Sensor Management and Information Flow Control for Multisensor Multitarget Tracking and Data Fusion

SENSOR MANAGEMENT AND INFORMATION FLOW CONTROL FOR MULTISENSOR MULTITARGET TRACKING AND DATA FUSION

By

D. AKSELROD, B.Sc., M.Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree of

Doctor of Philosophy

McMaster University

DOCTOR OF PHILOSOPHY (2008) MCMASTER UNIVERSITY

(Electrical and Computer Engineering) Hamilton, Ontario, Canada

TITLE:	**Sensor Management and Information Flow Control for Multisensor Multitarget Tracking and Data Fusion**
AUTHOR:	D. Akselrod B.Sc. Technion, Israel Institute of Technology Israel, 1999 M.Sc. Ben-Gurion University Israel, 2002
SUPERVISOR:	Dr. T. Kirubarajan
NUMBER OF PAGES:	xxvii, 175

Abstract

In this thesis, we address the problem of sensor management with particular application to using unmanned aerial vehicles (UAVs) for multitarget tracking. Also, we present a decision based approach for controlling information flow in decentralized multi-target multi-sensor data fusion.

Considering the problem of sensor management for multitarget tracking, we study the problem of decision based control of a group of UAVs carrying out surveillance over a region that includes a number of moving targets. The objective is to maximize the information obtained and to track as many targets as possible with the maximum possible accuracy. Uncertainty in the information obtained by each UAV regarding the location of the ground targets are addressed in the problem formulation. We propose an altered version of a classical Value Iteration algorithm, one of the most commonly used techniques to calculate the optimal policy for Markov Decision Processes (MDPs) based on Dynamic Element Matching (DEM) algorithms. DEM algorithms, widely used for reducing harmonic distortion in Digital-to-Analog converters, are used as a core element in the modified algorithm. We introduce and demonstrate a number of new performance metrics, to verify the effectiveness of an MDP policy, especially useful for quantifying the impact of the modified DEM-based Value Iteration algorithm on an MDP policy. Also, we introduce a multi-level hierarchy of MDPs controlling each of the UAVs. Each level in the hierarchy solves a problem at a different level of abstraction. Simulation results are presented on a representative multisensor-multitarget tracking problem showing a significant improvement in performance compared to the classical algorithm. The proposed method demonstrated robust performance while guaranteeing polynomial computational complexity.

Decentralized multisensor-multitarget tracking has numerous advantages over single-sensor or single-platform tracking. In this thesis, we present a solution for one of the main problems in decentralized tracking, namely, distributed information transfer and fusion among the participating platforms. We present a decision mechanism for collaborative distributed data fusion that provides each platform with the required data for the fusion process while substantially reducing redundancy in the information flow in the overall system. We consider a distributed data fusion system consisting of platforms that are decentralized, heterogenous, and potentially unreliable. The proposed approach, which is based on Markov Decision Processes with introduced hierarchial structure will control the information exchange and data fusion process. The information based objective function is based on the Posterior Cramér-Rao lower bound and constitutes the basis of a reward structure for Markov decision processes which are used, together with decentralized lookup substrate, to control the data fusion process. We analyze three distributed data fusion algorithms — associated measurement fusion, tracklet fusion and track-to-track fusion. The thesis also provides a detailed analysis of communication and computational load in distributed tracking algorithms. Simulation examples demonstrate the operation and the performance results of the system.

In this thesis, we also present the development of a multisensor-multitarget tracking testbed for simulating large-scale distributed scenarios, capable of handling multiple, heterogeneous sensors, targets and data fusion methods.

To my parents, my wife and my children.

Acknowledgements

I would like to thank everybody who have helped and supported me during those years.

I would like to sincerely thank my supervisor, Prof. T. Kirubarajan, for providing me with the opportunity to work in his team, his guidance and support during my research.

I would like to thank Dr. A. Sinha with whom I have worked all these years. I thank Claudia Goldman for her help and collaboration in the area of Markov Decision Processes. I would like to thank my committee members Prof. J. Reilly and Prof. Sirouspour for their guidance and valuable advice during my research. My special thanks to the staff of the department of Electrical Engineering at McMaster University, especially to Cheryl Gies, Helen Jachna, Cosmin Coroiu, and Terry Greenlay. And finally I would like to thank all my friends who have helped and supported me.

Contents

List of Tables

List of Figures

Nomenclature

Acronyms

ACI	Area Coverage Index
AMR	Associated Measurement Report
ATC	air traffic control
Dec-MDP	Decentralized MDP
DEM	Dynamic Element Matching
DFC	Data Fusion Centers
DFS	Data Fusion System
DHT	Distributed Hash Table
EKF	Extended Kalman filter
ESM	electronic support measures
FC	fusion centre
FN	fusion node

FIM	Fisher information matrix
FK	Fixed-kinematic
FP	Fixed-point
GMTI	Ground Moving Target Indicator
IMM	Interactive Multiple Model
IR	Infra Red
KF	Kalman filter
MAX OI	Maximal Opportunity Index
MDP	Markov Decision Process
MOI	Mean Opportunity Index
OI	Opportunity Index
PF	Particle Filter
PCRLB	posterior Cramér-Rao lower bound
pdf	probability density function
RMSE	root mean square error
RTT	round-trip-time
UAV	Unmanned Aerial Vehicle
VI	Value Iteration

Mathematical notations

A	set of MDP actions
a	MDP action
a_k	association vector
$\mathbb{E}(\cdot)$	expectation operator
$c(s)$	cost associated with being in state s
f_k	nonlinear state transition model at time k
F_k	state transition matrix at time k
$\hat{F}_k$	Jacobian of f_k
h_k	nonlinear measurement model at time k
H_k	linear measurement matrix at time k
J	Fisher information matrix
J_X	prior information matrix
$\mathcal{N}$	Gaussian distribution
I	expected information gain
K	space of the keys in
M	number of targets
$N_{meas/scan}$	number of associated measurements per scan
n_{data}	number of bits used to transmit the data
$P_{k\|k}$	covariance of estimate at time k

$P(s_{k+1} \mid s_k, a_k)$	MDP transition probability function
P_D	probability of detection
Q_k	covariance matrix of process noise at time k
$R(s_k, a_k)$	MDP reward function of state s and action a at time k
R_k	covariance matrix of measurement error at time k
$\dot{r}$	target's range rate
s	MDP state
S	set of MDP states
t	time
T	number of targets
T_s	sampling time interval
$V(s)$	MDP value function of state s
W	number of surveillance sectors
$\mathcal{U}[\cdot]$	uniform distribution
$\hat{x}$	estimated target state
x_k	target state at time k
x_k^t	target state of t-th target at time k
z_k	measurement vector at time k
$z_k(j, i)$	j-th measurement at the i-th sensor at time k
$z_k(i)$	measurements of i-th sensor at time k

Z_k	sequence of measurements up to time k
$\delta(\cdot)$	Dirac delta function
γ	MDP discount factor
λ	spatial density of the false alarms
λ_b	spatial density of the new targets
μ_{FA}	probability mass function of number of false alarms
μ_{NT}	probability mass function of number of new targets
π	MDP policy
ν_k	process noise at time k
ω_k	measurement noise at time k
σ	standard deviation of measurement error
Ψ	communication data rate
υ	false alarm distribution
$[\cdot]'$	transpose
$[\cdot]_t$	t-th block diagonal matrix
Δ_α^β	second-order partial derivative operator
∂	partial derivative operator

Chapter 1

Introduction

1.1 Motivation and Contribution of the Thesis

1.1.1 Sensor Management for Multisensor Multitarget Tracking

In this thesis, we consider the problem of sensor management with an application to optimization the information obtained by a number of unmanned aerial vehicles (UAVs) equipped with Ground Moving Target Indicator (GMTI) radars, carrying our surveillance over a region which includes a number of confirmed and suspected targets. The goal is to track both confirmed and potential targets present in the area.

The radars measure the location and movement parameters of the moving ground targets provided that the radars are in a certain vicinity of the targets; the surveillance region is divided into a number of sectors and it is assumed that a radar is able to detect the targets situated within the sector only when the UAV is itself located in the certain vicinity of the sector boundaries. Each UAV has to decide on the most optimal path to follow in order to cover as many targets as possible obtaining the best possible tracking information during its operation at the lowest possible cost. The operation of the UAV incurs a certain cost thus the sector that might contain more targets may eventually be a worse choice than a sector with less targets but located closer to the current location of the sensor or taking into account other considerations. Another issue to handle is that when a UAV decides on a certain move to perform, there is only a certain probability that it will succeed. Probabilities of its moving in wrong direction as a result of external conditions, limited controllability of the sensor movement in exact direction, navigation errors or any other reason can be specified.

Several UAV management algorithms were described in the literature. In [12] a system architecture for the target assignment and coordinated intercept problem was developed. The path-planning problem was solved via a Voronoi diagram and Eppstein's k-best paths algorithm. As an approach to finding an optimal solution for the team as a whole, a decomposition strategy was developed in [63] that lets team-optimal solutions to be calculated in a decentralized manner. A decision and control scheme that creates a model in which the gain is based on maximizing the expected number of targets found given some a priori information was presented in [44].

In that approach a feasible method of cooperation is achieved by considering other vehicles as stochastic elements. The problem of sensor management and coordinated sensor control in decentralized sensing networks was addressed in [40]. It builds on techniques established for the related problem of Decentralized Data Fusion. The methods are based on the use of mutual information gain measures to formulate local and global objective functions for the sensor network. The approach to cooperative control presented in [68] aims to decouple, in part, the central problems of formation maintenance and maneuver management. For this purpose it introduces to the group a virtual body which is a collection of linked, moving reference points. The approach in [45] presents a decentralized on-line control strategy that lets cooperative UAVs to engage multiple targets time-optimally. A UAV placement algorithm described in [81] and [82] proposes a Fisher information based approach for optimal GMTI sensor placement. An information based criterion, is used to select the path of an UAV such that the total information, obtainable by the sensors in the UAVs as a group, corresponding to the detected targets, is maximized. A single-level MDP approach for UAV management was presented in [1]. The criterion used for the policy update provides a robust solution without compromising the precision of the approach. Utilizing the approach of using information based objective function yielded good results when incorporated in the MDP reward structure. In [5] authors presented multi-level hierarchy of MDPs with each level in the hierarchy solving a problem at a different level of abstraction. The problem of sensor selection in multitarget tracking with an unknown associations of measurements to targets, and also with unknown and potentially time-varying number of targets was considered in [90].

