University of Southampton

ABSTRACT

Faculty of Engineering

Department of Electronics and Computer Science

Doctor of Philosophy

Parallel Tracking Systems

by Carlos Henrique de Oliveira Sá Hulot

Tracking Systems provide an important analysis technique that can be used in many different areas of science. A Tracking System can be defined as the estimation of the dynamic state of moving objects based on `inaccurate' measurements taken by sensors. The area encompasses a wide range of subjects, although the two most essential elements are estimation and data association. Tracking systems are applicable to relatively simple as well as more complex applications. These include air traffic control, ocean surveillance and control sonar tracking, military surveillance, missile guidance, physics particle experiments, global positioning systems and aerospace.

This thesis describes an investigation into state-of-the-art tracking algorithms and distributed memory architectures (Multiple Instructions Multiple Data systems - "*MIMD*") for parallel processing of tracking systems. The first algorithm investigated is the Interacting Multiple Model (IMM) which has been shown recently to be one of the most cost-effective in its class. IMM scalability is investigated for tracking single targets in a clean environment. Next, the IMM is coupled with a well-established Bayesian data association technique known as Probabilistic Data Association (PDA) to permit the tracking of a target in different clutter environments (IMMPDA). As in the previous case, IMMPDA scalability is investigated for tracking a single target in different clutter environments. In order to evaluate the effectiveness of these new parallel techniques, standard languages and parallel software systems (to provide message-passing facilities) have been used. The main objective is to demonstrate how these complex algorithms can benefit in the general case from being implemented using parallel architectures.

Table of Contents

List of Figures

Acknowledgements

I wish to thank all University of Southampton personnel and especially those in the Electronic and Computer Science Department. In particular, Tony Hey for his encouragement and demonstration of high spirits all of the time. Ed Zaluska for his supervision, encouragement, constructive criticism and friendship which he has given me throughout this thesis and without which this thesis would probably never have been finished. John Shaw for his advice and provision of radar data used as validation data. Flavio Bergamaschi, a very old friend, for his inquisitive mind and the many hours we spent together discussing the contents of this thesis and most of all for his unrestricted friendship. To my wife Mylene Melly for her unlimited encouragement, patience and love.

To everyone from the Laboratorio de Sistemas Integraveis da Universidade de Sao Paulo, who has pointed me in this direction and helped substantially in obtaining CNPq sponsorship. And finally to the Conselho Nacional the Desenvolvimento Cientifico (CNPq) from Brazil who has given me the chance to accomplish this work by providing the necessary financial support.

Chapter 1 : Introduction

1.1 Motivation

High-performance computers are in demand in many different areas of science. There is now a consensus that high-performance computing can be achieved by parallel architectures and processing. Many advances are still expected in the near future because there is an increasing demand for high performance as a direct consequence of the increase in problem complexity to be solved. This demand will require the utilization of different approaches and methodologies in terms of high-performance systems. This thesis is not directly related to the evolution of high-performance technology itself, but rather to the application of parallel technology to a specific and practical application, **Tracking Systems**.

Tracking system technology has undergone considerable evolution from the theoretical point of view. Unfortunately, this theoretical evolution has not been applied, until recently, to most commercial systems. As an example, Air Traffic Management systems (ATM) are still using outdated α-β trackers [Bozic79], [Schooler75], [Kalata84]. Nonetheless, in this particular field, a few advances are about to become reality with a more updated ATM system to be fully implemented and operational around year 2000 [BarShalom92a]. As a result, most of the new research achievements are now emerging in the specialized literature, however very few publications describe parallel implementations. Because many current tracking systems implementations have relied on the massively-increased computational power of modern computers to boost their very old (vintage in some cases) tracking system algorithm, these new theoretical advances will need even more computational power to achieve all the benefits. As a direct result parallel processing will inevitably play a major breakthrough in implementing these new tracking systems in the near future. The thesis is intended to increase the understanding of **Parallel Tracking System**. Example implementations on different parallel architectures will be explored in order to evaluate and establish some common factors in parallel tracking implementations. This thesis will demonstrated the feasibility of implementing tracking system on parallel platforms and show how this can be achieved. A special emphasis is given to ATM systems as a way of presenting the generic results.

1.2 Thesis Contents

This Chapter introduces the main objectives of the thesis, the thesis contents, along with the main achievements. The summary of the thesis contents that follows is given on a chapter basis.

Chapter 2 presents a brief review about parallel systems describing the technologies currently available. It then describes the hardware to be used to implement the tracking system to be developed in this thesis. This is followed by a brief presentation of the available message passing systems and our selection to perform the parallel implementation using the target hardware system.

Chapter 3 presents a summarized review about tracking systems explaining the main problems found in this area of application, starting with a very simple explanation of the overall tracking problem. The objective is to provide a precise understanding and overview about the problem to be tackled and the complexities that may be involved in tracking system. It summarizes the main elements currently in use to solve the tracking problem. This is followed by a description of ATM systems and how they make use of tracking systems. This description establishes the basic elements of any ATM system.

Chapter 4 describes and investigates a parallel implementation of a state-of-the-art tracking algorithm, the Interacting Multiple Model (IMM). The IMM algorithm is inherently parallel because it is based on multiple models. A first parallel implementation explores the IMM filter characteristics to determine the filter dependencies with relation to an increase in the number of models supported by the filter and the scalability. The IMM is implemented (parIMM) on a number of different parallel architectures and the results are presented in terms of speed-up and efficiency.

Chapter 5 also describes and investigates a parallel implementation obtained by coupling IMM with the Probabilistic Data Association technique. The PDA technique is a well-known and effective compromise for the tracking of targets in a clutter environment. In fact the PDA technique can be subdivided into two basic algorithms : single target (or simply Probabilistic Data Association - PDA) and multitarget (Joint Probabilistic Data Association). A single target version is implemented by taking IMM coupled with PDA.

Chapter 6 presents a summarized review of the main achievements obtained in the thesis, followed by a general overview of the main results of the work carried out along with suggestion for future developments.

The thesis contain also the following appendices :

- Appendix A presents all filter algorithms used in the thesis. It starts by giving the Kalman algorithm and the Extended Kalman algorithm.
- Appendix B presents the Interacting Multiple Model algorithm. The objective of this appendix is to present the necessary formulae to make the understanding of the thesis easier.
- Appendix C presents all the equations for the data association algorithm. It presents the Probabilistic Data Association equations for single target in a clutter environment.

1.3 Main Achievements

The main achievements of the thesis are described in Chapter 4 and Chapter 5. In Chapter 4 an IMM parallel implementation is made over a MIMD distributed memory architecture. The main achievement in this chapter is the simplicity of the implementation which is done using an Single Program Multiple Data (SPMD) strategy. It determines in a clear manner the minimum data set to be exchanged amongst processors. It introduces an optimization that will reduce the computational algorithm overhead by up to 50% as the number of models increases. In Chapter 5 a parallel implementation of the IMMPDA is shown. As far as it is known this is the first parallel implementation of the filter. It also shows the coupling of a two well-accepted models using the IMM algorithm in conjunction with PDA.

Chapter 2 : Parallel Processing

2.1 Introduction

Computational power increased has been highly utilized in all areas of science [Fox88]. The computer industry has undergone considerably development in many different directions in recent years. This has been translated into a vast number of research and development projects as well as the marketing of several proprietary parallel and high-performance computer architectures and systems. Many of these advances have been driven by the need to solve very complex scientific problems. On the other hand single processor computers continue to increase in power by using more and more elaborate methods in the Very Large Scale Integration (VLSI) fabrication process. However such VLSI technology will inevitably reach its limit, probably by the end of this century or the beginning of the next [Spec95]. The only foreseeable solution for such a limitation is concurrency. Although there are many different proprietary parallel and high-performance architectures available, these technologies are still under development.

Because of this the currently-available parallel technology is still very fragile with many weakness. Despite an almost general consensus about parallel terminology and requirements, most research is still tailored for particular applications. Only recently have a few standards started to appear, but many different flavours of parallel systems remain in common use. Each of them is trying to provide or convince the customer that it provides the best solution.

Nevertheless, in recent years parallel processing has advanced rapidly with an increased number of publications and related work. Although the discussion of the many characteristics and idiosyncrasies of parallel processing is not the main objective of this thesis, a brief review is necessary to establish the essential elements of parallel technology. This can be subdivided into hardware and software considerations. On the hardware side the parallel algorithms developed here target only one type of parallel machine architecture, namely Multiple Instruction Multiple Data (MIMD) [Quin87], [Fox88], [Hwang87]. However the granularity of two different architectures will be used whenever possible. The first one is based on transputers and the second is the architecture of the Meiko-CS2. On the software side, the first consideration is the type of communication mechanism. The most common mechanism for MIMD architectures is the message passing system which is discussed together with a few of the available options.

2.2 Hardware

There are a wide range of hardware choices available for parallel architectures and it is necessary to identify the major ideas behind architecture classification. This will be done in order to establish the architectures to be used in the context of all available parallel architectures.

Parallel computers can be classified as Single Instruction Multiple Data (SIMD) and Multiple Instruction Multiple Data (MIMD). In a SIMD architecture all processors execute the same instruction at the same time over multiple data. All processors are controlled or ‘slaved’ to a master control unit, which is generally a relatively complicated element [Hey87]. On the other hand with MIMD architecture each processor executes its instructions independent of the others. Although there is no complex control unit in the MIMD case, usually programming in such an architecture is more difficult than with a SIMD architecture. On the MIMD side two main alternatives exists depending on the memory implementation. They are either shared memory or distributed memory. The architecture to be used throughout this thesis is based on MIMD distributed memory technology.

The description to follow will concentrate on the two MIMD distributed memory architectures to be used during the development of this project.

2.2.1 Transputers

The Transputer is a microprocessor with its own local memory and with 4 communications links to provide point-to-point connection between processors [Inmos88]. There are also various hardware interfaces to permit the transputer to be used in virtually any application. In order to provide more flexibility in the point-to-point connection special hardware switches are available to give each transputer alternative ways to interconnect itself with other transputers. The transputer point-to-point communication link along with these special switches make the construction of transputer networks of almost any size and topology possible. Each transputer link operates independently and can provide a 10Mbits/second or 20Mbits/second transfer rate. A simplified view of the internal architecture of each transputer is illustrated in Figure 1. Some examples of possible transputer topologies are given in Figure 1. As it can be seen a transputer utilises an internal 32 bit bus to interconnect its components.

In the transputer architecture used in this thesis each processing node comprises a transputer T805, working at 25Mhz, with 4Mbytes of local memory and the necessary hardware to reconfigure the link connections. The transputer system architecture used to develop the parallel tracking system is made of a physically reconfigurable array of 8 transputers, connected to a IBM PC-compatible host.

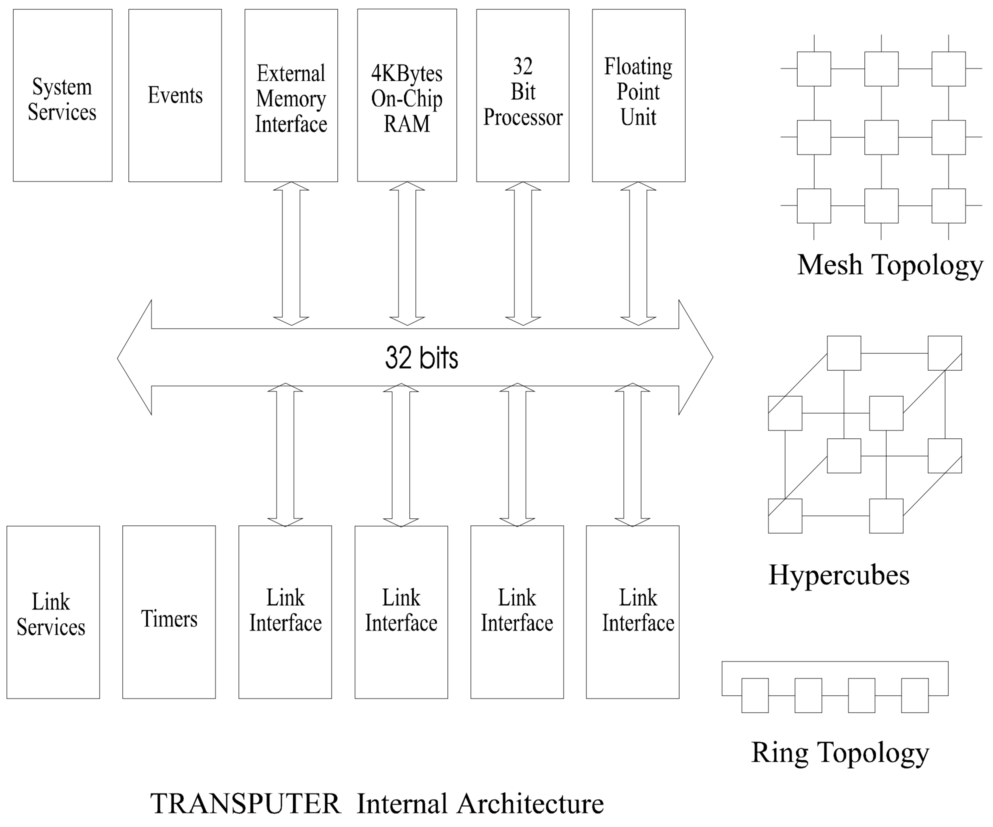
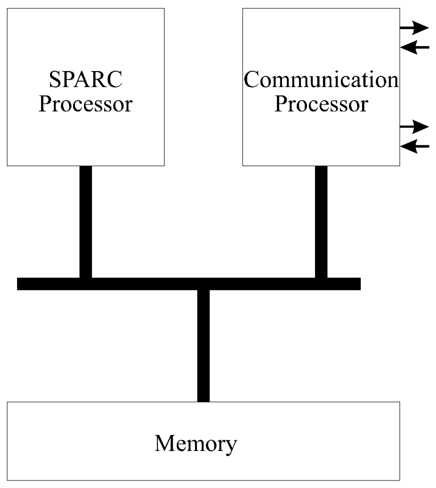


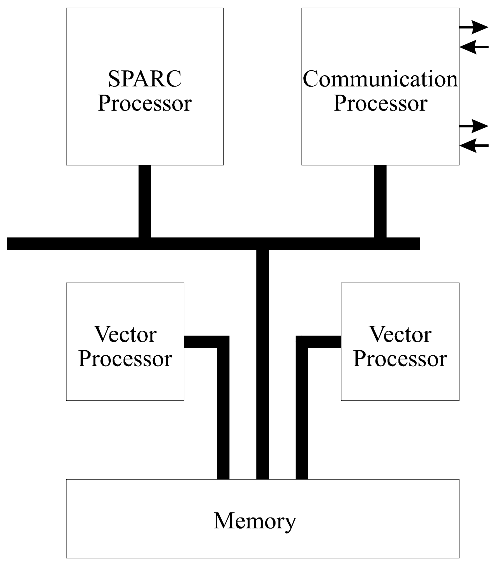
Figure 1 - Transputer Architecture

2.2.2 Meiko-CS2

The CS-2 architecture is too complex to be described in detail here and it provides a wide range of possibilities [CS2a]. In summary each processing node consists of a SPARC microprocessor unit with an optional two FUJTISU vector processor units, plus local memory and a specific proprietary communication processor. The units without the vector processors are called scalar elements, while the ones containing them are called vector elements. Figure 2 illustrates both basic elements. As it can be seen each scalar element is a fully operational computer and these units can have two possible configurations (optimised towards I/O intensive applications or to more intensive scalar processing). The vector element contains the two FUJTISU vector processors and the SPARC processor sharing a three-ported memory system. The memory is organised in 16 independent banks. The vector processors contain separate pipes for floating-point multiply, add and division and for integer operations. The multiplying and adding pipes can deliver 64 bits or two 32 bits results in IEEE format per cycle. The division pipe will take 8 cycles to deliver any result either in 64 bits or 32 bits using the IEEE format. The internal proprietary communication processor has a SPARC shared memory interface and 2 data links. Each link can provide a 50Mbytes/second transfer rate. The CS-2 parallel architecture is achieved by using the communications processor in each of its elements in conjunction with a full 8 by 8 crossbar switch as illustrated in Figure 3.

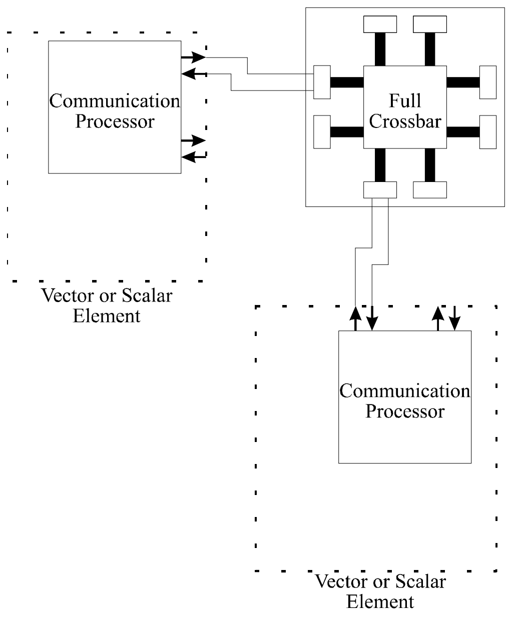


Scalar Element

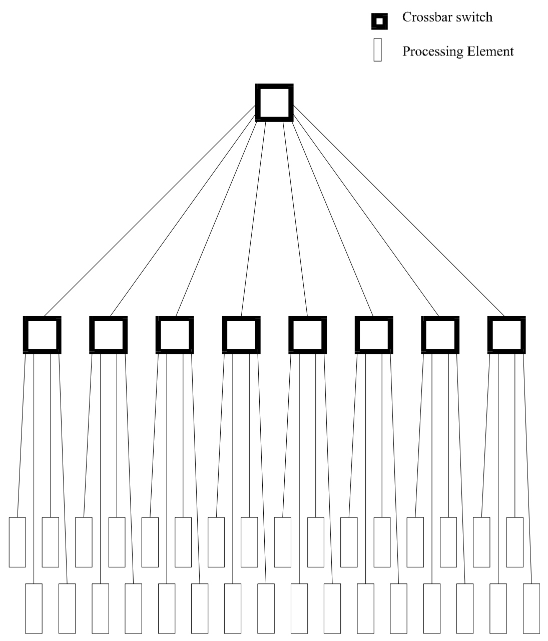


Vector Element

Figure 2 - CS-2 Main Components



CS-2 Inter-Processor Connection



CS-2 Network

Figure 3 - CS-2 Architecture

2.3 Software

Two main topics dominate the software issue, namely the communication mechanism to be used amongst parallel nodes and the choice of languages to implement the tracking algorithms.

2.3.1 Message Passing Systems

As previously described the only target architecture to be used in this thesis is the MIMD distributed memory system. This kind of architecture has been almost invariably used with message-passing systems. Conversely the majority of message-passing systems to date have been developed with distributed memory MIMD philosophy as a target. Therefore a brief background review of the message-passing technique is given, along with some of the available choices and the criteria used to select them for the purpose of developing the parallel applications of this project.

As a general overview of the message passing system it can be said that it provides two main elements for parallel processing, synchronisation and memory sharing. In other words this is equivalent to a number of independent sequential programs executing in parallel, and by using some sort of synchronisation they can be stopped in order to either obtain from or to provide information (data) to other programs. In a straightforward form a message-passing system consists of a minimal set of basic instructions to permit this data sharing. Usually it is based on constructs such as *send* and *receive,* with underlying synchronisation. As a matter of fact most of the available message-passing systems are built around these two constructs upon which more sophisticated constructs are built. This has generated a plethora of different implementations all with the objective of providing the developer easier ways to make full use of any particular parallel architecture. Most of these systems provide similar functionality but they all have their own individual idiosyncrasies. Therefore if a developer is careful enough to develop an application applying only the most basic constructs which are available in most of the message-passing systems, a migration from any particular architecture can be achieved without too much difficulty.

Amongst the early developments of message-passing systems are systems based on proprietary machines, for example CROS for the Caltech machines [Kolawa86b], IBM-EUI [Bala94], Intel NX [Pierce94], Meiko CS-2 [Barton94] and others. After these developments came more general message-passing systems targeting multiple-platforms to isolate the developer from the architectures. Typical examples of these multiple-platform message-passing systems are EXPRESS [Flower94], P4 [Butler92], PARMACS [Calkin94], PVM [Sunderam90], [Sunderam94] and others. Recently a joint committee of vendors, developers and researchers has reached a consensus upon a message-passing system standard to assist the development of more portable parallel applications. This standard is known as MPI (the Message Passing Interface) [MPI94].

The selection of the message-passing systems to be used for this project was based upon two basic criteria, availability and simplicity. For the transputer architecture two message-passing systems were selected, EXPRESS and PARAPET. For the CS-2 amongst the available choices there were PVM [CS2c], PARMACS, NX2, Elan Elite and MPI of which NX2 and MPI were selected.

EXPRESS

This system was developed by Caltech directly from CROS (Crystalline Operating System) [Kolawa86a]. It provides the developer with a easy-to-use library of low and high level communication primitives [Parasoft90a]. It also performs an automatic domain decomposition to map the physical topology into a logical topology. A transparent I/O system is supported by the Cubix library and a graphical interface by the Plotix library [Parasoft90b]. The system accommodates several different parallel implementation strategies, such as SPMD, Master-Slaves, multitasking. It is straightforward to use and provides a reliable system implementation environment.

PARAPET

ParaPET (Parallel Programming Environment Toolkit) [Debbage92a] is a set of tools developed in the Department of Electronics and Computer Science at the University of Southampton to provide message-passing system using a Single Program Multiple Data Model (SPMD) over transputer architectures [Debbage92b]. It is based upon the Virtual Channel Router [Debbage91] which is a lightweight communication mechanism. The tool is provided in a library format which is linked to a C program. The system isolates the developer from any underlying architecture constraints, such as node adjacency. However if required by the developer the system can provide access to lower-level mechanisms with improved efficiency. It is in fact a two layer system, with both low-level libraries and high-level libraries. The low-level library provides full access to the virtual channel interface, while the high-level interface provides simple constructs to send and receive messages from any node in the network.

MPI

This system is the result of a joint committee for the standardisation of message passing interfaces [MPI94]. It contains a number of high-level communication mechanisms. As in any standard the objective is to provide a common message-passing interface for different parallel architectures and in this way to accomplish straightforward migration of parallel applications. However it does not address other known problems such as I/O and multitasking.

NX

This is the message passing system created for the Intel parallel architectures and provides facilities such as topology independence and multiprocessing. It is implemented as a reduced and easy-to-use set of communication primitives linked to the parallel application as a library. The version used in this thesis is an emulation package for the CS-2 architecture [CS2b].

2.3.2 Development Languages

Two main languages were used to develop the applications in this thesis. Because no existing tracking algorithms were available for straightforward parallelization, a number of tracking algorithms had to be developed as part of the learning process and to develop the final tracking implementations of IMM and IMMPDA. This required the development of simplified algorithms which have been studied with a number of small experiments. MATLAB [Matlab93] was an essential tool to this development and consequently the overall learning process. Although MATLAB is not strictly a computer language, nevertheless it is a very powerful mathematical development tool. All of the algorithms shown in this thesis have been developed and tested for correctness and effectiveness by making use of MATLAB. The chosen language for the parallel implementation of the algorithms obtained was the ANSI-C language [Kernighan88], to facilitate migration to different platforms. Obviously, this does not mean that the identical program written in ANSI-C for a transputer architecture would run without any modifications in a CS-2 architecture. All of the message-passing constructs have to be replaced as part of the porting process. However by using ANSI-C these modifications were reduced to the message-passing constructs and corresponding connections. In this manner most of the algorithms developed here were carefully designed to indicate or isolate the machine or message-passing dependent components, to make porting a relatively straightforward task.

Chapter 3 : Tracking Systems

3.1 Introduction

Tracking technology has been used in many different areas of science ranging from physics experiments (e.g. using bubble-chambers) to military applications such as missile tracking, space-based weapon systems and civil applications such as surveillance systems, imaging processing, robotics, global position systems, and unmanned navigation systems. The objective of this thesis is to study tracking systems in general as far as possible. However, a real application has to be selected for a realistic demonstration implementation. The most-widely known commercial application of tracking systems is their use in Air Traffic Management systems (ATM), in particular air traffic control. Therefore ATM was chosen to be the basic application to demonstrate the feasibility of the parallel implementations. A few concepts are fundamental to understand better the use of tracking systems within ATM. The approach adopted here is to start with an overview of a general tracking problem. Next a discussion of ATM systems is provided to identify where tracking systems are localised. This contains a review of the use of tracking systems within ATM systems to help select basic applications to be used as a test bed for the parallel implementation demonstrators.

3.2 Tracking Problem Overview

All tracking systems are highly dependent on the type of application. However any tracking system comprises two main problems to be solved, namely *Filtering* and *Data Association* [Blackman86]. Both problems are in fact interdependent but the following examples will show them as two separate problems for simplicity and to illustrate the ideas more clearly. A more precise discussion about the interdependence is contained in the tracking application description, later on in this chapter.

3.2.1 Tracking Filter

The first problem in any tracking system is to obtain information about the object being observed and to illustrate this Figure 4 represents a target being observed by a sensor which will deliver information about that specific target to the tracking system. For simplicity, it is assumed that the sensor is located at the Cartesian origin and it rotates at a constant rate (T = sampling rate). As the sensor rotates it emits a very fine infinite radial beam that when obstructed by an object (the target) is immediately reflected back to the sensor. In this way the sensor detects any moving object every time its radial beam is reflected back. Because of imperfections in the sensors and other external elements, the reflections are corrupted by various kinds of noise. After signal processing the information contained in the reflected beam the sensor delivers, to an element called the “tracking filter” inside the tracking system, measurements in terms of range (r) and azimuth (θ) of the object being detected. As a consequence of the reflected beam signal being corrupted, the measurements delivered to the tracking system (range and azimuth) are also corrupted.

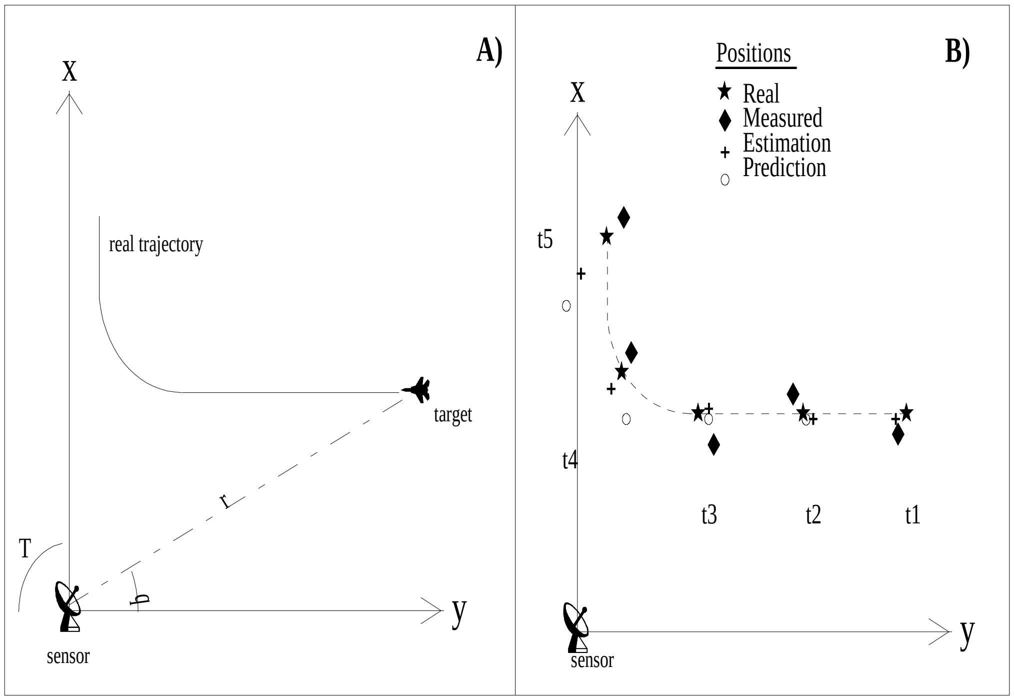


Figure 4 - Tracking Filter Problem

The whole purpose of the tracking filter is to provide a more accurate position, or estimate, from the corrupted measurements. At the same time the tracking filter will provide a prediction of where the target is expected to be in the future based on the current estimates. The tracking filter performs this estimation and prediction using mathematical models which are responsible for modelling the target trajectory or kinematics and the sensor measurement process. In the example shown the kinematics system model is a simple Cartesian straight-line constant-velocity movement and the sensor system is a straightforward conversion from Polar to Cartesian. The filter shown is assumed to be correctly initialised at t_1 and at t_2 to be already in track. Upon reception of the measurement in t_2 the filter performs the estimation and gives the resulting estimated position for t_2. At the same time the filter calculates the expected position (prediction) of the target for the next measurement (t_3). In t_3 after receiving the new measurement and having the previous prediction it calculates a new estimation for t_3 and a new prediction for t_3 . However when the measurement in t_3 arrives at the filter the previous prediction is significantly different. As a consequence the calculated estimate for t_4 is poor in comparison with the previous estimate. This problem continues for the next measurements until the target finishes its manoeuvre. In the above example, such filter can be considered as a simple α-β filter which is more precisely described below.

As it was assumed that measurements are delivered in range and azimuth and the filter is assumed to be Cartesian then a conversion is necessary and given by

$$\mathbf{z}=\begin{bmatrix} r \\ \theta \end{bmatrix}=\begin{bmatrix} \sqrt{\left(x^2+y^2\right)} \\ \arctan\left(y/x\right) \end{bmatrix}$$

which results in the following Polar-to-Cartesian measurement conversion function

$$x_m=r\cos(\theta)$$
$$y_m=r\sin(\theta)$$

Using these converted measurements the tracking filter loop is performed at each sensor target detection utilising the following equations:

current position estimation :

$$\mathbf{S}=\begin{bmatrix} x \\ y \end{bmatrix}=\begin{bmatrix} x_+ \\ y_+ \end{bmatrix}+\alpha\begin{bmatrix} x_m-x_+ \\ y_m-y_+ \end{bmatrix}$$

current velocity estimation :

$$\mathbf{V} = \begin{bmatrix} Vx \\ Vy \end{bmatrix} = \begin{bmatrix} Vx_+ \\ Vy_+ \end{bmatrix} + \frac{\beta}{T} \begin{bmatrix} x_m - x_+ \\ y_m - y_+ \end{bmatrix}$$

current position prediction :

$$\mathbf{S}_+ = \begin{bmatrix} x_+ \\ y_+ \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + T \begin{bmatrix} Vx \\ Vy \end{bmatrix}$$

current velocity prediction :

$$\mathbf{V}_+ = \mathbf{V}$$

where in the above equations **S** is the position state estimation that contains the Cartesian positions *x* and *y* of the target, $\mathbf{S}_+$ is the position state prediction containing the predicted positions x_+ and y_+ , **V** is the velocity state estimation with velocities *Vx* and *Vy* in the Cartesian plane, $\mathbf{V}_+$ is the velocity state prediction which is identical to **V**, and the estimation gain factor α and β for the estimation of the position and velocity states respectively [Gul89].

As can be seen the tracking filter performance depends on the values of α and β. If these coefficients are independent of the sampling rate, the filter is said to be stationary (fixed gain) otherwise it is non-stationary. The filter is said to be adaptive if the coefficients depend on the estimated value, otherwise is non-adaptive [Farina80]. As the filter assumed here is stationary, it starts to diverge whenever the target begins a manoeuvre. Because of the very nature of the filter chosen to estimate the target trajectory, fixed gain in this case, it is impossible to expect that this particular filter will give good estimates whenever the target manoeuvres. Therefore, the first problem in any tracking system refers to the selection of a good filter that can cope with most of the situations in the application where it is intended to be used. A common solution, (i.e. choice of filter for the above problem) is frequently based on a Kalman filter approach [Kalman60], [Kalman61], see also Appendix A for a basic description. In recent years an increasing number of papers have been published to cope with every possible situation [Gholson77], [Ricker78]. More recently a newly developed technique utilises a multitude of different Kalman filters concurrently to try to encompass most of the manoeuvres found in a specific tracking application. With this new technique, called *Multiple Model*, the most suitable filter is chosen adaptively according to the system state. The multiple model technique has also many variations and can be used in different applications. Amongst the *Multiple Model* techniques the **Interacting Multiple Model** (Appendix B) is the most cost-effective implementation developed to date and it has been chosen as the basic filter element of the current project.

3.2.2 Data Association

The second problem in Tracking System refers to the initiation, continuation and elimination of tracks, which is generally called data association. A typical example of data association continuation is illustrated in Figure 5.A. In the figure the two targets are moving at constant-velocity in a straight-line and it is assumed that two filters are already correctly initialized and they are each tracking the two targets. For simplicity the filters assumed are, as in the previous example, both α-β filters. While the two targets are well apart there is no difficulty in selecting which measurement belongs to each track, and in therefore performing the tracking filtering. In figures Figure 5.B, Figure 5.C and Figure 5.D three possible association problems are illustrated. In Figure 5.B when the two tracks get too close together and two measurements are received no simple decision is possible. A natural solution would be to associate with a particular track that measurement which is physically closer. However this does not guarantee that the correct association was made. Another possible situation ocurrs when there is a missing detection, as in Figure 5.B. In this case only one measurement is available to be associated with two tracks. And finally a more complicated situation arises when alongside the real measurements of the target position pertubations and clutter are also delivered by the sensor. In this case each target might have more than one measurement to choose from for the association process.

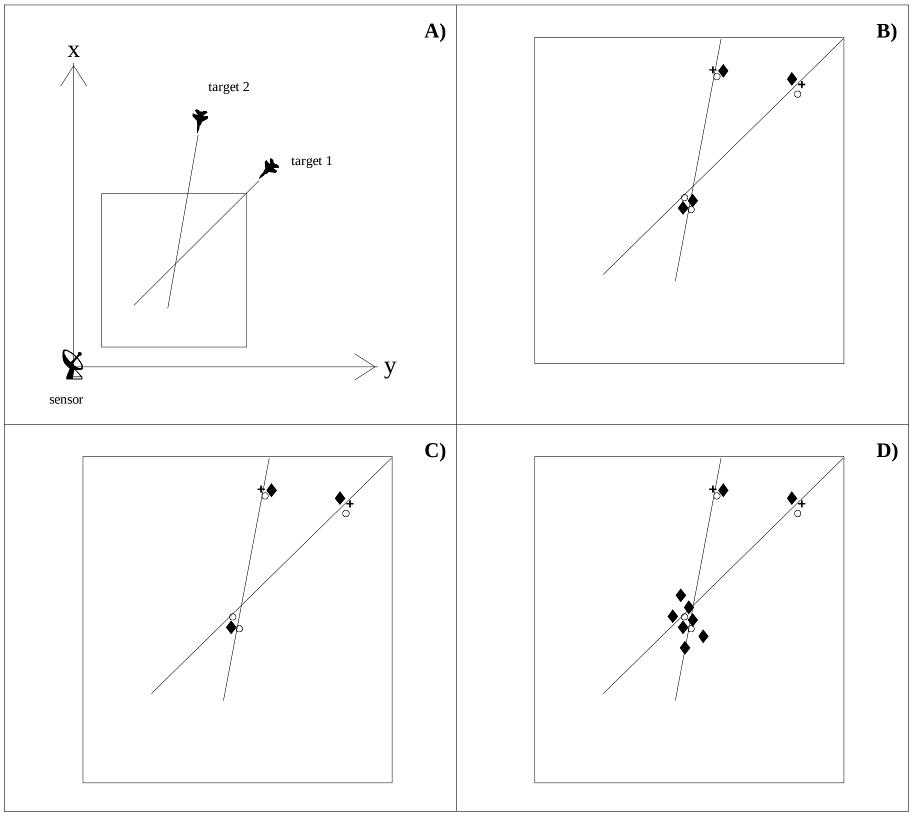


Figure 5 - Data Association Problem

In any of these examples the major problem is to select a specific mechanism to assist in the association process. Any simple mechanism depending on the application can lead to wrong associations which can then degrade the tracking until the track is lost. This is mainly caused by a divergence in the filter prediction position. The divergence occurs because the tracking filter received a wrong measurement from the data association and consequently the estimate and prediction delivered by the filter might not have any connection with the actual target position now or in the future. The prediction is then returned to the data association to determine where to expect the next measurements for that track, however because it is an incorrect prediction no correct measurement can be associated with the track. If this process continues, after a few detections the filter will be trying to estimate something that has no resemblance with the real target trajectory and consequently that track will be lost. In summary, data association continuation determines which measurements belong to which tracks utilising specific criteria to minimise the problems discussed above. There are two other related problems in data association, initiation and deletion of tracks. Initiation determines whenever a new track has to be established and hence a new tracking filter initiated. Deletion is the opposite problem when tracks cease to exist and hence their tracking filter should be terminated. As in the continuation case there are a number of different possibilities that can complicate these two processes. In practice these three data association problems cannot be considered independently because they are dependant on each other.

A number of different techniques have been employed to track targets [BarShalom88], [BarShalom78]. Each one of these techniques is adequate and possesses its own drawbacks. Initially the whole process was manual [Shaw86], but this can only permit the supervision and control of a few tracks and it also makes the system extremely limited. The first attempt to make an automatic tracking system dates from 1964 [Sittler64] and was a major breakthrough, establishing the data association process and the taxonomy currently used. The theory was based on a probabilistic approach, to make the association measurement process a more tractable one for implementation on digital computers. Bayesian probabilistic theory has been used in most of the more recent developments to determine the probability of each measurement association. Because no "a priori" information is known about the trajectories, in the case of an optimal solution, it is necessary to determine all possible associations and to store them until further measurements arrive. As they arrive, some associations can be discarded due to their low probabilities. At the same time, some of these associations have to be maintained while new ones are generated [Blackman86]. This method, if fully applied, results in a computer implementation that is not practical because of the exponential-increasing number of associations. A number of techniques have already been developed to reduce the increasing number of associations or hypotheses. These hypothese-reduction techniques are based on either merge or pruning of some of the hypotheses. Amongst these there are the Multiple Hypothesis Tracker [Reid79], Probabilistic Data Association [BarShalom78] and Joint Probabilistic Data Association [Fortmann83], see also Appendix C.

3.3 Air Traffic Control as a Tracking Application

3.3.1 Introduction

The proliferation and growing sophistication of civilian and military surveillance systems has generated a considerable interest in techniques to track an increasing number of objects whose measurements are obtained from as many different sensors as possible [BarShalom88]. Although numerous theoretical and technical advances have been made in recent years, most of them are still classified for military purposes as they were 40 years ago when the first work in this area appeared [Kailath68a].

