- [246] S. J. Yen and A. L. P. Chen. A graph-based approach for discovering various types of association rules. *IEEE Transactions on Knowledge and Data Engineering*, 13(5):839–845, September/October 2001.
- [247] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *Artificial Intelligence*, 10(5):673–685, September/October 1998.
- [248] Q. G. Zhang and A. Benveniste. Wavelet networks. *IEEE Transactions on Neural Networks*, 3(6):889–898, 1992.
- [249] Y. Zhang and L. Lu. A knowledge-based data fusion strategies for airborne multisensor surveillance system. In *Proceedings of IEEE Systems, Man and Cybernetics Conference*, pages 646–649. IEEE, 17-20 October 1993.
- [250] Z. Zhang and K. J. Hintz. OGUPSA sensor scheduling architecture and algorithm. In *Proceedings of the SPIE 1996, Signal Processing, Sensor Fusion, and Target Recognition V.*, volume 2755, pages 296–303. SPIE, April 1996. Orlando, Florida.
- [251] G. Zlotkin and J. S. Rosenschein. Mechanism design for automated negotiation and its application to task oriented domains. *Artificial Intelligence*, 86(2):195–244, 1996.

## Bayesian INDEX association techniques 111 classifier 205 network 102, 106, 113 Belief, desire and intention (BDI) 25, 190 Belief function 208 Bio-inspired algorithms 29, A data association 125 Acoustic sensor system 64-6, 70, 121 Blending of data 104 Active sensors 64 Best linear unbiased estimate Activities 24 (BLUE) 166, 168 Adaptive Boltzmann machine 38 backpropagation 39 learning 53 critic learning 51 network 126 interacting multiple model Brain 9-11, 36-8, 90, 154 (AIMM) 144 Brain-state-in-a-box (BSB) 38 learning rate 46 linear neuron 38 resonance theory (ART) 43 Case-based reasoning 60 Agents 20 Centralized proactive 21 fusion architecture 93, 96 social 21 Chaining for decentralized sensor backward 60 management 190 forward 60 Agent-oriented Chemical sensor 72 software development 22 Chromosomes 30-5 Aggregation 15, 101, Classification 40, 100, 178, 205 Algorithms Classifier see Bayesian biologically inspired 11, 29 Clustering 101 Artificial Cognitive immunology system 7, 210 informatics 90 intelligence 38 intelligence 90, 103 Assignment matrix 127 COMINT 65 Association

Command and control (C2)

Competing edge 123-4

Control 177

Competitive data 104-5,

Complementary data 104-5 Computational intelligence 103

Conjugate gradient decent 49

Convex combination 166, 175

Cooperative intelligence 153-5

20, 79, 106

## B Backpropagation algorithm 45

Awareness state 92

graph 121-4

Associative search learning 51

probability 116

with momentum 46, 55

Bar-Shalom and Campo combination 166, 168

| Correlation 98                     | strategies 30                       |
|------------------------------------|-------------------------------------|
| strength 112, 122                  | Euclidean                           |
| learning 52                        | distance 112                        |
| Covariance intersection 166-7, 175 | Expert system 59-61                 |
| Crossover 33-6                     |                                     |
| Cueing 185                         | F                                   |
|                                    | Fitness 31-2                        |
| D                                  | Fixed-gain filter 133-5             |
| Data                               | Foliage penetrating (FOPEN) 68      |
| alignment 86-7                     | Forward chaining 60                 |
| association 109                    | Frequency 64, 87, 105               |
| fusion 74, 79, 82, 84              | Fusion                              |
| incest 87                          | algorithm 164                       |
| structure 17-9                     | engine 89                           |
| Decentralized                      | of biologically inspired            |
| control 154, 212                   | algorithms 61                       |
| fusion architecture 93-4           | of multiple sources 86              |
| sensor management 190              | process 82, 86, 89, 97              |
| tracking 170                       | Fuzzy                               |
| Decision-making 181, 184-5         | logic 55, 62, 127                   |
| Decision tree 102                  | rule 40, 188                        |
| Defuzzification 58                 | system 55, 188                      |
| Delta-bar-delta (DBD) 39, 46       |                                     |
| Deghosting 83, 106, 121            | G                                   |
| Dempster-Shafer classifier 207     | Gas sensor 72                       |
| Deployment 14, 180-1               | Gating 111, 150-1                   |
| Descriptive technique 185          | Gauss-Newton 48                     |
| Diagrams                           | Genetic algorithm 31, 33, 36, 62    |
| software design 15                 | Genome 30                           |
| Distributed                        | Gradient methods                    |
| fusion architecture 94             | first order 45                      |
| problem solving 21                 | second order 47                     |
| Doppler see Radar                  | Graph-based                         |
| Dynamic situation 80               | data association 121                |
|                                    | inference 102                       |
| E                                  | Grossberg learning 44               |
| Electronic support measure (ESM)   |                                     |
| 65, 122                            | Н                                   |
| Electro-optic systems 66, 70       | Hebbian 52-3                        |
| ELINT 65                           | Hessian matrix 47-9                 |
| Elitism 35                         | Hierarchical fusion architecture 94 |
| Evidence combination 102           | High-level fusion process 82-4, 106 |
| Evolutionary                       | Human                               |
| computing 30-1                     | sensory system 65                   |
| programming 31                     | HUMINT 65                           |