In this thesis, we are interested to consider the UAV management problem using a tractable approach. In many solutions presented above a polynomial convergence of the proposed algorithms is not guaranteed or the computational complexity is not presented. The approach proposed in this thesis considers UAV placement task as a goal-oriented decentralized Markov Decision Process (Dec-MDP) with independent transitions and observations, solved by decomposing it to a number of Markov Decision Processes (MDPs). The individual Markov Decision Processes communicate with each other before they update their individual policies. Under certain conditions, our original problem, presented as a Dec-MDP may be decomposed to a number of MDPs which can be solved in polynomial time. Therefore, our approach will scale up in larger environments with larger number of targets better than algorithms with higher complexity. The criterion used for the policy update provides a robust solution without compromising the precision of the approach. Utilizing the approach of using information based objective function yielded good results when incorporated in the MDP reward structure.

Although a single-level MDP sensor management approach is certainly attractive in case of multiple targets as the optimization problem need not to be solved for each one of the targets individually, the decision regarding a UAV trajectory may be suboptimal. The additional proposed solution is multi-level hierarchy of MDPs controlling each of the UAVs. Each level in the hierarchy solves a problem at a different level of abstraction. As a result, a UAV will navigate more precisely in a populated MDP sectors of the higher level as they will be represented by a lower-level MDP process.

However, even using a multi-level MDP structure, the classical solution of MDP introduces a number of drawbacks to the practical problems solved using an MDP framework. Choosing just a single action for the given state out of several possible actions exposes the issue of choosing such action based merely on its yielding the highest value of state and eliminating another possible actions which yield the same or somewhat lower value. Also, because of inevitable noise components present in the system, the action yielding the highest value of state, would not necessary be the one yielding the highest value of state had the noise be eliminated from the system. In addition, choosing a single action for the given state causes a number of problems in practical applications. One of them is loop formation, especially taking into account loops mistakenly formed because of processing noisy observations data on the basis of which the policy was calculated. Another issue is decreasing the potential coverage of the area under surveillance thus reducing the likelihood of discovering new targets. In this thesis, we present a DEM based modification of a classical Value Iteration algorithm to calculate the optimal policy of an MDP. A number of new introduced performance metrics, such as Mean Opportunity Index, Maximal Opportunity Index and Area Coverage Index, verify the effectiveness of an MDP policy, especially for quantifying the impact of the modified DEM-based Value Iteration algorithm on the overall performance. The modified algorithm is applied to control a group of UAVs carrying out surveillance over a region that includes a number of moving targets and demonstrates accuracy improvement in position and velocity RMSE of the targets under surveillance. The proposed method provides robust performance while guaranteeing polynomial computational complexity.

1.1.2 Information Flow Control for Multisensor Multitarget Tracking and Data Fusion

Another problem considered in this thesis is information flow control for multisensor multitarget tracking and data fusion. Distributed tracking has many advantages over single-sensor (or single-platform) architectures. One major advantage is that a more complete, accurate and timely common tracking picture that covers targets beyond the visibility limits of a given platform can be shared among multiple platforms [7, 58, 83]. In addition, increased reliability is a benefit as well [65]. In distributed tracking systems individual sensors may be placed at considerable distances from one another. In this case, we face the problem of transferring significant amounts of data through communication channels that are subject to certain capacity limits, associated costs of data transfers, reliability and security issues.

A number of multisensor tracking systems and data fusion techniques have been introduced in the literature [65, 30, 7, 58, 83]. Several practical distributed tracking algorithms, including distributed track fusion, track fusion using tracklets and distributed composite tracking have been identified [65]. What is common to many of the above is that the data from each platform, which may correspond to a different type of data fusion or data format, are passed to all the other platforms. The assumption that the available data channel capacity will suffice to carry all the required data will not hold in applications that feature a large number of platforms with high volumes of data to be exchanged among them (e.g., a network of airports). A number of works present solutions for distributed sensor networks applications and network-centric data fusion where limited communication capacity is considered. An information graph approach is introduced in [31]. Using the information graph model, common information can be identified and removed to produce the optimal estimate. The approach of using graphical models was also used in [32] to develop distributed fusion algorithms reducing communication. An agent based fusion model is presented in [70], where agents model fusion entities' capabilities, expertise and intentions and perform fusion based on their intentions. They cooperate with each other to extract the relevant information in order to achieve their intentions. Issues associated with distributed multiple target tracking for ad hoc sensor networks are discussed in [33], which examines the applicability of tracking algorithms developed for traditional networks of sensors. A decision fusion rule based on the total number of detections made by local sensors, for wireless sensor networks with a large number of sensors was proposed in [66]. In [34], distributed fusion and communication management algorithms for target identification, where information graphs are used to select fusion architectures that minimize the effect of information double counting due to communication, were presented. Strategies to determine when a fusion agent should send its estimate to another fusion agent were also developed. Techniques to solve data association problems arising in distributed sensing scenarios were presented in [28], which introduced a communication-sensitive message-passing algorithm for achieving near-optimal performance with substantial savings in communication. A comparison of two different approaches for sensor selection for distributed tracking was presented in [22]. An approach for fusing information from diverse sources based on the quality of the source was presented in [97]. An algorithm for determining the quality of sensor data in the fusion process was presented in [19]. An information valuation metric for sensor networks was described in [78].

Our current work considers a distributed data fusion system consisting of platforms that are completely decentralized, independent and heterogenous. The platforms are also assumed unreliable, i.e., they may join or leave the network at any time, are not committed to share the information, and may have unreliable communication channels with limited capacity. When requesting specific information from such a platform we do not know if the requested information will be provided as we have only the statistical characteristics describing the ability of the platform to provide such information. Data transfer from a platform in the system may be interrupted at any moment. Similarly, a refusal to supply the specific data from a platform does not mean that the next request will be refused as well. We are considering a situation where the decisions regarding the data flow of the information in the distributed data fusion system should be made sequentially based on the currently observable state that reflects fully or partially the state of the environment. The result of each decision cannot be fully predicted, but can be accounted for using available statistical information before the next decision is made.

This thesis presents a decision mechanism that provides each platform with the required data for the distributed data fusion process subject to the available channel capacities and reducing redundancy in the information flow in the overall system. The proposed approach, which is based on Markov Decision Processes (MDP) and decentralized lookup substrate (i.e., the way to identify efficiently all the platforms in the distributed network that possess the required information), will control the data fusion and information exchange process based, among the other parameters, on information gain metrics of individual platforms, enhancing the total distributed system's reliability as well as that of each participating platform. The thesis includes a complete solution for the distributed data fusion architecture timely providing participating platforms with specific decisions regarding obtaining information that is needed for the data fusion.

In this thesis, we present three distributed data fusion algorithms — associated measurement fusion, tracklet fusion and track-to-track fusion [84, 37] — to be applied with the MDP-based data fusion system. We compare the performance based on the type of the data fusion used. In track-to-track fusion [26, 61] sensor measurements are used to update the tracks of each of the local trackers. The updated tracks are then communicated to the other platforms based on the pre-defined policy. Since compressed information (tracks, not measurements) is shared, using track-to-track fusion results in suboptimal performance. In addition to that, in the presence of process noise, this technique requires computation of the filter gains at the update instances of the received tracks. This step is required to compute the cross-correlation between the local and received tracks. Hence each local tracker would be required to compute the filter gains for all tracks at all local trackers. This makes the algorithm computationally burdensome. In [61, 9] an approximation technique is presented for the computation of cross-covariance matrices between tracks from different sources, which is used in this work. In the second fusion approach, the local tracks are first updated using the measurements available to the corresponding platform. After a few updates these tracks are decorrelated with the prior information and communicated for fusion to the other platforms. The decorrelated tracks are called tracklets in the literature [41].

The architecture involving tracklets is considered to be a special case of track fusion. Since they are decorrelated from the local tracks, the tracklets can be treated as measurements of the track states. In this case no further computation of the correlation between local tracks and tracklets is required. However, in the presence of process noise, the tracklets communicated by various platforms cannot be totally decorrelated from the local tracks. The degree of the residual correlation depends on maneuvering index [29] of the tracks. In the third fusion architecture, the sensor measurements are associated with the local tracks maintained in individual platforms and only then the associated measurements reports (AMRs) are communicated to the other platforms [8]. This architecture is easier to implement than the track fusion algorithms due to the independence of information from different sources. That stems from the fact that, unlike local tracks, measurements from different local platforms can be considered to be independent of each other. We present convergence and complexity analysis and demonstrate the proposed algorithm using simulation results. The proposed approach does not require any modification to the individual platforms or communication network connecting them beside the ability to fuse external data containing required information such as tracks, tracklets or AMRs with the platform's own data.

1.1.3 Multisensor Multitarget Testbed for Sensor Management and Data Fusion

In this thesis, we also present the development of a multisensor-multitarget tracking testbed for large-scale distributed (or network-centric) scenarios, capable of handling multiple, heterogeneous sensors and data fusion methods in a hierarchical architecture. The objective of the presented work was developing a state-of-the-art distributed estimation, tracking and fusion system testbed with advanced algorithms for multisensor-multitarget surveillance of air, littoral and sea targets. One crucial aspect of defense capability is the surveillance of mobile targets, in the air, ground or sea, within and near the borders. A large number of sensors, for example, radar, ESM or imaging, are used to gather information about the targets of interest. The gathered information or sensor reports are processed in one or more fusion centers by combining the data from multiple sensors. Typically, a realistic surveillance system is both hierarchical and distributed (or network-centric); that is, information fusion is carried out in a number of fusion centers and at different fusion levels (e.g., raw data, state estimates or decisions). In this work we propose to develop algorithms that solve some specific problems in a network-centric surveillance scenario. The proposed testbed will assist in developing

1. different architectures for distributed surveillance with heterogeneous sensors and ways to quantify their performances.
2. algorithms for preprocessing (e.g., registration, debiasing, out-of-sequence measurements, etc.) heterogeneous multisensor data.
3. estimators of multitarget states using distributed, heterogeneous sensor data.
4. fusion algorithms for combining raw sensor data, state estimates and target features in distributed and hierarchical architectures.
5. resource management algorithms in heterogeneous, multisensor scenarios.
6. a simulation environment to implement and evaluate the new algorithms developed in this project.

The proposed work enables a better utilization of existing sensor resources to extract the maximum information available in multisensor data about the targets of interest.