The theoretical and technological complexity of the recent advances has precluded their application to civil aviation. However with recent advances in aerospace and computer technology and the continuing increase in civil aviation transportation, many improvements are still expected in the near future for ATM [FAA85], [Fischetti86], [Brooker88]. A complete discussion of all these advances and how they can applied to improve ATM system is beyond the scope of this thesis. Nonetheless some of the advances in computer technology are being applied to ATM systems because hardware costs are rapidly diminishing and computational power is correspondingly increasing. In fact these advances have been exploited in recent years, but always by replacing old, relatively slow computers by similar ones with faster processing, preferably from the same manufacturer [Hunt87], [Goillau89]. Therefore it seems that the natural path to make use of high performance and parallel computer computational power allied with more advanced tracking systems will be the next major breakthrough in civil aviation. In practice, parallel technology has been available for some time but applied only to radar digital processing as described in [Bergland72], and more recently in [Franceschetti90], [Turner92], [Edward92]. On the other hand a major problem delaying current ATM system modernisation is the reluctance of ATM regulators and management to replace ATM tracking systems dating from 1960 and still using vintage α-β filters for modern systems with better tracking systems (algorithms and machines) [BarShalom92b]. This reluctance is caused by two basic facts: First, because current parallel technology has not achieved the necessary maturity to replace safely the old system in terms of the many strict security requirements used in ATM, such as hardware MTBF (Mean Time Between Failures) and software reliability; Second, because ATM regulators are extremely conservative and bureaucratic [Stix91]. Next an overview of ATM systems will be given. The objective is to show step-by-step each of the main components of an ATM system until the tracking subsystem to be used throughout this thesis is identified.

3.3.2 ATM Overview

ATM systems comprise many subsystems and consequently a number of different physical implementations exist [Shaw86], [Farina86]. These physical differences represent a wide range of choices influenced by a number of different factors such as the geographical location of the radars, type of radars employed, quality and type of their delivered data, communication mechanisms between the radars and the processing centres, computational power, technological limitations, economical constraints, etc. All of these elements in different proportions have contributed to the wide range of different ATM implementations. Thus Figure 6 represents a simplified illustration of a typical ATM system.

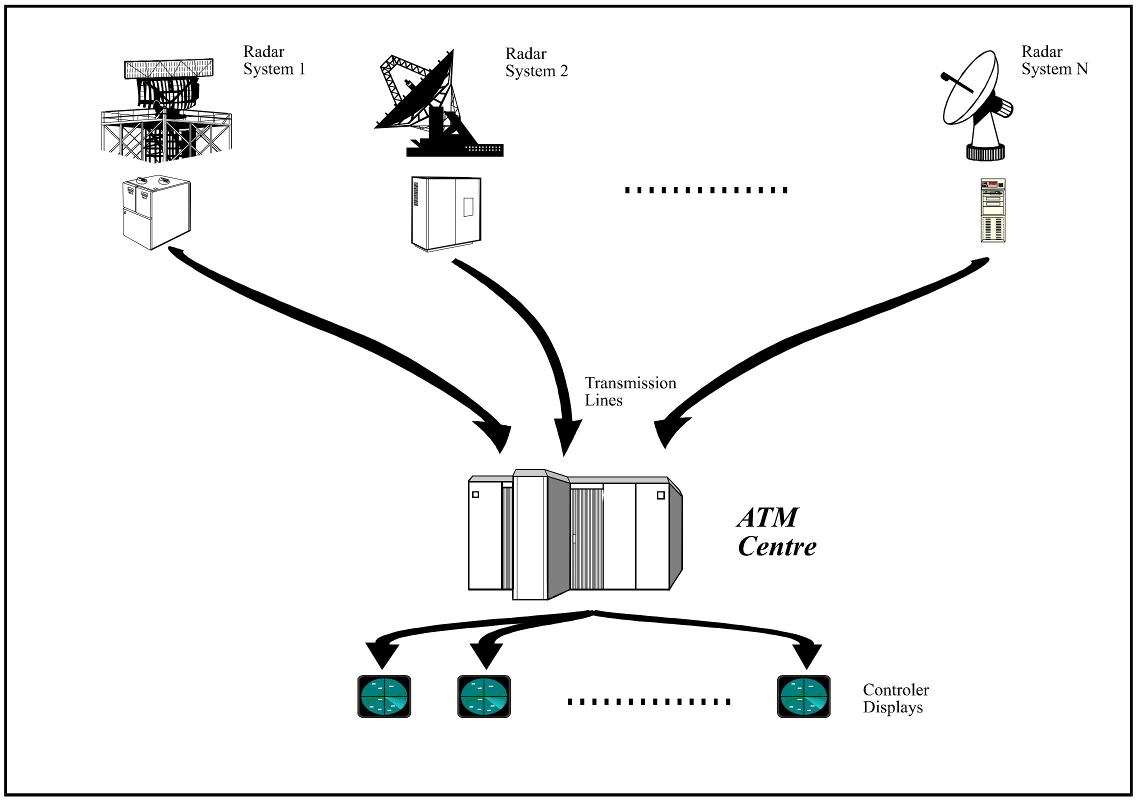


Figure 6 - Typical ATM System

The multiple sensors depicted in Figure 6 permit the implementation of mechanisms to improve the information provided by each sensor individually at the centre level [Gertz89]. The subject that studies sensor interconnection is known as multisensor Data Fusion [Blackman90]. Data Fusion in turn utilizes a number of different mechanisms to achieve a multiple-system synergy. For example, in the (already ageing) current ATM systems this synergy is achieved by combining many mental reasoning methods using manual aids [Waltz90] with automated tasks. On the other hand in a more modern and up-to-date ATM environment these mental reasoning and manual aids are expected to be automated and consequently transferred to computers. There are a vast list of possibilities to achieve such a fully automated system by employing state-of-the-art techniques. Again this will require a more open mind by ATM senior management to understand and to realize that these techniques will be introduced into the ATM environment in the near future. Because of the wideness and richness of this subject it goes far beyond the objectives of this thesis and therefore it will not be treated in detail here. Nonetheless it is important to note that any improvements that can be made in any of the stages in the ATM chain is beneficial to the overall system. It is also important to stress that multisensor data fusion along with high performance computing using parallel computers are two of the elements at the leading edge of ATM and will probably enable the next technological breakthrough for ATM systems.

This multiplicity consequently has resulted in a number of different logical architectures to make best use of all of the information provided by ATM subsystems. Amongst potential logical solutions, a possible one would be each radar system implementing a local tracking subsystem to improve the quality of the data received. This improved information would then be sent to a ATM centre where fusion of the information would be done. This fused data would require more sophisticated tracking filters and other mechanisms in order to take advantage of the diversity of information being obtained from several different sensors. A typical arrangement is illustrated in Figure 7 and before the logical distribution of ATM tasks is discussed, it is first necessary to establish the main reason why ATM systems possesses such a wide diversity. The main reason for such variety in ATM is the wide range of sensors being used, each presenting considerable diversity and at the same time having different geographical locations. In more concrete terms, radars can have different sampling rates, measurement precision and also measure different physical quantities (e.g. range and bearing versus bearing and frequency). These sensor differences make sensor alignment a necessity to bring them to the same precision and time. Also the sensors are usually geographically distributed over the surface of the Earth to improve the target information. This geographical distribution is usually arranged so that two radars cover the same area in space, providing an overlap.

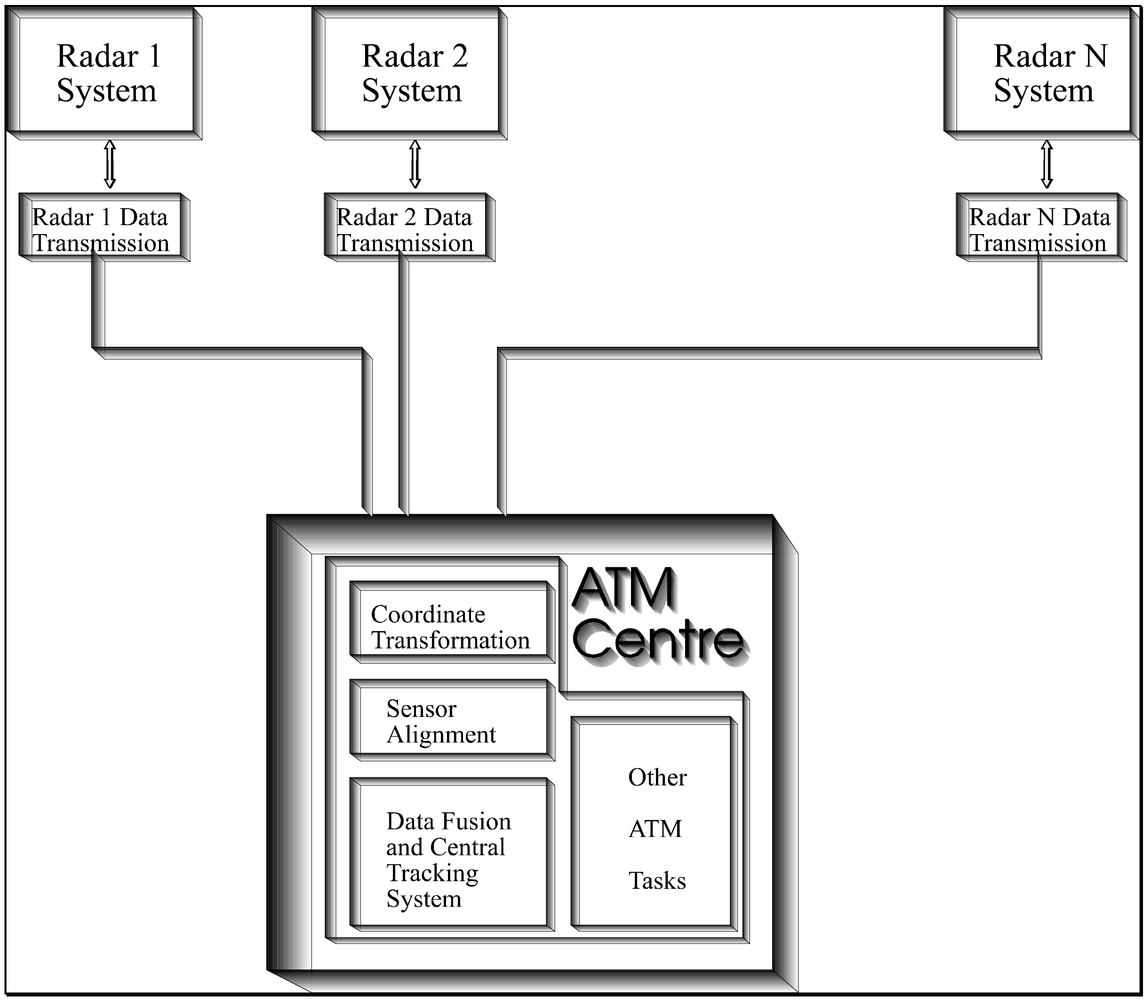

Radar 1 System
Radar 2 System
Radar N System
Radar 1 Data Transmission
Radar 2 Data Transmission
Radar N Data Transmission
ATM Centre
Coordinate Transformation
Sensor Alignment
Data Fusion and Central Tracking System
Other ATM Tasks

Figure 7 - ATM Logical Distribution

This overlapping means that one target is measured by two or more sensors and hence it is possible to improve the information about that particular target. Each radar works almost independent of each other, because they will measure a given target using their own local system co-ordinates. Now if each local radar information is sent to a ATM centre a co-ordinate transformation is required [Shank86] to bring all the sensors target information to a common co-ordinate system (e.g. stereographic projection [Mulholland82]). The illustration in Figure 7 does not give any details about this but as an example imagine that each radar measures the targets in its own local co-ordinate system. Each radar performs its own local tracking operation by using the local co-ordinate information in order to improve the target information. It would then send this improved filtered information in their local co-ordinates to the ATM centre. The ATM centre would then need to bring all local co-ordinates of each radar to a global co-ordinate system and then project these global measurements avoiding any geometric distortion. After that it will need to align all radar measurements which will then be sent to a global tracking system. A different arrangement is possible if after filtering the local co-ordinate measurements, each radar system converts this estimate into the global co-ordinate system prior to sending the data to the ATM centre. In that case the ATM centre would have only to perform the co-ordinate projection and sensor alignment. This task division can vary from different ATM systems depending on many factors. The list of factors is extensive, and sometimes the decision is not only guided by the best technological options, but will take into account other constrains that can limit the final resolution of the overall ATM system.

In any case, the central tracking system at the ATM centre is usually far more complicated and requires a more concrete and pragmatic approaches to achieve good results than isolated local radar tracking systems. Because the tracking system at each individual radar site is significantly simpler, such systems are the most logical candidate tracking system for the initial development of the generic parallel tracking algorithms in this thesis. Before discussing these tracking system, it is necessary to provide a brief introduction to radar systems in general.

3.3.3 Radar Systems

As indicated above there is a large variety of radar equipment available (for more information see e.g. [Giaccari79]). It is helpful to explain how data for tracking systems is obtained within radar systems. In order to do that, a brief discussion about this type of sensor is necessary, but the reader should bear in mind any attempt to separate the whole system into smaller pieces for a better understanding can lead to some oversimplifications. Radar systems are built for one essential purpose, to control aircraft movement. The tracking system in itself is only part of a control loop, therefore the full radar systems must be studied as a whole. However due to the overall complexity, it is inevitably broken down into subsystems for understanding and design purposes. In fact, this aspect has been forgotten in many descriptions about radar and in general sensor-based systems because the interpretation of these system as a whole is extremely lengthy and complicated in most cases [Blackman86].

A concise description of radar applications is given in Table 1 to illustrate a number of areas where radar systems are employed (see [Wheeler67], [Skolnik81], [Hovanessian84], [Brookner77], [Carpantier88], [Barton88], [Stevens88] for a more comprehensive review).

Air surveillance	Long-range early warning Ground-controlled intercept Acquisition for weapon systems Height finding and 3D radar Airport and air route
Space and missile surveillance	Ballistic missile warning Missile acquisition Satellite surveillance
Surface search and battlefield surveillance	Sea search and navigation Harbour and waterway control Ground mapping Intrusion detection Mortar and artillery location Airport taxiway control
Weather radar	Observation and prediction Weather avoidance Clear-Air turbulence detection Cloud visibility indicators
Tracking and guidance	Antiaircraft fire control Surface fire control Missile guidance Range instrumentation Satellite instrumentation Precision approach and landing
Astronomy and geodesy	Planetary observation Earth survey Ionospheric sounding

Table 1 - Radar Applications

The last factor to characterise radar systems are the basic elements contained in the system. There are many descriptions about the basic elements of radar systems [Farina80], [Barton88], [Farina85], [Hovanessian84], but all of them have the same simplified schematic as in Figure 8. Obviously specific applications will require different installations. For example, bistatic radars have two disjoint antennae, one for the receiver and other for the transmitter. More generally a multistatic system will comprise multiple transmitters coupled with multiple receivers, and also monostatic radars (transmitter and receiver in the same location) [Farina82]. A more precise description of antennae, transmitters, receivers and duplexers can be found in [Wheeler67], [Skolnik81].

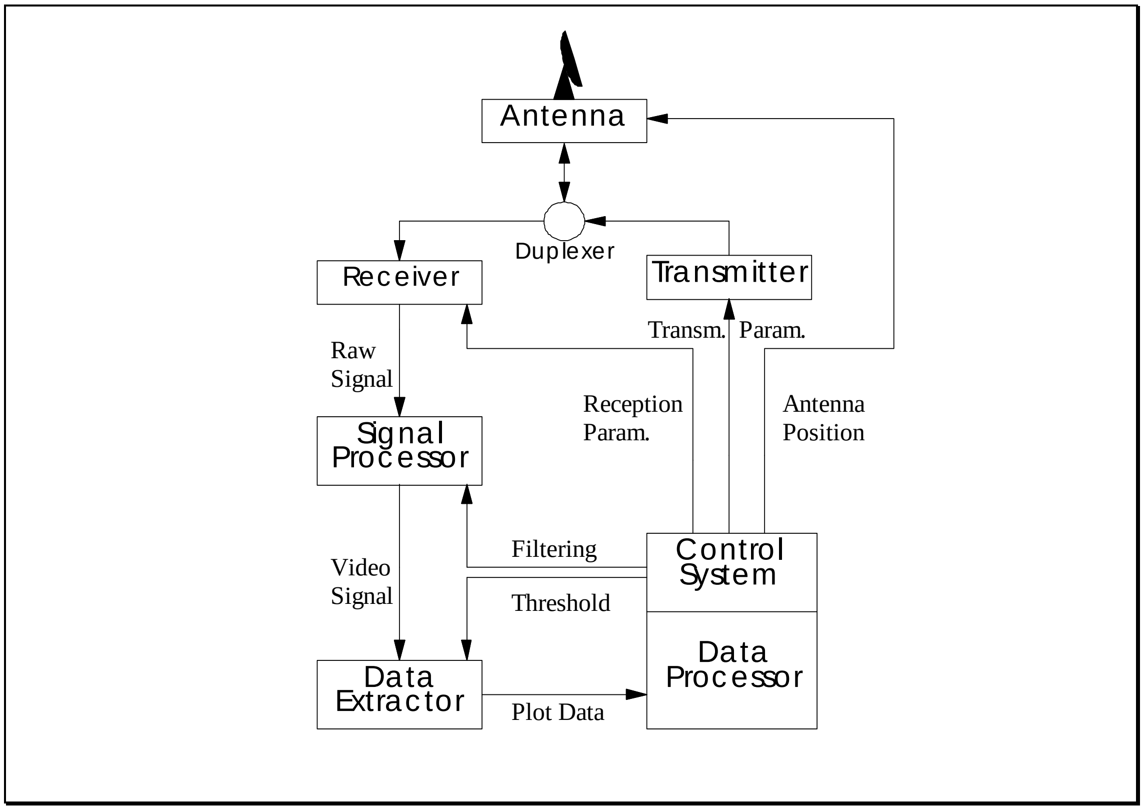


Figure 8 - Basic Elements of a Radar System

The signal processor is responsible for processing all of the signal echoed back to the radar antenna [Oppenheim78]. This component is crucial to obtain usable information that will be processed later by the tracking system. It will filter most of the radar return signal which is frequently corrupted by spurious returns caused by a variety of problems such as clutter, JEM (jet engine modulation), and ECM (electronic countermeasures) [Blackman86]. A number of techniques have been incorporated in the system in order to provide more reliable measurements, such as pulse compression, moving target indicator, pulse Doppler processing, moving target detector, constant false alarm rate, etc. [Barton88]. The main function of the signal processor is to process the obtained raw information from the receiver and to transform it into a more coherent form by means of digital signal processing. The scope of this activity is too large to be discussed here (a concise description of the main components and techniques for radar signal processing can be found in [Oppenheim78]). It is also important to point out that radar signal processors have been using parallel processing for some considerable time [Bergland72]. Until recently these techniques were based upon building relatively expensive dedicated hardware to cope with the computational demands of signal processing. Recently, more advanced and less costly parallel techniques have been used to implement these signal processing capabilities [Edward92] (in particular for Synthetic Aperture Radar (SAR) as described in [Franschetti90] and [Turner92]).

The information delivered by the signal processor still contains a large amount of data, which has to be further reduced in order to be processed by the radar data processor. At this stage, for example, a aircraft target will be seen as a sequence of detections from the same object, due to the discrete nature (pulsed) and antenna rotation of most Track-While-Scan (TWS) radars [Hovanessian73]. The next stage, the data extractor, is responsible for all necessary packing of related measurements into a smaller form, or "plot". In the aircraft example that information will be reduced to a single plot, referred to as a "target". Obviously some spurious information will also be contained in a plot, this will be referred to as "clutter". The data extractor attaches radar information to the plots. This information refers to range, azimuth, elevation (when available), radial velocity, and possibly other details (target signature). These plots together with the attached information are then passed onto the radar data processor [Farina85].

The radar data processor or tracking system, which is the main object of the present study, is responsible for all tasks related to interpreting and translating plot information into a coherent set that can be used by other computers/display systems/humans operators to provide control over the air space being observed. In fact the whole ensemble of the radar system can be seen as a bandwidth compressor as shown in Figure 9. This radar information compression is necessary to permit fast and easy comprehension by operators to provide fast responsiveness for different air traffic control conditions.

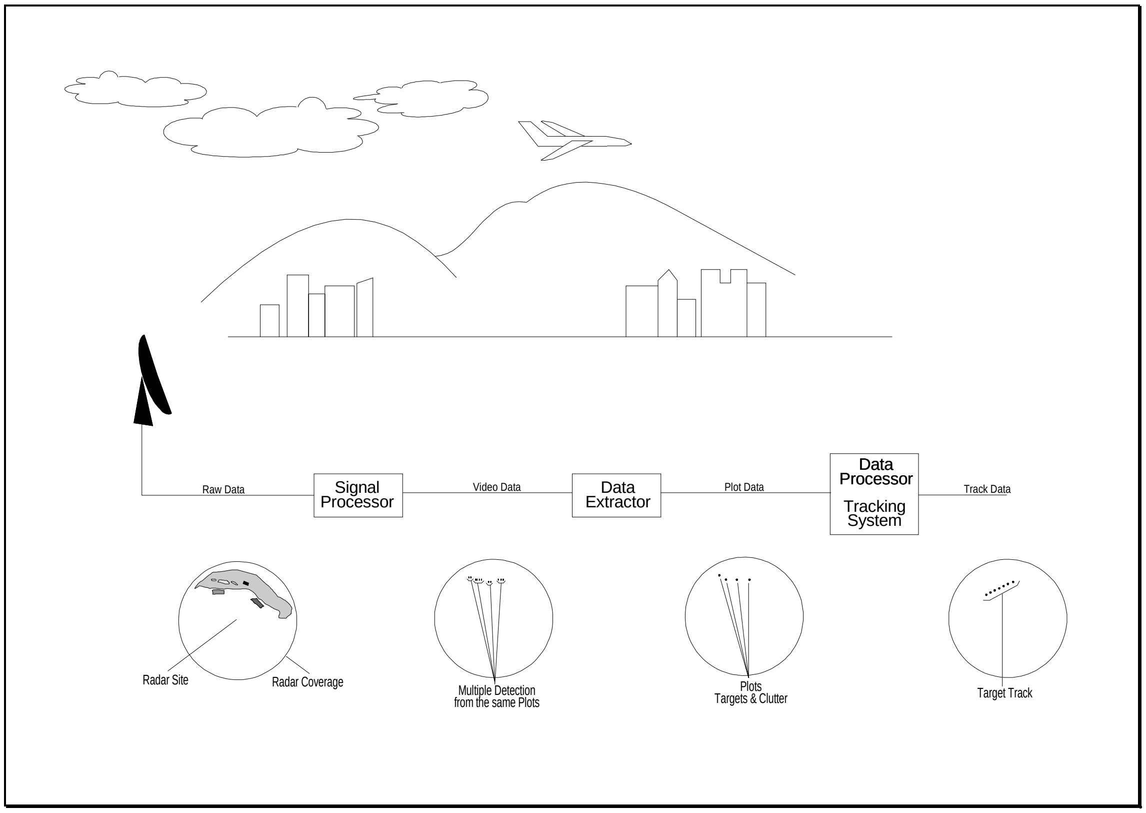


Figure 9 - Radar as a Bandwidth Compressor

As can be seen the radar data processor is the last stage in transforming the original information, raw data delivered by the receiver, into a more concise and useful format. The radar data processor is also called the radar tracking system and its basic operation is described next.

3.3.4 Radar Tracking System

Radar Tracking systems are used to reconstruct the trajectories of moving targets and to predict their future trajectories from the data provided by the rest of the radar system.

They have to determine when new tracks need to be created and old ones eliminated from the system. Two main elements are crucial in establishing the tracking system requirements, the target dynamics and available accuracy in trajectory measurements.

The dynamics of the targets [Willems90] in air traffic control are to some extent unpredictable, because of unplanned manoeuvres and weather conditions. Therefore, they can be tracked with relatively short smoothing times and predictions that are limited to a time not too far in the future. This is the opposite case to a target being tracked which obeys deterministic physical laws, as in the case of a satellite. The target trajectory is characterised, at any given time, by its current position and its derivatives [Goldstein80], [Sen79], [Lanczos70]. The target kinematics characterisation depends upon the target type. Target dynamics and performance varies enormously and Table 2 gives an idea of the wide range of velocities and accelerations that different targets can assume. Another important factor in the target kinematics characterisation is target manoeuvrability.

Target Type	**Velocity**	**Acceleration**
Military Aircraft	50m/s - 1000m/s	50m/s^2 - 80m/s^2
Civil Aircraft	50m/s - 300m/s	10m/s^2 - 60m/s^2
Short Range Missiles	2000m/s	150m/s^2
Long Range Missiles	7000m/s	600m/s^2
Satellites	4000m/s	10m/s^2
Ships	0 - 30m/s	2m/s^2
Land Vehicles	0 - 50m/s	4m/s^2

Table 2 - Typical Target Dynamic Characteristics

Modelling the dynamics of a target requires a number of simplifications to the original physical model in order to make it feasible for computer processing. These simplifications depend upon the target that the designer is trying to model. A common reason for simplification is to reduce the number of variables that represent the dynamics of the target (the computational costs increase as the number of variables increases). As an example, a common simplification for manned vehicles such as civil aircraft is to assume that the target follows a straight-line constant-velocity trajectory and well-behaved circular trajectories such as co-ordinated turns [Blom82a]. On the other hand, this is not the case when the target under consideration is a military aircraft or a missile. Despite any effort to perfectly model a target, there are also other unknown components that makes the modelling of target kinematics a difficult task, for example sudden and unexpected manoeuvres, unpredictable weather conditions, etc. These unknown components are frequently modelled as random perturbations that are incorporated into the target kinematics models [Singer70]. The correct modelling of the target dynamics will help to determine the efficiency and accuracy of the tracking system.

The target trajectory positions are measured by the radar and they are, in general for the purposes here, obtained in Polar co-ordinates, azimuth (❑) and range (❒). However due to the characteristics of the measurement process [Hovanessian84], [Hovanessian73], the azimuth measurement error is far more significant than the range error in determining the target correct position. These errors, in the same manner as in the kinematics modelling, play a major role in the quality of the tracking process. In other words, it does not matter how good the rest of tracking system algorithms is if the measurement data is excessively corrupted by noise. It is clear that there is a compromise between the quality of the tracking system and the quality of radar data. There are two type of errors associated with measured positions in radar systems [Gertz83]: measurements errors and bias errors. Measurements errors are unpredictable, uncorrelated, and unavoidable. Bias errors are consistent and can always be removed from the measurements, once detected and mapped (e.g. ground fixed clutter and mechanical misalignment of radars). Once again, in the case of measurements errors a common solution is to model them as random errors that are incorporated into the measurement model.

To summarize, modelling is a fundamental task in designing a tracking system, and it consists of separating phenomena into two parts: the first associated with the target under observation and the second that results from the measurement of the observed target. In other words, the target kinematics and the sensor reports or alternatively the *System Model* and the *Measurement Model*. This modelling usually requires a variety of mathematical tools to establish a satisfactory and convenient representation. The overall process is lengthy and is described in more detail in the following references [Schwartz75], [Sage79], [Gelb74], [Lanczos70], [Goldstein80], [Maybeck79].

Basic Elements

Different authors take different views of the parts of a tracker [Shaw86], [Farina85], [Hovanessian73], [Blackman86]. A tracking system in general can be broken down into four main task as illustrated in Figure 10.

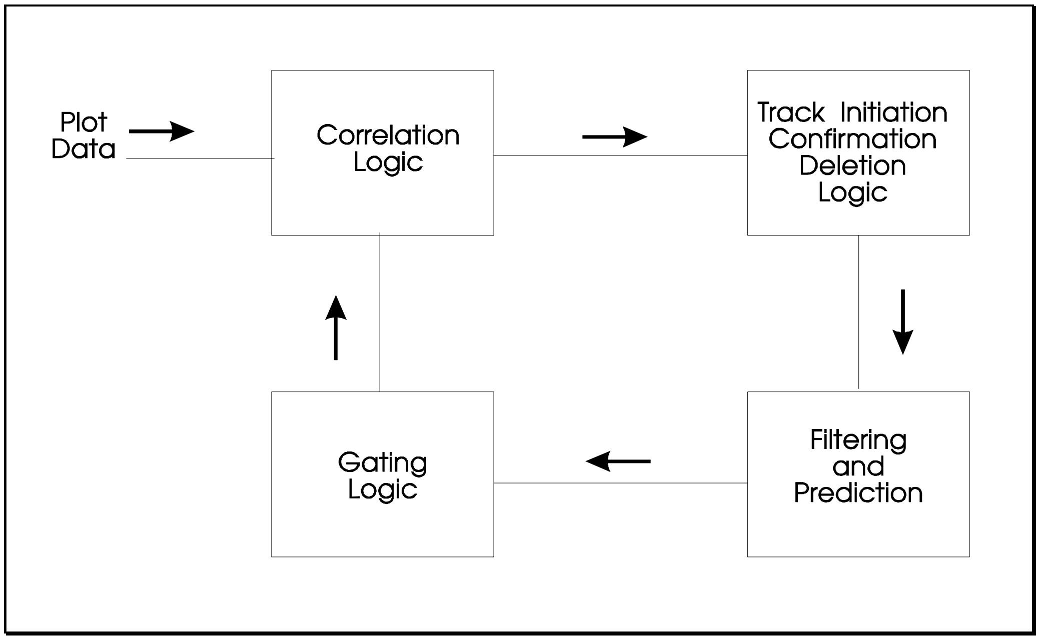


Figure 10 - Radar Tracking System

Before any discussion about the components of a tracking system can begin, some assumptions have to be made about the measurement process implemented by the rest of the radar system. It is assumed that the measurements are coming in time sequence without interruption. As an analogy imagine a TWS system collecting and delivering data with some small delay. A sectorization process [Farina85] is done in order to allow the real-time processing of the plots into tracks. Sectorization refers to the division of the radar coverage into identical azimuthal sectors. The number of sectors will depend on many factors such as typical number of plots per sector, computer memory, computer processing speed, etc. All processing is done on sector basis with the following logical description:

1. The radar was turned on some time ago starting from an assumed first sector or sector 1. It is now collecting data from sector 6. Therefore plot data has been already stored for sector 1 to 5. Suppose that the tracking program is running one step behind the radar data collection, i.e. while the plots from sector 6 is being collected, the tracking program is processing sector 5.
2. While the radar is collecting plots for sector 6, the tracking system is processing the deletion, initiation and maintenance of tracks using plots information from previous sectors (1 to 5). It first removes stationary clutter from sector 4 by looking simultaneously at two adjacent sectors (3 and 5) to avoid boundary problems. Sector 3 is processed to associate its plots with existing tracks, once again two adjacent sectors are also looked to avoid boundary problems. Next sector 1 is used to select the remaining plots to be used as tentative tracks.

The above description is an example of how a tracking system based on TWS radar copes with the arrival of continuous data. The important step for the purpose here is the association of plots to existing tracks, which in the above example occurs in sector 3. Again it is important to understand that the description given here is illustrative only and many details have been omitted. Also this is only one amongst many of the possible systems available to implement a tracking system.

Association (Correlation)

Correlation or association is concerned with the task of identifying which plots can be associated to which established tracks. The whole process consists of two stages, correlation and assignment. Correlation consists of finding all plots that can be associated with a particular track. If all plots are to be considered in association with a track, in a very dense or cluttered environment, this correlation will be a lengthy task. A common technique utilised to reduce the number of possible associations is known as *gating*. Gating consists of creating a geometric region around the predicted position of a track that will provide a *first cut* to selecting the most plausible plots to be associated with it. Note that the correlation is done on the basis of a plot inside the track gate. However, there are many possible combinations that make the overall process not as straightforward as set out above. In a very dense environment, many plots can occur in the same gate. On the other hand, a single plot can occur in many different gates. There are essentially two basic methods to permit the association, deterministic and probabilistic. Recently new techniques have been proposed in order to improve the association mechanism in dense, complicated environments. They are based on neural logic [Sengupta89] and the Dempster-Shafer method [Waltz90]. With the deterministic approach when conflicts arise a unique plot is sought inside the gate using distance criterion to make the assignment. The distance can be an Euclidean distance or a Bayesian probabilistic distance based on predicted characteristics of the track, the plot or both. This method is also known as nearest-neighbour algorithm [Farina85], [Blackman86], [Hovanessian84], and relies on the assumption that only one plot can be used to update the track.

Despite the straightforwardness of this method a number of pitfalls can occur. As an example imagine only one plot falling inside two gates from two different tracks with the plot having the same "distance" from both tracks. Another case would be having two plots inside a target gate with both plots having the same distance from the target predicted position (centre of the gate). A decision for the mentioned anomalies is, in general, implementation dependant and is usually deterministic when there are few occurrences. For more general situations where conflicting situations frequently arise, as mentioned above a more general method is employed. This method uses the Bayesian Probability Theory [Papoulis87] to determine the best solution. An optimal solution is frequently untractable because of the computational explosion of possible hypotheses and consequently high computational costs. A number of suboptimal techniques have been developed in recent years to prevent such a limitation. Three different approaches are known to date: *target-oriented*, *measurement-oriented* and *track-oriented* [Zhou95]. In the *target-oriented* approach a measurement is assumed to originate from a known target otherwise is considered as clutter. Typical examples are the Probabilistic Data Association (PDA) [BarShalom75] and Joint Probabilistic Data Association (JPDA) [BarShalom88], [Fortmann83].

The PDA deals with the tracking of a single target, already initiated, with clutter. The JPDA and variations as in [Zhou93], [Fisher89], [Chang84], can track multiple targets very close together, in the presence of clutter. The *measurement-oriented* approach assumes that each new measurement originates from either a known target, clutter, or new target. A typical example is the Multiple-Hypothesis-Tracking (MHT) algorithm [Singer74], [Reid79]. There is also a more simplified version of MHT, known as the branching algorithm [Smith75]. The essential concept of MHT is to postpone a decision, whenever any doubts about the association process occur, to a later stage when more information will be available. On the other hand, possible tracks are created and they will be used as soon as the new information becomes available. As new measurements arrive the hypothesis tree would expand and in order to prevent the explosion of possible hypotheses a mechanism based on merging and pruning of hypotheses is usually included in the method. Finally the *track-oriented* approach assumes that each track is either undetected, terminated, or associated with a measurement. This is a recent proposal and not many implementations of this method have been published.

Initiation/Confirmation/Deletion

This is the process of confirming the existence of a track whenever a certain condition arises. It is also used to eliminate tracks that for some reason are no longer valid tracks. And finally to determine which of the measurements left are considered to be possible new tracks, namely initiation. In some systems this task is still done manually [Shaw86] because of the complexity of the algorithms. The automatic initiation process is not a simple task and in fact it can be one of the most difficult in a dense and complex environment as pointed out in [Blackman86], [BarShalom83]. The Initiation and Confirmation stages are usually grouped as a track set-up process. The initiation stage constitutes all plots that were not used in existing tracks (firm tracks) or clutter (stationary tracks). Therefore, a initiation track can be started in any observation as long as the related plot does not belong to either a confirmed or stationary track. The confirmation stage is based on further observation and its relation with the initiation tracks. There are some rules to "confirm" the existence of a track. Again these rules are frequently based on probabilistic approaches. A simple and common technique known as the sliding window method (e.g. the probability of achieving m successes out of n consecutive events at least once by the Nth opportunity) is frequently used to initiate a track [Castella76].

A number of more complex approaches to track initiation have been proposed based on integer programming methods [Morefield77] and more sophisticated probabilistic rules [Blackman86], [BarShalom88]. The deletion process are generally based on very simple rules such as unsuccessful detection attempts (e.g. as a consecutive number of misses). However, again more complicated formulations based on probabilities can help to establish more general approaches [Sittler64]. There are also many other factors that should be taken into account, for example, the quality of the track will possibly determine the size and shape of the gate, and also the quality of the plot. As an example consider a track with high quality data and two plots within its gate. A first plot has high quality but is further from the predicted position than the second plot. The second plot, on the other hand, has a lower quality. Obviously some stronger criteria than distance (as Nearest-Neighbour - NN) should be found to permit the correct association for the above example, otherwise miscorrelation will happen and as a consequence tracking will be poor. These criteria are again heavily based on Bayesian probability.

Gating

As it has been shown in the correlation process, gating is a very important phase of the tracking process. The gates are generated from the filtering and prediction process (discussed below) and help to perform the correlation process. Gates are essentially determined by the variances of the measurement, the system co-ordinates and filter equations. Intuitively a gate is a region around the predicted position were the next measurement is expected to happen. The size will depend on the quality of the measurements (consequently their variances) and the quality is the estimation or prediction process. The shape will be strongly influenced by the system co-ordinate chosen. A more complete and formal description of gates can be found in [Blackman86] and [BarShalom88], where the filter theory for tracking systems is developed, in particular Kalman filtering.

Filtering and Prediction

Filtering and prediction are fundamental elements in any tracking system. These operations are implemented by means of digital filters and estimation theory [Gelb74]. As discussed above, the design of a tracking filter is strongly influenced by the mathematical models of the system and measurement. The design of the tracking filter also determines the overall performance of the tracking system. As an example, take the previous topic, correlation. The whole process can be jeopardised if the predicted position is too far away from where the next measurement is expected, assuming that the measurement is "correct". It also plays a very important role in determining the size of the gate. These two models are also strongly influenced by the co-ordinate model chosen [Mulholland82], [Shank86]. The co-ordinate system chosen also influences the dynamic equations of the target, because some of them are not inertial. It is natural to solve tracking kinematics equations in Polar co-ordinates (range, azimuth and elevation), because these are the natural measurements provided by the radar system (this is not completely correct as shown by [Farina85]). Most radar systems currently deployed deliver their measurements in Polar co-ordinates. As a consequence, a complicated filter is necessary to accomplish the tracking task. The use of Polar co-ordinates thus leads to the use of the more complicated Extended Kalman Filters for simple target trajectory tracking.