| Hybrid architecture 98                      | Learning                         |
|---|----------------------------------|
|   | algorithm 43-5                   |
| I   | rate 45-8                        |
| Identification 100                          | Levenberg-Marquardt method 48    |
| Identity 100                                | Likelihood function 110, 126     |
| Imagery see radar                           | Liquid state materials 63        |
| IMINT 65                                    | Locomotion 8                     |
| Immune system 210-13                        | Logic 80                         |
| Impact assessment 84, 90, 100               | Low-level fusion process 82      |
| Inference                                   |                                  |
| fuzzy linguistic 57                         | M                                |
| Information                                 | Man-made sensors 66              |
| fusion 97, 157                              | Magnetic                         |
| presentation 91                             | field 71                         |
| processing 153                              | sensor systems 71                |
| theory 99                                   | Measurement fusion 157, 164      |
| Intelligent                                 | Measurement-oriented approach    |
| fusion process 80-1                         | 114                              |
| system 5-11, 13, 212                        | Memory                           |
| Interactive multiple model 129-30           | cells 211                        |
| auto-tuning 142                             | structure for cognitive          |
| non-linear 144                              | intelligence processing 92       |
| Infrared search-track sensors (IRST)        | Microsensors 73                  |
| 69-70                                       | Millimeter-wave see Radar        |
|   | Moving target indicator 69       |
| J   | Multi-agent                      |
| Joint probabilistic data association filter | negotiation 21                   |
| (JPDAF) 111, 119-21                         | systems 21-2                     |
|   | Multiple                         |
| K   | hypotheses tracking (MHT)        |
| Kalman filter 136-8, 145, 161               | 110-1, 113-5,                    |
| decentralized 158                           | model approach 138               |
| Knowledge 80                                | model probabilistic data         |
| Knowledge-based                             | association filter               |
| data association method 127                 | (MMPDAF) 118, 130                |
| system 14, 61                               | platform 74, 88                  |
| Knowledge engineering 61                    | sensor integration 74            |
| Kohonen                                     | Mutation 30-6                    |
| network 38, 43                              |                                  |
| self organizing learning                    | N                                |
| algorithm 53                                | Nature 7, 125,                   |
| -   | Naturalistic decision making 103 |
| L   | Nearest neighbour 112-3          |
| Large-scale systems 106                     | Negotiation see Multi-agent      |
| Laser see Radar                             | Nerve cell 37                    |

| Network                              | R                               |
|--------------------------------------|---------------------------------|
| centric warfare 79, 106              | Radar                           |
| multilayer 43                        | continuous Doppler 69           |
| single layer 40, 43                  | imagery 68                      |
| topology 40                          | laser 69, 105                   |
| Network of nodes 155, 175            | millimeter-wave 64, 69          |
| Neural data association method 125   | pulse Doppler 69                |
| Neural networks                      | Radial basis function 39-40, 43 |
| feedforward 40                       | Reinforcement learning 50       |
| recurrent 40                         | Resource                        |
| Newton method 48                     | allocation 107                  |
| Normative technique 184              | management 85, 92               |
| Normalized distance function 167     | Retrieval strategies 102        |
|                                      | Robots 6, 156                   |
| 0                                    | Rule-based reasoning 7, 61      |
| Object assessment 97                 |                                 |
| Object-oriented software engineering | $\mathbf{S}$                    |
| 26                                   | Scheduling algorithm 183        |
| Object oriented modelling 14         | Selection                       |
| Ontology 213                         | Boltzman 35                     |
| Ontological engineering 61, 107      | Self-organizing                 |
| Operator 33, 35                      | behavior 154                    |
| OSINT 65                             | feature map 38                  |
|                                      | Semantic web 213                |
| P                                    | Sensemaking 91                  |
| Particle filter                      | Sensing platforms               |
| 129-30, 148, 201                     | multiple 74                     |
| Passive sensor systems 65-7, 110     | single 74                       |
| Pattern recognition 102, 107         | Sensor                          |
| Perceptron learning 44               | seismic                         |
| Physical models 102                  | systems 64-6, 73                |
| Plan recognition 107                 | Sensor management 177-81, 185   |
| Probabilistic                        | Situation                       |
| data association 115                 | assessment 84, 100              |
| multiple hypothesis tracking         | awareness 155                   |
| (PMHT) 106                           | Small-scale systems 106         |
| network 102                          | Speed-dependent gain 135        |
| Probability 34, 116,                 | Spiral development process 23   |
| Processor 93-5, 131                  | Software                        |
|                                      | agent-based system 19           |
| Q                                    | design for intelligent          |
| Quantum detector 70                  | engineering 26                  |
| Quickprop 39, 49                     | systems 13                      |
|                                      | Sonar sensor system 70          |

| State                                |  |  |
|--------------------------------------|--|--|
| estimation 99, 131, 157              |  |  |
| vector fusion 157, 165               |  |  |
| Statistical decision 102             |  |  |
| Storage 17<br>Supervized learning 43 |  |  |
| Sun Tzu 86                           |  |  |
| Swarm intelligence 7, 153            |  |  |
| Synthetic aperture 68                |  |  |
| Symmetre aperture of                 |  |  |
| T                                    |  |  |
| Target                               |  |  |
| dynamics 89, 137                     |  |  |
| tracking 129, 131-3                  |  |  |
| Thermal detector 70                  |  |  |
| Time                                 |  |  |
| alignment 87                         |  |  |
| difference 87                        |  |  |
| Track fusion 165-7                   |  |  |
| Track-oriented approach 115 Track    |  |  |
| quality 167                          |  |  |
| selection 166                        |  |  |
| Trend recognition 102, 107           |  |  |
| U                                    |  |  |
| Unified modelling language (UML) 14  |  |  |
| Unsupervized learning 52             |  |  |
| Utility function 184                 |  |  |
| ***                                  |  |  |
| W<br>Widrow-Hoff learning 44         |  |  |
| Work flow 91                         |  |  |
| WOLK HOW DI                          |  |  |