Different aspects of multisensor-multitarget tracking, the process of estimating the states of moving objects using the data from multiple sensors like, for example, radar, ESM or optical, has been handled by a number of researchers (see [7], [10], [20] for theory and applications). Typical applications of multisensor-multitarget tracking are in ground target tracking [55], air traffic control (ATC) [93], coastal monitoring [80], space surveillance [7], radar tracking in the presence of electronic countermeasures [21], [54], image-based tracking of tissue cells [56] and visual tracking [59], to name a few. Most practical systems consist of a number of heterogeneous and asynchronous sensors (i.e., different types of sensors resulting in measurements at different times) that may be geographically distributed. Then the task is to combine or fuse the noisy information from these different sensors and find the best possible state estimates of the targets of interest that are mobile. Multisensor tracking and fusion systems can be broadly categorized as centralized and decentralized (or distributed). In the centralized architecture, all raw data from individual sensors are sent to a single fusion center where they are statistically combined to obtain a single set of track estimates. In a centralized system, which produces the optimal (best) global estimates, the major bottleneck is the communication load to transfer the entire raw data from the sensors to the fusion center. In distributed scenarios multiple airborne or ship-mounted sensors, the centralized architecture may not be practicable [7]. In contrast, in the decentralized architecture, each sensor processes its own data and sends only the resulting estimates (and its covariances or uncertainties) to the fusion center. The fusion center carries out a track-to-track association to combine the estimates from different sensors. In some cases, the fusion center may send its fused estimates back to the individual sensors, in which case one has a centralized fusion system with feedback [7]. In addition the sensor-to-fusion center communication may be done "on-demand", that is, as needed. While the decentralized architecture requires less communication bandwidth, its performance is sub-optimal to that of the centralized architecture even if the track-to-track fusion is done optimally by accounting for all possible errors [26], [27] (results are available only for the case of homogeneous sensors). Simulation studies using homogeneous sensors have validated these results [27]. In a networked environment, as in many practical systems today, the architecture is hierarchical; that is, fusion may be carried out in more than one fusion center and in more than one level (e.g., raw data, estimates and decision) - the basic fusion architectures (centralized and decentralized) are just the building blocks in the overall fusion topology. Distributed sensor networks may require modification to the standard state estimation algorithms to handle the underlying fusion architectures. In addition, the hierarchical nature of a sensor/fusion system has to be taken into account. In [25] a Probabilistic multitarget tracker was proposed for distributed sensor networks. This work modifies the tracker to handle the centralized fusion architecture with multiple homogeneous sensors. In addition, the thesis presents a solution for a particular data association (the process of matching measurements to targets in the presence of measurement origin uncertainty) mechanism. In [39] and [50] a multiple model based estimation algorithm was proposed for multisensor tracking. In [57] a new algorithm for efficient multisensor fusion was proposed for multitarget tracking geographically distributed sensors. Again, [57] dealt with a specific data association mechanism using multidimensional assignment [38], [73]. In tracking a large number of targets using multiple sensors, one important task is sensor resource allocation. The idea is to select the best group, sequence or mode of sensors, with the minimal usage cost, which can be used to accomplish a given surveillance task [36]. Effective sensor resource management enables the best utilization of existing sensor resources while ensuring the accurate handling of the targets of interest in the surveillance region. A sensor resource manager can decide which sensor to use at which time and in which mode or with which parameters in order to track the targets with a given accuracy [21], [53], [54], [94]. Even with a single sensor, for example, a multifunction radar, resource management may be beneficial in selecting revisit time, mode, parameters [54] and even the platform location [14]. Note that the majority of existing resource management algorithms assume homogeneous sensors for sensor resource allocation.

Another important application of the presented testbed, especially for multisensor-multitarget tracking, is the evaluation of various algorithms on realistic scenarios. In [21] a realistic benchmark system was developed for evaluating different algorithms for single target tracking and single (multifunction) sensor resource management. It was later extended to two sensors and two targets [95]. In [27] a homogeneous multisensor-multitarget benchmark problem was used to evaluate tracking and fusion performances without sensor resource management. A realistic simulation or prototype environment for heterogeneous multisensor-mulitarget tracking, sensor resource management and data fusion performance evaluation is highly desirable.

1.2 Organization of the Thesis

This thesis is structured as follows: Chapter 2 presents solution to the problem of sensor management with an application to optimization the information obtained by a number of UAVs. Chapter 3 presents modified Value Iteration algorithm and Dynamic Element Matching based MDP for Distributed Data Fusion management. Chapter 4 present a solution for distributed information transfer and fusion among the platforms participating in distributed data fusion. Chapter 5 present the development of a multisensor-multitarget tracking testbed for simulating large-scale distributed scenarios for multisensor multitarget tracking. Chapter chapt:conclusion concludes the thesis.

1.3 Related Publications

1.3.1 Book chapter

D. Akselrod, M. McDonald, T. Lang, T. Kirubarajan, “Markov Decision Process Based Resource and Information Management for Sensor Networks”, in prepar. to appear in *Sensor Networks: Where Theory Meets Practice*, Ed. G. Ferrari, Springer, 2009.

1.3.2 Journal articles

1. D. Akselrod, A. Sinha and T. Kirubarajan, "Information Flow Control for Collaborative Distributed Data Fusion For Multisensor Multitarget Tracking", after 1st rev., *IEEE Transactions on Systems, Man, and Cybernetics.*

2. D. Akselrod, T. Lang, M. McDonald, and T. Kirubarajan, "Dynamic Element Matching based Hierarchical Sensor Management for Multisensor Multitarget Tracking", submitted to *IEEE Transactions on Aerospace and Electronic Systems.*

1.3.3 Conference publications

1. D. Akselrod and T. Kirubarajan, "Modified value iteration algorithm and dynamic element matching based MDP for distributed sensor management in multitarget tracking", *proc. 11th Intl. Conf. on Information Fusion*, Cologne, Germany, Jun. 30-Jul. 03, 2008

2. D. Akselrod, A. Sinha, and T. Kirubarajan, "Collaborative distributed sensor management and information exchange flow control for multitarget tracking using Markov decision processes", *SPIE Defense and Security Symposium*, Orlando, FL, Mar. 16-20, 2008

3. A. Sinha, D. Akselrod, D. Danu, T. Kirubarajan, M. Farooq, Z. Ding and

D, Brookes, "Network-Centric Mutisensor-Multitarget Tracking Testbed with Results from Real-Scenarios", *The Northern Watch Conference and Exposition on Arctic C4ISR (NWCEA 07)*, Halifax, Nova Scotia, Canada.

4. D. Akselrod, A. Sinha, and T. Kirubarajan, "Collaborative Hierarchical Markov Decision Processes Based Distributed Data Fusion and Collaborative Sensor Management for Multitarget Multisensor Tracking Applications", *IEEE Intl. Conf. Systems, Man, and Cybernetics*, Montreal, Canada, Oct. 7-10, 2007

5. D. Akselrod, A. Sinha, and T. Kirubarajan, "Collaborative distributed sensor management for multitarget tracking using hierarchical Markov decision processes", *SPIE Optics & Photonics Symposium*, San Diego, CA, Aug.. 26-30, 2007

6. D. Akselrod, A. Sinha, and T. Kirubarajan, "Collaborative Distributed Data Fusion Architecture Using Multi-Level Markov Decision Processes", *10th International Conference on Information Fusion*, Quebec, Canada, Jul. 9-12, 2007

7. D. Akselrod, A. Sinha, C. V. Goldman, and T. Kirubarajan, "Efficient Control of Information Flow for Distributed Multisensor Fusion Using Markov Decision Processes", *9th International Conference on Information Fusion*, Florence, Italy, Jul. 10-13, 2006

8. D. Akselrod, A. Sinha, T. Kirubarajan, M. Farooq, and Z. J. Ding, "Network-Centric Multisensor-Multitarget Tracking Testbed Based On Peer-To-Peer Communication," *Canadian Conference on Electrical and Computer Engineering (CCECE2006)*, Ottawa, ON, May 7-10, 2006

9. D. Akselrod, C. V. Goldman, A. Sinha, and T. Kirubarajan, "Collaborative Sensor Management for Multitarget Tracking Using Decentralized Markov Decision Processes", *SPIE Defense and Security Symposium, Orlando*, FL, Apr. 17-21, 2006

10. D. Akselrod, A. Sinha, T. Kirubarajan, M. Farooq, and Z. J. Ding, "A Large-Scale Network-Centric Multisensor-Multitarget Tracking Testbed," *8th ONR/GTRI Workshop on Target Tracking and Sensor Fusion*, Maui, Hawaii, Jul. 12-13, 2005

11. D. Akselrod, A. Sinha, T. Kirubarajan, M. Farooq, and Z. J. Ding, "A Distributed Multisensor-Multitarget Tracking Testbed for Maritime Surveillance", *Proc. SPIE Defense and Security Symposium*, Orlando, FL, Mar. 28-Apr. 1, 2005

Chapter 2

Markov Decision Processes for Sensor Management

In this chapter, we consider the problem of sensor management with particular application to using unmanned aerial vehicles (UAVs) for multitarget tracking. We study the problem of control of a group of UAVs equipped with Ground Moving Target Indicator (GMTI) radars, carrying out surveillance over a region that includes a number of confirmed and suspected moving targets. The objective is to maximize the information obtained and to track as many targets as possible with the maximum possible accuracy.

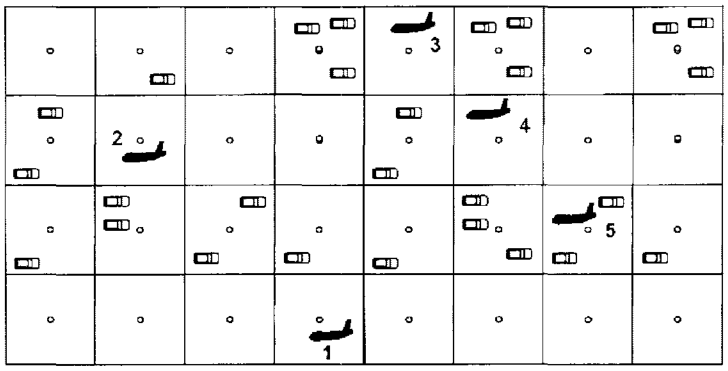


Figure 2.1: Area under surveillance containing a number of targets and sensors (UAVs).

2.1 Decision Based Sensor Management

2.1.1 Problem Formulation

Fig. 2.1 shows the area under surveillance divided into sectors that may contain a number of targets as well as sensors (UAVs) that need to track these targets. A sensor actions are contained within a set of possible actions $A = \{a_i | i = 1, ..., N\}$. A possible action set may include moves one sector to the North, West, East, or South, though it may include much more sophisticated actions.