Chapter 4 : Parallel Interacting Multiple Model

4.1 Introduction

The IMM algorithm has received great deal of attention in the literature [Blom82b], [Blom88], [Blom90a], [Blom90b], [Vacher92], [BarShalom92a], [Lerro93a], [Li93a], [Li93b], [Atherton94], [Lin93a], [Lin93b], [Lin93c]. It has been shown to be a relatively simple, cost-effective method for hybrid system estimation [Blom83a]. Hybrid systems comprise continuous-valued state elements that evolve in time according to one of the selected models. The selection or switching from one continuous model to another takes place according to a Markov chain with an appropriate transition probability matrix [Gillespie92]. The transition is the discrete state element in the hybrid system. The IMM has been shown to perform well in estimating the trajectory of very manoeuvrable objects [Blom82a]. Consequently, one of the main areas of application has been air traffic control and more generally Tracking System, although it can also be applied to other areas governed by hybrid systems [Blom83b]. As described previously, the application framework to be used to implement IMM in parallel (parIMM) is a simple air traffic control environment. The main objective of this section is to investigate the processor scalability and computational load of this algorithm. In a broader sense the scalability of the algorithm will vary, not only with an increase in the number of processors, but also with an increase in the number of models used to implement the IMM filter. Ultimately, the analysis needs to determine the internal component computational load according to the number of processors and number of models used to implement the algorithm.

In order to develop parIMM this section will not present the IMM algorithm equations because they have been extensively discussed in the literature [Li94]. The approach here is to assume that the reader is already familiar with Kalman filters and multiple model techniques as used in IMM. Only a generic (and as far as possible intuitive) description will be included to show the development of a parallel implementation, rather than concentrate on the mathematical complexity involved in the algorithm. A review of the necessary theoretical elements to introduce and explain the Interacting Multiple Model algorithm along with its equations in terms of the main components (Interaction and Mixing, Kalman Filters, Probability Update and Combine) is given in Appendix B. This review is not intended to be either complete or rigorous, being restricted to the introduction of the basic problems that have led to the development of a range of multiple model algorithms and in particular the most cost-effective in this class, the IMM algorithm.

The structure of this chapter is firstly to introduce the parallel model to be assumed for the parIMM implementation. It describes the correct selection and matching of the algorithm granularity to the architecture granularity. With a particular granularity selected it will show the minimum set of variables that are necessary to be exchanged amongst the parallel components. It will also show that more than one strategy exists to communicate this minimum set of variables amongst processors. The communication strategies are then explained taking as a reference the chosen architecture, granularity and minimum data set. Next an application based on air traffic control is described to test and validate the parIMM. The description gives the IMM filter parameters set-up followed by a description of the data test set simulator generation. After implementing the filter, the testing of parIMM is undertaken on different architectures with an increased number of processors. Therefore, the results are classified according to the architectures and presented in terms of the speed-up and efficiency achieved on each architecture. Also once the best communication strategy is determined for each architecture they are further analysed in terms of each of the internal computational elements to determine their computational load on the overall parallel implementation. Finally the conclusion gives a general overview about the achievements and drawbacks found during the development of the parIMM implementation.

4.2 Parallelizing the Interacting Multiple Model

A parallel implementation of the Interacting Multiple Model filter (parIMM) requires a number of different steps. Firstly, it requires the selection of the appropriate grain size for parallelization i.e. small, medium or larger grain. Secondly, depending on the selected grain size a strategy to obtain a parallel algorithm has to be determined from the original sequential algorithm. This section describes the criteria used to select the grain size and the approach to parallel implementation of the Interacting Multiple Model.

4.2.1 Grain Size Problem

The first important issue to discuss before selecting an appropriate grain size to implement the IMM on parallel architectures is the complexity of the algorithm. It is not an easy discussion because IMM complexity depends on many parameters and conditions. As a simple example take an IMM with few models (i.e. few Kalman filters). Although there are a small number of Kalman filters, they could be relatively complex and have a relatively high number of state variables. If this is the case the Kalman filters can become the most computational demanding component of the IMM algorithm. Hence, in this example the Kalman filters become the main element for parallelization. Kalman filter parallelization has been shown to be cost-effective if made on fine grain architectures [Gaston90], [Lawrie90], [Lawrie91]. Usually this fine grain architecture translates into systolic array implementations of the Kalman filter [McCanny89]. Another example is to consider the opposite case to the previous one, where a high number of models or Kalman filters are present and all of them are relatively simple and have a small number of state variables [Lin93a]. Clearly, this time the Kalman filters are not the main component of the overall computational load and therefore they are not the main subject of parallelization. Furthermore, there are many other factors that can make the IMM algorithm vary in complexity, e.g. an irregular transition probability matrix [Blom90b], utilisation of square root Kalman filters [Raghavan93] and high-order Extended Kalman Filters [Vacher92]. In summary, the IMM complexity study plays an essential factor in determining the parallel granularity selection to obtain the best compromise. The implementation presented here assumes Kalman filters that are computationally light-to-medium weighted and hence they are regarded as unit block elements around which parallelization will take place.

In view of the above explanation it is straightforward to conclude that the correct granularity is a large or coarse grain. In fact, the selection of the algorithm complexity and ultimately the final granularity cannot be done in isolation. The granularity of the architecture is a factor in selecting the appropriate algorithm granularity. The architecture considered in this work is a distributed memory MIMD architecture [Hwang87]. This architecture has a relatively large grain size which suggests the selection of a large algorithm grain size. This correct matching between algorithm grain size and architecture grain size is more likely to produce the best parallel efficiency [Fox88]. The selected grain size implies a distribution of the algorithm largest components, i.e. Interaction and Mixing, Kalman

Filters, Probability Update and Combine, amongst processors. This distribution can be done using essentially two different techniques. The first technique is the *Farm* approach and the second, almost a particular case of the first, is called *Single Program Multiple Data* (SPMD) [Quin87].

In the first approach (*farm*) a manager is responsible to partition the data and distribute them to specialised workers. In the second approach (SPMD) the same algorithm is implemented on each processor and it acts over different data without any particular manager. The *farm* approach depends on a heuristic distribution of the tasks which could lead to a task being a bottleneck in parallel algorithm performance [Atherton94]. The approach here is to implement the IMM algorithm in a SPMD way on two different granularity MIMD distributed memory architectures, namely transputers and the MEIKO CS-2. A manager-workers approach has been already used in two different types of MIMD architectures. In [Averbuch91a] and [Averbuch91b] a shared memory MIMD architecture has been used to implement the IMM, while in [Atherton94] a distributed memory MIMD architecture was used. Also a very interesting approach using an adaptive IMM (AIMM) was suggested for implementation on a parallel distributed memory MIMD architecture as described in [Munir95].

4.2.2 The parIMM Model

For the development of the algorithm it is assumed that the MIMD architecture has a number of processors $\boldsymbol{p}$ and the IMM filter has a number of models or filters $\boldsymbol{m}$, and with the only restriction that the number of processors do not exceed the number of filters, or $\boldsymbol{m} \geq \boldsymbol{p}$. The distribution of the $\boldsymbol{m}$ filters amongst the $\boldsymbol{p}$ processors is made statically and sequentially. This guarantees that processors are almost equally load balanced in terms of number of filters. After the distribution each processor $\boldsymbol{s}$ will have $\boldsymbol{p_S}$ filters. The algorithm performs as in the sequential case until external data is necessary from other processors or internal data needed by other processors has to be sent to them. To carry on with the description of the selected parallel model a few assumptions are necessary :

- The Probability Transition Matrix is known by all processors
- All mode probabilities (μ_i) of the IMM algorithm are also known by all processors all the time.
- And finally, for further simplification the index ***s*** referring to the processor will be also dropped and any processor will be assumed to have a set of internal filter parameters given by $\boldsymbol{p}_{in}$ and all other parameters corresponding to external filters on any other processor as an external set given by $\boldsymbol{p}_{out}$

The following table (Table 3) summarises where external data is needed and where internal data is ready to be sent to other processors for each IMM step

	Interaction and Mixing	Kalman Filters	Probability Update	Combine
Imported Values	$\hat{\mathbf{x}}_{out}$, $\hat{\mathbf{P}}_{out}$		μ^{p}_{out} , Λ_{out}	$\hat{\mathbf{x}}_{outt}$, $\hat{\mathbf{P}}_{out}$
Exported Values	$\bar{\mu}_{in}$	$\hat{\mathbf{x}}_{in}$, $\hat{\mathbf{P}}_{in}$, Λ_{in}	μ^{p}_{in}	

Table 3 - parIMM Exchanged Data

Considering the above table, the structure of the algorithm and its equations the following elements have to be exchanged amongst processors :

$$\hat{\mathbf{x}}, \hat{\mathbf{P}}, \ \mu \ , \ \Lambda$$

In order to reduce the number of exchanged data items, consider now the following product

$$\mu^{p}_{in} = \bar{\mu}_{in} \ \Lambda_{in}$$

which will be called from now on the *partial probability update*.

The above analysis has determined the minimum amount of information which has to be exchanged in order to accomplish the parallelism. This is the estimates, the respective covariances and their partial mode probability or

$\hat{\mathbf{x}}, \hat{\mathbf{P}}, \boldsymbol{O}$

These internal values of a processor are generated at different points in the parallel algorithms as can be seen in Table 3. On the other hand, the external values for the same processor are needed at different points as shown in the same table. To accommodate this, three different methods are utilised to implement the algorithm on a parallel architecture.

Asynchronous

This mechanism allows data items to be exported to other processors as soon as they have been produced. This approach broadcasts to every other processor the estimates, their covariances and the corresponding partial probability updates just after they have been calculated. This mechanism implies that each processor will be interrupted in different phases of its internal processing to receive data from other processors. However these data may not be needed inside that processor at the moment of its arrival. Consequently, a buffering mechanism has to be implemented to hold the received data until the recipient processor is ready to utilise them.

Synchronous

This mechanism allows data items to be imported from other processors on a demand basis. This approach requires that a requesting processor demands data from any other processor whenever is needed inside it to pursue further calculations. This mechanism implies that each processor sending a request for data in other processors will have to wait until that data becomes available on the requested processor. This mechanism does not require any special kind of buffering inside the parIMM algorithm.

Block Synchronous

This mechnism allows each processor to exchange data items in a synchronous block mode. This approach calculates all its relevant Kalman filters parameters, namely estimates and respectives covariances along with their associated partial probability mode updates and accumulates them internally in a block of data that will be later broadcast to all other processors. After all processors reached the same stage they engage themselves in a broadcast of their internal block of data to all other processors. In summary every processor exchanges its block of data with every other processors in the architecture. This in fact implies the creation of a new module as illustrated in the Figure 11.

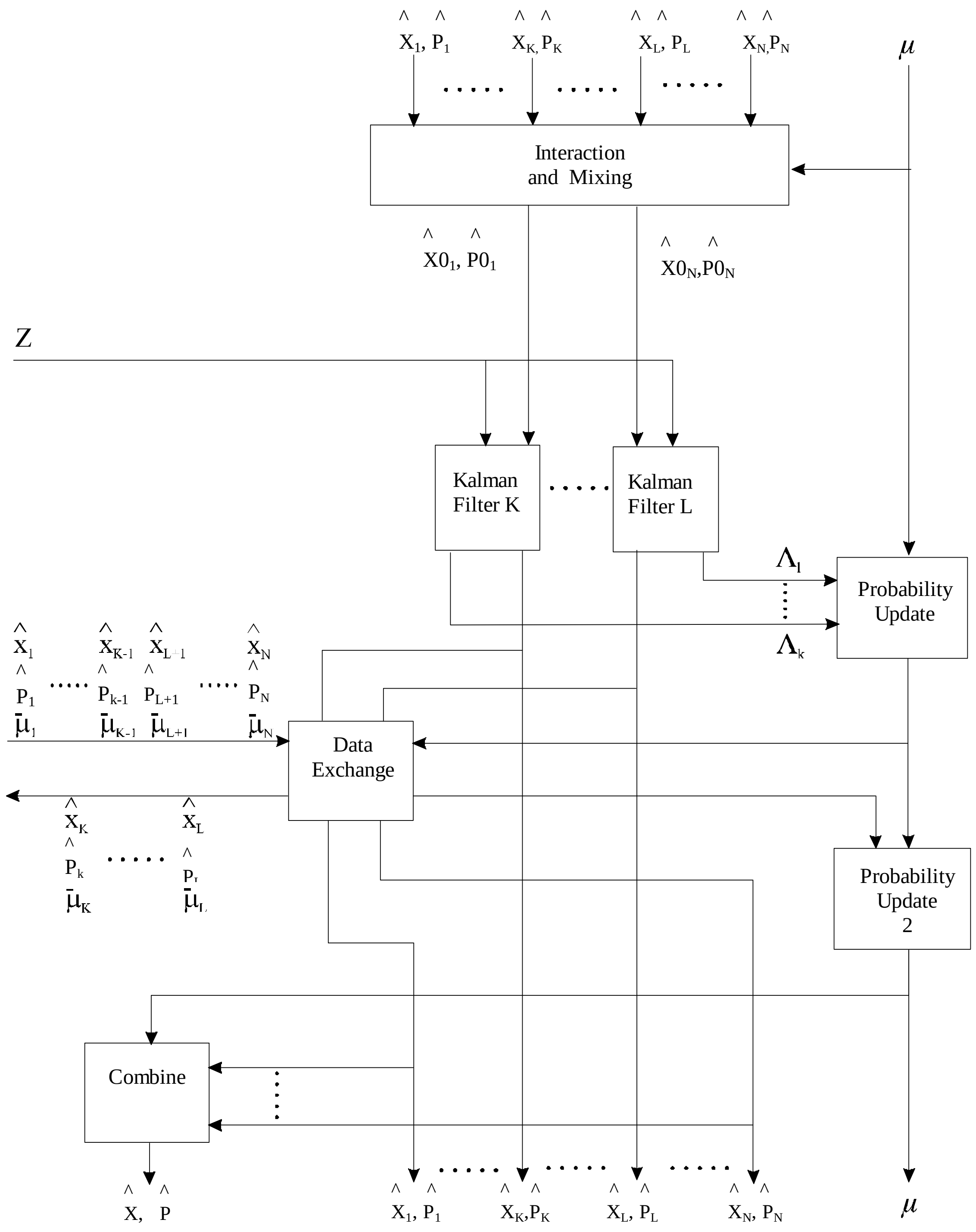


Figure 11 - parIMM

4.3 Implementations

4.3.1 parIMM set-up

The IMM has been mainly used to implement relatively complex tracking filters for air traffic control systems [Blom84a], [Blom92], [Vacher92], [Li93a] that are based on a limited number of models, usually 2 or 3 (straight line and co-ordinate turn movements), with variations to accommodate different kinematics and measurements. In order to obtain a large number of Kalman filters a different approach from these applications will be taken. This can be done using different constant acceleration models covering different ranges of acceleration. Each of these accelerations models has a different trajectory, e.g. straight-line constant-velocity requires no acceleration while turn requires an acceleration. It is also assumed that these accelerations during turns are almost constant. A similar approach is taken by [Averbuch91a], [Averbuch91b] using a different MIMD architecture based on general-purpose shared-memory hardware.

Kinematics Model

The target considered is describing a trajectory in a bidimensional plane (x-y plane). The trajectory can be modelled as uniform accelerated movement at any time, but with different accelerations. This set of different x and y accelerations will help to model different trajectory kinematics states. Despite the fact that the set of accelerations can be as large as required, it cannot model all types of trajectories as in the case of applications where constant turn rates are normal because they do not have constant acceleration, although for the parallel testbed purposes this is an acceptable limitation and a fairly crude approximation to the real kinematics. These constant acceleration models can be generally characterised by the following general differential equation.

$$\frac{\partial^2 \mathbf{s}}{\partial t^2} = \mathbf{a} + \blacksquare$$

where **s** is the target position, **a** is a constant acceleration value and ▰ is a zero-mean Gaussian white noise to account for any small variations in the acceleration due to system disturbances (environmental and target). The final discrete state equation is obtained by writing the above equation in state space form. The state vector is augmented to account for bias problems caused by mismatch between true and discrete values of the accelerations, as in [Averbuch91a], [Averbuch91b]. After solving the continuous equation, it is then possible to obtain a discretized state-space equation. The final discretized state-space equation is given by

$$\mathbf{x}(k+1) = \mathbf{A}\,\mathbf{x}(k) + \mathbf{B}\,\mathbf{u} + \mathbf{C}\,\nu$$

where in the above

$$\mathbf{A} = \left\{\begin{matrix} 1 & 0 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & T & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}\right\}, \qquad \mathbf{B} = \left\{\begin{matrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \\ 0 & 0 \\ 0 & 0 \end{matrix}\right\}, \qquad \mathbf{C} = \left\{\begin{matrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \\ 1 & 0 \\ 0 & 1 \end{matrix}\right\}$$

and

$$\mathbf{x} = \left[x \; y \; \dot{x} \; \dot{y} \; \ddot{x} \; \ddot{y}\right]', \qquad \mathbf{u} = \left[a_x \;\; a_y\right]', \qquad ■ = \left[■_x \;\; ■_y\right]'$$

The constant accelerations a_x and a_y are going to be selected from a set of accelerations that will compose the different models for the IMM filter. The mechanism to generate different number of models to implement IMM is straightforward. Given a maximum (A_{max}) and a minimum (A_{min}) acceleration and an increment (A_{inc}) that divides the range (A_{max} - A_{min}) exactly, a different set of accelerations are obtained as follows:

Accelerations are replicated for x and y axis and the final number of models is the combination of the x and y acceleration. Six different sets of accelerations were used to implement the parIMM as shown in the table below

Set No.	Max. Acc. (m/s^2)	Min. Acc. (m/s^2)	Incr. Acc.	No. of Models	Total No. of Models in x-y
1	40	- 40	40	3	9
2	40	- 40	20	5	25
3	40	- 40	10	9	81
4	40	- 40	8	11	121
5	40	- 40	5	17	289
6	40	- 40	4	21	441

Table 4 - Acceleration Ranges

The system noises are assumed constant and equal for all the models and the covariance is assumed to be, as in [BarShalom88],

$$\mathbf{Q} = E\left[■_i ■_j' \right] = \mathbf{CC}' q ♎_{ij}$$

where $\sqrt{qT} = 0.2 \; m/s^2$ is the acceleration rate of change and $♎_{ij}$ is Kronecker’s delta

The variable *T = 4 seconds* represents the radar sampling rate and it is constant throughout the experiment.

Measurement Model

The sensor is assumed to deliver measurements in Polar co-ordinates (❐, ❑) and given by

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

where

$$\mathbf{z}(k) = \begin{bmatrix} \square(k) \\ \square(k) \end{bmatrix},$$

$$\mathbf{H}\left(\mathbf{x}(k)\right) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \arctan\left({}^{y}\!/\!{}_{x} \right) \end{bmatrix},$$

$$\blacklozenge = \begin{bmatrix} \blacklozenge_{\square} \\ \blacklozenge_{\square} \end{bmatrix}$$

The measurement system is the same one for all models and the measurement noises w_X and w_Y are assumed constant and equal for all the models. Its covariance is given by

$$\mathbf{R} = E\left[\blacklozenge\,\blacklozenge'\right] = \begin{Bmatrix} \blacklozenge_{\square}^{2} & 0 \\ 0 & \blacklozenge_{\square}^{2} \end{Bmatrix}$$

where $\blacklozenge_{\square} = 30\,m$ and $\blacklozenge_{\square} = 0.9\,mrad$ represent the errors in range and azimuth respectively [Stevens88].

Transition Matrix

As previously described in the IMM algorithm the transition between models, i.e. different accelerations, is given by a Markov process [Blom90a]. The transition probability values determination is the same as in [Moose75]

$$p\left[m_i(k+1) \mid m_j(k)\right] = \begin{cases} prob & , i = j \\ \dfrac{1 - prob}{m - 1} & , i \neq j \end{cases}$$

where prob = 0.95 represents the probability of staying in the same model.

4.3.2 Data Test Set

To test the application described above it is necessary to simulate the measurements produced by the sensor. In order to do that a trajectory simulator software [Farina86] was implemented using MATLAB [Matlab93]. This software takes as input the trajectory segments, such as straight-line constant-velocity, straight-line constant-acceleration and co-ordinate turns, and sensor information such as sensor sample time, sensor error azimuth error, maximum range. The trajectory comprises a number of different input kinematics segments as described below.

1. Initial conditions: Position = (10000m,25000m); Velocity = (1.0m/s,1.0m/s).
2. Straight Line Constant Acceleration: Acceleration=(8.0m/s^2, 8.0m/s^2); Distance=2400m.
3. Straight Line Acceleration: Acceleration=(4.0m/s^2, 4.0m/s^2); Distance=1200m.
4. Coordinate Turn: Turn Angle=(45^o); Turn Rate=(1.0^o /s).
5. Straight Line Constant Velocity: Distance=10000m.
6. Coordinate Turn: Turn Angle=90 ; Turn Rate=3.0/s).
7. Straight Line Constant Velocity: Distance=5000m.
8. Coordinate Turn: Turn Angle=45° ; Turn Rate=3.0°/s.
9. Straight Line Constant Velocity: Distance=10000m
10. Coordinate Turn: Turn Angle=45° ; Turn Rate=-3.0°/s.
11. Straight Line Constant Acceleration: Acceleration=-2.0m/s^2 ; Distance=10000m.
12. Straight Line Constant Acceleration: Acceleration=2.0m/s^2 ; Distance=10000m.
13. Straight Line Constant Velocity: Distance=10000m.
14. Coordinate Turn: Turn Angle=270° ; Turn Rate=3.0°/s.
15. Straight Line Constant Velocity: Distance=5000m.
16. Coordinate Turn: Turn Angle=45°; Turn Rate=2.0°/s.
17. Straight Line Constant Acceleration: Acceleration=-6.0m/s^2 ; Distance=6000m.

The trajectory simulator software generates two files:

- Cartesian Co-ordinates (x, y) real data, containing the x and y position and Vx and Vy velocities and the Ax and Ay accelerations. This data is uncontaminated or has noise added.
- Measurement data file containing Polar co-ordinates obtained from the real Cartesian position according to the prescribed conversion equation. Measurements are then

contaminated with noise to simulate the radar measurement process. Added noise is obtained randomly for each new measurement conversion.

4.3.3 Optimisations

Because of the particular transition probability format some simplifications of the total number of operations can be achieved. The transition probability matrix has in practical terms only two elements, namely the transition probability to stay in the same model and the transition to change to another model, as in the equation described above in the transition matrix section and assuming the following notation

$$\begin{aligned} \square_{stay} &= prob \\ \square_{chng} &= (1 - prob)/m \\ \square_{diff} &= \square_{stay} - \square_{chng} \end{aligned}$$

As has been shown [Atherton94] and can be confirmed by analysis of the IMM equations, in particular the Interaction and Mixing step, an increase of the number of models to implement the IMM filter imposes a substantial overhead in that step. However the above, particular format of the transition probability matrix simplifies the IMM Interaction and Mixing equations considerably in terms of the number of floating point operations. In order to make the explanation of the simplifications introduced more straightforward the following two new variables are introduced into the Combination step.

$$\hat{\mathbf{x}}_{comb} = \sum_{i=1}^{m} \bigcirc_i \hat{\mathbf{x}}_i$$

$$\hat{\mathbf{P}}_{comb} = \sum_{i=1}^{m} \bigcirc_i \hat{\mathbf{P}}_i$$

Now returning to the Interaction and Mixing step and rewriting it in terms of the two variables just shown results in

$$\bar{\bigcirc}_j = \square_{diff}\,\bigcirc_j + \square_{chng}\sum_{i=1}^{m}\bigcirc_i$$

Consequently using the results from the Combine step and those from above with some manipulation the mixed estimates and its respective covariances can be further reduced to

$$\hat{\mathbf{x}}_{Oj} = \frac{1}{\bar{\bigcirc}_j}\left(\square_{diff}\,\hat{\mathbf{x}}_j + \square_{chng}\,\hat{\mathbf{x}}_{comb}\right)$$

$$\begin{aligned}\hat{\mathbf{P}}_{Oj} = \frac{1}{\bar{\bigcirc}_j}\Bigg[\square_{diff}\left(\hat{\mathbf{P}}_j + \left[\left(\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_{Oj}\right)\left(\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_{Oj}\right)'\right]\right) + \\ ] \\ \square_{chng}\left(\hat{\mathbf{P}}_{comb} + \sum_{i=1}^{m}\left[\left(\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_{Oj}\right)\left(\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_{Oj}\right)'\right]\right)\Bigg]\end{aligned}$$

This simplification brings a reduction up to almost 50% in the floating point number of operations performed in the Interaction and Mixing step. This reduction reaches its peak when the number of models used is more than 25, along with a state vector with elements between 4 and 8. On the other hand if a small number of filters is used, typically between 2 and 5, along with a state variable with the same number of components, 4 to 8, the percentile gain is modest or almost negligible. However the important point here is the gain obtained when the number of models used in the filter is increased. It is important because as noted previously the Interaction and Mixing step is the dominant computational element as the number of models increases.

4.4 Results

4.4.1 Introduction

The parIMM has been implemented in two different distributed memory MIMD architecture, as previously described. Results are therefore presented by architecture and the final format of the graphs presented is explained below. Because the parIMM has been implemented using three different communications strategies where each one was tested against six sets of different acceleration ranges allied with a variable number of processors there is a considerable amount of output data to be analysed. In order to keep the results presentation in a concise format initially only the speed-up and efficiency graphs for each of the architectures is shown. Next the best communication strategies are selected for further analysis, based on this results. This analysis will show the computational contribution at each of the algorithm components in relation to the final computational cost according to the number of processors and ultimately the number of models. Finally the two architectures are compared to determine the overall performance and to select which one is more suitable for the parallel implementation of IMM.

The speedup and efficiency presented throughout this thesis are given by the following definition [Fox88]:

$$\text{Speedup}\left(S_p\right) = \frac{\text{Sequential Processig Time}\left(T_s\right)}{\text{Parallel Processing Time}\left(T_p\right)}$$

where on the above T_S is the execution time for the sequential program executing on a single processor. And T_p is the execution time for the parallel program executing on p processors. The efficiency is defined as

$$\text{Efficiency} = \frac{\text{Speedup}\left(S_p\right)}{\text{Number of Processors}\left(p\right)}$$

4.4.2 Validation of Tracking Results

In order to ensure the IMM filter tracking capability, the sequential filter was initially implemented with MATLAB. Validation of the filter was performed for each of the deterministic acceleration sets. Each set was tested against the trajectory specified in Section 4.3.2 perturbed with noise defined using Monte Carlo techniques, for a total of 50 different executions. The results for each set were analysed in terms of the RMS errors against the true trajectory for the x and y position and velocity. The criteria adopted to define an acceptable level of tracking was set at a level comparable to that presented in [Averbuch91a], [Averbuch91b] and [Atherton94]. Each execution of the filter was then evaluated to ensure that acceptable tracking was maintained throughout the trajectory and the RMS error calculated. Finally, the RMS errors for an 50 execution were summed and an addition evaluation performed to confirm that the overall performance of the filter was satisfactory for that data set. Once the sequential filter had been validated over all data sets, the MATLAB simulation, was again used to output the estimated, predicted and RMS errors values for position and velocity. These values were produced for each of the 50 different executions and stored for use during the parallel filter execution to validate the parallel implementation against the sequential results. In every execution the results generated by the parallel filter were identical to the sequential simulation (to the accuracy of the 64 bit floating point representation used)

4.4.3 Transputers

The implementation for the transputer platform was made using the architecture and message passing mechanism described in Chapter 1, namely an hypercube and EXPRESS. The efficiency graphs are shown in Figure12 and speed-up graphs in Figure12. First let us analyse the transputer parallel implementation of IMM from the efficiency side. As can be seen the efficiency drops quite dramatically for parallel implementation where the number of models used to implement the IMM filter is less equal than 25. In particular for a 9 model implementation there is a loss of almost 40% in efficiency when the implementation goes from a single processor to two processors. It is also noticeable that for implementations with less than 81 models there are a few anomalous results, which are caused by the message passing system being used (EXPRESS) (the efficiency for 3 processors is slightly better than for 2 processors in a 9 model implementation, with a similar result occurring for 81 models and also with 25 models but this time with 5 processors). Now a similar analysis can be done by looking at the speed-up. In Figure 13 is clear that after the introduction of a second transputer to solve the problem no real speed-up is in fact achieved. Despite the fact that again when a third processor is introduced a small improvement is reached, no real speed-up is gained for a 9 model implementation.

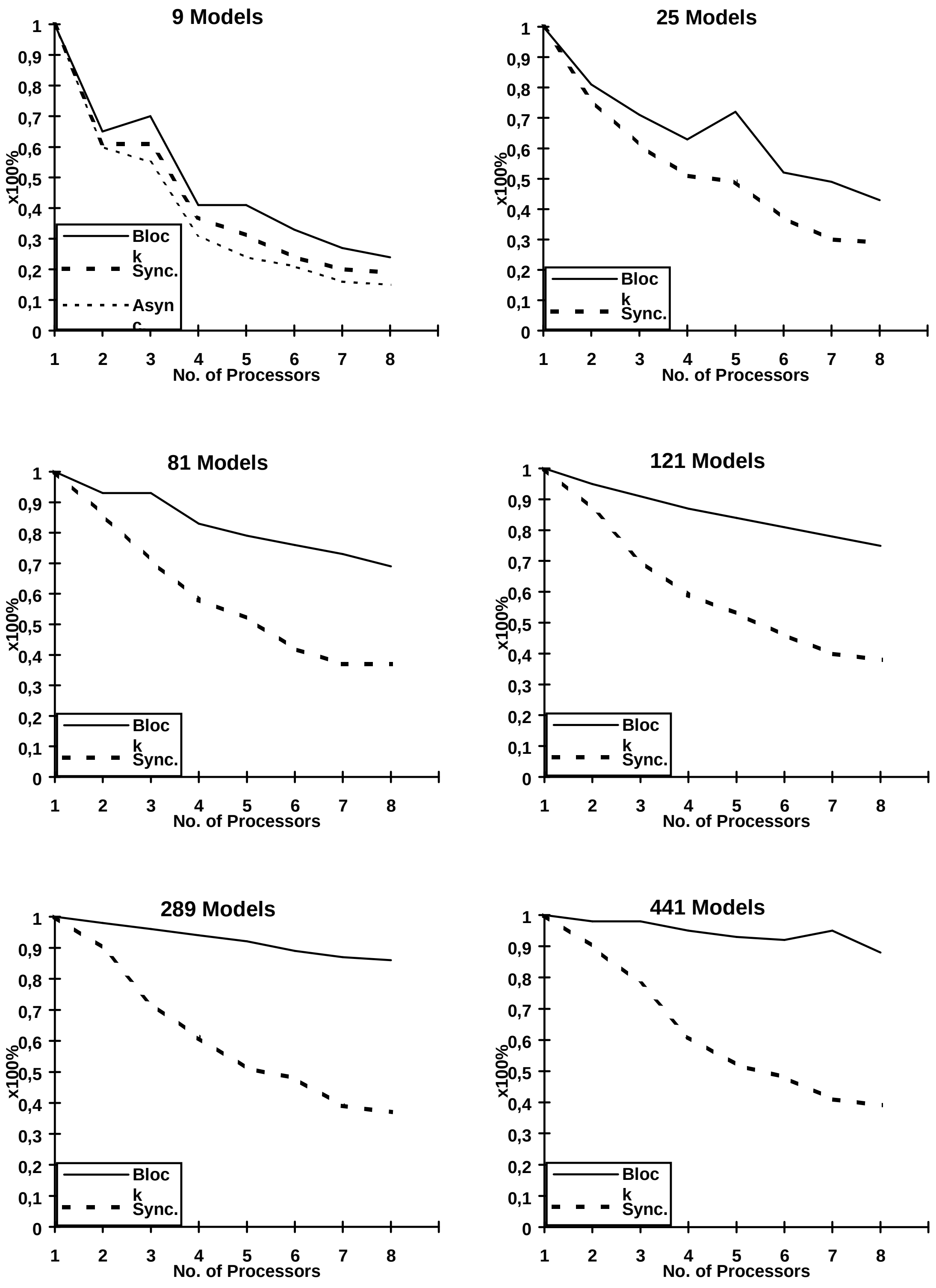
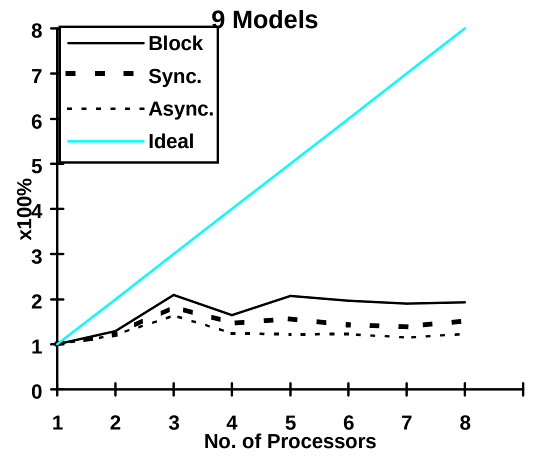
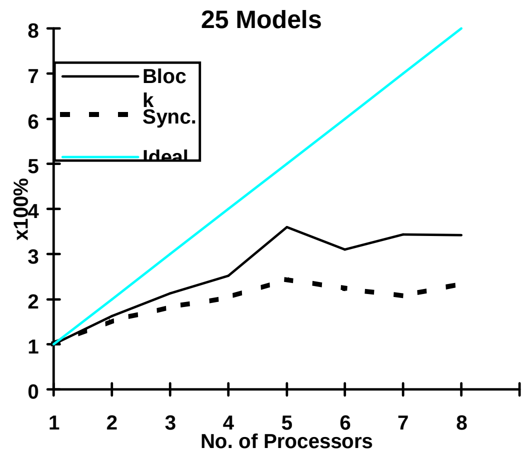
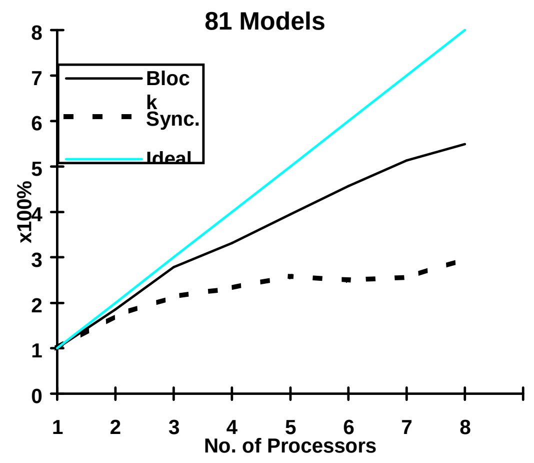
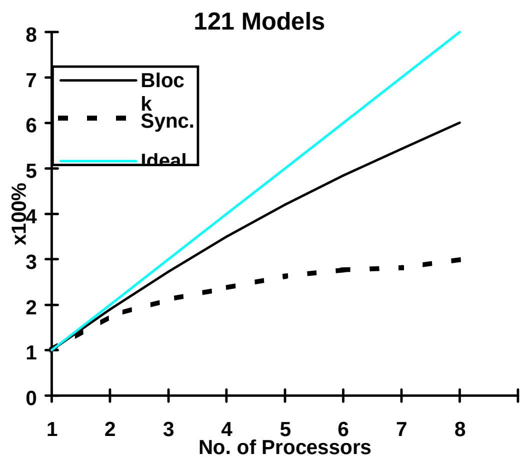


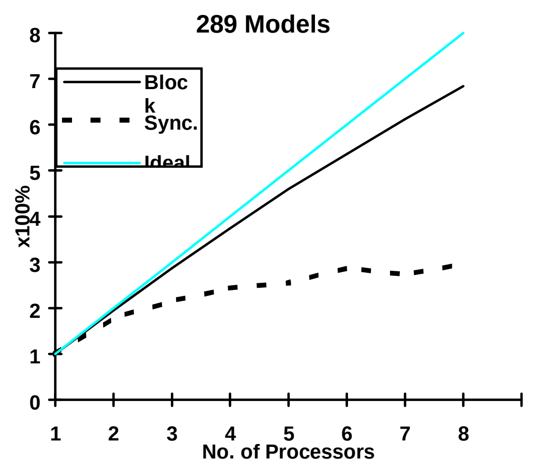
Figure12-Efficiency Graphs for Transputers











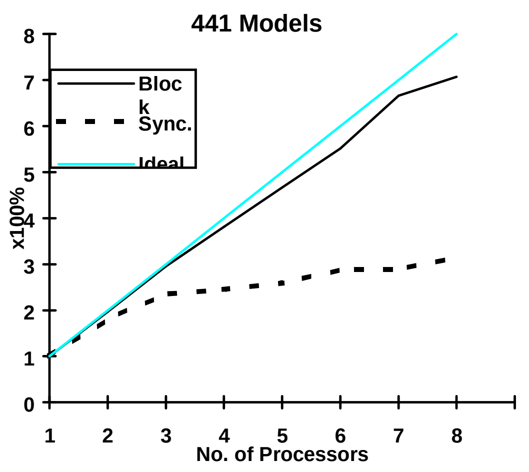


Figure 13 - Speedup Graphs for Transputers

Amongst the communication strategies the ***block*** mechanism has been shown to be the most effective, particularly when the number of models is high. Therefore this mechanism was chosen to be analysed further and to determine the computational cost of each IMM component as a function of the number of models and number of processors. Figure 14 shows the contribution of each IMM component per processor for each implementation using different numbers of models. For instance in the single processor case for a 9 model IMM implementation, the Interaction and Mixing component is responsible for 68% of the overall processing which is consistent with the results presented in [Atherton94]. The graphs confirm the results previously shown in the efficiency and speed-up graphs for implementations with a small number of models. It is the Data Exchange component which is responsible for the loss in efficiency. The increase of the number of models minimises this problem, but there is an increased contribution from the Interaction and Mixing component. Because of this the optimisation discussed in Section 4.3.3 was introduced to reduce the number of floating point operations.