We assume that with each such action is associated a probability P of reaching the intent of that action which besides the action itself depends on the current state of the sensor, e.g. the physical location of the sensor and on the goal to be reached. It means that the sensor moving to the north could actually have moved to the east.

In each of its possible states the sensor receives a reward $R(s)$. The performance of the sensor will be measured by a sum of rewards for the states visited. Therefore, the performance measure will be based on utility function of the sensor's states history: $U_h([s_0, s_1, ..., s_n])$. The abbreviated objective function is thus formulated as:

$$\pi^* = argmax_{\pi}\mathbb{E}\left[\Sigma_{t=0}^{\infty} U_h([s_0, s_1, ...])|\pi\right] \tag{2.1}$$

where π^* is the policy of actions maximizing the expectation of the utility function U_h of all the sequence of states $[s_0, s_1, ...]$ which resulted from following that policy.

2.1.2 Sensor Management as a Decision Mechanism

Choosing to take a specific move introduces a new decision to be made regarding the one to follow and so on. The decision to take a move in a certain direction may not be carried out as planned because of external conditions or errors in navigation. After this situation is discovered and the current state is updated, the UAV will need to take a new decision taking its current state into account. What is important to note is that in such a case, the optimal configuration is not achievable by a single decision or solution, but is actually a multi-stage process where to each state of the system corresponds the optimal (or more precisely, the most optimal taking into account the information available to the UAV) action that can be taken in the given state.

In the problem we are considering, we do not seek a solution at a particular time only before formulating a new optimization problem to find the solution at the next step. We are rather looking for an approach, or multi-stage allocation process [13], which, once having solved an optimization problem we have formulated, will always know how to provide us with the next step taking into account the outcome of the previous one. It means that we are considering a situation where the decisions should be made sequentially or in a recursive manner. The result of each decision cannot be fully predicted, but can be accounted for using available statistical information before the next decision is made. The objective is to minimize the cost of our actions (or, alternatively, maximize a utility function) [52, 17]. In the following section we show how the formal model of Markov Decision Process can be incorporated to formulate such decision process.

2.1.3 Sensor Management as a Markov Decision Process

Reinforcement learning is the way of achieving a goal by learning from interactions with an environment [52, 88]. An agent is taught how to behave by means of reward and punishment without being exactly informed how to act. In the reinforcement-learning model, an agent interacts with the environment that surrounds it via perception and action. On each step the agent:

- Receives, fully or partially, the current state, s, of the environment.
- Then it decides on an action to take, a which is the agent's output.
- The action it takes modifies the state of the environment

- The value of this transition is passed to the agent via a scalar reinforcement signal, r.

The agent should choose actions that maximize the expectation of sum of values of the reinforcement signal, which can take one of the following forms:

- Finite-horizon model:

$$E(\sum_{t=0}^{N} r_t) \tag{2.2}$$

  This model is applicable in the cases where the exact number of steps is known.

- Infinite horizon discounted model:

$$E(\sum_{t=0}^{\infty} \gamma^t r_t) \tag{2.3}$$

  Any reward received in the future is discounted with the discount coefficient γ.

- Average-reward model (optimizing the long-run average reward):

$$\lim_{h \longrightarrow \infty} E(\frac{1}{h} \sum_{t=0}^{h} r_t) \tag{2.4}$$

The agent does this by hit-and-miss methods, using one of reinforcement learning approaches. After performing an action an agent is notified about the value of the immediate reward resulting from this action, r, as well as its next state, s. The agent is not told which action is better in the current situation. The only feedback the agent receives is the reward and information about the next state. In order for the agent to act optimally, it should collect experience.

In the problem we are considering, we do not seek a solution at a particular time only before formulating a new optimization problem to find the solution at the next step. We are rather looking for an approach, or multi-stage allocation process [13], which, once having solved an optimization problem we have formulated, will always know how to provide us with the next step taking into account the outcome of the previous one. It means that we are considering a situation where the decisions should be made sequentially or in a recursive manner. The result of each decision cannot be fully predicted, but can be accounted for using available statistical information before the next decision is made. The objective is to minimize the cost of our actions (or, alternatively, maximize a utility function) [52, 17].

2.1.4 Markov Decision Processes

A Markov Decision Process (MDP) is a stochastic process controlled by a decision maker. An infinite horizon fully observable MDP is defined by the model $M = < S, A, P, R >$ where S is the finite set of world states, A is the finite set of actions, P is the transition probability function, and R is the real-valued reward function.

- S is the finite set of world states.
- A is the finite set of actions.
- P is the transition probability function. $P(s_{t+1}|s_t, a_t)$ is the probability of moving from state $s_t \in S$ to state $s_{t+1} \in S$ when the agent performs action a. The transition model is stationary. The model is Markov if the transitions are independent of the previous states or actions. That is, $P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_0 ... s_t, a_0 ... a_t)$.
- R is the real-valued reward function. $R(s_{t+1}|s_t, a_t)$ is the award obtained by the system when the agent executes action a_t in state s_t resulting in a transition to state s_{t+1}. In special cases, R can be defined as either $R(s_t, a_t)$ or $R(s_t)$.

In an MDP, a policy π is a mapping from states to actions. Then, $\Pi : S \rightarrow A$. Because of the Markov property, the action taken depends on the current state only, not on any of the previous states. The MDP model is solved using dynamic programming approaches, which have been described in [13, 16] among other works.

2.1.5 Dynamic Programming Solution Approach for Infinite Horizon Discounted MDP

The MDP problem can be solved using the theory of dynamic programming. The basic problem of decision for the infinite horizon case [17] can be expressed as follows.

Given a stationary discrete-time process,

$$s_{t+1} = f(s_t, a_t, w_t), \qquad t = 0, 1, ..., \tag{2.5}$$

where s_t is the state, a_t is the control or action that depends only on the current state s_t, and w_t is the random disturbance, the problem is to find a stationary policy π that maps each $s_t \in S$ to $a_t \in A$, which, given an initial state s_0, maximizes the following value function given by

$$\begin{aligned} V_\pi(s_0) &= \lim_{N \longrightarrow \infty} \mathbb{E}\{ \sum_{k=0}^{N-1} \gamma^k R(s_{t+k}, a_{t+k}) | s_t = s_0 \} \\ &= \lim_{N \longrightarrow \infty} \mathbb{E}\{ R(s_t, a_t) + \gamma \sum_{k=0}^{N-1} \gamma^k R(s_{t+k+1}, a_{t+k+1}) | s_t = s_0 \} \\ &= \mathbb{E}\{ R(s_t, a_t) + \gamma V_\pi(s_{t+1}) | s_t = s_0 \} \\ &= \sum_{s_{t+1} \in S} P(s_{t+1} | s_t = s_0, a_t) \left[R(s_t, a_t) + \gamma V_\pi(s_{t+1}) \right] \end{aligned} \tag{2.6}$$

where γ is a discount factor, $0 < \gamma < 1$. The random disturbance w_t is represented by the transition probability P in an MDP. Then, the optimal value function is defined as

$$V^*(s) = \max_{\pi \in \Pi} V_\pi(s), \quad s \in S \tag{2.7}$$

where the policy π is optimal if $V_\pi(s) = V^*(s)$ for all the states $s \in S$. The optimal V^* is unique and is the solution of the Bellman equation [13, 16]

$$V^*(s_t) = \max_{a \in A} \mathbb{E}\{R(s_t, a_t) + \ \gamma V^*(f(s_t, a_t, w_t))\}, \ for \ all \ s_t \in S \tag{2.8}$$

Discount factor γ expresses the dependence of the value function on current rewards over future rewards. Using the elements of the MDP model defined in Section 2.1.4, (2.8) can be expressed as

$$V^*(s_t) = \max_{a \in A}(R(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_t, a_t, s_{t+1}) V^*(s_{t+1}), \ \forall s_t \in S \tag{2.9}$$

Given the optimal value of the function, the optimal policy is

$$\pi^*(s_t) = \arg\max_{a \in A} R(s_t, a_t) + \gamma \sum_{s_{t+1} \in \ S} P(s_t, a_t, s_{t+1}) V^*(s_{t+1}) \tag{2.10}$$

One of the existing techniques to find the optimal policy is the value iteration method, which is based on the Bellman equation [88].

The iteration can be stopped when the difference between two successive values of state $\varepsilon = |V(s_{t+1}) - V(s_t)|$ is less than a pre-defined error Θ [17]. According to [96], for any state s

$$V^*(x) - V(x) \leq \frac{2\varepsilon}{1-\gamma} \tag{2.11}$$

Given the discount factor γ and the pre-defined error $V^*(x) - V(x)$, we can calculate the difference between two successive values of the state $\varepsilon = |V(s_{t+1}) - V(s_t)|$, which, when reached, will indicate that the algorithm can be stopped. Computational complexity of MDPs was analyzed in [69] and it was shown that, under any of the three cost criteria [69, 62], namely, the expected cost to target, expected discounted cumulative cost, and average expected cost per stage, the problem is P-complete.

2.1.6 Policy Update and Termination Criteria

When a certain policy is calculated, it is assumed that during its execution the conditions and the state of the environment that contributed to the specific policy do not change significantly, except for the changes in the environment as a direct consequence of the actions that are taken. For example, transition probability P and reward R matrices are supposed to reflect the environment all the time during the policy execution. Also, it is assumed that the locations of all information sources have not been changed. Obviously, these assumptions do not hold, at least not for a long period of time, which means that the policy should be re-evaluated from time to time as shown in Figure 2.2 taking into account any new information that might

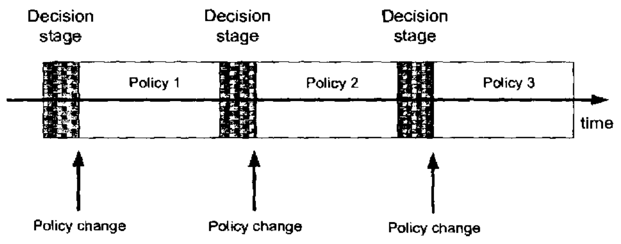


Figure 2.2: Policy update stages.

has arrived.

This approach is similar to sampling an analog signal, where the sampling frequency reflects the dynamics of the signal, specifically its highest harmonic. In our case, the environment change rate will also influence the policy update rate. In some cases, *premature policy termination* will be performed after the threshold is reached in the difference between the assumed environment change rate and the actual one, prior to the planned policy update time. We define a criterion for a premature policy change as exceeding the maximum allowed difference between the state of the world as we observed it at the beginning of the current policy and the updated one during the policy execution.

2.1.7 Solving Decentralized MDPs

The problem of sensor management can be presented as a decentralized operation of a group of decision-makers lacking full observability of the global state of the system which can be modeled as a Decentralized Markov Decision Process (Dec-MDP). It has been shown that solving optimally a Dec-MDP in NEXP-complete [18, 75]. In

[47] special classes of Dec-MDPs were considered. It was shown that decentralized problems with independent transitions and observations and goal-oriented behavior can reduce the complexity to polynomial. Specifically, it was shown that deciding a goal-oriented Dec-MDP with independent transitions and observations, with one global goal state and with uniform cost is P-complete.

In our original decentralized problem the transitions and observations can be considered independent and all the decision makers are bounded by a global goal of covering all the targets obtaining desired tracking metrics. Thus, we can decompose it into a number of local problems presented by the corresponding MDPs. We assume that the global goal of covering all the targets, which binds together all the UAVs, is split into a number of corresponding local goals of covering targets corresponding to the respective MDP process. Our task also complies with *No Benefit to Change Local Goals* property described in [47]. This property is inherent to our problem as we describe the global state as such that all the targets are covered and tracked. Naturally, in our case none of the decision makers would find it beneficial to change the local goal as it will violate the very purpose of its actions.

When a certain policy is calculated, it is assumed that during its execution the conditions and the state of the environment that contributed to the specific policy do not change significantly, except for the changes in the environment as a direct consequence of the action that are taken. For example, transition probability P and reward R matrices are supposed to reflect the environment all the time during the policy execution. Also, it is assumed that the locations of all information sources have not been changed. Obviously, this assumption does not hold, at least not for a long period of time, which means that the policy should be re-evaluated from time to time taking into account any new information that might have arrived. This approach is similar to sampling an analog signal, where the sampling frequency reflects the dynamics of the signal, specifically its highest harmonic. In our case, the environment change rate will also influence the policy update rate. In some cases, *premature policy termination* may be introduced after a threshold is reached in the difference between the assumed environment change rate and the actual one, prior to the planned policy update time. We define a criterion for a premature policy change as the maximum difference between the state of the world as we observed it at the beginning of the current policy and the updated one during the policy execution. Currently the policy update rate is defined according to the following criterion - during the policy execution the fastest target will not move from its location corresponding to the beginning of the policy further than the distance at which the probability of detection of this target is less than 50%. This approach yields good results and robust behavior of the algorithm despite is heuristic nature. As the future work, the authors work on implementation of the approach described in [46] which provides a tractable practical approach for situations in which communication among the decision makers is possible, thus the agents of the goal-oriented Dec-MDP have the ability to share information with each other operating separately between communications. The proposed in [46] approximation method, based on mechanisms for communication, computes off-line the policy of communication among the decision makers.

2.2 MDP-based Structure for Multitarget Tracking

The overall problem of sensor management can be formulated as a Dec-MDP process possessing certain qualities mentioned above which allows us to decompose it to a number of MDPs. We specify below all the elements of these MDPs which are solved optimally to attain the most desirable solution to the decentralized optimization problem. As we have mentioned, the problem solved by each of the composing MDPs is an optimization of the information obtained by a moving sensor carrying our surveillance - search and tracking - over a region which includes a number of confirmed and suspected targets. The overall surveillance region is divided into a number of MDP sectors of which the corresponding MDP is responsible for a certain part. In this thesis, we divide the area of surveillance into 25 equal sectors which we refer to as MDP sectors. Any target which is situated within the boundaries of the area will belong to one of these 25 sectors. All the UAVs act within the boundaries

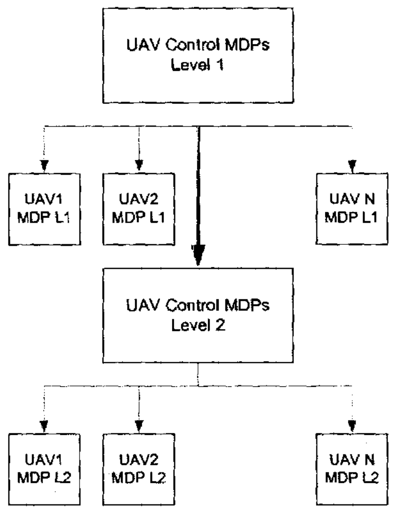


Figure 2.3: UAV control MDP scheme.

of these sectors when each one is responsible for the designated area. The division to MDP sectors is independent of the division to sectors to be scanned by a UAV. It is assumed that the sensor is able to detect the targets situated within the MDP sector only when the sensor is itself located within a certain vicinity from the sector boundaries. The sensor has to decide on most optimal path to follow in order to cover as many targets as possible during its operation at the lowest cost. A separate MDP is designated to a UAV at each of the levels as shown in the UAV control MDP scheme (Figure 2.3).

- Set of states S. The state of the MDP corresponding to each of the UAVs is composed from its location, combined with state of all populated sectors that the UAV is responsible for. A sector is considered populated if it contains at least one target in it. Once a certain sector has been visited by the UAV, its status will be cleared. Once status of all the populated sectors is cleared, the statuses will be asserted again and the UAV will continue with the surveillance.

- Set of actions A. The set A contains all the possible actions that UAV can take in order to do its next move. In our case we have four actions in the set, each one instructs it to move to North, South, West, or East direction to the center of the adjacent sector.

- Transition probabilities $Pr(s_{t+1}|s_t, a_t)$.

  The UAV has a priori information regarding the targets it needs to visit as well as information regarding the additional information related in visiting a specific sector - such as bad weather and other factors which may reduce its chance of survival, chance of getting to the correct location, etc. For example if we know that in the vicinity of a certain sector there blow high winds, the chance of the UAV to arrive to the desired exact location will be significantly lower compared to another region. All such information is eventually translated to the transition probability matrix which specifies the probability of transition to a specific state s_{t+1}, provided the transition is done from another state s_t while performing a certain action a_t.

- Real-valued reward function on states R(s). $R(s)$ contains the value of the immediate reward associated to being in a certain state. $R(s)$ is a tool to specify priorities in getting to specific information source, rather than to others. We can express R as:

$$R(s) = \Theta r(s) - (1 - \Theta)c(s), \quad 0 \leq \Theta \leq 1 \tag{2.12}$$

where $r(s)$ is the revenue associated with being in state s and $c(s)$ is the cost associated with it. Coefficient Θ balances between considerations of importance to reach the state s as well as the revenue this will produce and the considerations of the cost this will incur. For example, in vitally important or mission-critical applications Θ will be higher than when considering commercial usage.

In the above, $r(s)$ and $c(s)$ are expressed as

$$\begin{aligned} r(s) = \sum_{i=1}^{N} r_i^w(s) r_i^{re}(s) &= R_s^{wT} R_s^{re}, \\ r_1^w(s) + \ldots + r_N^w(s) &= 1 \end{aligned} \tag{2.13}$$

$$\begin{aligned} c(s) = \sum_{j=1}^{M} c_j^w(s) c_j^{ce}(s) &= C_s^{wT} C_s^{ce}, \\ c_1^w(s) + \ldots + c_M^w(s) &= 1 \end{aligned} \tag{2.14}$$

where $R_s^{re} = [r_1^{re}(s), \ldots, r_N^{re}(s)]^T$ and $C_s^{ce} = [c_1^{ce}(s), \ldots, c_N^{ce}(s)]^T$ are vectors containing contributing element of the state revenue and state cost, respectively.

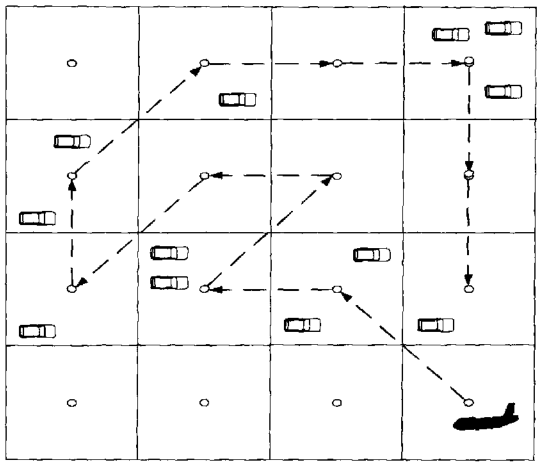


Figure 2.4: The surveillance area is divided into a number of MDP sectors. Each MDP sector may contain a number of targets an informative value of which during one radar scan is represented collectively.

Vectors $R_s^w = [r_1^w(s), ..., r_N^w(s)]^T$ and $C_s^w = [c_1^w(s), ..., c_N^w(s)]^T$ contain weights, which control the influence of R^{re} and C^{ce} elements on the revenue $r(s)$ and the cost $c(s)$, respectively. In general, both cost and revenue weights may depend on the state, but in many applications they can be independent of the system state s.

Fig. 2.4 shows a grid of MDP sectors designated on the surveillance area. When reaching a neighboring MDP sector after taking a previous action, according to the state s, a UAV chooses an action a which corresponds to one of the possible flight directions.

Figure 2.5 shows the pseudo code of the UAV control algorithm.

```
function UAVControlAlgorithm
    1. Initialization: Designate surveillance area. Set initial parameters
    2. UAV movement control.
        (a) If (NoPolicySet Or ReachedPolicyUpdate)
            ChangePolicy:
                i. Build updated list of sectors: (a) populated Or (b) unscanned over time limit
                ii. Calculate updated TransitionProbabilityMatrix $P$
                iii. Calculate updated RewardFunction $R$
                iv. Calculate MDP policy $\pi$
        (b) ExecuteManeuver
                • BoundaryCheck
                  Until(ReachedNewSector)
                  ContinueManeuver according to $a(s)$
                • NewSectorDecisionPoint
                  If(ReachedPolicyUpdate)
                  ChangePolicy
                  Else
                  DetermineNewAction
                  ExecuteManeuver
    3. Scan decision control.
    4. Process obtained measurements.
```

Figure 2.5: UAV control algorithm.