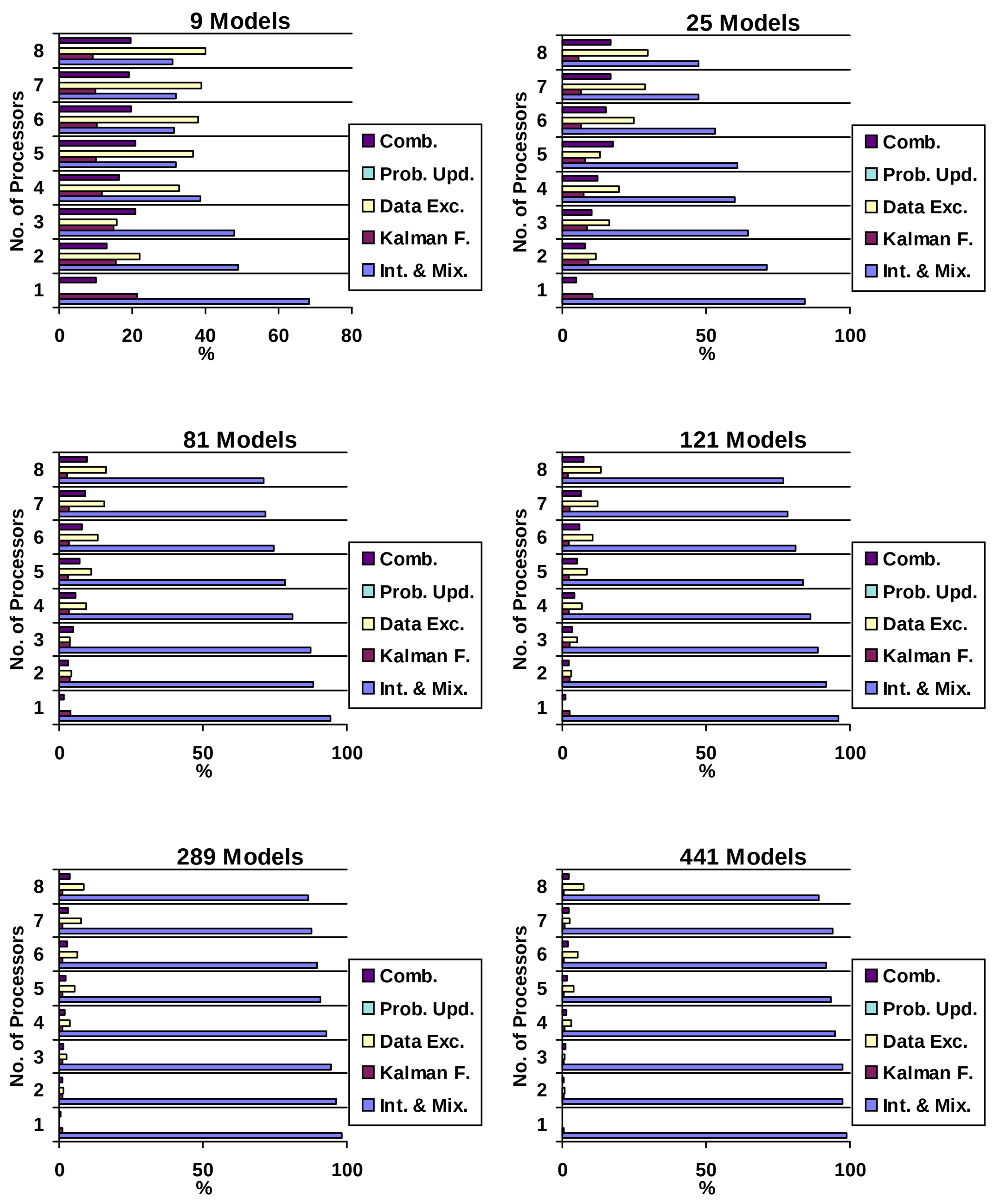


Figure 14 - Component Computational Load for Transputers

Many other different implementations were evaluated using this architecture in order to determine the main differences with different architectures and different message passing. Although they are not shown in detail the results are described next. In order to compare the experiments that follow the block strategy over a fully logical connected architecture using a physical hypercube with EXPRESS as the message passing system was taken as the reference implementation. A first experiment was made by using the most effective communication mechanism, ***block***, and by taking advantage of the hypercube architecture to implement the communication amongst the SPMD distributed IMM. Therefore there was a correct matching between the logical and physical architecture. The results obtained were essentially identical with almost negligible differences in efficiency and speed-up when compared with the reference implementation. A second measurement was done by removing all software optimisations included in the parallel IMM implementation and by using the fully connected architecture with the block strategy. The suppressed optimisations consisted of removing all Interaction and Mixing optimisations described above and the replacement of all scalar operations by a general-purpose matrix operations package [Press88], [Golub83] (addition, multiplication, inversion, etc.). This implementation was 5 to 12 times slower than the reference implementation. A third measurement was made by utilising a different message passing philosophy based on the Virtual Channel Router and PARAPET environment. The results showed that EXPRESS is slightly faster than PARAPET by about 10%.

4.4.4 Meiko CS-2

The Transputer implementation can be considered to be on a low cost, small MIMD architecture and has been shown to be suitable for parallel implementation of a parallel SPMD IMM implementation as long as the number of IMM filters is medium to high. Now a significantly more expensive and potentially larger MIMD architecture, the Meiko CS-2, is analysed for the parallel implementation of IMM. Again the efficiency and speed-up are shown in Figure 15 and Figure 16 respectively.

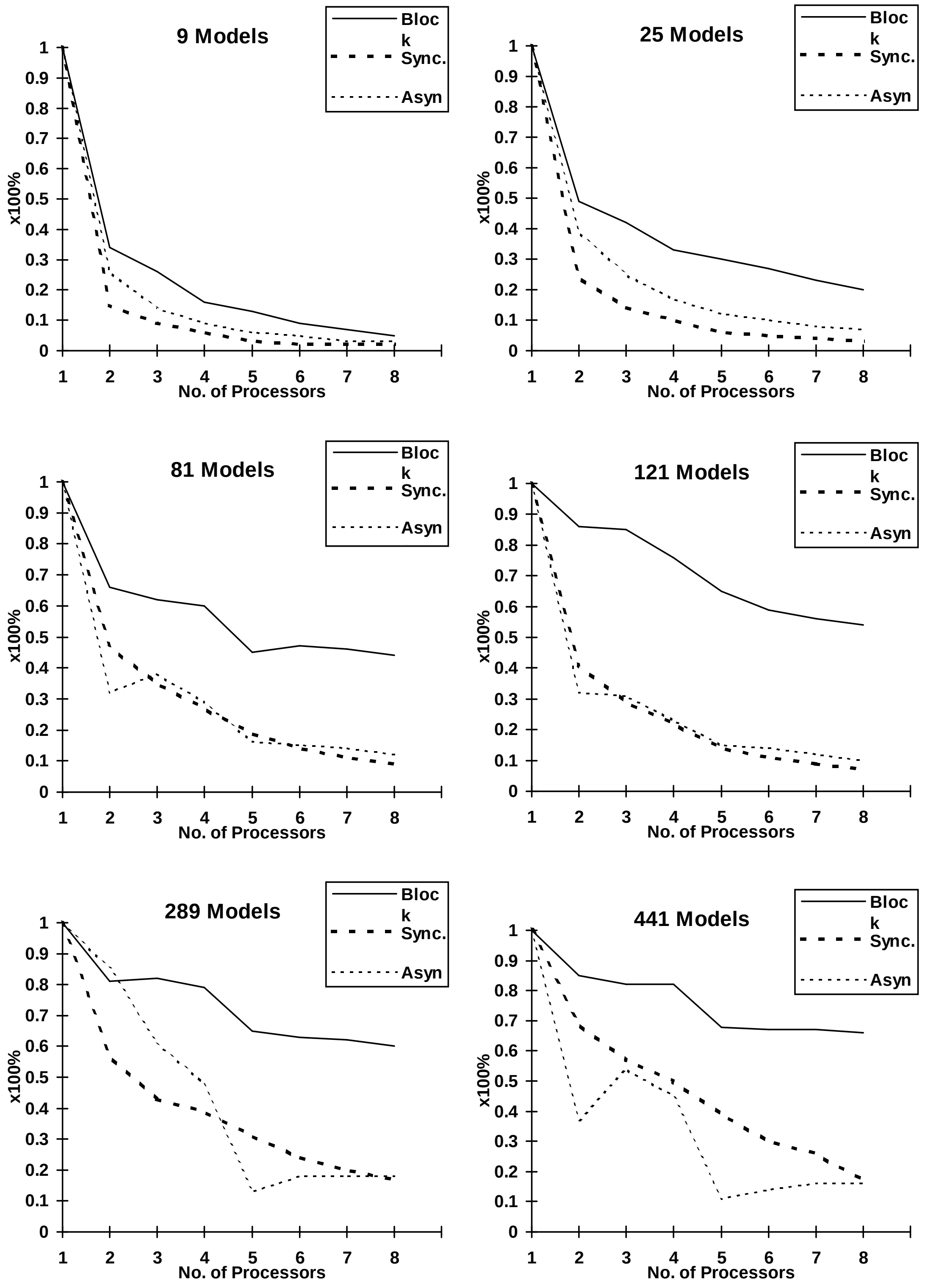


Figure 15 - Efficiency Graphs for Meiko CS-2

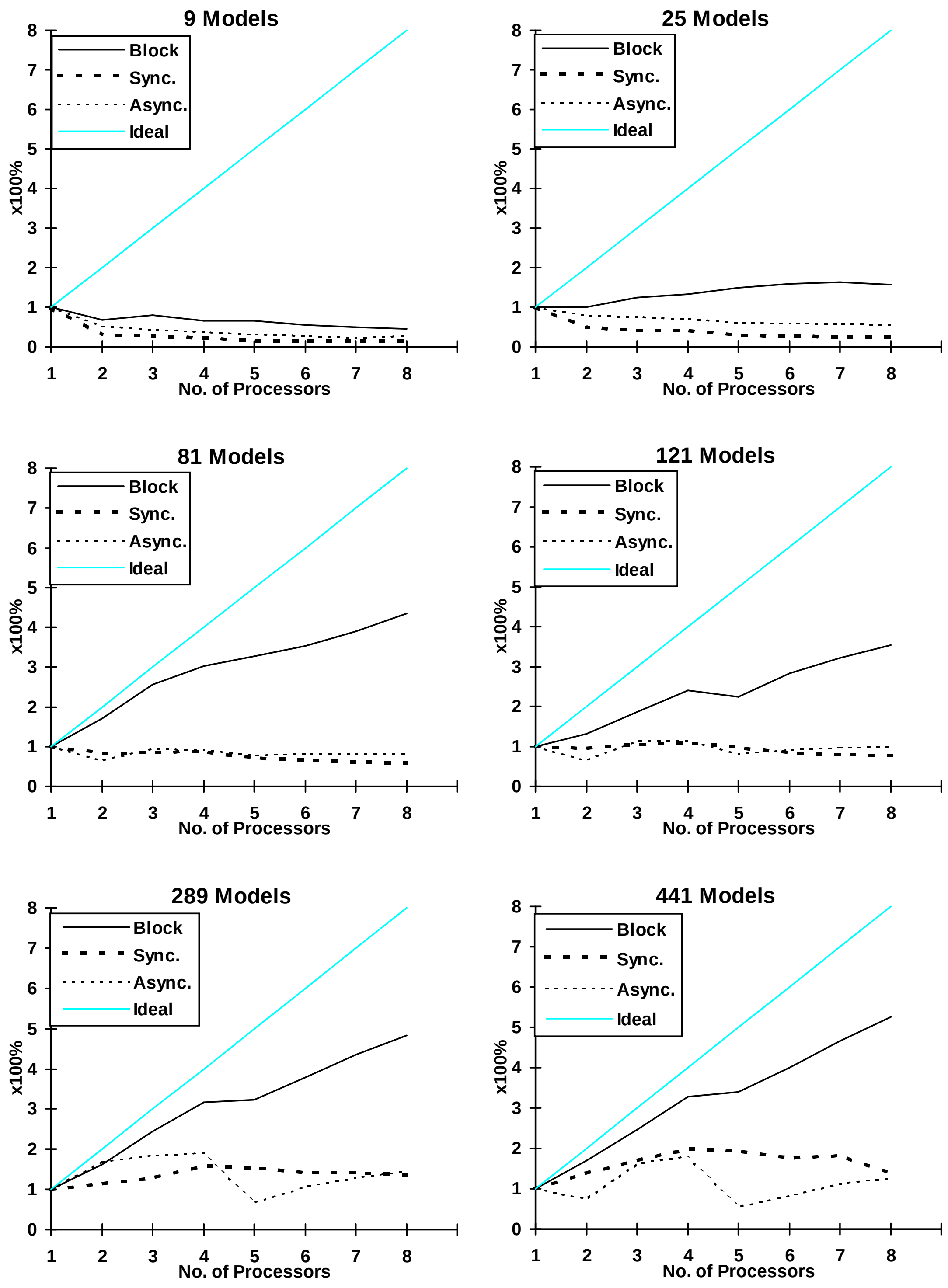


Figure 16 - Speedup Graphs for Meiko CS-2

As can be seen the efficiency drops quite dramatically for the parallel implementations in almost all the cases independent of the number of models used to implement the IMM filter. In particular, for model implementations with a small number of filters, less than 121, there is a strong loss of efficiency when the implementation goes from monoprocessing to dual processing (e.g. in the case of 9 models the efficiency drops on almost 70%). It is also interesting to observe the results for implementations with 81 models or more, which show that whenever a parallel implementation goes from 4 to 5 processors, there is a noticeable drop in efficiency. This loss in efficiency is probably caused by the unevenness of the processors in the CS-2 architecture (scalar and vector processors). A similar analyses can be made by looking at the speed-up graphs in Figure 16. It clearly shows that now reasonable speed-up is achieved independent of the number of models used to implement the IMM filter. The situation is only minimised for the 441 models case until the fifth processor is introduced, when by the introduction of the vector processors the drop in efficiency is noticeable.

Despite the poor quality of the parallel implementation of IMM using the Meiko CS-2, the block strategy has been shown to be the best of the communication strategies. Therefore as in the previous case, it was chosen to be analysed further in terms of the computational load for each of the IMM algorithm components. Figure 17 shows the contribution of each IMM component per processor for each different number of model implementations.

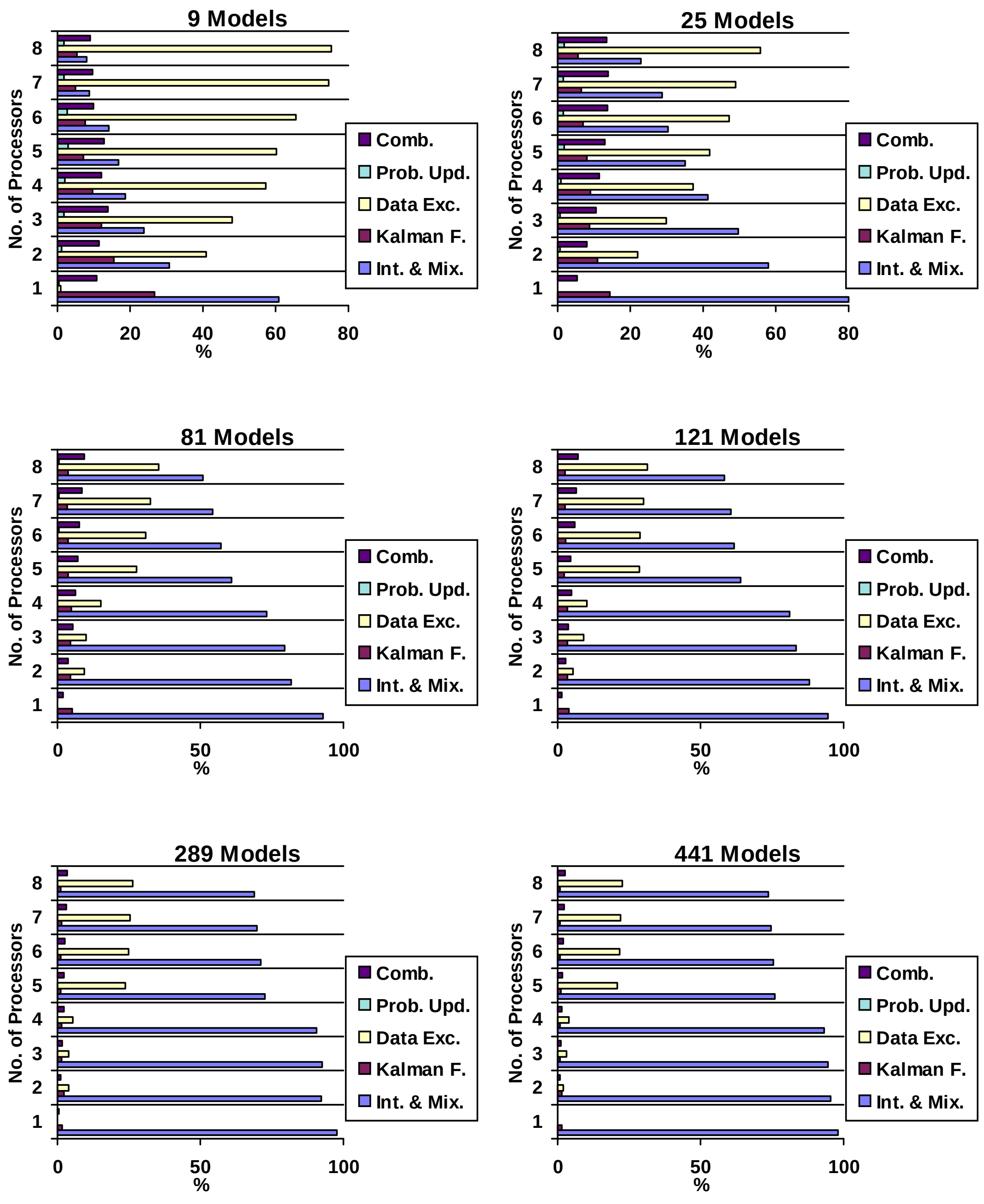


Figure 17 - Component Computational Load for Meiko CS-2

For instance in the single processor case for a 9 model IMM implementation the Interaction and Mixing component is responsible for 61% of the overall processing which again is in accordance with the results presented in [Atherton94]. There is in fact a difference between the contribution of each component for a single processor in each of the architectures (Transputers and CS-2). For example in the above mentioned 9 model implementation the contribution due to Interaction and Mixing for the Transputer architecture is 68% while in the CS-2 it is 61%. A similar disparity occurs for the other algorithm components. This difference in the percentage contribution is caused by the different compilation optimisations inherent to each system. Once again these graphs confirm the results previously shown. The Data Exchange component consumes most of the computational time and consequently reduces the potential parallel efficiency. It is also noticeable that for implementations with more than 25 models, the Data Exchange is increased when the architecture goes from 4 to 5 processors, or in other words by the introduction of the vector processors.

Two other experiments were made with the CS-2 architecture to evaluate the performance of parallel implementations of IMM on this architecture. A comparison was made against a reference implementation. The reference was the implementation based on the block communication strategy. The first experiment suppresses all optimisations made into the algorithm, as in the transputer case. The results has shown a similar increase in the overall processing time, which again is 5 to 12 times slower. A second experiment was made by changing the message passing software. The selected message passing software was the new standard Message Passing Interface (MPI) [MPI94] (this MPI implementation is derived from CHIMP [Alaisdair94]). The results has shown a loss in performance of 4 to 6 times worse than when using the MPSC message-passing emulation software. In addition the overall CHIMP-MPI implementation system is at present inconsistent and extremely unreliable as it is still under development.

4.5 Conclusions

The results presented above took a long time to derive, which will be first justified before discussing the IMM parallel implementation. The main elements that have contributed to such a long delay in concluding the parallel IMM implementation are identified. On the other hand this long learning and development process has, to some extent, introduced many beneficial factors to the general understanding of the two main topics being researched, namely tracking and parallel systems. The main reason for the delays were caused by a number of drawbacks found during the development of the project. These can be divided into three distinct classes of problems, theoretical, systemic and algorithmic.

The first problem refers to the inherent theoretical complexity of the algorithms. As a matter of fact a detailed understanding of the theory involved in estimation is essential if a successful parallel implementation is desired. This implies that the parallelization goes beyond the simple translation of IMM equations into programs. It is also part of one of the main objectives of this research, which is the full understanding of the theory involved in the development of the IMM algorithm. In that sense this objective has been fully achieved.

The second problem refers to the idiosyncrasies of the parallel architectures and message passing systems. A detailed description of all related drawbacks goes beyond the scope of this thesis and besides, it is not helpful to discuss individual problems faced with different platforms and message passing systems as they can be seen as part of the overall development process. However, some of the major issues have contributed to the long development time taken to obtain a parallel implementation of IMM will be identified. On the architecture side, the main problem was hardware instability. In the case of the CS-2 case this is quite noticeable because there is an almost total lack of system support, or at least a very long response time to problems reported. On the transputer case the main problem was initially very difficult to detect. The origin of the problem was unknown and unpredictable which led to a considerable amount of time reconfiguring and setting up the system to ensure that everything was all right. Recently one of the problems was localised and the appropriate corrective action taken. On the software side (the message passing systems) the main problems were the software defects and inconsistencies found in all the packages used. This for instance can be illustrated by the absence of the asynchronous results for the transputer implementation, where despite the fact that the algorithm functions perfectly for 9 models it could not be made to function for any other case. A large number of corrective actions were tried but the system always responded in an unpredictable behaviour.

The third problem has to do with the actual implementation of a parallel algorithm. Again, different problems occurred that prevented the final version from being completed in a shorter time. Amongst those, a typical one was the floating-point round-off problem. This problem refers to different numerical results obtained in the probability update step. Here if the probabilities summation is done in different order on each processor different results are produced. This example is particularly true with the asynchronous communication mechanism implementation. With this implementation the partial probabilities arrive in a nonordered way. If the summation of the partial values is done in order of arrival time, each processor eventually ended up with different probability mode updates. This difference is small initially but it can grow as the algorithm progresses resulting in the estimates diverging on each processor.

The three classes of problems described are inherently difficult by themselves. However when they occur simultaneously (which was normally the case), any attempt to detect and to correct the problem becomes a lengthy and painful process. The above problems also brought a particular insight into each of the mentioned problem areas. From the theoretical point of view it has helped considerably in the general understanding of the many different aspects of such complex theory and its applications. From the system problems a deeper understanding of the parallel architectures and message passing systems was achieved. From the parallel implementation, the comprehensive analyses of the IMM algorithm has led to an excellent understanding of all the components of the algorithm and their interaction. As a result different approaches for the parallel implementation of this algorithm can be now explored, if a more efficient architecture is chosen.

A number of conclusions can be drawn from the current parIMM implementation on two different platforms, namely the Transputer and the Meiko CS-2. These conclusions can be divided into two different levels. On a higher level the conclusions will focus on a general approach that combines both implementations. On a lower level, conclusions will focus on each particular architecture implementation.

Initially parIMM discussion establishes the minimum set of parameters necessary to be exchanged to obtain a IMM parallel implementation. This minimum set includes the local estimates and respective covariances and the partial probability mode update. It also determines the three possible communication mechanisms to implement IMM in parallel using the selected granularity over the chosen architectures.

This experiment has also provided a successful implementation of a parallel Interacting Multiple Model which explores these three possible communication mechanism alternatives when implemented over a MIMD architecture with distributed memory. Amongst the three communication mechanism used, namely, block synchronous, synchronous and asynchronous, the first has been shown to be the most efficient independent of the platform used.

Also a new set of equations was created in order to minimise the number of floating point operations in the IMM filter. These new equations could bring a reduction of up to nearly 50% compared with the original version of Interaction and Mixing step equations. This assertion holds true provided that a large number of models is used along with a reasonable number of state variables to implement the parallel IMM.

In cases where the time taken must be minimised, faster implementations could have be achieved by utilising a simpler message passing system. However development, implementation and debugging time would have been increased. In fact it was noticeable, independent of the message passing system used, that commercial packages still lack adequate reliability, and if the application is time-critical, the use of a simplified special-purpose communication mechanism is recommended instead of general-purpose message passing tools.

Despite the fact that transputer execution times were much slower than CS-2 times, this architecture provides a better match to the parIMM algorithm than the CS-2. In the transputer case some improvements could still be achieved using the same message passing system and the hypercube implementation. Faster times could have been obtained by simultaneously exchanging information over all available physical communication links. In the case of the CS-2, in particular the implementations with a small number of models, no obvious improvements can be identified because of the extremely poor performance on most of the parIMM algorithms.

Chapter 5 : Parallel Interacting Multiple Model with Probabilistic Data Association

5.1 Introduction

In the previous chapter the IMM was parallelized using the SPMD approach instead of the more usual farmer-workers approach [Atherton94], [Averbuch91a], [Averbuch91b]. In order to pursue the investigation of tracking algorithms implemented on parallel system a different approach is taken here. The approach presented here differs from the previous chapter in two ways. First, instead of using a multitude of similar system models differing only in their plant noise, a two model IMM filter will be implemented. Second, an heuristic approach will be used to implement the filter. The two system model IMM to be implemented is well known and discussed in the literature [Li93a]. Such systems have proved to be amongst the best ones developed so far for air traffic control applications [Baret90]. In order to improve the quality of the filter in adverse conditions the IMM filter is coupled with the Probabilistic Data Association [BarShalom75] technique to permit the tracking of targets in clutter environments. The resulting filter is, therefore, an IMMPDA, which is initially investigated in relation to its tracking effectiveness in air traffic control applications. Next the IMMPDA filter is implemented on a parallel system to determine its scalability. The resulting parallel implementations, parIMMPDAs, are presented and analysed to show the obtained parallel efficiency and speed-up. In summary this chapter has two objectives: firstly, to investigate the tracking capabilities of a two-system-model based IMMPDA algorithm with different clutter densities; second to investigate the resulting scalability on a parallel MIMD distributed architecture.

As in the previous chapter the resulting algorithm mechanisation and equations will not be reviewed as they are extensively presented in the literature [BarShalom75], [Blom82a], [Houles89] and [Li93a]. Once again the approach is designed to demonstrate the important points to obtain the final filter set-up. A brief review of the basic equations for a simple Probabilistic Data Association problem is shown in Appendix C and for the IMM in Appendix B. A brief discussion of the important points to be considered when coupling these two widely accepted techniques is also included.

The structure of this chapter is initially to introduce the problem of coupling the Interacting Multiple Model with the Probabilistic Data Association. It briefly describes the necessary formulation to combine them conveniently. Once the structure of the final filter is determined from this coupling it is then possible to select the heuristic task distribution and consequently the parallel approach to be taken. Because of the nature of the filter and the parallel implementation strategy to be used very few options are possible as will be shown. After the determination of the possible implementation options there follows a descriptions of the filter set-up and the data set to be used to test the final filter. The filter set up describes the necessary parameters and constants used in the filter. The data test set description explains, initially, how the clutter was generated followed by the clutter densities used during the filter test. The filter implementation results are then analysed at two different levels. The first level is at the tracking level where the filter effectiveness in real air traffic control situations with different clutter densities will be demonstrated. The second level refers to the computational implementation, which is made on parallel system. The results are also presented in a two-level format, first the algorithm tracking effectiveness in different cluttered environments followed by its parallel speed-up and efficiency. Finally a conclusion is given based on the results obtained.

5.2 Parallelizing the Interacting Multiple Model coupled with Probabilistic Data Association

Because a limited number of filters are now used to implement this tracking filter, it is not necessary to look at the granularity problem, as it is assumed from the previous chapter results that a coarse granularity will be used. The problem, in order to select the best parallel strategy, is to select the task distribution in an heuristic way. However, this begins with a general description about the filter major components, namely IMM and PDA, in order to show how the coupling of these two well-accepted techniques is achieved. This section describes the coupling criteria and the parallel model used to implement the IMMPDA filter.

5.2.1 IMM and PDA Coupling

The IMMPDA is essentially almost identical to the IMM model presented in the previous chapter. However the PDA mechanisation is introduced into the Kalman equations as described in [BarShalom88] with the following two other important features:

1. A gating mechanism
2. A PDA-based likelihood calculation.

The gating mechanism is used in conjunction with the filter to select the measurements to be taken into consideration by the IMMPDA Kalman filters. The basic gating equations are given in Appendix C and not repeated here. The two models used in the IMM are producing two different measurement predictions and associated covariances as part of their IMM Kalman filter mechanisation. As a result the predicted measurement to be used in the gating mechanism should be a mixture of the measurement predictions from both filters. The mixing is based on the assumption that the measurement prediction is given by the following estimate :

$$\begin{aligned}\hat{\mathbf{z}}(k|k-1) &= E\left[\mathbf{Z}(k)\,|\mathbf{Z}^{\mathbf{k}-1}\right]\\ &= \sum_{i=1}^{2}\sum_{j=1}^{2} E\left[\mathbf{Z}(k)\,|M_i(k), M_j(k-1), \mathbf{Z}^{\mathbf{k}-1}\right] P\left\{M_i(k), M_j(k-1), |\mathbf{Z}^{\mathbf{k}-1}\right\}\end{aligned}$$

To conclude, the gating mechanism is necessary to determine the associated predicted measurement covariance. The choice is to use the one with the largest determinant to guarantee that a large number of measurements fall inside the validation region which corresponds to the worst case, or a more widespread covariance [Houles89].

$$\begin{cases}\mathbf{S}(k) = \mathbf{S}_1(k) & \det\left[\mathbf{S}_1(k)\right] \geq \det\left[\mathbf{S}_2(k)\right]\\ \mathbf{S}(k) = \mathbf{S}_2(k) & \textit{otherwise}\end{cases}$$

In the conventional IMM algorithm the likelihood function is calculated assuming a Gaussian distribution for only one measurement. In the IMMPDA algorithm all measurements used in the filter must be used to determine the likelihood function. To establish this, we start from the essential assumption concerning the likelihood function, which is

$$\Lambda_j = p\left\{\mathbf{Z}(k) | M_j(k), \mathbf{Z}^{\mathbf{k}-1}\right\}$$

and for the IMMPDA case this results in

$$\Lambda_j = \frac{\gamma_0(m_k)}{V^m(k)} + \sum_{i=1}^{m_k} \frac{\mathrm{N}(\nu_i; 0; \mathbf{S})}{V^{m_k - 1}\, P_G} \gamma_i(m_k)$$

where in the above

$$\gamma_i(m_k) = \begin{cases} \dfrac{1}{m_k} P_D\, P_G \ , & i = 1, \ldots, m \\ 1 - P_D\, P_G \ , & i = 0 \end{cases}$$

and all other parameters are described in the Appendix C and in more detail in [BarShalom88].

As a result Figure 18 resents a essential decomposition of the resulting IMMPDA components already broken down with a view to parallel implementation. Based on Figure 18 a strategy to implement the algorithm in parallel can be established. Once again, as in the previous chapter the targeted granularity is coarse grain. The reason are essentially identical to the SPMD case. However as shown in Figure 18 only two models (filters) are present in the IMMPDA implementation which makes an SPMD approach attainable in only two processors in a parallel architecture. Although the coarse granularity will be used to implement the parIMMPDA, an heuristic approach will be applied. This approach will introduce a limitation in the breaking up or distribution of tasks amongst processors. If the overall IMM is divided too much, the resulting granularity will be too small, contradicting the initial coarse granularity principle described in the previous chapter. By observing Figure 18 it seems possible using a straightforward approach to distribute each component in a processor in the parallel architecture. It is obvious that this will cause a mismatch between the architecture granularity and the algorithm granularity, as a consequence of not extracting the best from the MIMD parallel architecture being used and thus inevitably resulting in a poor parallel efficiency. To this problem the heuristic distribution needs to be made with caution. The simplest approach is to start with minimal heuristic distributions as described next.

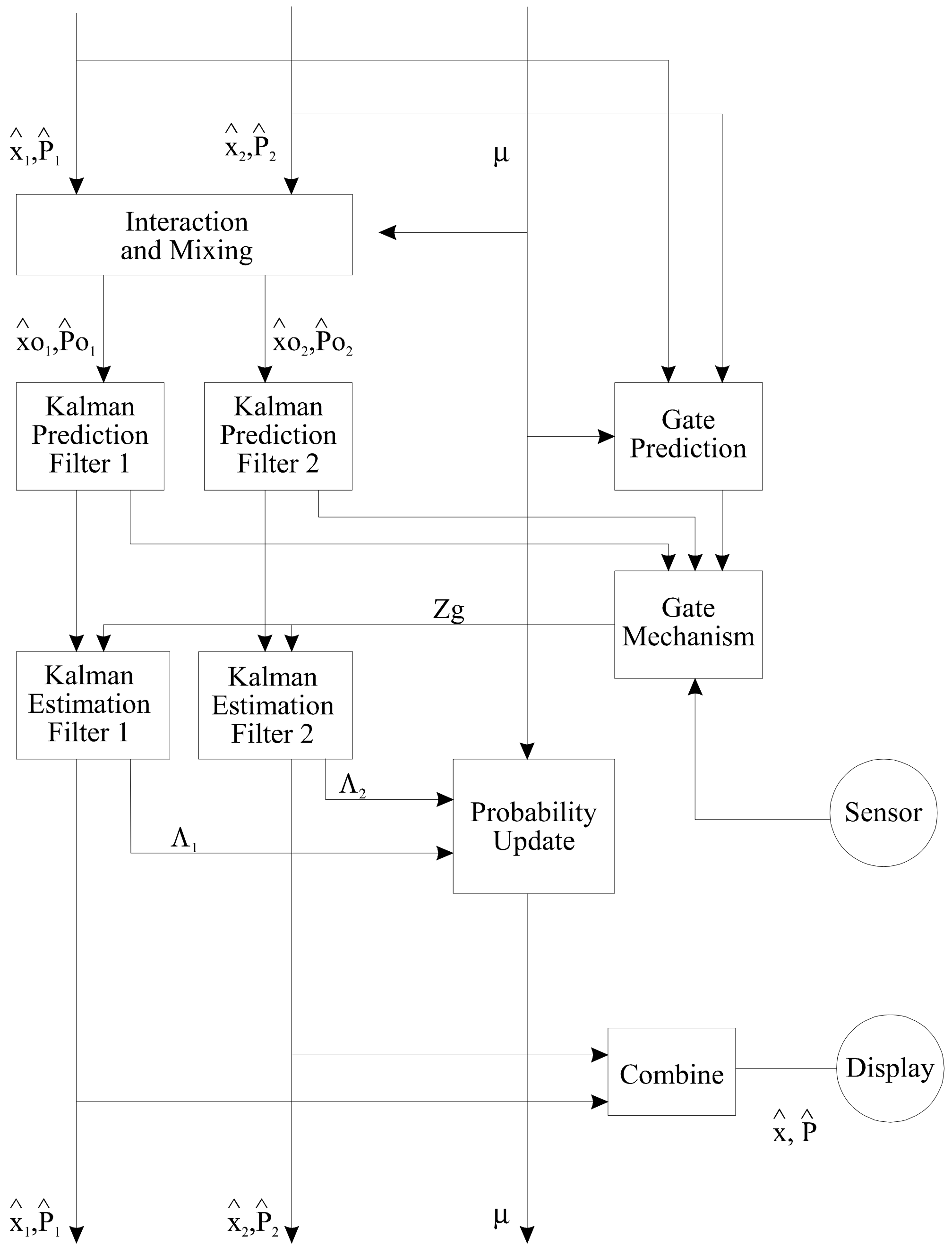


Figure 18 - IMMPDA

5.2.2 The parIMMPDA models

The heuristic task distribution approach was attempted only for two and three processors as described below. The gating mechanism is responsible for selecting only those sensor measurements that are taken into consideration to be processed further by the IMMPDA filter. Because of that, in heavy clutter environments the gating mechanism will receive many measurements from the sensor. It would be unwise to have the gating mechanism separate from where the sensor measurements are sent. Therefore, it has to be in the processor where the sensor measurements are processed. This will also reduce the amount of measurements that eventually will be sent to the other processors for further processing. Two possible task distributions have been devised.

The first distribution utilises 2 processors and the task distribution is done by allocating each of the filters to each processor. Also in order to reduce the amount of information exchanged between the two processors, the Interaction and Mixing module is present in both processors, as well as the Probability Update module. In this manner the Data Exchange module presented in the previous parallel IMM implementation is used once again. Its utilisation permits both processors to obtain the minimum set of data, determined in the previous chapter, that is necessary to perform the IMM mechanisation independently on each processor. The gating mechanism being in the first processor will also require information about the predicted measurement covariance generated in the second processor. In turn the first processor after performing the gating over the measurements sent by the sensor will send only those measurements lying inside the validation gate to the second processor. By allocating the gating mechanism to the first processor the amount of measurement sent to the second processor is reduced. It is also convenient to have the system model with less computational overhead allocated the same processor. In the IMMPDA filter implementation the less computational-intensive Kalman filter is the one corresponding to the linear target kinematics model, or straight-line constant-velocity, while the more computationally intensive is the non-linear model or constant co-ordinated turn. Therefore, the linear model along with the gating mechanism is allocated to the first processor (processor 1), and the non-linear model to the second processor (processor 2).

Both models are described in more detail in the next section. Finally the task distribution over two processors is illustrated in Figure 19.

The second task distribution makes use of three processors. The gating mechanism is allocated entirely to the first processor (processor 1), while the two filters are allocated to the two remaining processors (processors 2 and 3). Once again in order to reduce amount of information exchanged between the two processor, processors 2 and 3, the Interaction and Mixing and Probability Update modules are present in both processors. The Data Exchange module is again used in processors 2 and 3 to allow both processors to obtain the minimum set of data to perform the IMM mechanisation independently on each processor. The gate prediction in processor 1 requires the estimations and probability modes of the two Kalman filters running in the other processors. As processors 2 and 3 are both producing these data, processor 2 has been chosen to send them to processor 1. This selection has not been arbitrary, it has been determined by the Kalman filter being used in that processor. The Kalman filter in processor 2 is the one with less computational intensive that corresponds to straight line constant velocity model or linear model. As in the previous implementation the gating mechanism will require information about the predicted measurement covariance obtained in processors 2 and 3. The final filter distribution is shown in Figure 20.