2.3 Expected Information Gain Based Reward Structure of MDP for Sensor Management

2.3.1 Posterior Cramér-Rao Lower Bound

In this work, the authors utilize the approach of using information based objective function [81, 82, 89, 90]. The objective function is based on the Posterior Cramér-Rao lower bound (PCRLB) [92]. Let $\hat{X}(Z)$ be an unbiased estimate of a r-dimensional random state vector X, based on measurement vector Z. The PCRLB for the estimation error is defined to be the inverse of the Fisher Information Matrix (FIM) [91], J, providing a lower bound of the estimation error:

$$C(k) \triangleq \mathbb{E}\{[\hat{X}_k(Z_k) - X_k][\hat{X}_k(Z_k) - X_k]'\} \geq J(k)^{-1} \tag{2.15}$$

From (2.15) it is seen that $C(k) - J(k)$ is a positive semi-definite matrix. The $r \times r$ dimensional Fisher information matrix J is defined as:

$$J \equiv \mathbb{E}\{[\nabla_x \log(p(Z, X))][\nabla_x \log(p(Z, X))]'\} \tag{2.16}$$

where $p(Z, X)$ is the joint probability density function of (Z, X) and $\mathbb{E}$ denotes expectation over (Z, X).

For a general dynamic system

$$X_{k+1} = f[k, X_k] + v_k \tag{2.17}$$

$$Z_k = h[k, X_k] + \omega_k \tag{2.18}$$

with mutually independent white process noise v_k and measurement noise ω_k, the sequence $\{J_k\}$ of posterior information for estimating X_k is given by the following recursion [92]:

$$J(k+1) = D_k^{22} - D_k^{21}(J(k) + D_k^{11})^{-1}D_k^{12} \tag{2.19}$$

where

$$D_k^{11} = \mathbb{E}\{ -\Delta_{X_k}^{X_k} \log p(X_{k+1}|X_k)\} \tag{2.20a}$$

$$D_k^{12} = \mathbb{E}\{ -\Delta_{X_k}^{X_{k+1}} \log p(X_{k+1}|X_k)\} \tag{2.20b}$$

$$D_k^{21} = [D_k^{12}]' \tag{2.20c}$$

$$D_k^{22} = \mathbb{E}\{ -\Delta_{X_{k+1}}^{X_{k+1}} \log p(X_{k+1}|X_k)\} + \mathbb{E}\{ -\Delta_{X_{k+1}}^{X_{k+1}} \log p(Z_{k+1}|X_{k+1})\} \tag{2.20d}$$

and $\Delta_{\Psi}^{\Theta} = \Delta_{\Psi}[\Delta_{\Theta}]'$ is the operator of second-order partial derivative whose (i, j)th term is given by

$$\Delta_{\Psi}^{\Theta}(i,j) = \frac{\partial^2}{\partial\Psi(i)\partial\Theta(j)} \tag{2.21}$$

$\Psi(i)$ and $\Theta(i)$ are the i-th components of vectors Ψ and Θ, respectively.

2.3.2 PCRLB for Multitarget Tracking

Assuming independently moving targets, the linear state equation can be decomposed into M equations (see [89]):

$$X_{k+1}^{j} = F_k^j X_k^j + \nu_k^j \quad j = 1,\ 2, \ldots,\ M \tag{2.22}$$

In the equations above, M is a number of targets, F_k^j is the state transition matrix, ν_k^j is the process noise of target j and Q_k^t is its covariance matrix.

For the linear system (2.22), (2.20) can be expressed as follows [77]:

$$D_k^{11} = F_k' Q_k^{-1} F_k \tag{2.23a}$$

$$D_k^{12} = -F_k' Q_k^{-1} = [D_k^{21}]' \tag{2.23b}$$

$$D_k^{22} = Q_k^{-1} + J_Z(k+1) \tag{2.23c}$$

Using the Matrix Inversion Lemma it can be shown [77] that (2.19) and (2.23) are equivalent to the following recursion:

$$J^j(k+1) \;=\; [Q^j_k + F^j_k J^j(k)^{-1}(F^j_k)']^{-1} + J^j_Z(k+1) = J^j_X(k) + J^j_Z(k+1) \tag{2.24}$$

where $J^j_X(k)$ expresses the prior information and $J^j_Z(k+1)$ the measurement contribution, respectively. Assuming measurement origin uncertainty, the l-th measurement at the s-th sensor at time step k is given by either

$$Z^s_k = [h^s_k]^j(X^l_k) + [\omega^s_k(l)]^j \tag{2.25}$$

if the measurement originated from target j or by

$$Z^s_k = v^s_k(l) \tag{2.26}$$

if it is a false alarm. In the equations above, $[h^s_k]^l$ is a non-linear function, $[\omega^s_k(l)]^j$ is a zero mean Gaussian random variable with covariance R_k and $v^s_k(l)$ is distributed uniformly across the surveillance region. It can be shown (see [49, 48]) that for sensor s

$$J^j_{z_s}(k) = \mathbb{E}\{([H^s_k(X^j_k)])' R^{-1}_k [H^s_k(X^j_k)]\} \tag{2.27}$$

where R_k is the measurement noise covariance and (a,b)-th element of matrix $[H^s_k(X^j_k)]$

is given by:

$$[H_k^s(X_k^j)]_{(a,b)} = \frac{\partial [h_k^s]^j(a)}{\partial x_k^j(b)} \tag{2.28}$$

The expectation $\mathbb{E}[\cdot]$ in (2.27) is with respect to the state $X(k)$. We can see thus that (2.19) is equivalent to:

$$J^j(k+1) = J_X^j(k) + J_Z^j(k+1) = J_X^j(k) + \mathbb{E}\{([H_{k+1}^i(X_k^j)])' R_{k+1}^{-1} [H_{k+1}^i(X_k^j)]\} \tag{2.29}$$

which has the same form as the Extended Kalman Filter (EKF) covariance matrix propagation equation except for the expectation operator. For measurement equation (2.25) in linear form

$$Z_k^s = H_k^s X_k + \omega_k^s \tag{2.30}$$

J_{z_s} is given by

$$J_{z_s}(k) = [H_k^s]' R_k^{-1} H_k^s \tag{2.31}$$

where H_k^s is a measurement matrix Assuming a linear state equation (2.22)

$$J(k+1) = [F_k J(k)^{-1} F_k' + Q_k]^{-1} + [H_k^s]' R_k^{-1} H_k^s \tag{2.32}$$

With $J(k)^{-1} = P_{k|k}$ we see that (2.32) is the covariance update equation ([10]) equivalent to:

$$\begin{aligned} P(k+1|k+1)^{-1} &= P(k+1|k)^{-1} + [H_k^s]' R_k^{-1} H_k^s \\ &= [F_k P(k|k) F_k' + Q_k]^{-1} + [H_k^s]' R_k^{-1} H_k^s \end{aligned} \tag{2.33}$$

PCRLB based scalar performance measure can be based on a number of measures such as trace, determinant or the maximum eigenvalue. In this thesis, we defined determinant as a scalar performance measure of PCRLB which is proportional to the volume of the rectangular region enclosing the minimum achievable covariance ellipsoid. The objective function corresponding to M targets for a given sensor is thus given by:

$$I(k) = \sum_{j=1}^{M} \log |J_j(k|k)| = \sum_j \log |P_j(k|k)^{-1}| \tag{2.34}$$

where $P_j(k|k)$ is the posterior covariance matrix of the state vector corresponding to target j at time k.

The reward associated with a state of a UAV corresponding to the specific MDP sector is expressed as the expected information gain corresponding to this MDP sector. One MDP sector may contain a number of surveillance sectors, i.e. discrete geometric units a number of which that can be covered during one UAV radar scan. Then, the reward of the MDP sector comprising W surveillance sectors is

$$\sum_{f=1}^{W} I_{f\ m,n}^{scan}(k,s) \tag{2.35}$$

$I_{f\ m,n}^{scan}(k,s)$ is the expected information gain from the surveillance sectors to be covered by a UAV. If there are $N_{m,n}(k)$ targets in surveillance sector (m,n) at time step k, then the expected information gain by sensor s from this sector is given by

$$\begin{aligned} I_{m,n}^{scan}(k,s) &= \sum_{j=1}^{N_{m,n}(k)} \log|J_{s,j}(k|k)| - \log|J_{s,j}(k|k-1)| \\ &= \sum_{j=1}^{N_{m,n}(k)} \log|J_{s,j}(k|k-1) + \pi_D(k,s,j)J_{z_s}(k)| - \log|J_{s,j}(k|k-1)| \end{aligned} \tag{2.36}$$

where $J_j(k|k-1)$ is the predicted information matrix and $J_{s,j}(k|k)$ is the updated posterior information matrix corresponding to target j. $\pi_D(k,s,j)$ is the target detection probability expressing the reduction in the target originated measurement contribution to potential information gain ([81]). For a GMTI radar, a target will not be detected if the magnitude of the target's measured range rate falls below a threshold $\dot{r}_{min}$. Therefore, $\pi_D(k,s,j)$ is given by

$$\pi_D(k,s,j) = 1 - P\{|\dot{r}(k,s,j)| < \dot{r}_{min}|\dot{r}(k,s,j|k-1), \sigma_{\dot{r}}(k,s,j|k-1)^2\} \tag{2.37}$$

where $\dot{r}(k,s,j)$, $\dot{r}(k,s,j|k-1)$ and $\sigma_{\dot{r}}(k,s,j|k-1)$ are the measured range rate, predicted range rate and the variance of the predicted range rate respectively. If there are no detected targets in the sector (m,n), $I^{scan}_{m,n}(k,s)$ is given by

$$I^{scan}_{m,n}(k,s) = \log|J_{m,n}(k,s)| - \log|J^0_{m,n}| \tag{2.38}$$

where $J^0_{m,n}$ is the prior information matrix for sector (m,n) and $J_{m,n}(k,s)$ is the expected updated information matrix corresponding to sector (m,n), when scanned by sensor s. $J_{m,n}(k,s)$ is expressed as follows:

$$J_{m,n}(k,s) = J^0_{m,n} + \tilde{\pi}_D(k,s,m,n)\tilde{\pi}_{new}(k,m,n)\tilde{H}(k,s,m,n)'\tilde{R}(k,s,m,n)^{-1}\tilde{H}(k,s,m,n) \tag{2.39}$$

where $\tilde{\pi}_{new}(k,m,n)$ is the probability of detecting a new target in sector m,n at time k and $\tilde{\pi}_D(k,s,m,n)$ is the probability of detection of that target. $\tilde{H}(k,s,m,n)$ and $\tilde{R}(k,s,m,n)$ are measurement and measurement covariance matrices respectively calculated for a target located in the center of the surveillance sector.

2.3.3 Simulation Results

In this section, we present simulation results obtained from a 60 minute simulation involving a number of UAVs and targets under surveillance. The surveillance region dimensions were 42 by 42 km. The simulation included 30 targets and 5 UAVs. The sensors that the UAVs in the simulations were equipped with were of GMTI type. The state of the target j was of the following form:

$$\mathbf{x}^j = [x^j \ \ \dot{x}^j \ \ y^j \ \ \dot{y}^j]' \tag{2.40}$$

where x^j and y^j are 2-D Cartesian coordinates of the target j and $\dot{x}^j$ and $\dot{y}^j$ are its velocity components. We convert the original measurements obtained from the sensor s in the form

$$[r(k,s,j) \ \ \theta(k,s,j) \ \ \dot{r}(k,s,j)]' \tag{2.41}$$

where $r(k,s,j)$, $\theta(k,s,j)$ and $\dot{r}(k,s,j)$ are the range, the azimuth angle and the range rate of the target j supplied by sensor s at time t_k respectively, to the measurement vector of the following form:

$$\mathbf{z}^j = [x^j \ \ y^j \ \ \dot{r}^j]' \tag{2.42}$$

where $\dot{r}^s$ is the speed of target j. The original measurement vector (4.16) is assumed to contain independent additive Gaussian noise with variances denoted as σ_r^2, σ_θ^2 and $\sigma_{\dot{r}}^2$ respectively. The measurement covariance matrix $R(k,s,j)$ corresponding to the converted measurement is given by [7]

$$R(k,s,j) = \begin{bmatrix} R_{1,1} & R_{1,2} & 0 \\ R_{1,2} & R_{2,2} & 0 \\ 0 & 0 & \sigma_{\dot{r}}^2 \end{bmatrix} \tag{2.43}$$

The elements of R_L are

$$R_{1,1} = r^2\sigma_\alpha^2 \sin^2\alpha + \sigma_r^2 \cos^2\alpha \tag{2.44}$$

$$R_{2,2} = r^2\sigma_\alpha^2 \cos^2\alpha + \sigma_r^2 \sin^2\alpha \tag{2.45}$$

$$R_{1,2} = (\sigma_r^2 - r^2\sigma_\alpha^2)\sin\alpha\cos\alpha \tag{2.46}$$

$$\tag{2.47}$$

The overall surveillance region has been divided into a number of clusters equal to the number of the UAVs deployed. The UAVs move at a constant speed of 75 m/s. In the simulation one point track initialization [98] is applied. The track maintenance is performed by measurement to track association, performed by the auction algorithm [15], and track update using a Kalman filter. A white noise acceleration model is assumed for the targets with process noise standard deviation of 1 m/s^2. The measurement noise standard deviations are σ_r = 10m, $\sigma_\theta = 10^{-3}$ rad and $\sigma_{\dot{r}}$ = 1m/s.

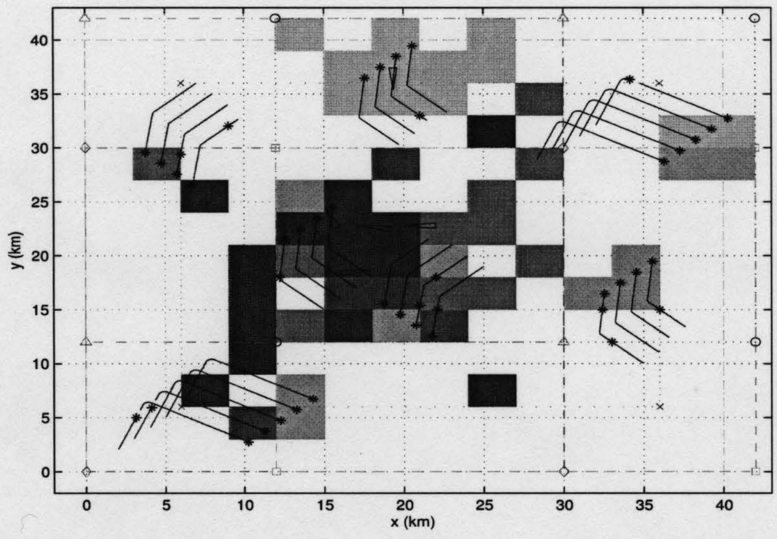


Figure 2.6: The trajectories of the 30 targets. Colored squares designate currently scanned sectors.

Fig. 2.6 shows the trajectories of all targets. Colored squares designate currently scanned sectors. Fig. 2.7 shows the trajectories of the five UAVs and the targets. Each UAV is responsible for a designated area and during the policy execution time tracks the targets in this area. For the demonstrated simulation scenario the total surveillance area was divided into five equal overlapping regions. During the current policy execution UAVs record the information regarding the targets they track. Occasionally, UAVs may spot targets located in the regions responsibility on which belongs to other UAVs. In this case during the policy execution this information is transferred to the corresponding UAV. When performing a policy update, each UAV takes into consideration the targets that have been spotted by it, by other UAVs as well as the targets potentially residing in the areas that have not been scanned

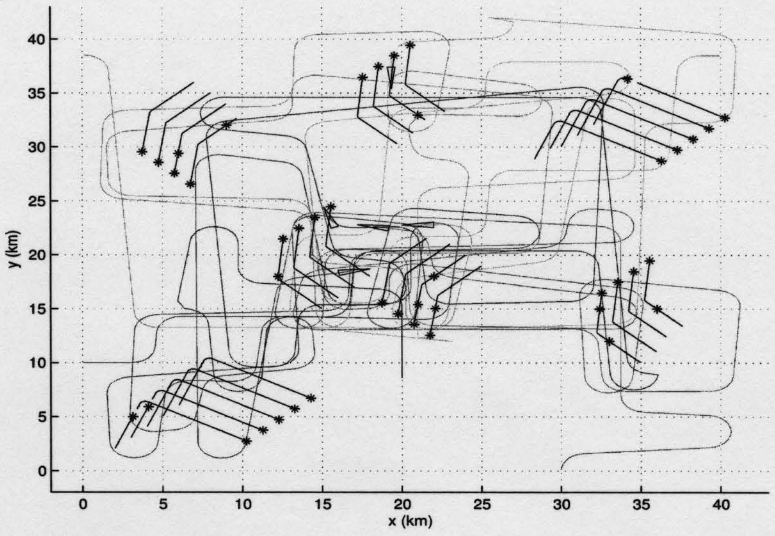


Figure 2.7: The trajectories of the 4 UAVs and the 12 targets between two consecutive policy updates.

during the current policy execution. Then, each UAV forms a list of the sectors, either populated by confirmed targets or unscanned ones and using information based objective function approach described above, reward of each MDP sector is calculated for the next policy. Fig. 2.16 shows the scan decisions for one of the UAVs during four consecutive cycles. Ten sectors were chosen for scan during each cycle. As can be seen from Fig. 2.7, UAVs do not always succeed to move exactly as planned. One of the reasons of it in the current simulation is as follows. When a UAV needs to change its course, it starts a coordinated turn in corresponding direction. As a result, at the end of the coordinated turn, the direction of the UAV may deviate from the planned one. Thanks to the MDP approach any outcome will have the corresponding optimal action to take.

Group no.	Tar no.	Position RMSE (m)	Velocity RMSE (m/s)
1.	1	12.7	1.5
2.	6	7.6	1.2
3.	10	7.8	1.2
4.	14	9.5	1.3
5.	19	12.6	1.5
6.	23	9.8	1.3
7.	27	8.9	1.3

Table 2.1: Performance metrics in single-layer MDP hierarchy

Controlled by a corresponding MDP, a UAV can perform a coordinated turn with angular turn rate of up to 0.05 rad/s. The targets move at different speeds. They include maneuvering targets and move-stop-move ones. The scan time used in this simulation is 5 sec. Fig. 3.10 show four snapshots of the simulation at the times of 1, 5, 10 and 15 min respectively. The UAVs are depicted as triangles with the sharp angles pointing the direction of their movement. The lines represent the tracks and the stars show the last updates position of the detected targets. The colored sectors represent surveillance sectors that have been scanned during the last scan time.

All the targets were belonging to one of the seven different groups. Fig. 2.9 to Fig. 2.15 shows 50 Monte Carlo run position and velocity RMSE results of the tracks corresponding to the targets from groups one to seven, respectively. Table. 2.1 shows the summary of the performance metrics in single-layer MDP hierarchy. The results demonstrate the ability of the proposed approach to maintain an acceptable level of accuracy for different targets.

Fig. 2.16 shows scan decisions for one of the UAVs during four consecutive cycles.

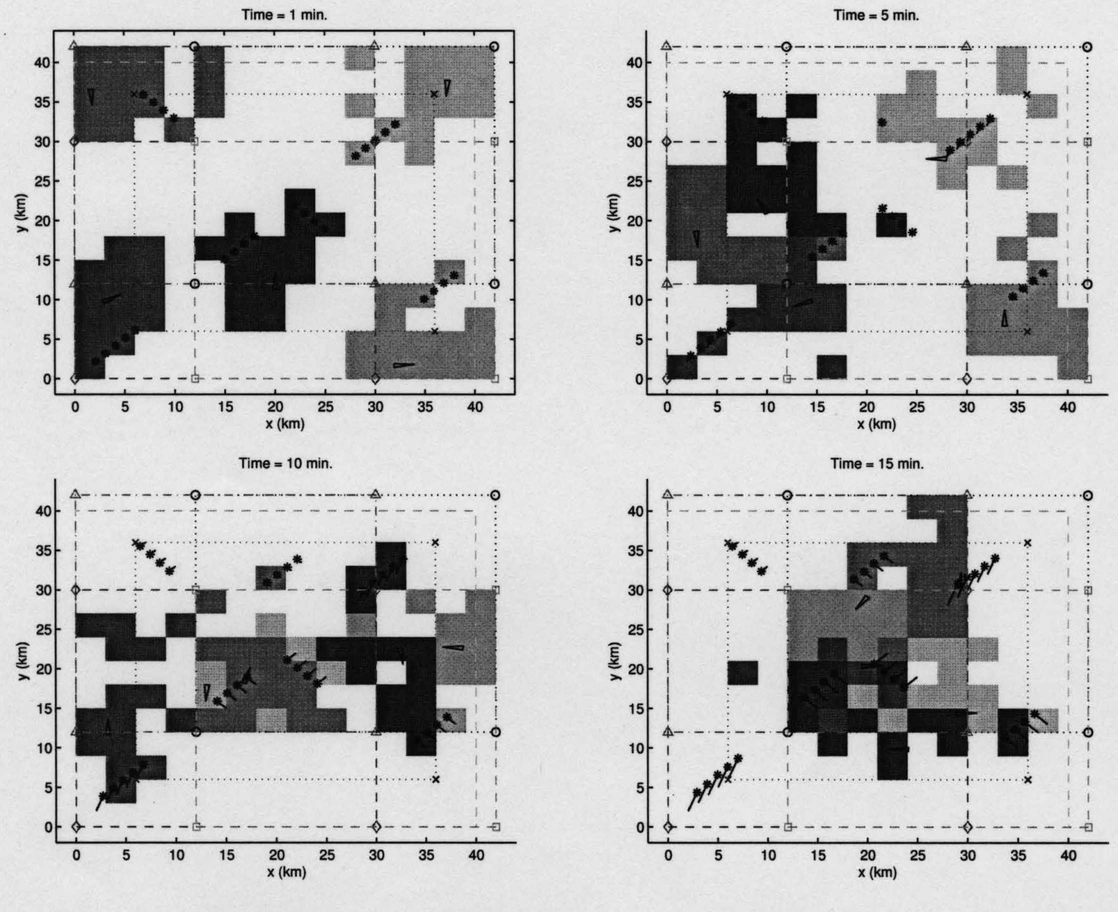


Figure 2.8: Snapshots depicting UAV locations and target tracks. The UAVs are depicted as triangles. The lines represent the tracks and the stars show the last updates position of the detected targets. The colored sectors represent surveillance sectors that have been scanned during the last scan.