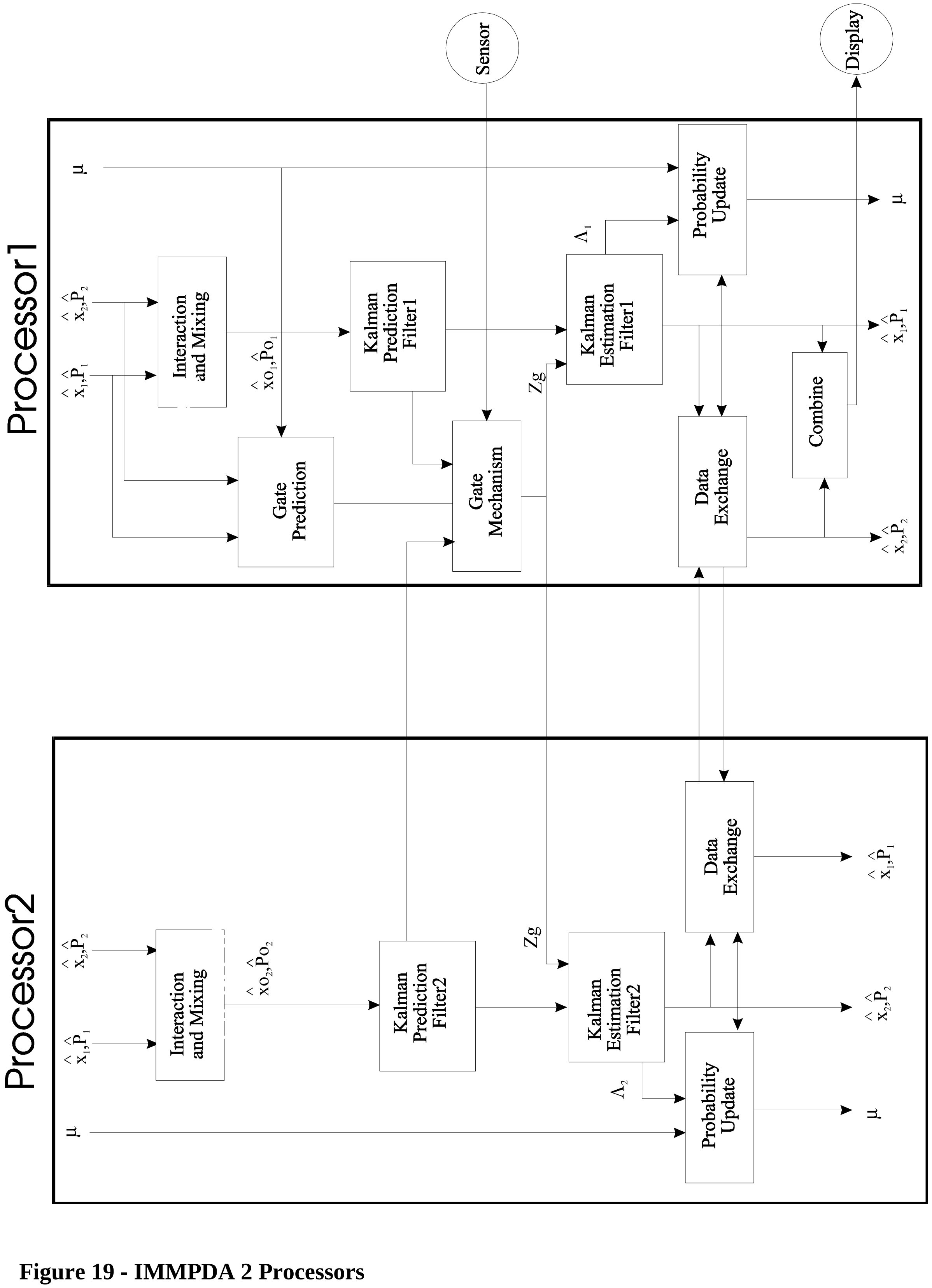


Figure 19 - IMMPDA 2 Processors

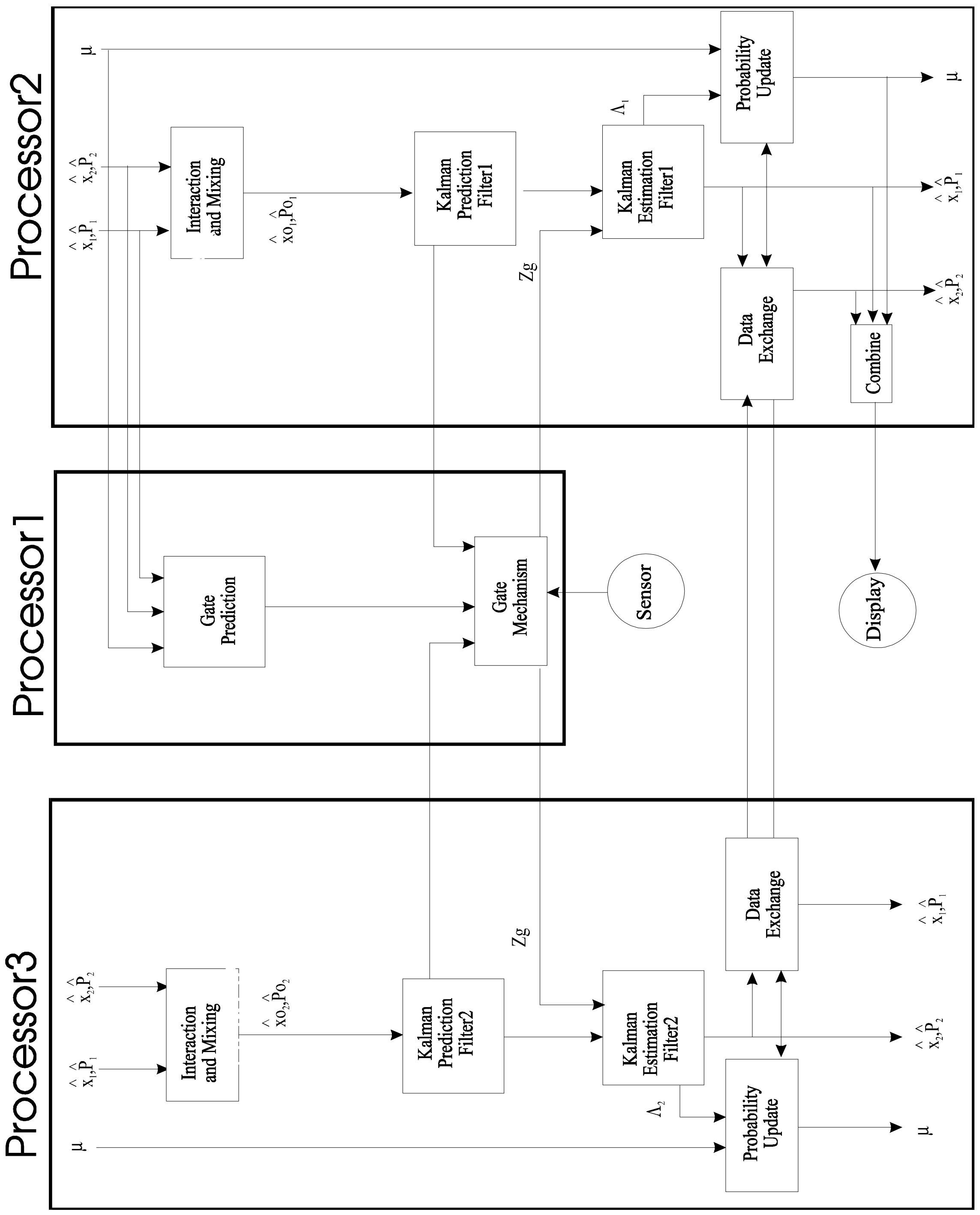


Figure 20 - IMMPDA 3 Processors

5.3 Implementations

5.3.1 parIMMPDA Set-up

The two system models to be described have been widely accepted as a reasonable compromise for implementing tracking filters in air traffic control applications. They have been shown in [Barret90] to perform well in various real air traffic control tracking situations. This filter will also be used in the air traffic control system which will be implemented in the next generation of ATM [Vacher92]. A similar filter has also been implemented in other air traffic control applications as in [Lerro93a] and [Blom92] (the former is also going to be part of a future air traffic control system). All of the elements necessary to set-up the resulting IMMPDA filter will now be described

Straight-Line Constant-Velocity Kinematics Model (SLCV)

The straight line constant velocity trajectory can be modelled as an uniform constant velocity movement in the x-y Cartesian plane with some small variations in velocity that are modelled by a Gaussian white noise. They are represented by the following first order discrete time state equations

$$\mathbf{x}(k+1) = \mathbf{A}\,\mathbf{x}(k) + \mathbf{C}\,\nu$$

where in the above

$$\mathbf{A} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} T^2/2 & 0 \\ 0 & 0 \\ 0 & T^2/2 \\ 0 & 0 \end{bmatrix}$$

and

$$\mathbf{x} = \begin{bmatrix} x & V_x & y & V_y \end{bmatrix}', \qquad \blacksquare = \begin{bmatrix} \blacksquare_x & \blacksquare_y \end{bmatrix}'$$

where in the above, *T* is the sensor sampling period, x and y denote the Cartesian co-ordinates, V_x and V_y the velocity, and ν_x and ν_y are zero mean Gaussian white noise sources used to model small accelerations caused by any external disturbances such as turbulence, wind change, etc..[Li93a]. The process noise variances in each co-ordinate are assumed to identical and their values given by

$$\sigma_{\nu_x} = \sigma_{\nu_y} = 4 m / s^2$$

Constant Co-ordinated Turn Rate Kinematics Model (CCTR)

The constant co-ordinate turn rate trajectory can be modelled as an uniform constant velocity, constant turn rate movement in the x-y Cartesian plane with some small variations in velocity that are modelled as a Gaussian white noise. They are represented by the following non-linear discrete time state equations

$$\mathbf{x}(k+1) = \mathbf{A}(\mathbf{x}(k)) + \mathbf{C}\nu$$

where in the above

$$\mathbf{A} = \begin{bmatrix} 1 & \frac{\sin(\omega T)}{\omega} & 0 & -\frac{1-\cos(\omega T)}{\omega} & 0 \\ 0 & \cos(\omega T) & 0 & -\sin(\omega T) & 0 \\ 0 & \frac{1-\cos(\omega T)}{\omega} & 1 & \frac{\sin(\omega T)}{\omega} & 0 \\ 0 & \sin(\omega T) & 0 & \cos(\omega T) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \\ 0 & 0 \end{bmatrix}$$

and

$$\mathbf{x} = \begin{bmatrix} x & V_x & y & V_y & \omega \end{bmatrix}', \qquad \blacksquare = \begin{bmatrix} \blacksquare_x & \blacksquare_y \end{bmatrix}'$$

where in the above T is the sensor sampling period, x and y denoting the Cartesian co-ordinates, V_x and V_y the velocity, ω the constant turn rate, and ν_x and ν_y are zero mean Gaussian white noise used to model uncertain accelerations in x and y caused by any small variation in ω[Lerro93b]. The process noise variances in each co-ordinate are assumed to be identical and their values given by

$$\sigma_{\nu_x} = \sigma_{\nu_y} = 2\, m / s^2$$

Measurement Model

The sensor considered delivers measurements in Polar co-ordinates (❒, ❑) and given by

$$\mathbf{z}(k) = \mathbf{H}\big(\mathbf{x}(k)\big) + ◆$$

where

$$\mathbf{z}(k) = \begin{bmatrix} ❒(k) \\ ❑(k) \end{bmatrix},$$

$$\mathbf{H}\big(\mathbf{x}(k)\big) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \arctan\left({}^{y}\!/_{x} \right) \end{bmatrix},$$

$$◆ = \begin{bmatrix} ◆_{❒} \\ ◆_{❑} \end{bmatrix}$$

The measurement system is the same one for all models and the measurement noises w_x and w_y are assumed constant and equal for all the models. Its covariance is given by

$$\mathbf{R} = E\left[◆\, ◆' \right] = \begin{Bmatrix} ◆_{❒}^2 & 0 \\ 0 & ◆_{❑}^2 \end{Bmatrix}$$

where ♦ ❐ $= 30m$ and ♦ ❑ $= 0.9\,mrad$ represent the errors in range and azimuth respectively [Stevens88].

A second order extended Kalman filter (see Appendix A) is also used to cope with the non-linearity in the measurement model. The CCTR filter also utilises a second-order Kalman approximation to linearize its non-linear system model equation.

The switch from a manoeuvre model (CCTR) to the constant velocity (SLCV) is possible by setting the turn rate state variable to zero. Conversely, the switch from the constant model (SLCV) to the manoeuvre model (CCTR) is possible by changing the turn rate from zero to a value which is a random variable modelled as Gaussian with a distribution $\mathrm{N}\left(\mu_\omega, \sigma_\omega^2\right)$. Note that other jump techniques could be used as discussed in [Lerro93a]. The Gaussian distribution assumed is given by

$$\mu_\omega = 0^\circ / s$$
$$\sigma_\omega = 0.2^\circ / s$$

Transition Matrix

The model switching probabilities are represented by a Markov chain transition matrix given by

$$\pi = \begin{bmatrix} \pi_{SLCV \to SLCV} & \pi_{SLCV \to CCTR} \\ \pi_{CCTR \to SLCV} & \pi_{CCTR \to SLCV} \end{bmatrix}$$

where in the above a transition probability $\pi_{1 \to 2}$ represents the probability of going from model 1 to model 2. The following probabilities were used for the above matrix for the filter implementation:

$$\pi_{SLCV \to SLCV} = 0.9$$
$$\pi_{SLCV \to CCTR} = 0.1$$
$$\pi_{CCTR \to CCTR} = 0.8$$
$$\pi_{CCTR \to SLCV} = 0.2$$

The initial mode probabilities for both models were assumed to be :

$$mu_{SLCV} = 1.0$$
$$mu_{CCTR} = 0.0$$

Probabilistic Data Association Parameters

The gate is normally a g-sigma ellipsoid [Fortmann83] and the P_G is the probability that the correct measurement lies inside the gate. The probability of a measurement falling inside the validation region is given by

$$P_G = 0.98$$

and the corresponding g-sigma for a two dimensional gate (extract from chi-square distribution table) is given by

$$g^2 = \gamma = 9.21$$

For simplicity, the probability of detection is assumed to be equal to 100 percent or

$$P_D = 1.0$$

Track Initialisation

The track initialisation is done assuming a non-cluttered environment with a two-point difference technique [BarShalom88]. The clutter is introduced into the experiment after the third sensor measurement sample simulation.

5.3.2 Data Test Set

To test the application just described it is necessary to simulate the measurements produced by the sensor including also the generation of clutter. Once again the trajectory simulator explained in the previous chapter was used to generate in an identical fashion simulated measurements from sensors. However another module was introduced to generate clutter around the obtained simulated measurements. This module was built using MATLAB and the clutter generation implemented in the same manner as [BarShalom75]. The simulated trajectory in a clean environment for testing the above filter comprises 3 constant-velocity segments and two constant-turn rate as illustrated in Figure 21. In order to perform Monte Carlo testing on the filter a set of 50 identical trajectories but with different real position perturbations were generated. The filter is initially tested against the clean trajectory using Monte Carlo testing, to demonstrate the filter quality in clean environment and to compare and validate with results already presented in [Barret90] and [Vacher92]. Such comparison is possible because the IMMPDA filter when used in a clean environment is identical to an IMM filter.

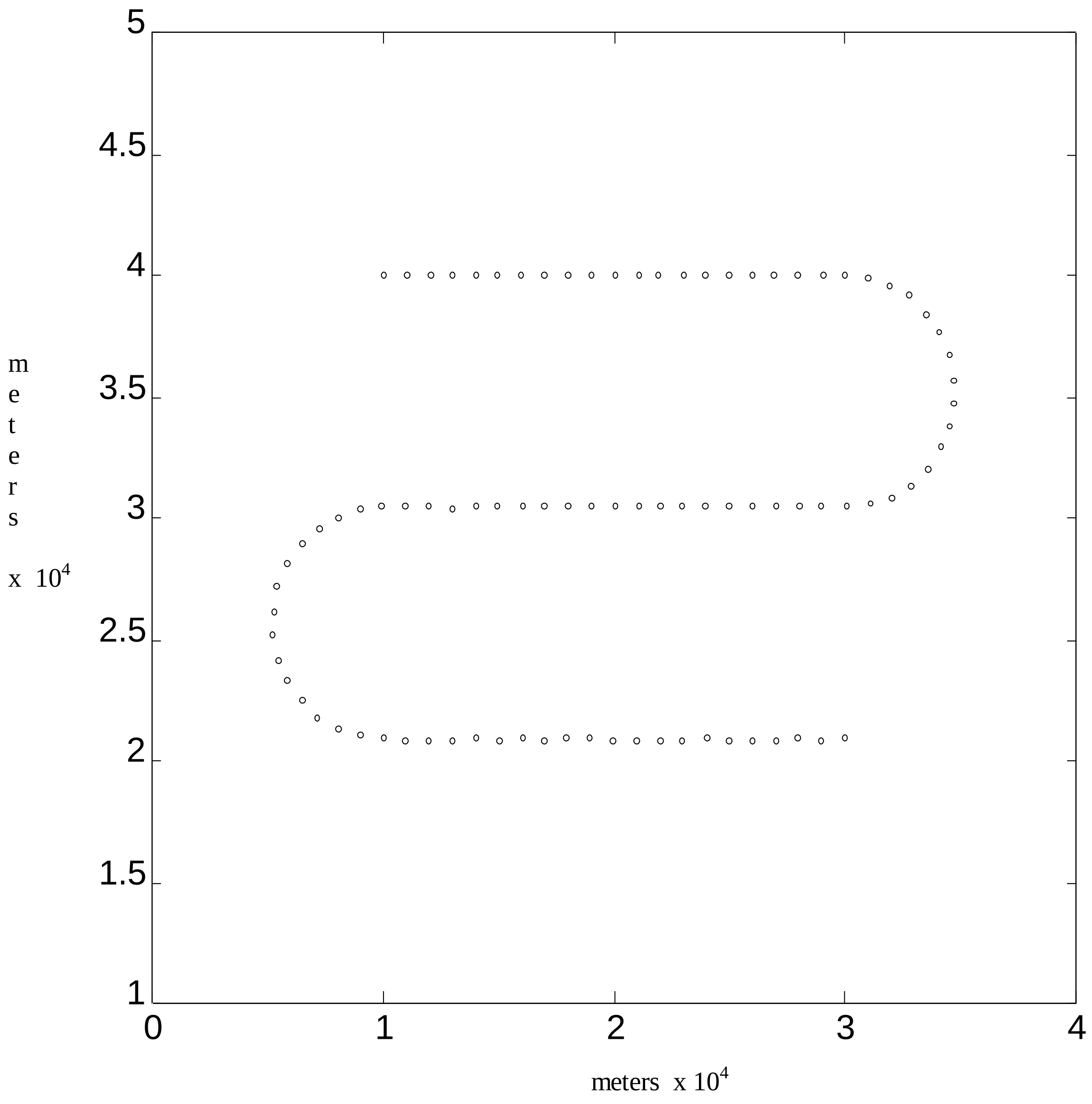


Figure 21 - Trajectory without clutter

The clutter module built utilises the noise measurements obtained above to generate simulated clutter. The module generates new measurements including the original ones with added spurious measurements in any particular clutter density desired. In order to test the effectiveness of the IMMDA filter in different clutter densities, three densities clutter were simulated, their values are shown in Table 5

	No Clutter	Light Clutter	Medium Clutter	Heavy Clutter
Density $\left(\frac{\text{points}}{\text{meter miliradians}}\right)$	0.0	0.000001	0.000013	0.0006

Table 5 - Clutter Densities

5.3.3 Other Considerations

In order to evaluate the parallel performance of the filter, two different sets of clutter were used. The first set contains the clutter densities shown in Table 5 and they are called from now on Lambda0, Lambda1 and Lambda2. The second set also contains the same clutter densities as the previous set, however a larger number of points has been generated by the simulation. This increase correspond to a proportional increase in the area over which the points are spread in order to keep the densities. This does not introduce any alteration to the tracking results, it permits the parallel scalability to be analysed as the number of measurement increases. The second set contains the subsets Lambda3, Lambda 4 and Lambda5. Both sets are designed to replicate the typical tracking environment that the IMMPDA filter will have to cope with, Figure 22, Figure 23, and Figure 24 show the typical cluttered trajectory for the first set.

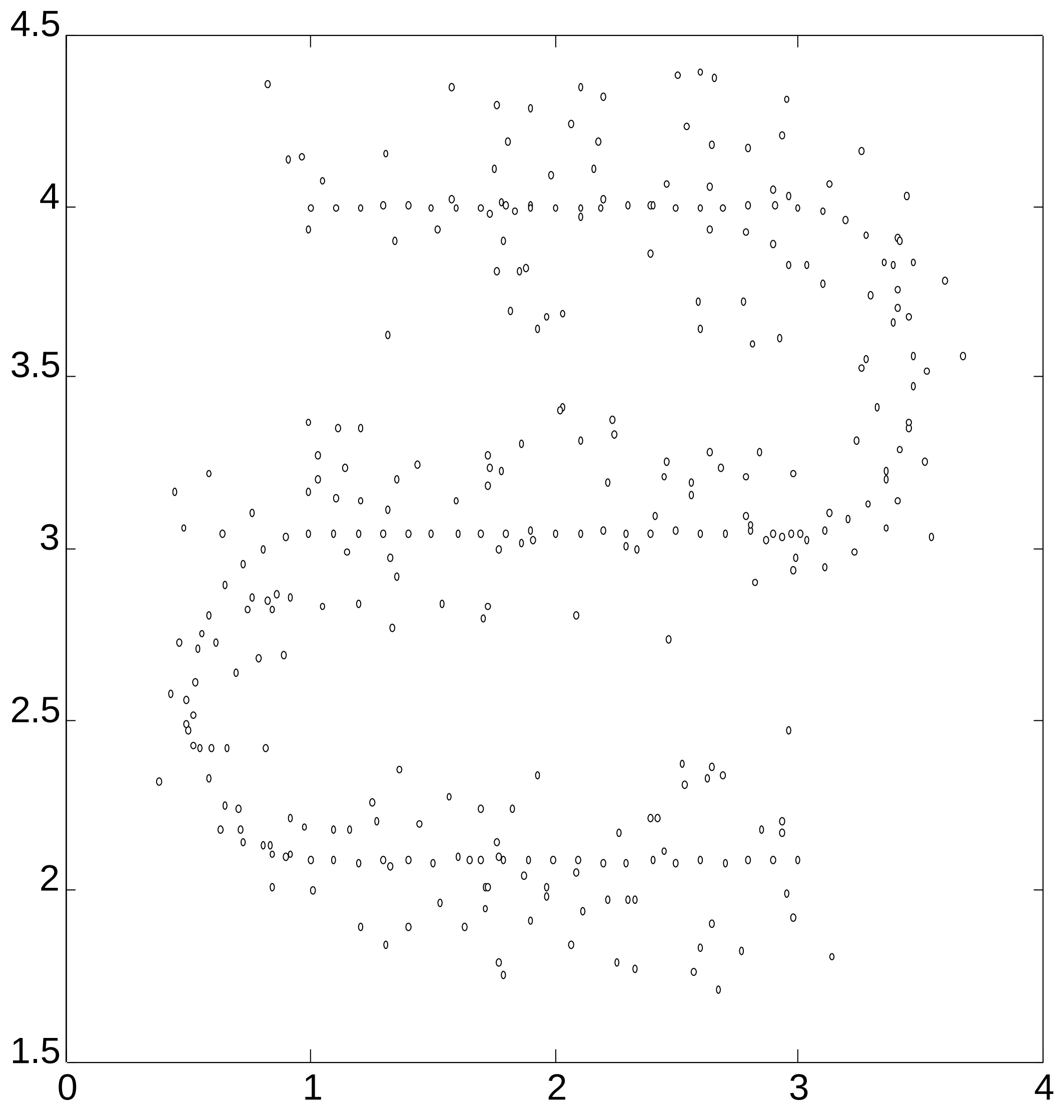


Figure 22 - Light Clutter

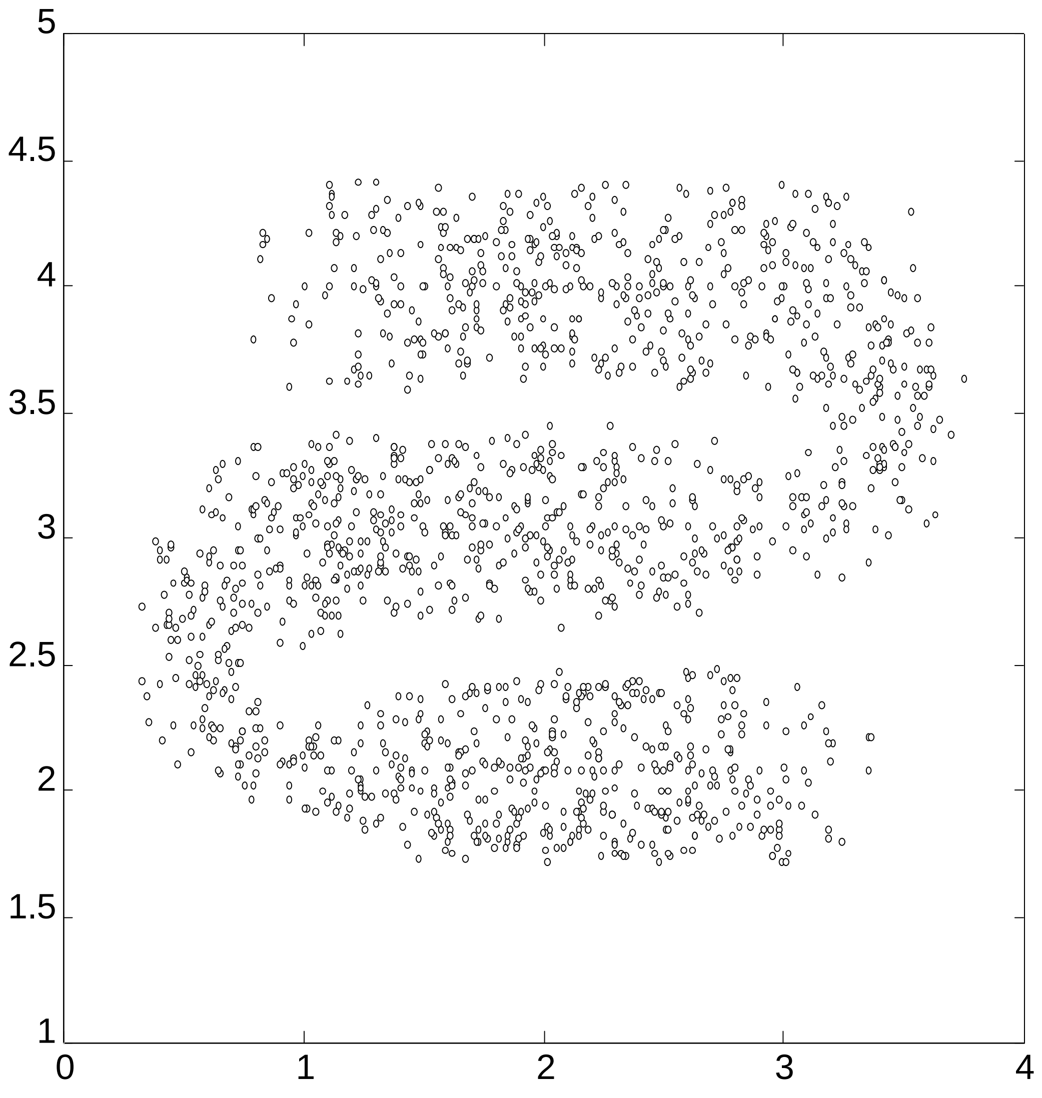


Figure 23 - Medium Clutter

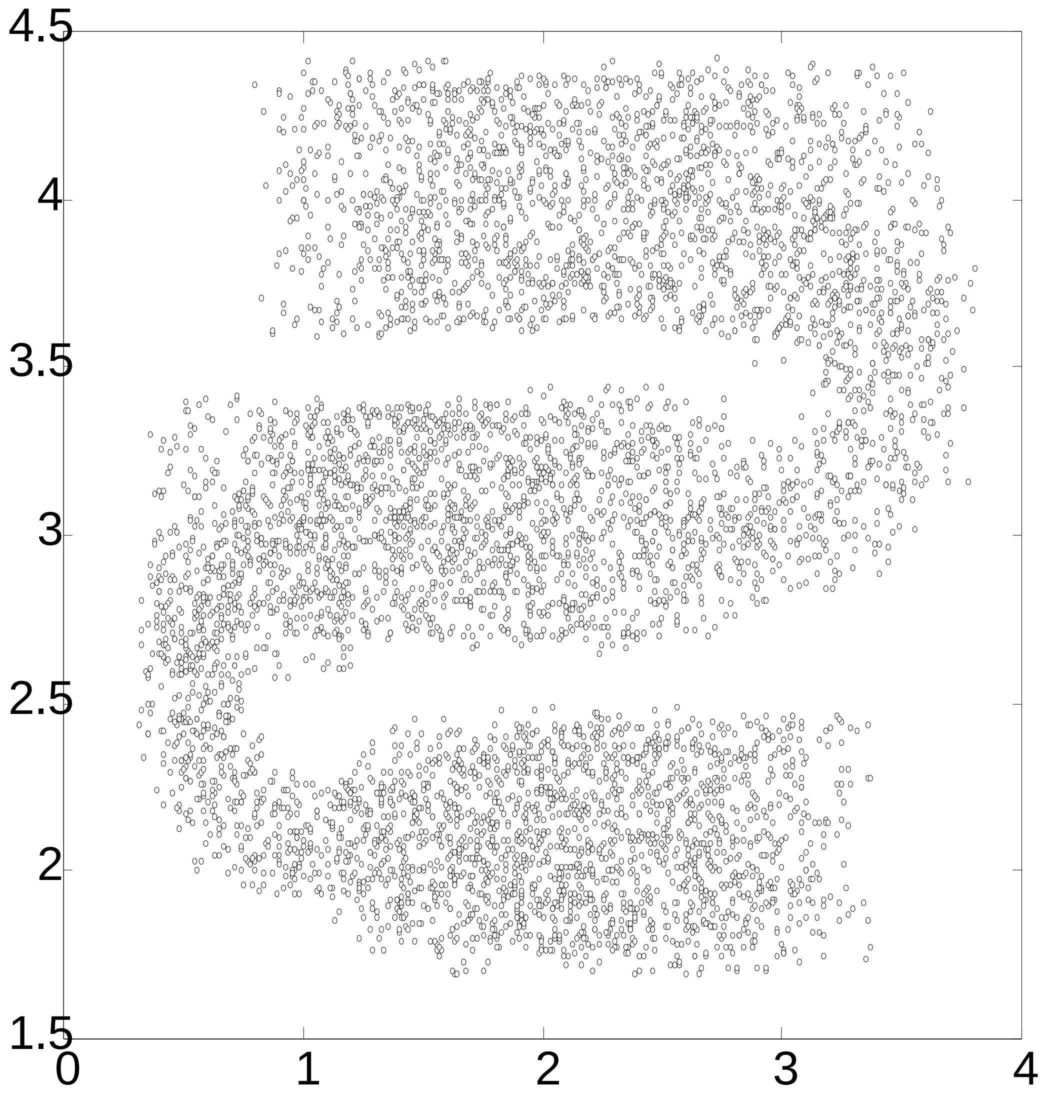


Figure 24 - Heavy Clutter

5.4 Results

5.4.1 Introduction

The first part of the results has the objective of demonstrating the tracking quality of the filter. Consequently, for simplicity, the filter is initially configured and tested over a single processor and tested against the simulated trajectory and its effectiveness determined. The second part of the results refers to the parallel implementation speed-up and efficiency. Therefore the two parallel implementations previously shown are tested, and their speed-up and efficiency calculated utilising the previously described set of measurements.

5.4.2 Validation of Tracking Results

In a similar fashion to the methodology explained in Section 4.4.2, the algorithm IMMPDA were first simulated in sequential form using MATLAB. These simulations confirmed that there was a possibility of *track loss* under some circumstances. Two different types of problems occur that can be considered as *track loss*. The first type is characterised as pure *track lost* and the second as *track divergence*. For the first type or pure *track loss* again two possible alternatives occur and this usually happens at the end of the first turn. In the first alternative the track is lost when the gate mechanism generates such a small gate that no measurements lies inside the gate. The second possibility occurs during the switch from one model to another, in this case from CCTR to SLCV which is too slow and as a consequence generates a bad predicted position and poor mode probabilities. As these values are used in the calculation of the new gate position and dimensions wrong values for the gate are produced, this results in a gate being too far from the next real measurement with a wrong size. The second type of *track loss* or *track divergence* can be characterised as loosing the track by increasing the number of points being incorporated by the filter. This becomes critical again at the end of the first turn and the beginning of the second straight line movement where more and more points are used to update the filter. This can be translated as an increase in the uncertainty about where the real trajectory measurements actually are. This in fact just reflects the probabilistic nature of the filter, that whenever there is an uncertainty about the correct position of the track the gate size is increased in order to obtain more information.

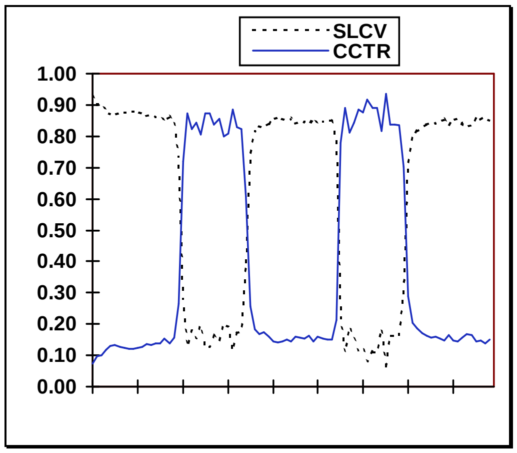
In the simulation, for a given set of filter parameters satisfactory tracking was maintained as the clutter was increased until a critical level was reached, when tracking would then be lost. The main direction of the research project was to investigate the parallel implementations of the state-of-the-art tracking filter, rather than to improve the effectiveness of the filtering results. In this context it is sufficient to ensure that adequate tracking is maintained over the range of data selected for the performance tests. A careful matching of the filter parameters and the level of clutter was therefore conducted to ensure that tracking was maintained throughout the 50 executions with random data used for each evaluation.

The detailed checking of the parallel implementation results was identical to the method described in Section 4.4.2, thus again demonstrating correct operation of the parallel filter against the sequential MATLAB simulation.

5.4.3 IMMPDA Effectiveness

Initially the IMMPDA filter was run against a trajectory without any clutter to determine its capability to detect the manoeuvres and its overall tracking qualities. Next the three levels of densities were used to test the filter tracking capability. The results are presented in 4 sets of graphs. Each set contains a specific parameter which is analysed against the 4 different clutter density environments used, i.e. no clutter, light clutter, medium clutter, heavy clutter (Lambda0, Lambda1 and Lambda2). In the first set of graphs the mode probability for each of the models, the SLCV model and the CCTR, is shown. It is followed by the square error position in x and y graphs, and finally the square error velocity in x and y graphs.

Mode Probabilities for Models SLCV and CCTR



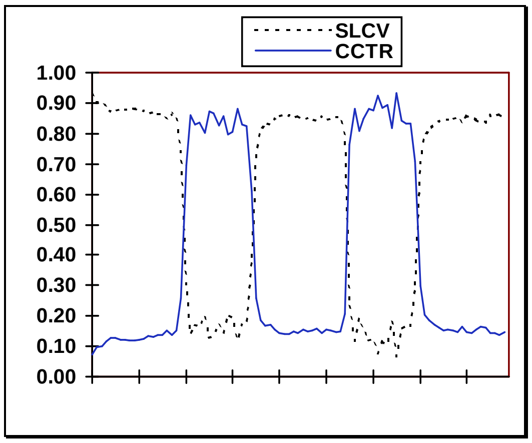
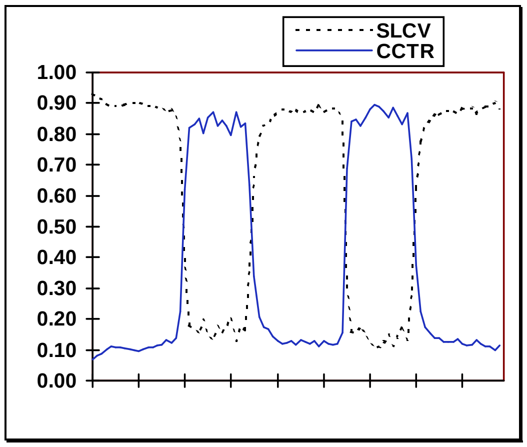


Figure 25 - No Clutter

Figure 26 - Light Clutter



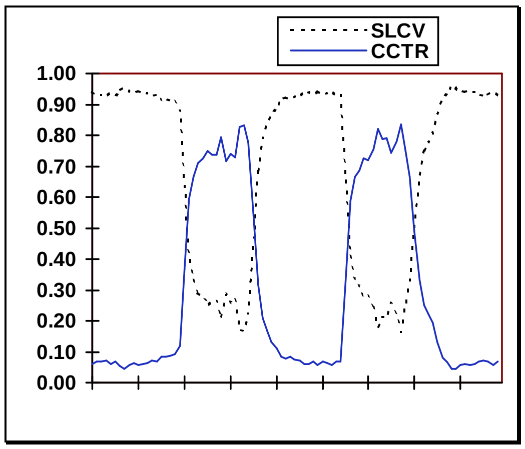
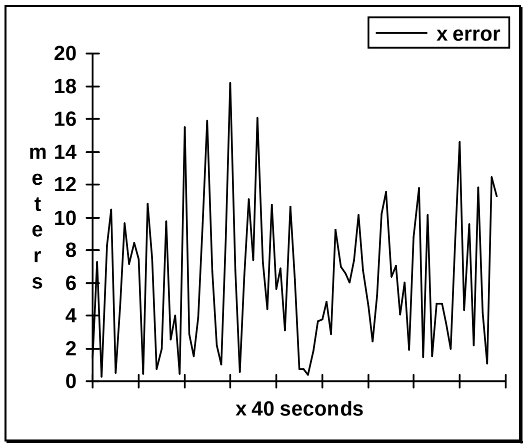


Figure 27 - Medium Clutter

Figure 28 - Heavy Clutter

Square Root Error in x Position



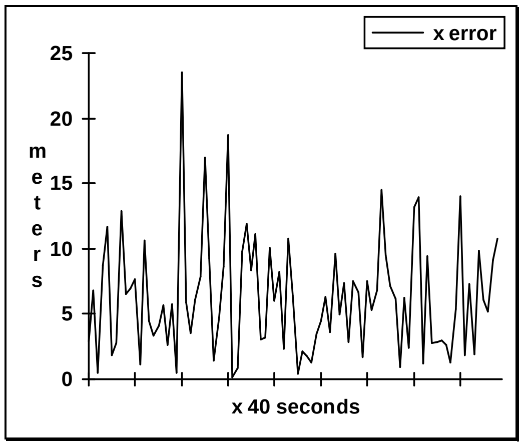
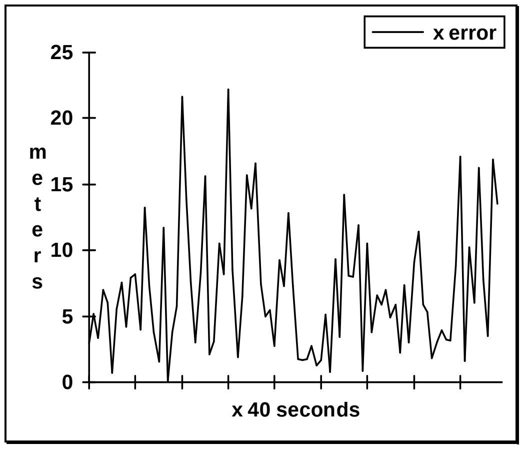


Figure 29 - No Clutter

Figure 30 - Light Clutter



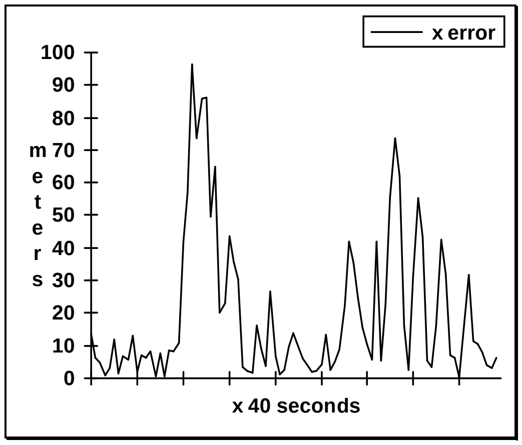


Figure 31 - Medium Clutter

Figure 32 - Heavy Clutter

Square Root Error in y Position

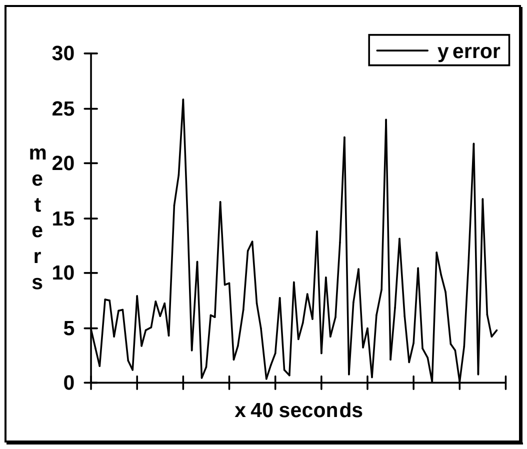


Figure 33 - No Clutter

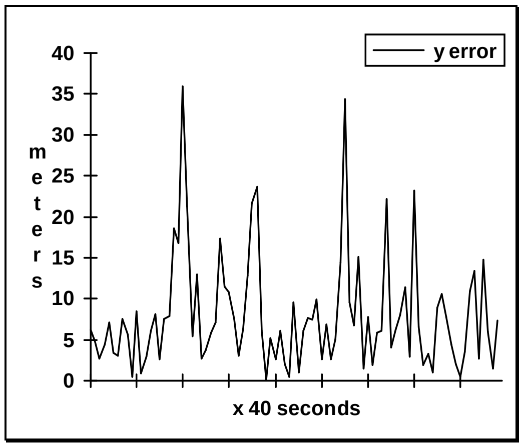


Figure 34 - Light Clutter

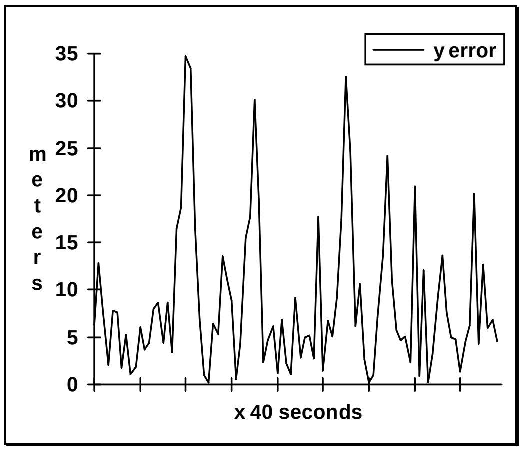


Figure 35 - Medium Clutter

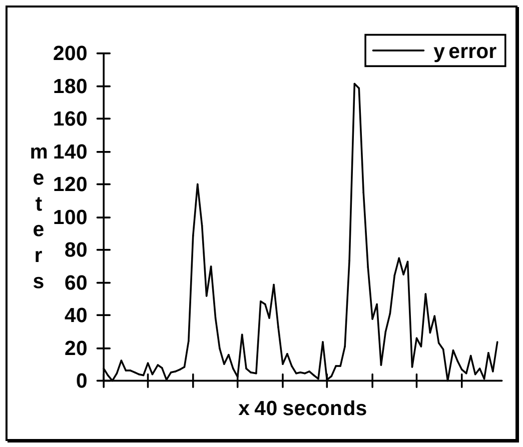
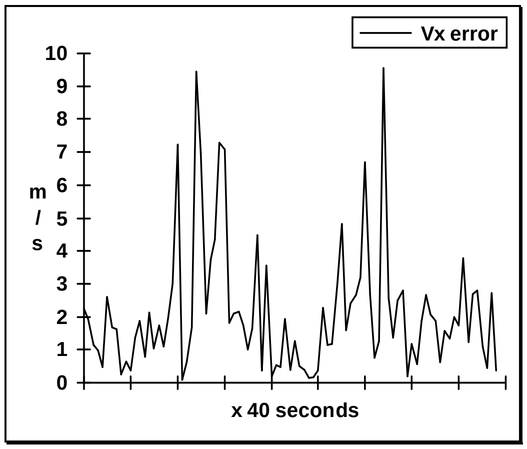


Figure 36 - Heavy Clutter

Square Root Error in x Velocity



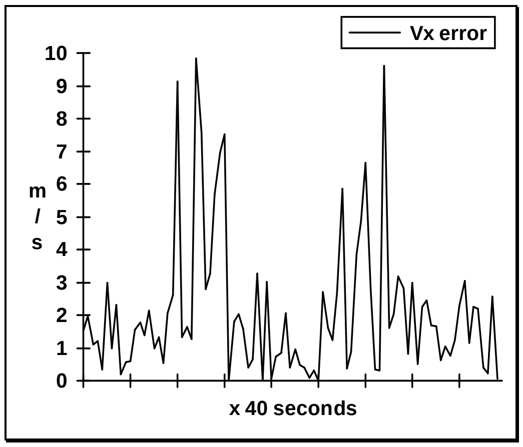
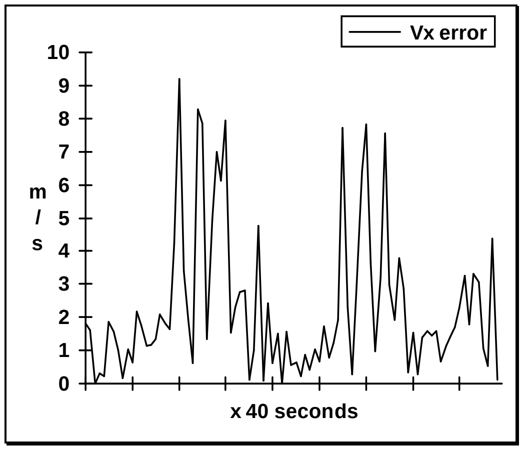


Figure 37 - No Clutter

Figure 38 - Light Clutter



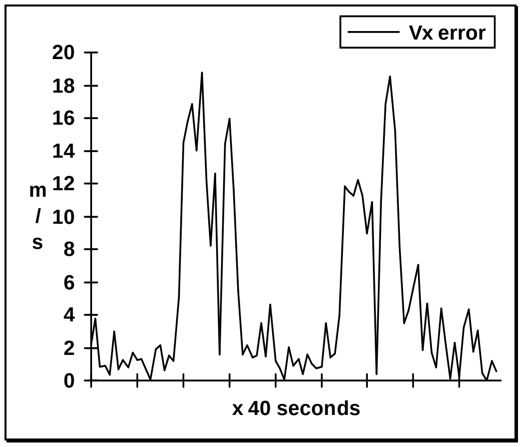


Figure 39 - Medium Clutter

Figure 40 - Heavy Clutter

Square Root Error in y Velocity

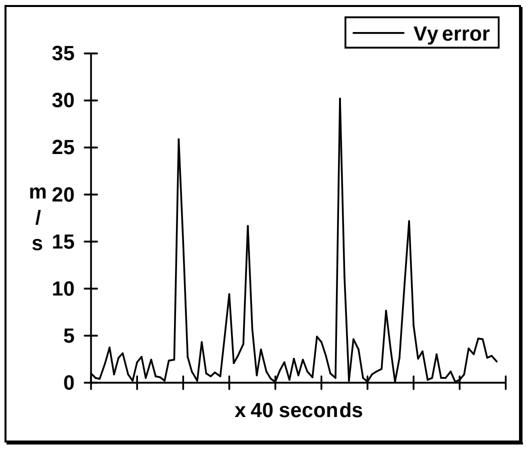


Figure 41 - No Clutter

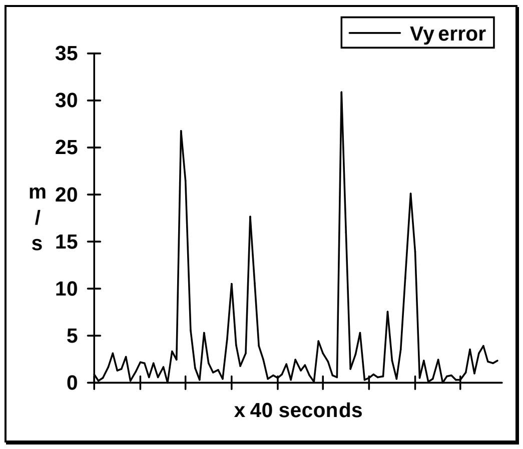


Figure 42 - Light Clutter

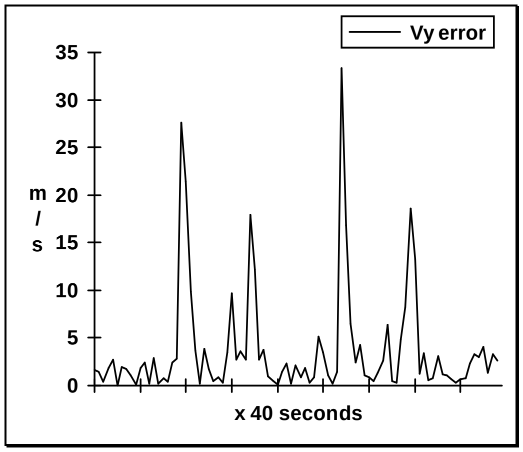


Figure 43 - Medium Clutter

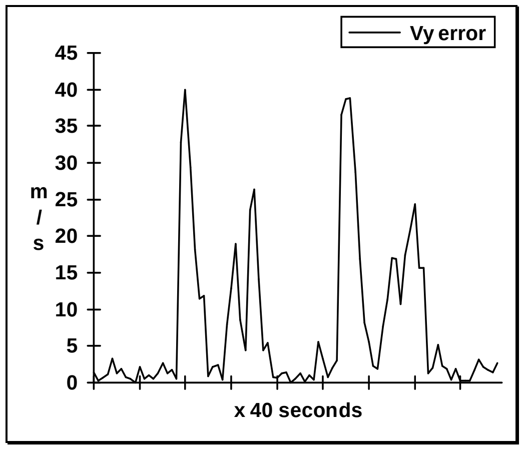


Figure 44 - Heavy Clutter

5.4.3 IMMPDA Transputer Implementations

This is the second part of the analyses and comprises the two parallel implementations previously shown. The implementations were made using one, two and three transputers using the task distribution shown previously. The message-passing system utilised to exchange data amongst processors was once again the EXPRESS package, although this time the more lower-level functions to exchange data provided by the package were used in order to obtain better performance.

As previously described the tests were realised running each experiment 50 times. Table 6 shows the average time in milliseconds to execute the IMMPDA filter for the available implementations

No. of Transputers	***No Clutter***	***Lambda0***	***Lambda1***	***Lambda2***	***Lambda3***	***Lambda4***	***Lambda5***
1	2374	2388	2395	2460	2386	2442	3078
2	1831	1841	1847	1893	1841	1896	1926
3	1571	1582	1588	1635	1582	1638	1668

Table 6 - IMMPDA Execution time (milliseconds)

Next by using the same criteria as in the previous chapter to calculate the speed and efficiency the following speed-up and efficiency results are obtained from the Table 6. First the speed-up.

No. of Transputers	***No Clutter***	***Lambda0***	***Lambda1***	***Lambda2***	***Lambda3***	***Lambda4***	***Lambda5***
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	1.297	1.297	1.297	1.299	1.296	1.287	1.598
3	1.511	1.509	1,508	1.505	1.508	1.491	1.845

Table 7 - IMMPDA Speed Up

and second the efficiency.

No. of Transputers	***No Clutter***	***Lambda0***	***Lambda1***	***Lambda2***	***Lambda3***	***Lambda4***	***Lambda5***
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	0.771	0.771	0.771	0.770	0.772	0.777	0.626
3	0.661	0.662	0.663	0.664	0.663	0.671	0.542

Table 8 - IMMPDA Efficiency

As it can be seen from the above tables (Table 6, Table 7 and Table 8) there is little difference in the results for almost all clutter densities, with the exception of the test based on the data set in Lambda5. In this case processing time is above average, and consequently its speed-up is higher and efficiency lower. Next, to illustrate the speed-up and efficiency for the parallel implementation of the IMMPDA Figure 45 and Figure 46 show the above tables in a graph format. Because most of the values are so close together it is almost impossible to distinguish one from the other, with the exception of the one for Lambda5.

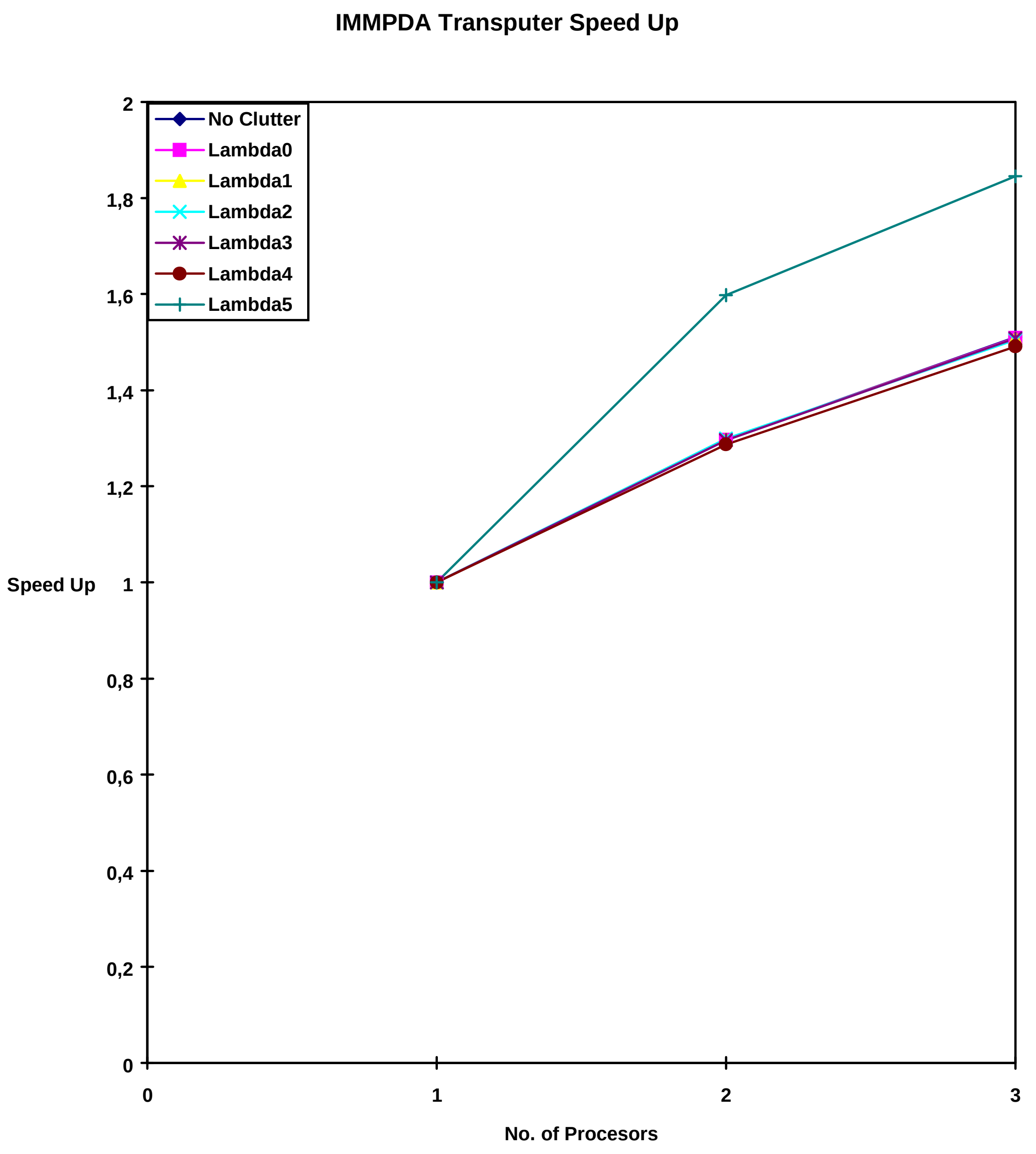


Figure 45 - IMMPDA Speed Up for Transputers

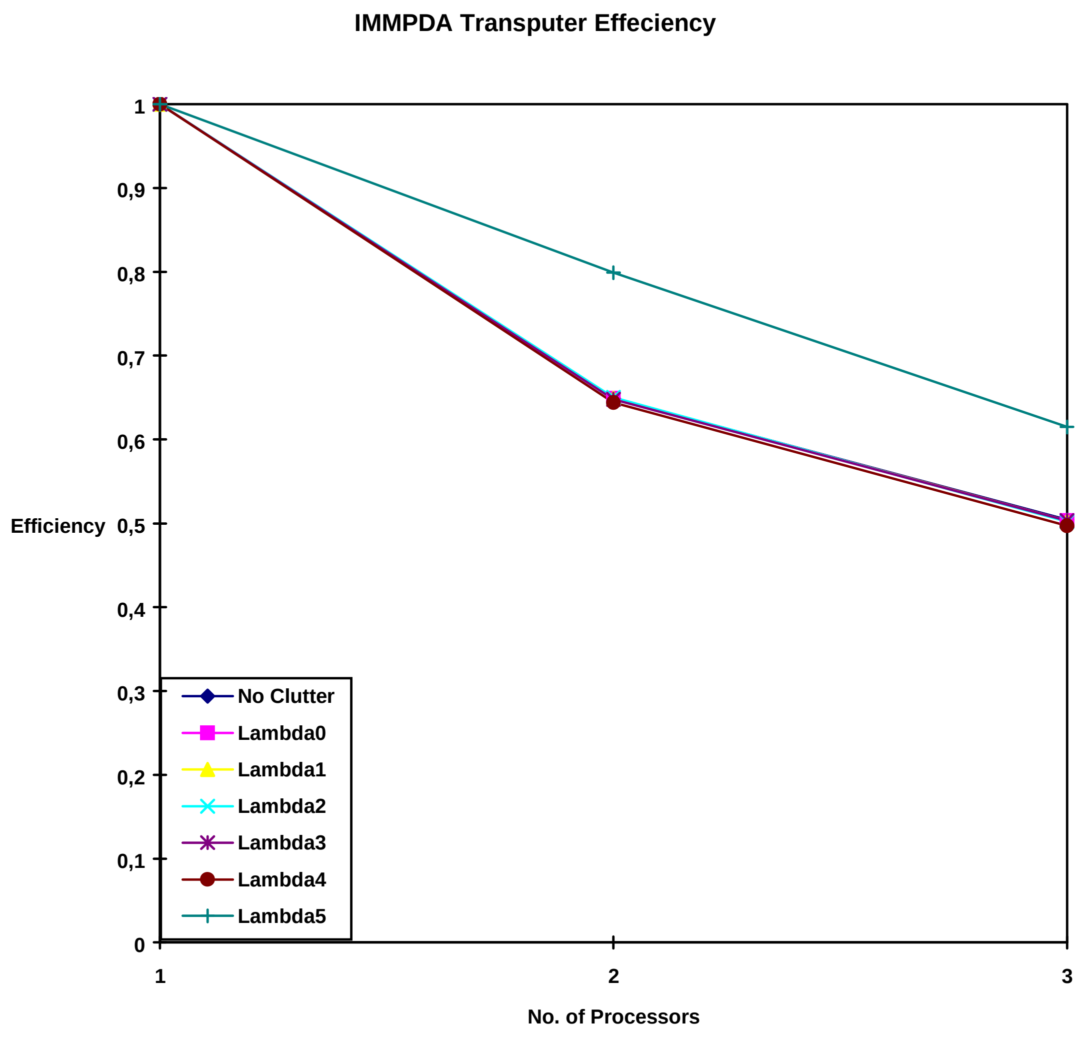


Figure 46 - IMMPDA Efficiency for Transputers

5.5 Conclusions

The conclusions presented here will follow an identical structure as used in the previous section where the results were shown. Therefore, initially, the IMMPDA filter effectiveness is discussed followed by a discussion about the parallel implementation of the IMMPDA.

In the previous section the filter effectiveness results were shown for a particular filter parameters set-up selection. As previously described during the selection of these parameters the IMMPDA filter demonstrated a strong sensitivity to small variations in parameters such as, the process noises for both models, SLCV and CCTR, the values in the transition matrix and finally the gate probability. This sensitivity could be observed by fixing all the other parameters and changing only one of the above mentioned variables. In most of the cases this change could cause the filter to diverge or in other words to loose tracking of the target. In fact, the set up values presented are the best ones found after many interactions. This selection took many interactions where these parameters were carefully changed until the best result was achieved. Obviously this sensitivity is most pronounced when clutter is present. It seems clear that the main problem is caused by the transition from the CCTR to SLCV models. Therefore, a more reliable filter for any possible turn situation should be considered, and also a more careful analysis should be made in the corresponding filter for the CCTR model. One possible solution in order to reduce the number of approximations caused by the use of traditional extended Kalman filter is by using a similar technique as described in [Lerro93b] to improve the filter corresponding to the CCTR mode. In the reference a different approximation is used in the non-linear measurement model that performs better, within certain conditions, than the traditional extended Kalman filter. Other solution for civil aviation would be to use different turn models with fixed turn rates, each one corresponding to typical civil aircraft turns as in [Vacher92].

In the second part of the previous section the parallel implementations results were presented in terms of efficiency and speed-up. Although the efficiency and speed-up achieved in practice for this very cost-effective tracking filter in this particular set-up is not very large, it is necessary to remind the reader that transputers offer a very attractive cost-performance relation. For a two transputer filter implementation an efficiency of 65% was achieved, while in the three transputer implementation an efficiency of more than 50% was obtained. This means that the utilisation of either two or three transputers to implement the IMMPDA filter is an economically viable option taken into consideration the low prices of this type of hardware and the comparatively good price-performance relation. Also assessing the results from a different perspective, transputers have been shown to have a very good overall performance in the processing of IMMPDA filters. For instance the three processor implementation took about 1.6 seconds to process information which in real life would take about 6 minutes to acquire and process, taking into account that the sensors sampling rate is 4 seconds and 91 samples were made. Using a simple extrapolation this is equivalent to estimating that this particular implementation could handle up to 240 targets in real time (obviously no track initialisation overhead is included). However these figures are reasonably close to the ones that could be expected from an accurate analysis.

In summary, despite a large number of variations of IMM filters found in the literature, the filter presented here is the first attempt to couple the IMM as described in [Barret90] and [Lerro93a] with a well-known suboptimal Bayesian technique for data association known as the Probabilistic Data Association technique. At the same time this is the first attempt to have the resulting filter implemented over a parallel architecture based on transputers. As a result the filter effectiveness of the obtained IMMPDA tracking filter is good, given certain specific conditions, although for a more general situation an improved turn model should be incorporated into the final IMMPDA. On the parallel effectiveness side the IMMPDA filter has demonstrated a very good cost-performance characteristics.

Chapter 6 : Conclusions

6.1 Summary

As set out in the introduction this thesis is not directly related to the evolution of high-performance parallel technology itself, but rather to the application of parallel technology to a specific and practical application, **Tracking Systems**. Therefore the main objective of this research has been to select suitable state-of-the-art tracking algorithms, for example the IMM, to create a parallel algorithm to be implemented over distributed memory MIMD architectures. The investigation shown here has proved that tracking filters, essential elements in any tracking system and in particular the IMM algorithm, can and have been successfully implemented on parallel architectures. It also demonstrates that it is worthwhile to undertake further development to implement more sophisticated and functional tracking filters and ultimately entire tracking systems utilising parallel technologies.

6.2 Achievements

The main achievements of this thesis can be found in the two experiments described in Chapter 4 and Chapter 5. These experiments implemented, in two different approaches, the state-of-the-art tracking filter, the Interacting Multiple Model. The first experiment considered a filter comprising similar systems models and an SPMD strategy was used to obtain a parallel algorithm. The second experiment considered a two-model IMM filter coupled with the Probabilistic Data Association technique. The main achievements of these two implementations are set out below separately.

In Chapter 4 the IMM was implemented using the same models as in [Averbuch91a], [Averbuch91b] and [Atherton940]. The implementation shown in this thesis differs from those above because of the use of a more straightforward parallel strategy to implement the algorithm over a MIMD architecture. This strategy is based on a SPMD approach while the ones mentioned use the manager-workers approach. This strategy is not only more straightforward but also provides good performance and speed-up when compared with the more complex alternative approaches. An exact comparison is difficult because each implementation has approached the problem from a different perspective. For instance, the nine models case implemented over three processors, which provides one of the best efficiencies cannot be compared with [Averbuch91a] and [Averbuch91b] because they do not vary the number of models, despite using three processors in a shared memory MIMD architecture. They have obtained their results by varying the number of processors but using a fixed number of models (12 models). On the other hand in [Atherton94] two different numbers of models are used, nine and thirteen models, however in this experiment the number of processors is not varied, although an almost-identical MIMD architecture (transputers) is used. Also in [Atherton94] an interesting approach (time-slipping) was introduced to increase the efficiency and speed-up. Nevertheless, a simplified comparison with these results is shown below.

	parIMM	[Averbuch91a]	[Atherton94]	[Atherton94] (time-slipping)
No. of Processors	**3**	**3**	**4**	**4**
No. of Models	**9**	**12**	**9**	**9**
Speed-up	**71%**	**67%**	**59%**	**78%**

Table 9 - IMM Comparison

The experiments detailed in Chapter 4 used 3 different techniques to exchange data amongst processors. Amongst those the blocking mechanism has been shown to be the most cost-effective for the SPMD strategy. The SPMD strategy requires processors from time to time to exchange information in order to complete their tasks. This experiment demonstrates a clear and concise minimum data set necessary to accomplish the SPMD strategy by exchanging this set amongst processors. Finally as described above, none of the other implementations have explored implementing the IMM with a large number of models to study the parallel issues related to this. The experiment in Chapter 4 have shown that by having a large number of models efficiency and speed-up are greatly influenced by the Interacting and Mixing module. As a consequence a numerical simplification was determined that can provide a reduction of up to 50% in the number of numerical operations. This simplification is a particular format of the transition matrix [Mouse75], and it has been incorporated into the most time consuming component of the IMM mechanisation, namely the Interaction and Mixing module.

In Chapter 5 IMM was implemented coupled with the PDA technique by making use of two well-accepted system models [Barret90] and then the resulting filter (parIMMPDA) was implemented on a distributed memory MIMD architecture. First, the parIMMPDA was analysed in terms of its tracking capability when clutter is present, and finally the efficiency and speed-up were analysed. The filter implemented cannot be easily compared with other filters implemented in the literature. For instance in [Barret90] and [Vacher92] the IMM filter is implemented using the same system models as the parIMMPDA, however it is not coupled with a PDA technique. Also in [Li93a] IMM is built utilising similar systems models but again no provision is made to cope with clutter which is handled by the PDA component. Once again in [Lerro93a] IMM is built using very similar system models as used in parIMMPDA, but in this case the final filters components of the IMM use more information provided by the sensors (debiased information) to improve the tracking performance and in addition the resulting filter is not coupled with the PDA technique. In [Houles89] the IMM is coupled with the PDA, although in this experiment the resulting filter utilises different system models. Therefore, a comparison between parIMMPDA and the above references is difficult.

The parIMMPDA shown here appears to be the first attempt to couple the IMM utilising the system models described in [Barret90] with the PDA technique. Because of this particular characteristic the first analysis is directed to the filter tracking performance when clutter is present. The results have shown that the final filter effectiveness has proved to be sensitive when high density clutter and manoeuvre occur at the same time. This sensitivity needed a fine tuning procedure to adjust the resulting parIMMPDA filter in order to maintain target tracking under heavy clutter environments. The parIMMPDA is therefore not suitable for applications where heavy clutter is probable. Nevertheless, the filter has good performance when light to medium clutter is present and, as expected, it has an identical behaviour to the filter described in [Barret90] when there is clutter. This filter could be used in civil aviation where high-density clutter is extremely unlikely. In this case the parIMMPDA filter would provide a good compromise when clutter could lead to a simple IMM failing to maintain target tracking. Besides, the computational overhead imposed by the coupling of the IMM and PDA techniques is almost negligible which makes this implementation even more attractive.

The final aspect analysed from the parIMMPDA filter was the implementation on a distributed memory MIMD architecture. As before comparison now is further complicated because, as far as it is known, there are no other parallel implementations of IMM coupled with PDA. Therefore, the parIMMPDA presented in Chapter 5 is also the first attempt to parallel implement the IMM coupled with PDA. The implementation is made on a distributed memory MIMD architecture based on transputers. Because of the reduced number of filters comprised in the parIMMPDA, an heuristic strategy is used to exploit the parallelism of the filter, in contrast with the previous experiment where a SPMD strategy was tried. The parallel results presented have shown that either the two or three transputer architecture represents a good balance between computational efficiency and implementation costs. A parallel efficiency of 65% and 50% is achieved for two and three transputers configurations respectively. Also, an almost independent parallel efficiency was achieved despite the increase in the clutter density. This reinforces the concept of parIMMPDA for civil aviation applications using a parallel implementation. Obviously this filter should be part of a more complete tracking system which would comprise features for track initialisation as well as the capability to handle multiple targets as suggested next.

6.3 Future Work

This section takes a wider perspective and consider the next steps to be taken if further research in this area is to be continued. There is a large number of possibilities to research further in this field. However, taking into account the main objective of this thesis, i.e. Tracking System, a natural evolution of the current work would be to extend the experiments to multiple targets and afterwards to multiple targets and multiple sensors.

As a continuation to the work carried out here the first step would be to improve the tracking capabilities of the IMMPDA filter to cope with heavy clutter environments, as it has been shown that the mentioned filter is not completely effective in such circumstances. A direction for such an improvement would be to provide a better mechanism to increase the filter convergence when a turn is detected. This could be either achieved by using more sophisticated models to improve the overall tracking capability of the IMM filter or by some empirical technique that will depend on the application. If more sophisticated models are to be used, then the final resulting filter will benefit from implementation on a parallel system. Another possible solution could be to have 4 different models as suggested in [Lerro93a]. Again this also will work in favour of increasing the parallel efficiency factor. In all cases this filter should be provided with a mechanism for track initiation. As already mentioned this can be a time consuming mechanism, and a parallel implementation would be desirable for such a mechanism. This initiation mechanism could be done as suggested in [BarShalom92]

All studies carried out in this thesis considered a single target and a single sensor and therefore further work could address the multiple target tracking problem [Atherton90]. As a first step to a multiple target tracking system, the parIMMPDA filter could be used in an environment where individual tracks are kept disjoint (non-crossing). A simple configuration for such a multiple tracking system could be based on the Caltech multiple tracker [Gottschalk87a], [Gottschalk87b], [Gottschalk87c], [Baillie87], [Cao88], [Gottschalk90], which is widely available. This could be done by substituting the Caltech initiation component and the steady-state Kalman filter for our previously suggested initiation mechanism and the IMMPDA filter. One of the early studies in this research project resulted in the Caltech algorithm being migrated and modified for the transputer architecture used in Chapter 4 and Chapter 5 using the EXPRESS message passing system.

In order to cope with the problem of crossing tracks the Probabilistic Data Association mechanism would have to be replaced by the Joint Probabilistic Data Association mechanism (JPDA) [Fortmann83], [BarShalom88]. This filter implementation is an attractive tracking choice, although the complexity is significantly higher when compared with alternative related works already presented in the literature for multiple target tracking such as [Gul89], [Atherton90], [Kurien86], [Pattipati90]. There are relatively few related works on this subject (IMM+JPDA) and research to determine the capability of this filter along with the parallel implementation issues would be an interesting area for further research. Again, as in the case of the IMMPDA, the IMMJPDA would have to go through the same development cycle. Initially a simple filter with simple models would have to be implemented to determine the filter tracking capability in different clutter environments. At the same time a parallel implementation should be developed to determine any bottlenecks in the system. Improvements would then be incorporated to improve the tracking capabilities. A tracking initiation mechanism is also necessary, to complete the full tracking system utilising the IMMJPDA (a more sophisticated mechanisms because of the multiple targets).

As a final development the inclusion of multiple sensors capability should be included. This feature could be achieved by using a simple approach as used in [Houles89]. The resulting multiple sensor IMMJPDA filter (multiple targets and multiple sensors) would then be implemented on a parallel system. Because of the complexity of the filter this parallel implementation would be a very interesting research topic, where many different aspects of the implementation could be analysed in order to provide a highly cost-effective solution for tracking systems.

6.4 Conclusions

The issues of implementing reliable and improved tracking systems are not straightforward and require careful implementation if the resulting filter is to be used for real applications. The work presented here has demonstrated that, despite this inherent complexity and intricacy, reliable approaches can be utilized to implement tracking filters on parallel systems. The parIMM has demonstrated that straightforward SPMD technique can provide good parallel efficiency within certain limits. The parIMMPDA has demonstrated two different aspect of the implementation. Firstly parIMMPDA does not introduce any computational overhead at the expense of improving the tracking filter capability in clutter environments. Secondly a heuristic approach to parallel implementation can also lead to good parallel efficiency. In summary this thesis has shown that the two different strategies to implement the state-of-the-art IMM and IMMPDA tracking filters have both provided good parallel results, and both implementations have been shown to be cost-effective when implemented with compatible parallel architecture granularities.

Finally, as suggested above, the work developed in this thesis can be used in further developments towards a more complex and complete tracking system. Such a tracking system would comprise features to cope with multiple targets and multiple sensors. To conclude, the thesis provides a demonstration that parallel technology is certainly a viable solution for future tracking system implementations.

Appendix A

Kalman Filters

A.1 Introduction

The sole purpose of this appendix is to present the Kalman equations, without taking into consideration many of the theoretical complexities and passages involved in the process to reach its final algorithmic format. This particular approach is taken because Kalman filter theory is vast and comprises many aspects which would make a full derivation of the filter a tedious and lengthy exercise, beyond the scope of this thesis. This assertion can be simply illustrated by looking at the number of related research and publications in this field since its introduction. Therefore, for simplicity and conciseness only the final filter equations are shown together with initial principles. Nonetheless, in order to illustrate the wide range of research around this well-accepted filtering technique a review of related Kalman work is given in order to demonstrate this large spectrum of research. After this short revision the two most-used Kalman mechanizations are presented. A first version deals with the linear case, the original formulation of the Kalman filter. The second version is an approximation to deal with particular cases of non-linear systems which is commonly known as the Extended Kalman filter. It is also important to mention that these two Kalman filter summaries are heavily based on the treatment in [BarShalom88]. Because the equations are presented in their final format anyone wishing to understand the basic principles should refer to [BarShalom88], [Bozic79], or [Gelb74] for a simple and practical approach. If a more rigorous approach is sought this can be found in other references [Blakrishnan87], [Maybeck79] and [Chui87]

The Kalman filter [Kalman60], [Kalman61] is a recursive filter that does not need to store previous measurements to perform an estimation. Due to this characteristic, it has been widely used in a range of applications [Hutchison84], notably in the control [Maybeck79] and signal processing [Schwartz75] fields, particularly for aerospace applications [Pearson74], [Singer71]. In fact the Kalman filter is part of linear estimation [Kailath80], and has its roots on the initial work of Gauss [Gauss63]. A very good historical perspective of this intricate theoretical evolution can be found in [Sorenson70], [Kailath74]. Also a good source of information to establish the interconnection and theoretical evolution of linear estimation can be found in a sequence of papers [Kailath68a], [Kailath68b], [Kailath71a], [Kailath71b], [Kailath73a], [Kailath73b], [Kailath73c]. Since its introduction in 1960 a number of improvements and modifications have been devised to permit the utilization of this filter in a wide range of areas. Despite the effectiveness of the filter in providing good estimates for linear systems theoretically, unfortunately in the 60s and 70s computer implementations were not very reliable due to limitations of the available processors and related technological issues. As a consequence a number of simplifications were initially introduced to make use of Kalman principles viable. This was the case described in, for example, [Bierman73], [Friedland73], [Castella74]. These simplifications still hold true for current implementations where simple processors are needed for economical reasons as in the case described in [Baheti86].

Another related problem with the early implementations of complete Kalman filters was the lack of processing units with high-precision floating-point operations. As a consequence many algorithms were also developed to overcome this precision problem which caused the filter to diverge. Kalman filter divergence [Fitzgerald71] is mainly caused by the lost of covariance matrix properties (positive semi-definitiveness) [Bierman77]. In order to avoid such instability problems a number of algorithms have been generated, most of them based on matrix factorization properties [Bellantoni67], [Andrew68], [Carlson73]. All this effort was necessary to permit the utilization of the filter in an increasing number of more complex applications. However some specific real-time applications require complicated and complex Kalman filters. For such cases the demand for computational power of the Kalman filters can be high as shown in [Mendel71].

This has stimulated much research in recent years to applying parallel processing to solve these computationally-demanding Kalman filter applications. This research has been carried out in order to understand the parallel structures present in the filter as demonstrated in [Jover86], [Hashemi87], [Lee88] and [Rhodes90]. At the same time many different parallel architectures and granularities have been used to implement the Kalman filter ranging from large architectures to VLSI implementations. An early example of a large architecture can be found in [Ohallaron88], [Baheti88] where the Warp [Annaratone87] computer was used to implement the filter. In contrast, for very demanding applications a VLSI implementation of the Kalman filter exists as shown in [Sung87]. However many other approaches (heuristic and systolic) exist such as [Lawrie90] and [Lawrie91] where three different types of platforms were used to implement the filter. Also the systolic approach has been widely accepted as a good methodology to implement the filter as demonstrated in [Gaston89] and [Gaston90]. Despite the high precision floating-point operations present in most of the platforms currently available factorized Kalman algorithms were also implemented in parallel as described in [Carlson90]. There are other Kalman implementation descriptions utilizing different processors and languages, as in the case of [Tan88] where the filter is implemented in a digital signal processor, or as in [Kee91] where a transputer is used instead, and in [Smith91] where Matlab has been used to implement the filter. As in can be seen from the above discussion there are a large number of publications and research in the field of Kalman filter theory and applications. This is the main reason why the Kalman equations presented here in very a highly condensed form.

A.2 Summary of Kalman Filter Equations

The Kalman Theory is applied to quantities that vary dynamically with a time-linear deterministic/stochastic model of a system [Lawrie90], [Gelb74], [Chui87], [BarShalom88]. Consider a discrete-time, linear, dynamic system described by the state space equation [DeRusso66] given by

<u>System model</u>

$$\mathbf{x}(k+1) = \mathbf{F}(k)\mathbf{x}(k) + \mathbf{G}(k)\mathbf{u}(k) + \Gamma(k)\mathbf{v}(k)$$

where **x**(k) is the system state-vector at time *k*, **F**(*k*) is the state transition matrix, **G**(*k*) is the input matrix, **u**(k) is the system forcing function or deterministic control input known, **v**(k) is the system noise sequence with zero-mean, white, Gaussian distribution with covariance given by **Q**(*k*), or

$$E[\mathbf{v}(k)] = 0$$

$$E\left[\mathbf{v}(i)\,\mathbf{v}(j)'\right] = \mathbf{Q}(k)\,\delta_{ij}$$

Now, consider also that measurements are related to the above equation by

<u>Measurement Model</u>

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

where **H**(*k*) is the measurement matrix, and **w**(*k*) is a sequence of zero-mean, white Gaussian measurement noise terms with covariance **R**(*k*) or

$$E\left[\mathbf{w}(k)\right] = 0$$
$$E\left[\mathbf{w}(i)\,\mathbf{w}(j)'\right] = \mathbf{R}(k)\,\delta_{ij}$$

Also the initial state **x**(0) is assumed to be Gaussian with estimate $\overset{\wedge}{\mathbf{x}}(0|0)$ and its covariance $\overset{\wedge}{\mathbf{P}}(0|0)$ is assumed to be available.

One of the main uses of the Kalman filter is to obtain the optimal estimation [Deutsch65], [Gelb74] of the system state-vector conditioned by all past measurements,

$$\overset{\wedge}{\mathbf{x}}(k\,|\,k) = E\left[\mathbf{x}(k)|\mathbf{Z}^{\mathbf{k}}\right]$$

along with its associated state estimate covariance given by

$$\overset{\wedge}{\mathbf{P}}(k\,|\,k) = E\left[\left(\mathbf{x}(k) - \overset{\wedge}{\mathbf{x}}(k)\right)\left(\mathbf{x}(k) - \overset{\wedge}{\mathbf{x}}(k)\right)'\,|\,\mathbf{Z}^{\mathbf{k}}\right]$$

where in the above equations $\mathbf{Z}^k$ denotes the measurements up to and including sample k, or

$$\mathbf{Z}^{\mathbf{k}} \equiv \left\{ \mathbf{Z}(k) \right\}_{j=1}^{k}$$

and the set of measurements obtained at time k is denoted by

$$\mathbf{Z}(k) = \left\{ \mathbf{z}_i(k) \right\}_{i=1}^{m_k}$$

The above is simplified to **z** *(k)* when only one measurement is available.

This optimisation is performed in the least-squares sense and results in the Kalman filter. A complete description of all the steps in obtaining the following updating structure or, simply, the Kalman filter formulas can be found, for example, in [Chui87], [Miller87], [Blakrishnan87].

<u>State Prediction</u>

$$\hat{\mathbf{x}}(k+1|k) = \mathbf{F}(k)\hat{\mathbf{x}}(k|k) + \mathbf{G}(k)\mathbf{u}(k)$$

<u>State Covariance Prediction</u>

$$\hat{\mathbf{P}}(k+1|k) = \mathbf{F}(k)\mathbf{P}(k|k)\mathbf{F}(k)' + \mathbf{Q}(k)$$

<u>Measurement Prediction</u>

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

<u>Residual or Innovation</u>

$$\nu(k+1) \equiv \tilde{\mathbf{z}}(k+1|k) = \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1|k)$$

<u>Innovation Covariance</u>

$$\mathbf{S}(k+1) = \mathbf{H}(k+1)\hat{\mathbf{P}}(k+1|k)\mathbf{H}(k+1) + \mathbf{R}(k+1)$$

Kalman Gain

$$\mathbf{W}(k+1) = \frac{\hat{\mathbf{P}}(k+1|k)\,\mathbf{H}(k+1)'}{\mathbf{S}(k+1)}$$

State Update

$$\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathbf{W}(k+1)\,\nu(k+1)$$

State Update Covariance

$$\hat{\mathbf{P}}(k+1|k+1) = \hat{\mathbf{P}}(k+1|k) - \mathbf{W}(k+1)\,\mathbf{S}(k+1)\,\mathbf{W}(k+1)'$$

A.3 Summary of Extended Kalman Filter Equations

Consider, as in the previous section, a system model that for simplicity has no control input and which can be expressed by a nonlinear relation of the form

System Model

$$\mathbf{x}(k+1) = \mathbf{f}[k, \mathbf{x}(k)] + \Gamma(k)\,\mathbf{v}(k)$$

with measurements defined by

Measurement Model

$$\mathbf{z}(k) = \mathbf{h}[k, \mathbf{x}(k)] + \mathbf{w}(k)$$

Again to apply Kalman filter theory, system and measurement noises must be white noise and have a Gaussian distribution with known covariance matrices as in the linear case. Similarly to the linear case, the main purpose is to obtain the optimal estimation of the system state-vector conditioned by all past measurements. This time the functions are non-linear, although there are complicated methods to cope with this a more simple approach is sought [Jazwinski70]. A typical solution is to approximate the non-linear functions by linear ones. Although there a different number of ways to overcome the above nonlienarity [Miller82], a common method is to expand the nonlinear functions $\mathbf{f}(.)$ and $\mathbf{h}(.)$ in a Taylor series [Kreyszig88] around the latest estimate $\hat{\mathbf{x}}(k|k)$. As a consequence, different filter orders can be obtained upon truncation of terms in the Taylor series. Thus, for example, a filter that truncates the series in the first Taylor series term is known as a first order extended Kalman filter (FOEKF). Identically a filter with truncation on the second term is known as a second order extended Kalman filter (SOEKF), and so on. Because the filters used in the thesis are either FOEKF or SOEKF, the second order filter is presented. The first order filter follows as a particular case of the second order. The following elements are necessary to understand the extended Kalman filter mechanization:

1. Cartesian basis vectors for a n-dimensional space

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ . \\ . \\ \mathbf{e}_n \end{bmatrix} = \textit{Identity Matrix}\,(n \; x \; n)$$

where $\mathbf{e}_\mathbf{i}$ has i-th component equal to unity and all other elements are zero

2. Gradient operator

$$\nabla_\mathbf{x} = \begin{bmatrix} \partial/\partial x_1 \\ \partial/\partial x_2 \\ . \\ . \\ \partial/\partial x_n \end{bmatrix}$$

3. Jacobian of a function $f(\mathbf{x})$

$$f_\mathbf{x}(\mathbf{x}) \equiv \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \left[\nabla_\mathbf{x} f'(\mathbf{x})\right]'$$

4. Hessian of a function *f*(**x**)

$$f_{\mathbf{xx}}(\mathbf{x}) \equiv \frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^2} = \nabla_{\mathbf{x}} \nabla_{\mathbf{x}}{}' f(\mathbf{x})$$

Thus the Extended Kalman filter equations can be summarised as :

<u>State Prediction</u>

$$\hat{\mathbf{x}}(k+1|k) = \mathbf{f}\left(k, \hat{\mathbf{x}}(k|k)\right) + \frac{1}{2}\sum_{i=1}^{n_x} \mathbf{e}_i \; trace\left[\mathbf{f}_{\mathbf{xx}}{}^i\left(k, \hat{\mathbf{x}}(k|k)\right)\hat{\mathbf{P}}(k|k)\right]$$

<u>State Covariance Prediction</u>

$$\hat{\mathbf{x}}(k+1|k) = \mathbf{f}_{\mathbf{x}}\left(k, \hat{\mathbf{x}}(k|k\right)\hat{\mathbf{P}}(k|k) +$$

$$+\frac{1}{2}\sum_{i=1}^{n_x}\sum_{j=1}^{n_x} \mathbf{e}_i \mathbf{e}_j{}' \; trace\left[\mathbf{f}_{\mathbf{xx}}{}^i\left(k, \hat{\mathbf{x}}(k|k)\right)\hat{\mathbf{P}}(k|k)\mathbf{f}_{\mathbf{xx}}{}^j\left(k, \hat{\mathbf{x}}(k|k)\right)\hat{\mathbf{P}}(k|k)\right] + \mathbf{Q}(k)$$

<u>Measurement Prediction</u>

$$\hat{\mathbf{z}}(k+1|k) = \mathbf{h}\left[k, \hat{\mathbf{x}}(k+1|k)\right] + \frac{1}{2}\sum_{i=1}^{n_z} \mathbf{e}_i \; trace\left[\mathbf{h}_{\mathbf{xx}}{}^j\left(k, \hat{\mathbf{x}}(k+1|k)\right)\hat{\mathbf{P}}(k+1|k)\right]$$

<u>Residual or Innovation</u>

$$\nu(k+1) \equiv \tilde{\mathbf{z}}(k+1|k) = \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1|k)$$

<u>Innovation Covariance</u>

$$\mathbf{S}(k+1) = \mathbf{h}_{\mathbf{x}}(k+1)\hat{\mathbf{P}}(k+1|k)\mathbf{h}_{\mathbf{x}}(k+1)$$

$$+\frac{1}{2}\sum\sum \mathbf{e}_i \mathbf{e}_j{}' trace\left[\mathbf{h}_{\mathbf{xx}}{}^i\left(k, \hat{\mathbf{x}}(k+1|k)\right)\hat{\mathbf{P}}(k+1|k)\mathbf{h}_{\mathbf{xx}}{}^j\left(k, \hat{\mathbf{x}}(k+1|k)\right)\hat{\mathbf{P}}(k+1|k)\right]$$

$$+ \mathbf{R}(k+1)$$

all the other equations to evaluate the Kalman filter remain the same, i.e.

Kalman Gain

$$\mathbf{W}(k+1) = \frac{\hat{\mathbf{P}}(k+1|k)\,\mathbf{H}(k+1)'}{\mathbf{S}(k+1)}$$

State Update

$$\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathbf{W}(k+1)\,\nu(k+1)$$

State Update Covariance

$$\hat{\mathbf{P}}(k+1|k+1) = \hat{\mathbf{P}}(k+1|k) - \mathbf{W}(k+1)\mathbf{S}(k+1)\,\mathbf{W}(k+1)'$$

Appendix B

Interacting Multiple Model

B.1 Introduction

The Kalman filters previously shown provide an optimum estimate as long as the object being observed behaves according to the prescribed model. However in the real world, quite often system dynamics cannot be translated into just one model, but require several different models corresponding to different system regimes. An ideal system would have a finite set $M = \{1, 2,..., m\}$ of different models to cope with all possible regimes and it would obey one out of the m available models at any time. The difference in these models is related to previous system equations in Appendix A by assuming different parameters for those equations. In the particular case of the measurement model multiple sensors can be achieved by using a mapping function H, for the linear case, or h(.), for the non-linear case, for each of the type of sensor being used. However in the thesis only one type of sensor will be used, therefore the measurement equation remains as in the equations previously shown for the linear and non-linear cases, although multiple sensors are also possible [Blom90b][Li93a].

An aircraft trajectory being detected by a scan radar is a good example of a system that fits the above description. The trajectory comprises a sequence of different flight modes, such as left and right turn, uniform motion (straight line constant velocity, straight line constant acceleration), climb and descent motions. The pilot controls when and how to change from one model to another. Nevertheless the pilot decision can be guided by some basic rules (e.g. the flight plan) and from the Air Traffic Control point of view the switch depends only in the current mode of flight. This type of mode switching is best characterised as a Markov process [Blom83b], [Gillespie92], [Papoulis87]. A Markov process can be simply described as a process that “its future is conditionally independent of its past given its present", in other words to change from a current model to a new model the process only depends on its present model. The set $\boldsymbol{M} = \{1,2,...,m\}$ of models in a Markov process is known as a discrete Markov chain. The switch from any current model to any future model is governed by probabilities. These probabilities are stored in a matrix format known as transition matrix where the elements are given by

$$p\left\{m_j\left(k+1\right)|m_i\left(k\right)\right\} = \pi_{ij}$$

where π_{ij} is the probability of changing from model *i* at instant *k* to model *j* at instant *k*+1. The determination of these probabilities can be obtained theoretically by the studying the mean time of stay on each particular model [Barret90]. Another approach is to assume that the probability of stay on the current model is as any "a priori" probability value, for example *prob* and the probability to change to any other possible model equal [Moose75] or

$$\pi_{ij} = \begin{cases} prob & , i = j \\ \dfrac{\left(1-prob\right)}{\left(m-1\right)} & , i \neq j \end{cases}$$

A system composed of a system model as described in the above equations and a Markov chain is also known as a hybrid system [Blom82a]. The purpose here, once again as in the Kalman case, is to obtain an improved estimate and its respective covariance using all models on this hybrid system. An optimal hybrid system solution for *m* models has a growing number of hypotheses at each new measurement arrival. These hypotheses grow exponentially with time as m^t [Tugnait82] and this is illustrated in Figure 47.

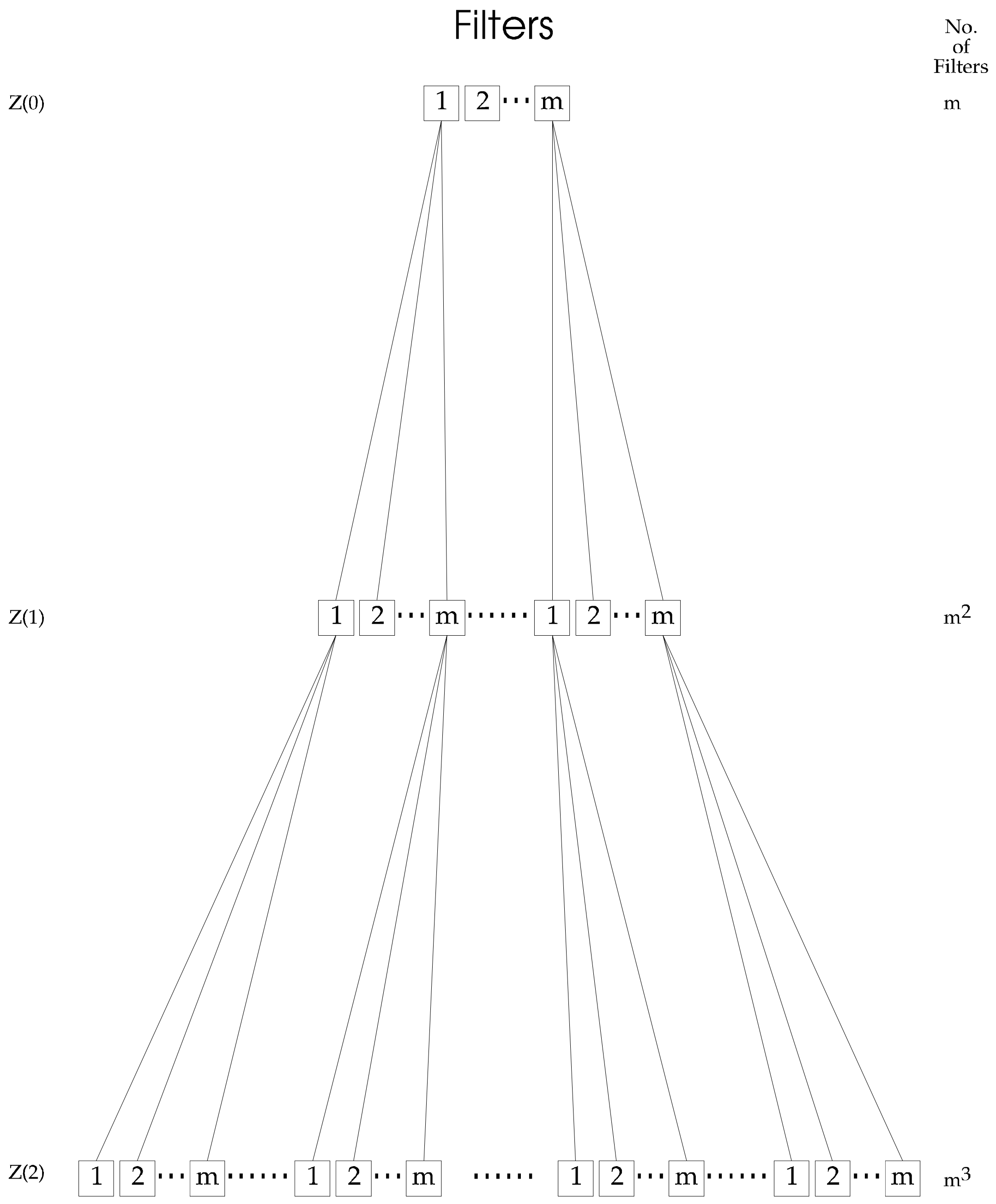

Filters
No. of Filters
Z(0)
1 2 ··· m
m
Z(1)
1 2 ··· m ······ 1 2 ··· m
m2
Z(2)
1 2 ··· m ······ 1 2 ··· m ······ 1 2 ··· m ······ 1 2 ··· m
m3

Figure 47 - Multiple Model Approach

Such a growth is obviously unacceptable for computer implementation and this has generated a number of different techniques in order to make such multiple model approaches viable for computer implementation. These techniques are based on hypothesis reduction by making use of either pruning or merging of hypothesis. The underlying concept in both approaches is to prevent the explosion of the total number of hypothesis and to keep them at a fixed or nearly fixed number as in shown in Figure 48.

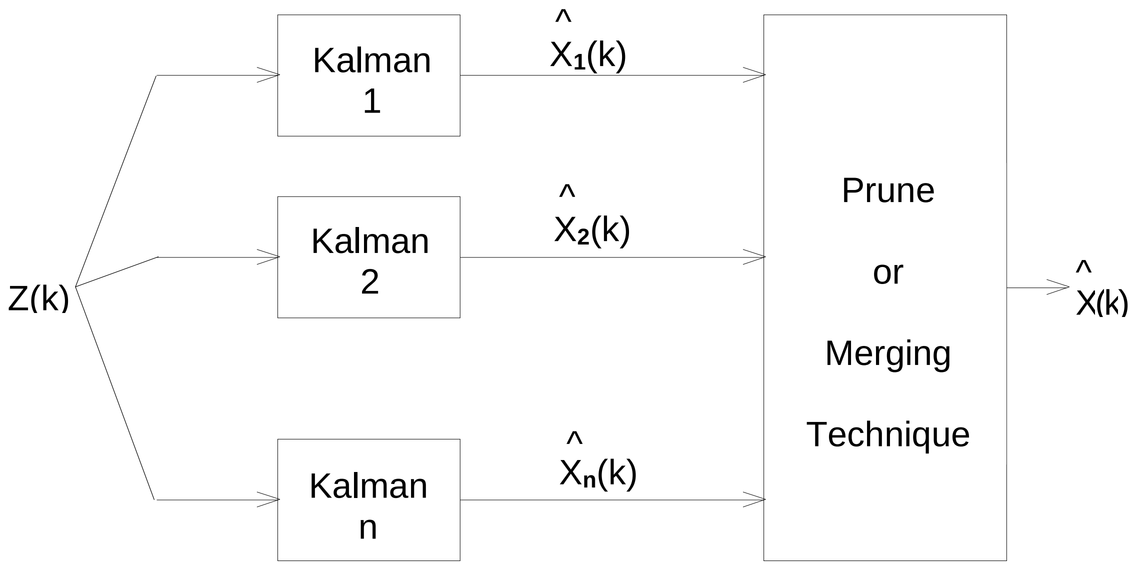


Figure 48 - Multiple Model Reduction

Pruning methods are based on discarding hypotheses that are below a certain probability level. Again the selection of this level has originated a number of different pruning algorithms [Tugnait82]. The merging method is based on binding hypotheses together that are within certain probability levels [Chan82]. Similarly to pruning, it has originated a number of different merging algorithms [Moose75]. Amongst the merging algorithms the most important one, until recently, was the Generalised Pseudo Bayes (GPB) [Chan79]. The GPB algorithm merges those hypotheses that have similar probabilities given some fixed time window (fixed depth hypotheses merging). Consequently GPB with *m* filters and a depth of *k* past measurements will have m^k Kalman filters at any time. A common notation to show the GPB depth is GPB(k) or GPBk. A GPB2 (m^2 Kalman filters) has been shown to be the best choice when compared to the algorithms previously mentioned in terms of performance and final precision. However in many real time applications GBP2 can still compromise the final performance making its use unacceptable. The Interacting Multiple Model (IMM) developed by H.A. Blom [Blom82a] solved the performance problem. The IMM algorithm performs as well as a GPB2 but with the computational cost of a GPB1. A more precise description about IMM performance in practical applications without making use of Monte Carlo simulations can be found in [Li93a].

The IMM has been successfully used in implementing air traffic control system [Li93b], [Vacher92], [Blom84b] and in other problems where multiple and different sensors along with highly-manoeuvrable targets are present [Blom88], [Dufour92], [Blom92]. The IMM is a recursive algorithm which consists of four steps that are cycled at each new measurement arrival. The algorithm is shown in Figure 49 and explained below. For a fully detailed description of the IMM algorithm, see [Blom90a].

B.2 The IMM Algorithm

Step 1 - Interaction and Mixing

This step is the key element that distinguishes IMM from other GPB algorithms. The merging of hypothesis is done in this stage to prevent the explosion. The first step in the Interaction and Mixing is to calculate the predicted mode probability or probability normalisation constant for each model. The predicted mode probability for a particular model is obtained by summing all the multiplication of the previous mode probability for any other models by the corresponding transition matrix probability from going from any other models to that particular one. Based on the predicted mode probability the mixed probability can be easily obtained by dividing the product of the previous model probability and correspondent probability of changing to that specific model by the corresponding probability normalisation constant. Using the mixed probabilities it is possible to mix all the previous filter outputs, namely estimates and its covariances. By mixing them, a previous mixed estimates and covariance is obtained for each model and they can be now used as input to the next filter step (Kalman Filters). This can be summarised as :

$$\mu_{i\,i/j}\,(k-1|k-1) \equiv p\left\{M_i(k-1)|M_j(k),\mathbf{Z^k}\right\} \qquad (\text{using Bayes})$$

$$= \frac{p\left\{M_j(k)|M_i(k-1),Z^{k-1}\right\}p\left\{M_i(k-1)|Z^{k-1}\right\}}{p\left\{M_j(k)/Z^{k-1}\right\}}$$

With the new mixing weights one can now calculate the new estimates for each of the models by mixing all the models estimations from each of the previous Kalman filters outputs weighted by their correspondent mixed probability.

$$\hat{\mathbf{x}}_{Oj}(k-1|k-1) = \sum_{i=1}^{m} \hat{\mathbf{x}}_i(k-1|k-1)\,\mu_{i|j}(k-1|k-1)$$

The covariance corresponding to the above estimation is also given by the same mixing principle, thus

$$\hat{\mathbf{P}}_{Oj} = \sum_{i=1}^{m} \mu_{i|j}(k-1|k-1) \left\{ \begin{array}{l} \hat{\mathbf{P}}_i(k-1|k-1) + \\ \left[\hat{\mathbf{x}}(k-1|k-1) - \hat{\mathbf{x}}_{Oj}(k-1|k-1)\right] . \\ \left[\hat{\mathbf{x}}(k-1|k-1) - \hat{\mathbf{x}}_{Oj}(k-1|k-1)\right]' \end{array} \right\}$$

Step 2 - Kalman Filter

The previous mixed estimates and their covariances are used as input to each of the correspondent Kalman filter. This is done together with the input of the current in the same way as a single Kalman filter. Because there are a multitude of filters these calculations can all be performed in parallel. The Kalman algorithms to be used here for each model j can be either of the two Kalman mechanisations previously shown:
Kalman Filter (KF).
Extended Kalman Filter First Order (FOEKF).
Extended Kalman Filter Second Order (SOEKF).

For the IMM algorithm the necessary Kalman filter outputs are not only each model current estimation $\hat{\mathbf{x}}_j(k)$ and correspondent covariance $\hat{\mathbf{P}}_j(k)$ but also the filter likelihood Λ_j. The likelihood function of each filter is calculated using the innovation $\xi_j(t)$ and its covariances $S_j(t)$ and given by

$$\Lambda_j = p\left\{z(k) | M_j(k), \mathbf{Z}^{\mathbf{k}-1}\right\} = N\left(\overset{\wedge}{\mathbf{z}}(k+1|k); 0, \mathbf{S}_j\right)$$

As a result this likelihood calculation is incorporated in the previous set of Kalman equations and the Kalman filter performing in parallel should return the three variables mentioned above.

Step 3 - Probability Update

Using the likelihood function of each filter the model probabilities update can now be determined by

$$\mu_j(k) = p\left\{M_j(k) | Z^k\right\} = \frac{p\left\{z(k) | M_j(k), Z^k\right\} p\left\{M_j \; | Z^{k-1}\right\}}{p\left\{z(k) | Z^{k-1}\right\}}$$

Step 4 - Combination

The final step calculates, for output purposes, a unique or combined state estimate and its covariance. The output is based on adding all models estimated output weighted with their model probability, or

$$\overset{\wedge}{\mathbf{x}}(k|k) = \sum_{i=1}^{m} \mu_i \, \overset{\wedge}{\mathbf{x}}_i(k|k)$$

and its corresponding covariance is

$$\overset{\wedge}{\mathbf{P}}(k|k) = \sum_{i=1}^{m} \mu_i(k) \left\{ \overset{\wedge}{\mathbf{P}}_i(k|k) + \left[\overset{\wedge}{\mathbf{x}}_i(k|k) - \overset{\wedge}{\mathbf{x}}(k|k)\right]\left[\overset{\wedge}{\mathbf{x}}_i(k|k) - \overset{\wedge}{\mathbf{x}}(k|k)\right]' \right\}$$

The algorithm returns to the first step and proceeds to the next cycle with a new measurement. The IMM algorithm representation is shown in Figure 49.

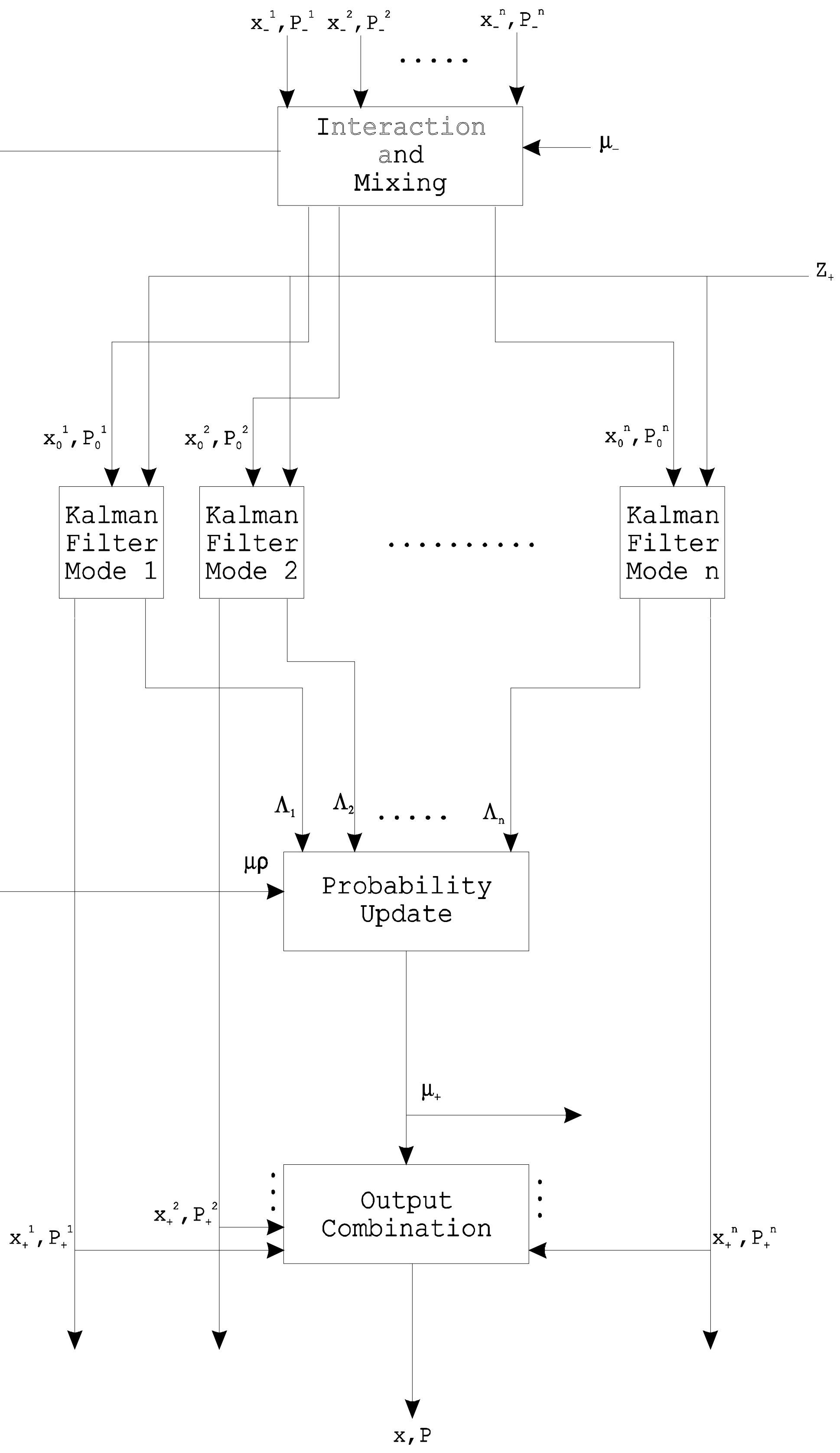


Figure 49 - IMM Diagram

Appendix C

Probabilistic Data Association

C.1 Introduction

As described in the previous two appendixes, the objective here is to depict all of the formulae necessary to understand the parallel implementations carried out in this thesis. Once again the probabilistic data association equations will be presented in a summarized way, with references to the further comprehension of the theoretical aspects involved in using these equations for real life applications. Both methods to be presented are based on the Bayesian approach [BarShalom78]. The method presented refers to a suboptimal method to track one known target in a cluttered environment.

C.2 Probabilistic Data Association

The main advantage of Probabilistic Data Association technique is to reduce the computational overload that would result if a more general approach was to be used, such as the Multiple Hypotheses [Reid79] approach based on an optimal Bayesian estimator where the number of hypothesis grows exponentially as the measurements are received. In that sense the PDA is a suboptimal Bayesian estimator [BarShalom78] which takes into account all measurements that may have originated from the target. The final form of the PDA algorithm (PDA) was first presented in [BarShalom75]. The algorithm is intended to track one object of interest, “the target”, in the presence of spurious objects, “clutter”. The first step of the PDA is to select amongst all measurements those that lie inside a validation region (gate) whose centre was predicted by the filtering algorithm (Kalman). This phase is known as the gating phase [Blackman86] and it is used to reduce the number of possible target association measurements. Those measurements that are in a Mahalanobis distance sense less or equal than a threshold distance (inside the gate) are considered to be the most probable candidates for the track update. Once these gated measurements have been determined, they are all taken into account to update the track. The next phase and the core of the method consists of establishing a probabilistic weight for all those validated measurements. Each measurement will have a different weight corresponding to its likelihood in the updating process according to its probabilistic distance. In summary the essential elements of the PDA are the gating determination and measurements weight selection. A major limitation of the PDA algorithm is the lack of a track initiation mechanism, which has to be provided separately and this has been shown to be not straightforward [BarShalom90]. The PDA mechanization is incorporated into the estimation filter as described below (extracted from [BarShalom75] and [BarShalom88])

PDA Gating

Before the PDA can be applied a gating process is performed over all received measurements at an instant k. The gating process is used to select those measurements lying in the neighbourhood of the target predicted position. It has been used in most of the tracking algorithms developed up to date [Kurien90]. A precise equation for the gating process is given by

$$\left[\left(\mathbf{z}(k)-\overset{\wedge}{\mathbf{z}}(k)\right)\mathbf{S}^{-1}\left(\mathbf{z}(k)-\overset{\wedge}{\mathbf{z}}(k)\right)'\right]\langle\ \gamma$$

This type of gating is commonly known as *ellipsoidal gating*, although other simpler gate shapes exist [Blackman86]. The ellipsoid is in fact a probability concentration or a statistical region around the target predicted position. The parameter γ is obtained from X^2 (chi-square) distributions [Kreyszig88].

After the gating, fewer measurements are taken as candidates for updating the filter, which means only those lying inside the gate. These measurements cumulative up to time k can be represented as

$$\mathbf{Z}^{\mathbf{k}} \equiv \{\mathbf{Z}(k)\}_{j=1}^{k}$$

and the set of validated measurements at time k by

$$\mathbf{Z}(k) = \{\ \mathbf{z}_i(k)\ \}_{i=1}^{m_k}$$

where in the above m_k is the number of measurements lying inside the validation gate.

It is also assumed that the state is normally distributed (Gaussian) conditioned upon the latest estimate and corresponding covariance, or

$$p\{\mathbf{x}(k)|, Z^{k-1}\} = N\left(\mathbf{x}(k); \hat{\mathbf{x}}(k|k-1); \mathbf{P}(k|k-1)\right)$$

<u>PDA Weights</u>

Amongst those validated measurements (m_k) there is possibly a correct one and the others are false measurements and/or clutter or alternatively none of them is correct. This events can be expressed as

$$\chi_i(k) = \{\mathbf{z}_i(k) \text{ is the correct measurement}\}_{i=1}^{m_k}$$

and

$$\chi_0(k) = \{\text{none of the measurements is correct}\}$$

with probabilities given by

$$\beta_i(k) = p[\chi_i(k)|\mathbf{Z^k}]$$

Based on the above assumptions about the events they are mutually exclusive and exhaustive, thus

$$\sum_{i=0}^{m_k} \beta_i = \sum_{i=0}^{m_k} p[\chi_i(k)|\mathbf{Z^k}] = 1$$

The best estimate for the target position is given by a conditional mean based upon all measurements that with some non-zero probability may have originated from the target of interest. This can expressed as

$$\hat{\mathbf{x}}(k|k) = E\left[\mathbf{x}(k)|\mathbf{Z}^{\mathbf{k}}\right] = \sum_{i=1}^{m_k} p\left[\chi_i(k)|\mathbf{Z}^{\mathbf{k}}\right] E\left[\mathbf{x}(k)|\chi_i(k),\mathbf{Z}^{\mathbf{k}}\right]$$

Therefore, by making use of the Bayes' total probability rule [Papoulis87] yields

$$\hat{\mathbf{x}}(k|k) = \sum_{i=0}^{m_k} \beta_i(k)\hat{\mathbf{x}}_i(k|k)$$

where $\hat{\mathbf{x}}_i(k|k)$ is the update estimate conditioned on the event that one of the measurements is correct. Making use of the above equations and assuming that i measurement is the correct one results in

$$\hat{\mathbf{x}}_i(k|k) = \hat{\mathbf{x}}(k|k-1) + \mathbf{W}(k)\,\nu_i(k)$$

with

$$\nu_i(k) = \mathbf{z}_i(k) - \hat{\mathbf{z}}(k|k-1)$$

Assuming that when none of the measurements is correct, i.e. i = 0, the estimate is given by

$$\hat{\mathbf{x}}_0(k|k) = \hat{\mathbf{x}}(k|k-1)$$

by now combining the equations for $\hat{\mathbf{x}}_i(k|k)$ and $\hat{\mathbf{x}}_0(k|k)$ in the equation for $\hat{\mathbf{x}}(k|k)$ makes the final expression for the combined estimate as

$$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + \mathbf{W}(k)\,\nu(k)$$

where

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

In a similar way the corresponding covariance can be obtained and it is given by

$$\hat{\mathbf{P}}(k|k) = \beta_0\,\hat{\mathbf{P}}(k|k-1) + [1-\beta_0][\mathbf{I}-\mathbf{W}(k)\mathbf{H}(k)]\hat{\mathbf{P}}(k|k-1) + \mathbf{W}(k)\left[\sum_{i=1}^{m_k}\beta_i(k)\,\nu_i(k)\,\nu_i'(k) - \left(\nu(k)\,\nu(k)'\right)\right]\mathbf{W}(k)'$$

The main objective now is therefore to obtain expressions for the a *posteriori* or association probabilities or PDA weights $\beta_i(k)$. A complete description of how they can be determined is detailed in [BarShalom75] and [BarShalom88], only the final form is shown here. The association probabilities can be written as

$$\beta_j(k)=\mathbf{P}\left\{\chi_i(k)|\mathbf{Z}(k),m_k,\mathbf{Z}^{k-1}\right\}, \qquad i=0,1,2,\ldots,m_k$$

from Bayes' rule the above is rewritten as

$$\beta_i(k)=\frac{\mathbf{p}\left[\mathbf{Z}(k)|\chi_i(k),m_k,\mathbf{Z}^{k-1}\right]\mathbf{P}\left\{\chi_i(k)|m_k,\mathbf{Z}^{k-1}\right\}}{C}$$

where C is the normalisation constant. The probability for a measurement being the correct one is given by

$$\mathbf{p}\left[\mathbf{z}_i(k)|\chi_i(k),m_k,\mathbf{Z}^{k-1}\right]=\frac{\mathrm{N}\left(\nu_i(k);0;\mathbf{S}(k)\right)}{P_G}$$

where P_G is the probability that the correct measurement falls inside the gate (gate probability). Therefore the total probability assuming that one of the measurements is the correct one and all the others are incorrect (m_k - 1) results in

$$\mathbf{p}\left[\mathbf{Z}(k)|\chi_i(k),m_k,\mathbf{Z}^{k-1}\right]=\frac{\mathrm{N}(\nu_i;0;\mathbf{S})}{V^{m_k-1}\,P_G} \qquad i=1,2,\ldots,m_k$$

The total probability when all measurements falling inside the gate are incorrect is given by

$$\mathbf{p}\left[\mathbf{Z}(k)|\chi_i(k),m_k,\mathbf{Z}^{k-1}\right]=\frac{1}{V^{m_k}} \qquad i=0$$

The probabilities of the event χ_i conditioned on the number of measurements falling inside the gate is given by [Bar-Shalom88]

$$\mathbf{P}\left\{\chi_i(k)|\,m_k,\mathbf{Z}^{k-1}\right\}=\mathbf{P}\left\{\chi_i(k)|\,m_k\right\}=$$
$$=\begin{cases}\dfrac{1}{m_k}P_G\,P_D\left[P_G\,P_D+(1-P_G\,P_D)\dfrac{\mu_F(m_k)}{\mu_F(m_k-1)}\right] & i=1,2,\ldots,m_k\\[2ex] \dfrac{(1-P_G\,P_D)}{\left[P_G\,P_D+(1-P_G\,P_D)\dfrac{\mu_F(m_k)}{\mu_F(m_k-1)}\right]}\,\dfrac{\mu_F(m_k)}{\mu_F(m_k-1)} & i=0\end{cases}$$

In the above equation P_D is the probability that the correct measurement is detected (detection probability). The function $\mu_F(m_k)$ is the probability mass function for the spurious measurements and it can be either Poisson or uniform distributed. These distribution results in two different models for the probability mass function that can be represented by

1. Parametric Model

$$\mu_F(m_k)=\exp(-\lambda\,V)\frac{(\lambda\,V)^{m_k}}{m_k\,!}$$

which is a Poisson distribution.

2. Nonparametric Model

$$\mu_F(m_k)=\frac{1}{N}$$

which is a Uniform distribution.

Because only the nonparametric model is of interest in this thesis, after some manipulation the resulting association probabilities are :

$$\beta_i(k) = \frac{e_i}{b + \sum_{j=1}^{m_k} e_j} \qquad i = 1, 2, \ldots, m_k$$

$$\beta_0(k) = \frac{b}{b + \sum_{j=1}^{m_k} e_j}$$

where

$$b = \frac{m_k}{V}\frac{(1-P_D\,P_G)}{P_D\,P_G}$$

and

$$e_i = \frac{N(\nu_i;0;\mathbf{S})}{P_G}$$

with P_G as the probability of the correct measurement to be inside the gate, P_D as the target detection probability and V is the volume of the elliptical validation region (gate) or

$$V(k) = c_{nz}\,\gamma^{nZ/2}\sqrt{\|\mathbf{S}(k)\|}$$

where *nz* is the dimension of the measurement vector z and the corresponding c_{nz} is the volume of the *nz* dimensional unit hypersphere (c1=2, c2= ,c3=4 /3, etc..) [Fortmann83].

References

[Alasdair94] Alasdair,R., Bruce,A., Mills,J.G. and Smith,A.G. - *CHIMP Version 2.0 User Guide*. University of Edinburgh, March 1994.

[Andrews68] Andrews,A. - *A Square-Root Formulation of the Kalman Covariance Equations*. AIAA Journal; Vol.6, No. 11, June 1968, pp65-6.

[Annaratone87] Annaratone,M., Arnould,E., Gross,T., Kung,H.T., Lam,M., Menzilcioglu,O. and Webb,J. - *The Warp Computer: Architecture, Implementation, and Performance*. IEEE Transactions on Computers;Vol. 36, December 1987, pp1523-38.

[Atherton90] Atherton, D.P., Gul, E., Kountzeris, A. and Kharbouch, M.M. - *Tracking Multiple Targets using Parallel Processing*. IEE, Pt. D, No. 4, 1990, pp225-31`.

[Atherton93] Atherton,D.P. and Hussain,D.M.A. - *Development of a Multiple Target Tracking Algorithm on Transputers*. Parallel Processing in a Control System Environment, Edited by Eric Rogers and Yun Li. Prentice Hall, Inc.; Chapter 8; 1993.

[Atherton94] Atherton,D.P. and Lin,H.J. - *Parallel Implementation of IMM Tracking Algorithm using Transputers*. IEE Proceedings of Radar, Sonar Navigation; Vol. 141, No. 6, December 1994, pp325-332.

[Averbuch91a] Averbuch,A., Itzikowitz,S. and Kapon,T. - *Parallel Implementation of Multiple Model Tracking Algorithms*. IEEE Transactions on Parallel and Distributed Systems; Vol. 2, No. 2, April 1991, pp242-51.

[Averbuch91b] Averbuch,A., Itzikowitz,S. and Kapon,T. - *Radar Target Tracking - Viterbi versus IMM*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 27, No. 3, May 1991, pp550-63.

[Baheti86] Baheti,R. - *Efficient Approximation of Kalman Filter for Target Tracking*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 22, No. 1, January 1986, pp8-14.

[Baheti88] Baheti,R.S. and O'Hallaron,D.R. - *Efficient Parallel Implementation of Target Tracking Kalman Filter*. Proc. 27th Conference on Decision and Control; Austin, December 1988, pp376-81.

[Baillie87] Baillie,C.F., Gottschalk,T.D. and Kolawa,A. - *Comparisons of Concurrent Tracking on Various Hypercubes*. Caltech;C3P-568.

[Bala94] Bala,V.et al. - *The IBM External User Interface for Scalable Parallel Systems*. Parallel Computing. Elsevier Science B.V.; 20, 1994; pp445-462.

[Barret90] Barret,I. - *Synthese d'Algorithms de Poursuit Multi-Radars d'Avions Civils Manouvrants*. Thèse de Doctorat, Ecóle Nationale Supérieure de l'Aéronautique et de l'Espace, 1990.

[BarShalom75] BarShalom,Y. and Tse,E. - *Tracking in a Cluttered Environment with Probabilistic Data Association*. Automatica; Vol. 11, 1975, pp451-60.

[BarShalom78] BarShalom,Y. - *Tracking Methods in a Multitarget Environment*. IEEE Transactions on Automatic Control; Vol. 23, No. 4, August 1978, pp618-26.

[BarShalom83] BarShalom,Y. and Birmiwal, K. - *Consistency and Robustness Evaluation of the PDAF for Target Tracking in a Cluttered Environment*. Automatica; Vol. 19, July 1993, pp431-7.

[BarShalom88] BarShalom,Y. and Fortmann,T.E. - *Tracking and Data Association*. Mathematics in Science and Engineering; Academic Press 1988; Vol. 179.

[BarShalom90] BarShalom,Y., Chang,K.C. and Blom,H.A.P. - *Automatic Track Formation in Cutter with a Recursive Algorithm.* Multitarget-Multisensor Tracking: Advanced Applications, Volume 1. Edited by Yaakov Bar-Shalom. Artech House,Inc. 1990; Vol. 1.

[BarShalom92] BarShalom,Y. - *Multitarget-Multisensor Tracking: Applications and Advances-Volume 2*. Artech House,Inc. 1992; Vol. 2.

[BarShalom92a] Bar-Shalom,Y., Chang,K.C. and Blom,H.A.P. - *Tracking Splitting Targets in Clutter by Using an Interacting Multiple Model Joint Probabilistic Data Association Filter*. Multitarget-Multisensor Tracking: Applications and Advances, Volume 2. Edited by Yaakov Bar-Shalom. Artech House,Inc. 1992.

[Barton88] Barton,D.K. - *Modern Radar Analysis*. Artech House,Inc. 1988.

[Barton94] Barton,E., Cownie,J. and McLaren,M. - *Message Passing on the Meiko CS-2*. Parallel Computing. Elsevier Science B.V.; 20, 1994, pp497-507.

[Bellantoni67] Bellantoni,J.F. and Dodge,K.W. - *A Square Root Formulation of the Kalman-Schmidt Filter*. AIAA Journal; Vol. 5, July 1967, pp1309-14.

[Bergland72] Bergland,G.D. and Hunnicutt,C.F. - *Application of a Higly Parallel Processor to Radar Data Processing*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 8, No. 2, March 1972, pp161-7.

[Bierman73] Bierman,G.J. - *A Comparison of Discrete Linear Filtering Algorithms*. IEEE Transactions on Aerospace and Electronic Systems;Vol. 9, No. 1, January 1973, pp28-37.

[Bierman77] Bierman,G.J. - *Factorization Methods for discrete Sequential Estimation*. Mathematics in Science and Engineering; Academic Press 1977; Vol. 128.

[Blackman86] Blackman,S.S. - *Multiple Target Tracking with Radar Applications*. Artech House,Inc. 1986.

[Blackman90] Blackman,S.S. - *Association and Fusion of Multiple Sensor Data*. Multitarget-Multisensor Tracking: Advanced Applications, Volume 1. Edited by Yaakov Bar-Shalom. Artech House,Inc. 1990.

[Blakrishnan87] Blakrishnan,A.V. - *Kalman Filtering Theory*. Optimization Software, Inc. 1987.

[Blom] Blom, H.A.P. - *"Application of a Probabilistic Tracking Filter - A First Evaluation with Emphasis on Air Traffic Control"*. NLR Report TR 82088, Part I and Part II, The Netherlands, 1982.

[Blom82] Blom,H.A.P. - *Application of a Probabilistic Tracking Filter : A First Evaluation with Emphasis on Air Traffic Control - Part 1 : Evaluation Results*. National Aerospace Laboratory NLR. NLR TR 82088 U Part I; Amsterdam , The Netherlands.

[Blom83a] Blom, H.A.P. - *Comparison of a Jump-Diffusion Tracker with a Kalman Tracker - An Evaluation with Emphasis on Air Traffic Control*. NLR Report TR 83063 U, The Netherlands, 1983.

[Blom83b] Blom, H.A.P. - *Markov Jump-Diffusion Models and Decision-Making-Free Filtering*. NLR Report MP 83067 U, The Netherlands, 1983.

[Blom84a] Blom,H.A.P. - *A Sophisticated Tracking Algorithm for ATC Surveillance Radar Data*. Proceedings of International Radar Conference;Paris, 1984, pp393-8.

[Blom84b] Blom,H.A.P. - *An Efficient Filter for Abruptly Changing Systems*. Proceedings 23th Conference on Decision and Control; Las Vegas, December 1984, pp656-8.

[Blom88] Blom,H.A.P. and BarShalom,Y. - *The Interacting Multiple Model Algorithm for Systems with Markovian Switching Coefficients"*. IEEE Transactions on Automatic Control; Vol. 33, No. 8, August 1988.

[Blom90a] Blom H.A.P. - *Bayesian Estimation for Decision-Directed Stochastic Control*. NLR Report, TP 90039 U, The Netherlands, 1990.

[Blom90b] Blom H.A.P., Hogendoorn R.A., and van Schaik, F. J. - *Bayesian Multi-Sensor Tracking for Advanced Air Traffic Control Systems*. Aircraft Trajectories Computation-Prediction-Control, AGARD AG-301, Vol. 2, May 1990.

[Blom92a] Blom.,H.A.P., Hogendoorn,R.A. and van Doorn,B.A. - *Design of a Multisensor Tracking System for Advanced Air Traffic Control*. Multitarget-Multisensor Tracking: Applications and Advances, Volume 2. Edited by Yaakov Bar-Shalom. Artech House,Inc. 1992.

[Bozic79] Bozic,S.M. - *Digital and Kalman Filtering*. Edward Arnold 1979.

[Brooker88] Brooker,P. - *Demand, Capacity and Traffic Management*. CAA-Civil Aviation Authority, April 1988;DORA-8809.

[Brookner77] Brookner,E. - *Radar Technology*. Artech House,Inc. 1977.

[Butler92] Butler,R. and Lusk,I. - *User's Guide to the P4 Programming System*. Argonne National Laboratory, Technical Report ANL-92/17, 1992.

[Calkin94] Calkin,R., Hempel,R., Hoppe,H. and Wypior,P. - Portable Programming with the PARMACS Message Passing Library. Parallel Computing. Elsevier Science B.V.; 20, 1994.

[Carlson73] Carlson,N.A. - *Fast Triangular Formulation of Square-Root Filter*.AIAA Journal; Vol. 5, September 1973, pp1259-65.

[Carlson90] Carlson,N.A. - *Federate Square Root Filter for Decentralized Parallel Processes*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 26, No. 3, May 1990, pp517-25.

[Carpantier88] Carpantier,M.H. - *Principles of Modern Radar Systems*. Artech House,Inc. 1988.

[Castella74] Castella,F.R. and Dunnebacke,F.G. - *Analytical Results for the x,y Kalman Tracking Filter*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 10, No. 6, November 1974, pp891- 5.

[Castella76] Castella,F.R. - *Sliding Window Detection Probabilities*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 11, No. 6, November 1976, pp815-9.

[Chan79] Chan,Y.T., Hu,A.G.C. and Plant,J.B. - *A Kalman Filter Based Tracking Scheme with Input Estimation*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 15, No. 2, March 1979, pp237-44.

[Chan82] Chan,Y.T., Plant,J.B. and Bottomley,J.R.T. - *A Kalman Tracker with a Simple Input Estimator*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 18, No. 2, March 1982, pp235-41.

[Chang84] Chang, K.C. and Bar-Shalom,Y. - *Joint Probabilistic Data Association for Multitarget Tracking with possibly Unresolved Measurements and Maneuvers*. IEEE Transactions on Automatic Control; Vol. 29, No. 7, July 1984, pp585-94.

[Chui87] Chui,C.K. and Cheng,G. - *Kalman Filtering with Real Time Applications*. Springer- Verlag 1987.

[CS2a] Meiko CS-2 - *Meiko CS-2 Documentation Guide*. Meiko, S1002-00C101 CVS1.7, 1993.

[CS2b] Meiko CS-2 - *MPSC: Tagged Message Passing & Global Reduction*. Meiko, S1002-10M108 CVS1.6, 1993.

[CS2c] Meiko CS-2 - *Parallel Virtual Machine (PVM) User's Guide and Reference Manual* - *Version 3.0*. Meiko, 1993.

[Debbage91] Debbage,M., Hill,M.B. and Nicole,D.A. - *Virtual Channel Router version 2.0 User Guide*. ESPRIT Project PUMA P2701 working paper no. 25, University of Southampton, June 1991.

[Debbage92a] Debbage,M. and Hill,M.B. - *ParaPET: A Parallel Programming Environment Toolkit*. University of Southampton, June 1992.

[Debbage92b] Debbage,M., Hill,M.B. and Nicole,D.A. - *An Interface to a Reliable Packet Delivery Service for Parallel Systems*. University of Southampton, April 1992.

[DeRusso66] DeRusso,D.M., Roy,R.J. and Close,C.M. - *State Variables for Engineers*. John Wiley & Sons 1966.

[Deutsch65] Deutsch,R. - *Estimation Theory*. Prentice Hall,Inc. 1965.

[Dufour92] Dufour,F. and Mariton,M. - *Passive Sensor Data Fusion and Maneuvering Target Tracking*. Multitarget-Multisensor Tracking: Applications and Advances, Volume 2. Edited by Yaakov Bar-Shalom. Artech House,Inc. 1992.

[Edward92] Edward,M.N. and Thompson,H.A. - *Array Processors for Real Time Radar Signal Processing*. Parallel Computers and Transputer Applications; Vol. I and II, October 1992, pp945-54.

[FAA85] FAA-Federal Aviation Administration - *An Overview of the Advanced Automation Program*. US Department of Transportation Mar 1985;FAA/APP85.

[Farina80] Farina,A. and Pardini,S. - *Survey of Radar Data Processing Techniques in Air Traffic Control and Surveillance Systems*. IEE Proceedings; Vol. 127, No. 3, June 1980, pp190-204.

[Farina82] Farina,A. - *Multistatic Tracking and Comparison with Netted Monostatic System*. IEE International Conference on Radar 82; October 1982, pp183-7.

[Farina85] Farina,A. and Studer,F.A. - *Radar Data Processing*. Volume 1 - Introduction and Tracking. Research Studies Press,John Willey & Sons 1985; Vol. 1.

[Farina86] Farina,A. and Studer,F.A. - *Radar Data Processing*. Volume II - Advanced Topics and Applications. Research Studies Press,John Willey & Sons 1986; Vol. 2.

[Fisher89] Fisher, J.L. and Casasent, D.P. - *Fast JPDA multitarget tracking algorithm*.Applied Optics; Vol. 28, No. 2, January 1989, pp371-6.

[Fischetti86] Fischetti,M.A. and Perry,T.S. - *Our Burdened Skies*. IEEE Spectrum; November 1986, pp36- 80.

[Fitzgerald71] Fitzgerald,R.J. - *Divergence of the Kalman Filter*. IEEE Transactions on Automatic Control; Vol. 16, No. 6, June 1971, pp736-47.

[Flower94] Flower,J. and Kolawa,A. - *EXPRESS is not just a Message Passing System; Current and Future Direction in EXPRESS*. Parallel Computing. Elsevier Science B.V.; 20, 1994, pp597-614.

[Fortmann83] Fortmann,T.E., BarShalom,Y. and Scheffe,M. - *Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association*. IEEE Journal of Oceanic Engineering; Vol. 8, No. 3, July 1983, pp173-84.

[Fox88] Fox ,G., Johnson, M., Lyzenga, G., Otto, S., Salmon, J. and Walker, D. - *Solving Problems on Concurrent Processors: Volume I and Volume II*. Prentice-Hall, Inc. 1988.

[Franceschetti90] Franceschetti,G., Mazzeo,A., Mazzocca,N., Pascazio,V. and Schirinzi,G. - *An Efficient SAR Parallel Processor*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 27, No. 2, March 1991, pp343-52.

[Friedland73] Friedland,B. - *Optimum Steady-State Position and Velocity Estimation Using Noisy Sampled Position Data*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 9, No. 6, November 1973, pp906-11.

[Gaston89] Gaston,F.M.F. and Irwin,G.W. - *A Systolic Approach to Square-Root Information Filtering*. International Journal of Control;Vol. 5, No. 1, 1989, pp225-48.

[Gaston90] Gaston,F.M.F. and Irwin,G.W. - *SystolicKalman Filtering: An Overview*. IEE Proceedings. Vol. 137, Pt. D, No. 4, July 1990.

[Gauss63] Gauss,K.F. - *Theoria Motus - Theory of the Motion of Heavenly Bodies Moving About the Sun in Conic Sections*. Dover Publications, Inc. 1963.

[Gelb74] Gelb,A. - *Applied Optimal Estimation*. The Analitic Sci. Corp.; The MTI Press 1974.

[Gertz83] Gertz,J.L. - *Mode S Surveillance Netting*. Lincoln Laboratory - DOT/FAA/ Nov 1983;PM-83-17.

[Gertz89] Gertz,J.L. - *Multisensor Surveillance for Improved Aircraft Tracking*. The Lincoln Laboratory Journal; Vol. 2, No. 3, 1989, pp381-96.

[Gholson77] Gholson,N.H. and Moose,R.L. - *Maneuvring Target Tracking Using Adaptive State Estimation*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 13, No 3, May 1977, pp310-7.

[Giaccari79] Giaccari,E. and Nucci,G. - *A Family of Air Traffic Control Radars*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 15, No. 3, May 1979, pp378-96.

[Gillespie92] Gillespie,D.T. - *Markov Processes - An Introduction for Physical Scientists*. Academic Press, Inc. 1992.

[Goillau89] Goillau,P.J. - *A System Analysis Study of Air Traffic Control: Identification of Future ATC Computer Assistence Concepts for NERC*. Royal Signals and Radar Establishment Oct 1989;AD4/89/10.

[Goldstein80] Goldstein,H. - *Classical Mechanics*. Addison-Wesley Publishing Company 1980.

[Golub83] Golub,G.H. and Van,Loan,C.F. - *Matrix Computations*. North Oxford Academic 1983.

[Gottschalk87a] Gottschalk,T.D. - *Multiple Target Track Initiation on a Hypercube*. Caltech; C3P-398.

[Gottschalk87b] Gottschalk,T.D. - *A New Multitarget Tracking Model*. Caltech; July 1987; C3P- 480.

[Gottschalk87c] Gottschalk,T.D. - *Concurrent Multiple Target Tracking*. Caltech; C3P-567.

[Gottschalk90] Gottschalk,T.D. - *Concurrent Multi-Target Tracking*. In The Fith Distributed Memory Computing Conference - Proceedings and Applications. IEEE Computer Society Press; Vol. 1, April 1990, pp85-8.

[Gul89] Gul,E. and Atherton,D.P. - *Transputer Implementation for Multiple Target Tracking*. Microprocessors and Microsystems; Vol. 13, No. 3, April 1989, pp188-94.

[Hashemi87] Hashemi,R.H., Roy,S. and Laub,A.J. - *Parallel Structures for Kalman Filtering*. Proceedings of 26th Conference on Decision and Control; Los Angeles-USA, December 1987, pp1476-81.

[Hey87] Hey,A.J.G. - *Parallel Decomposition of Large Scale Simulations in Science and Engineering*. SHEP86/87 - 2.

[Houles89] Houles,A. and BarShalom,Y. - *Multisensor Tracking of a Maneuvring Target in Clutter*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 25, No. 2, March 1989, pp176-89.

[Hovanessian73] Hovanessian,S.A. - *Radar Detection and Tracking Systems*. Artech House,Inc. 1973.

[Hovanessian84] Hovanessian,S.A. - *Radar System Design and Analysis*. Artech House,Inc. 1984.

[Hunt87] Hunt,V.R. and Kloster,G.V. - *The Federal Aviation Administration's Advanced Automation Program*. IEEE Computer; February 1987, pp14-82.

[Hutchinson84] Hutchinson,C.E. - *The Kalman Filter Applied to Aerospace and Electronic Systems*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 20, No. 4, July 1984, pp500-4.

[Hwang87] Hwang,K. and Brigss,F.A. - *Computer Architecture and Parallel Processing*. McGraw-Hill Book Company, 3rd. edition, 1987.

[Inmos88] Inmos - *The Transputer Data Book*. INMOS Databook Series, 1988.

[Jazwinski70] Jazwinski,A.H. - *Stochastic Processes and Filtering Theory*. Math. in Sci. and Eng.; Academic Press 1970; Vol. 64.

[Jover86] Jover,J.M. and Kailath,T. - *A Parallel Architecture for Kalman Filter Measurement Update and Parameter Estimation*. Automatica; Vol. 22, No. 1, 1986, pp43-57.

[Kailath68a] Kailath,T. - *An Innovations Approach to Least Squares Estimation; Part I: Linear Filtering in Additive Noise*. IEEE Transactions on Automatic Control; Vol. 13, No. 6, December 1968, pp646-55.

[Kailath68b] Kailath,T. and Frost,P. - *An Innovations Approach to Least Squares Estimation; Part II: Linear Smoothing in Additive Smoothing in Additive Noise*. IEEE Transactions on Automatic Control; Vol. 13, No. 6, December 1968, pp655-60.

[Kailath71a] Kailath,T. and Frost,P.A. - *An Innovations Approach to Least Square Estimation; Part III: Nonlinear Estimation in White Gaussian Noise*. IEEE Transactions on Automatic Control; Vol. 16, June 1971, pp217-26.

[Kailath71b] Kailath,T. and Gessey,R.A. - *An Innovations Approach to Least Squares Estimation; Part IV: Recursive Estimation Given Lumped Covariance Functions*. IEEE Transactions on Automatic Control; Vol. 16, December 1971, pp720-7.

[Kailath73a] Kailath,T. and Geesey,R. - *An Innovations Approach to Least Square Estimation; Part V: Innovations Represetations and Recursive Estimation in Colored Noise*. IEEE Transactions on Automatic Control; Vol. 18, October 1973, pp435-53.

[Kailath73b] Kailath,T. - *An Innovations Approach to Least Square Estimation; Part VI: Discrete Time Innovations Representations and Recursive Estimation*. IEEE Transactions on Automatic Control; Vol. 18, December 1973, pp588-600.

[Kailath73c] Kailath,T. and Aasnaes,H. - *An Innovations Approach to Least Square Estimation; Part VII: Some Applications of Vector Autoregressive Moving Average Models*. IEEE Transactions on Automatic Control; Vol. 18, December 1973, pp601-7.

[Kailath74] Kailath,T. - *A View of Three Decades of Linear Filtering Theory*. IEEE Transactions on Information Theory; March 1974, pp146-81.

[Kailath80] Kailath,T. - *Linear Systems*. Prentice Hall,Inc. 1980.

[Kalata84] Kalata,P.R. - *The Tracking Index:A Generalized Parameter for Alpha-Beta and Alpha-Beta-Gama Target Trackers*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 20, No. 2, March 1984, pp174-82.

[Kalman60] Kalman,R.E. - *A New Approach to Linear Filtering and Prediction Problems*. Transactions of ASME - Journal of Basic Engineering; Vol. 82, March 1960, pp35-45.

[Kalman61] Kalman,R.E. and Bucy,R.E. - *New Results in Linear Filtering and Prediction Theory*. Transactions of ASME - Journal of Basic Engineering; Vol. 83, December 1961, pp95-107.

[Kee91] Kee,R.J. and Irwin,G.W. - *Transputer Implementation of Tracking Kalman Filters*. IEE International Conference on Control 91; Vol. II, No. 332, 1991, pp861-6.

[Kernighan88] Kernighan, D. and Ritchie, D.M. - *The C Programming Language* Englewood Cliffs N.J. Prentice Hall, 1988, 2nd Edition

[Kolawa86a] Kolawa,A. and Zimmerman,B. - *CrOS III Manual*. Caltech; 1986, C3P-253.

[Kolawa86b] Kolawa,A. - *Implementation of a High Performance Crystalline Operating System on the Intel iPSC Hypercube*. Caltech Concurrent Computing Project Memo, 1986, C3P-247.

[Kreyszig88] Kreyszig,E. - *Advanced Engineering Mathematics*. John Wiley & Sons 1988.

[Kurien86] Kurien,T., Allen,T.G., Washburn,Jr. and R.B. - *Parallelism in Multitarget Tracking and Adaptation to Multiprocessor Architectures*. Proceedings 9th MIT/ONR Workshop on C3 Systems; December 1986, pp45-51.

[Kurien90] Kurien,T. - *Issues in the Design of Practical Multitarget Tracking Algorithms*. Multitarget-Multisensor Tracking: Advanced Applications, Volume 1. Edited by Yaakov Bar-Shalom. Artech House,Inc. 1990; Vol. 1.

[Lanczos70] Lanczos,C. - *The Variational Principles of Mechanics*. University of Toronto Press 1970.

[Lawrie90] Lawrie,D.I. - *Parallel Processing Algorithms and Architectures for the Kalman Filter*. British Library Document Supply Centre; Ph.D. Thesis, University College of North Wales, Bangor, August 1990.

[Lawrie91] Lawrie,D. and Fleming,P. - *Fine Grain Parallel Processing Implementations of Kalman Filter Algorithms*. IEE International Conference on Control 91; Vol. II, No. 332, 1991, pp867-70.

[Lee88] Lee,E.K.B. and Haykin,S. - *Parallel Implementation of the Tracking Kalman Filter*. Proceedings of ICASSP 88; Vol. 1, No. 5, 1988, pp2092-5.

[Lerro93a] Lerro,D. and Bar-Shalom,Y. - *Interacting Multiple Model with Target Amplitude Feature*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 29, No.2, April 1993.

[Lerro93b] Lerro,D. and Bar-Shalom,Y. - *Tracking with Debiased Consistent Converted Measurement Versus Extended Kalman Filter*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 29, No.3, July 1993.

[Li93a] Li,X.R. and Bar-Shalom,Y. - *Design of an Interacting Multiple Model Algorithm for Air Traffic Control Tracking*. IEEE Transactions on Control Systems Technology; Vol. 1, No. 3, September 1993.

[Li93b] Li,X.R. and Bar-Shalom,Y. - *Performance Prediction of the Interacting Multiple Model*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 27, No. 3, July 1993.

[Li94] Li,X.R. and Bar-Shalom,Y. - *Estimation and Tracking: Principles, Techniques and Software*. Artech House Radas Library, 1994.

[Liebelt67] Liebelt,P.B. - *An Introduction to Optimal Estimation*. Addison-Wesley Publishing Company 1967.

[Lin93a] Lin,H.J. - *Investigation of Manoeuvring Target Tracking using Interacting Multiple Model Algorithms*. D.Phil Thesis, University of Sussex, 1993.

[Lin93b] Lin,H.J. and Atherton,D.P. - *An Investigation of Interacting Multiple Model Algorithm for the Manoeuvring Target Tracking*. First Regional IEEE Conference Aerospace Control Systems; May 1993, pp113-17.

[Lin93c] Lin,H.J. and Atherton,D.P. - *An Investigation of the Selected Filter Interacting Multiple Model Algorithm for Tracking Manoeuvring Targets*. Proceedings of the 32nd Conference on Decision and Control; San Antonio, Texas-USA, 1993, pp930-35.

[Matlab93] Matlab - *MATLAB User's Guide-Version 5.0.* The MathWorks, Inc. 1993.

[McCanny89] McCanny,J., McWhirter,J. and Swartzlander Jr, E. - *"Systolic Array Processors"*. Prentice-Hall, 1989.

[Maybeck79] Maybeck,P.S. - *Stochastic Models,Estimation and Control*. Math. in Sci. and Eng.; Academic Press 1979/1982; Volumes 1, 2 and 3.

[Mendel71] Mendel,J.M. - *Computational Requirements for a Discrete Kalman Filter*. IEEE Transactions on Automatic Control; Vol. 16, No. 6, December 1971, pp748-58.

[Messina90] Messina,P. et,al. - *Benchmarking Advanced Architecture Computers*. Concurrency: Practice and Experience; Vol. 2, No. 2, June 1990.

[Miller82] Miller,K.S. and Leskiw,D.M. - *Nonlinear Estimation with Radar Observations*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 18, No. 2, March 1982, pp192-200.

[Miller87] Miller,K.S. and Leskiw,D.M. - *An Introduction to Kalman Filtering with Applications*. R.E.Krieger Publishing Company 1987.

[Moose75] Moose,R.L. - *An Adaptive State Estimation Solution to the Maneuvering Target Problem*. IEEE Transactions on Automatic Control; Vol. 20, No. 5, June 1975.

[Morefield77] Morefield,C.L. - *Application of 0-1 Integer Programming to Multitarget Tracking Problems*. IEEE Transactions on Automatic Control; Vol. 22, No. 6, June 1977, pp302-11.

[MPI94] M.P.I. Forum - *MPI: A Message-Passing Interface Standard*. March 1994.

[Mulholland82] Mulholland,R.G. and Stout,D.W. - *Stereographic Projection in the National Airspace System*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 18, No. 1, January 1982, pp48- 57.

[Munir95] Munir,A. and Atherton, D.P. - *Adaptive Interacting Multiple Model Algorithm for Tracking a Manoeuvring Target*. IEE Proceedings of Radar, Sonar and Navigation; Vol. 142, No. 1, February 1995, pp11-17.

[OHallaron88] O'Hallaron,D. and Baheti,R.S. - *Parallel Implementation of a Kalman Filter on the Warp Computer*. Proceedings of ICPP 88; Vol. 3, 1988, pp108-11.

[Oppenheim78] Oppenheim,A.V. - *Applications of Digital Signal Processing*. Prentice Hall, Inc. 1978.

[Papoulis87] Papoulis,A. - *Probability, Random Variables and Stochastic Processes*. Second Edition, McGraw-Hill, Inc. 1987.

[Parasoft90a] Parasoft - *EXPRESS C - Reference Guide, Version 3.0*. Parasoft Corporation, 1990

[Parasoft90b] Parasoft - *EXPRESS 3.0 - User's Guide*. Parasoft Corporation, 1990.

[Pattipati90] Pattipati,K.R., Kurien,T., Lee,R. and Luh,P.B. - *On Mapping a Tracking Algorithm onto Parallel Processors*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 26, No. 5, September 1990, pp774-91.

[Pearson74] Pearson,III,J.B. and Stear,E.B. - *Kalman Filter Applications in Airborne Radar Tracking*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 10, No. 3, May 1974, pp319-29.

[Pierce94] Pierce,P. - *The NX Message Passing Interface*. Parallel Computing. Elsevier Science B.V.; 20, 1994.

[Press88] Press,W.H., Flannery,B.P., Teukolsky,S.A. and Vettering,W.T. - *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press 1988.

[Quin87] Quin, M.J. - *"Design Efficient Algorithms for Parallel Computers"*. MacGraw-Hill Book Company, 1987.

[Reid79] Reid,D.B. - *An Algorithm for Tracking Multiple Targets*. IEEE Transactions on Automatic Control; Vol. 24, No. 6, December 1979, pp843-54.

[Raghavan93] Raghavan,V., Pattipati, K.R. and Bar-Shalom, Y. - *Efficient L-D Factorization Algorithms for PDA, IMM, and IMMPDA Filters*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 29, No. 4, October 1993.

[Rhodes90] Rhodes,I.B. - *A Parallel Decomposition for Kalman Filters*. IEEE Transactions on Automatic Control; Vol. 35, No. 3, March 1990, pp322-4.

[Ricker78] Ricker,G.G. and Willians,J.R. - *Adaptive Tracking Filter for Maneuvering Targets*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 14, No. 1, January 1978, pp185-93.

[Sage79] Sage,A.P. and Melsa,J.L. - *Estimation Theory with Applications to Communication and Control*. Robert E. Krieger Publishing Company 1979.

[Schooler75] Schooler,C.C. - *Optimal Alfa-Beta Filters for Systems with Modelling Inaccuracies*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 11, No. 6, November 1975, pp1300-6.

[Schwartz75] Schwartz,M. and Shaw,L. - *Signal Processing: Discrete Spectral Analysis, Detection and Estimation*. McGraw-Hill, Inc. 1975.

[Sen79] Sen,A.K. - *Tools of Kinematics: A Critical Review*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 15, No. 1, January 1979, pp40-6.

[Sengupta89] Sengupta,D. and Iltis, R. A. - *Neural Solution to the Multitarget Tracking Data Association Problem* .IEEE Transactions on Aerospace and Electronic Systems; Vol. 25, No. 1, January 1989, pp96-108.

[Shank86] Shank,E.M. - *A Coordinate Conversion Algorithm for Multisensor Data Processing*. Lincoln Laboratory - DOT/FAA August 1986;PM-86-37.

[Shaw86] Shaw,J.M. - *A Review of Radar Data Processing Methods for ATC*. Royal Signals and Radar Establishment; July 1986; Report 86010.

[Singer70] Singer,R.A. - *Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 6, No. 4, July 1970, pp473-83.

[Singer71] Singer,R.A. and Behnke,K.W. - *Real-Time Tracking Filter Evaluation and Selection for Tactical Applications*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 7, No. 1, January 1971, pp100-10.

[Singer74] Singer,R.A., Sea,R.G. and Housewright,K.B. - *Derivation and Evaluation of Improved Tracking Filter for Use in Dense Multitarget Environments*. Transactions on Information Theory; Vol. 20, July 1974.

[Sittler64] Sittler,R.W. - *An Optimal Data Association Problem in Surveillance Theory*. IEEE Transactions on Military Electronics; Vol. 8, April 1964, pp125-39.

[Skolnik81] Skolnik,M.I. - *Introduction to Radar Systems*. McGraw-Hill, Inc. 1981.

[Smith75] Smith,P. and Buechler,G. - *A Branching Algorithm for Discriminating and Tracking Multiple Objects*. IEEE Transactions on Automatic Control; Vol. 20, February 1975, pp101-4.

[Smith91] Smith,P.R. and Dailly,C. - *Optimal Filtering and Kalman Filtering Using MATLAB*. IEE International Conference on Control 91; Vol. II, No. 332, 1991, pp734-7.

[Sorenson70] Sorenson,H.W. - *Least Squares Estimation from Gauss to Kalman*. IEEE Spectrum, July 1970, pp63-8.

[Spec95] Spectrum - *Technology 1995, Analysis and Forecast Issue*. IEEE Spectrum, whole issue, January 1995.

[Stevens88] Stevens,M.C. - *Secondary Surveillance Radar*. Artech House,Inc. 1988.

[Stix91] Stix,G. - *Trends in Transportation - Along for the Ride?*. Scientific American; Vol. 265, No. 1, July 1991, pp76-85.

[Sunderam90] Sunderam,V. - PVM: A Framework for Parallel Distributed Computing. Concurrency: Practice and Experience; Vol. 2, No. 4, December 1994.

[Sunderam94] Sunderam, V., Geist, G.A., Dongarra, J. and Manchek, R. - *The PVM Concurrent Computing System: Evolution, Experiences, and Trends*. Parallel Computing. Elsevier Science B.V.; 20, 1994, pp531-45.

[Sung87] Sung,T. and Hu,Y. - *Parallel VLSI Implementation of the Kalman Filter*. IEEE Transactions on Aerospace and Electronic Systems; Vol. 23, No. 2, March 1987, pp215-24.

[Tan88] Tan,J. and Kyriakopulos,N. - *Implementation of Tracking Kalman Filter on a Digital Signal Processor*. IEEE Transactions on Industrial Electronics; Vol. 35, No. 1, February 1988, pp126-34.

[Tugnait82] Tugnait,J.K. - *Detection and Estimation for Abruptly Changing Systems*. Automatica, Vol. 18, No. 5, 1982, pp607-615.

[Turner92] Turner,S.P. - *A Parallel Architecture for Flexible Real Time SAR Processing*. Parallel Computers and Transputer Applications; Vol. I and II, October 1992, pp419-34.

[Vacher92] Vacher,P., Barret,I., Gauvrit,M. - *Design of a Tracking Algorithm for an Adanced Air Traffic Control System*. Multitarget-Multisensor Tracking: Applications and Advances, Volume 2. Edited by Yaakov Bar-Shalom. Artech House,Inc. 1992.

[Waltz90] Waltz, E. and Llinas, J. - *Multisensor Data Fusion*. Artech House, Inc. 1990.

[Wheeler67] Wheeler,G.J. - *Radar Fundamentals*. Prentice Hall,Inc. 1967.

[Willems90] Willems, P.Y. - *Aircraft Dynamics for Air Traffic Control* Aircraft Trajectories Computation-Prediction-Control, AGARD AG-301 Vol. 2, May 90.

[Zhou93] Zhou,B. and Bose, N.K. - *Multitarget Tracking in Clutter: Fast Algorithms for Data Association* .IEEE Transactions on Aerospace and Electronic Systems; Vol. 29, No. 2, April 1993, pp352-63.

[Zhou95] Zhou, B. and Bose, N.K. - *An Efficient Algorithm for Data Association in Multitarget Tracking* .IEEE Transactions on Aerospace and Electronic Systems ; Vol. 31, No. 1, January 1995, pp458-68.