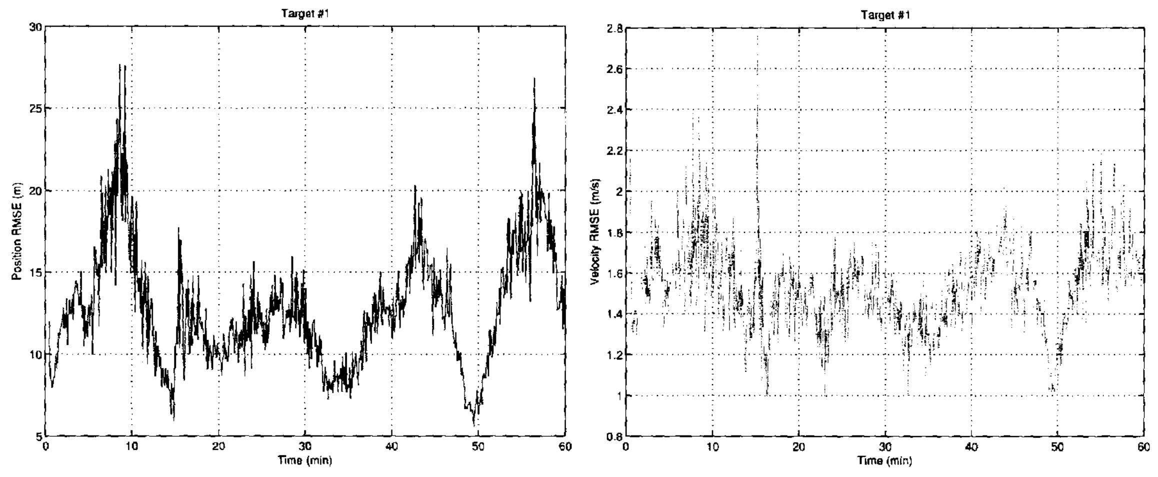


Figure 2.9: The position and velocity RMSE of target 1, group 1.

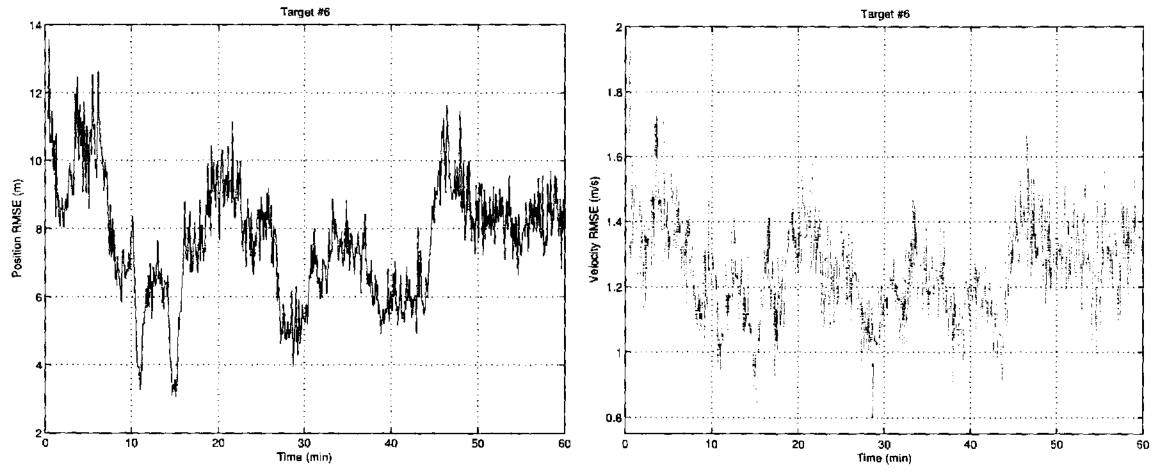


Figure 2.10: The position and velocity RMSE of target 6, group 2.

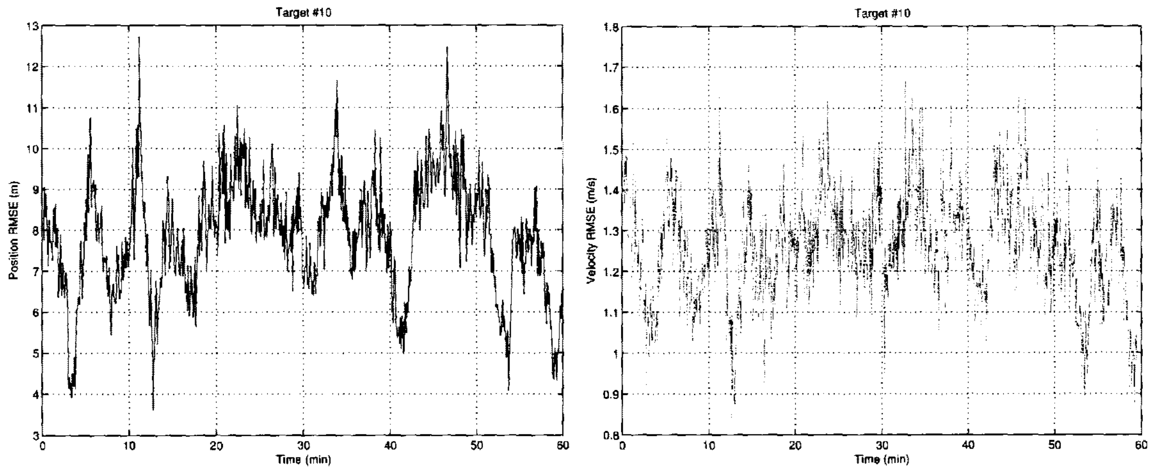


Figure 2.11: The position and velocity RMSE of target 10, group 3.

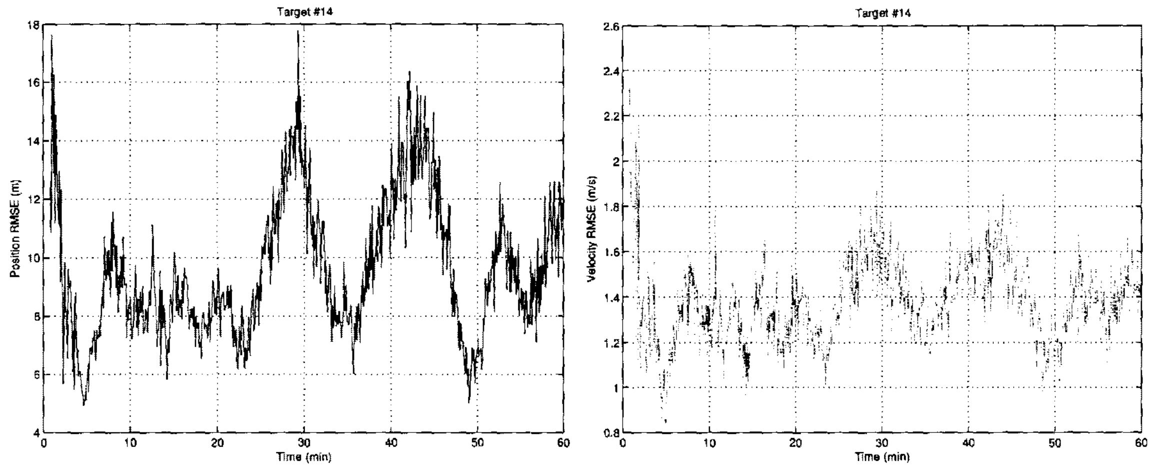


Figure 2.12: The position and velocity RMSE of target 14, group 4.

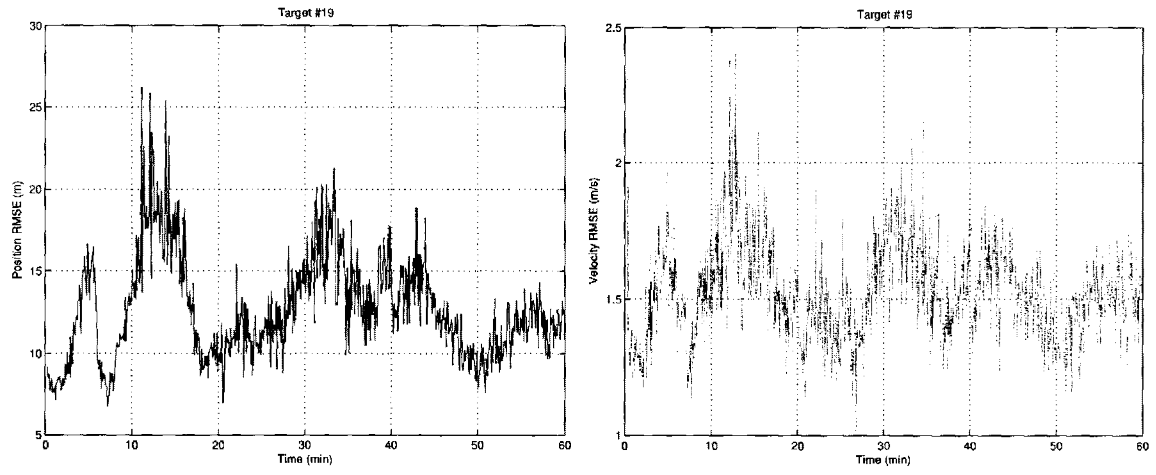


Figure 2.13: The position and velocity RMSE of target 19, group 5.

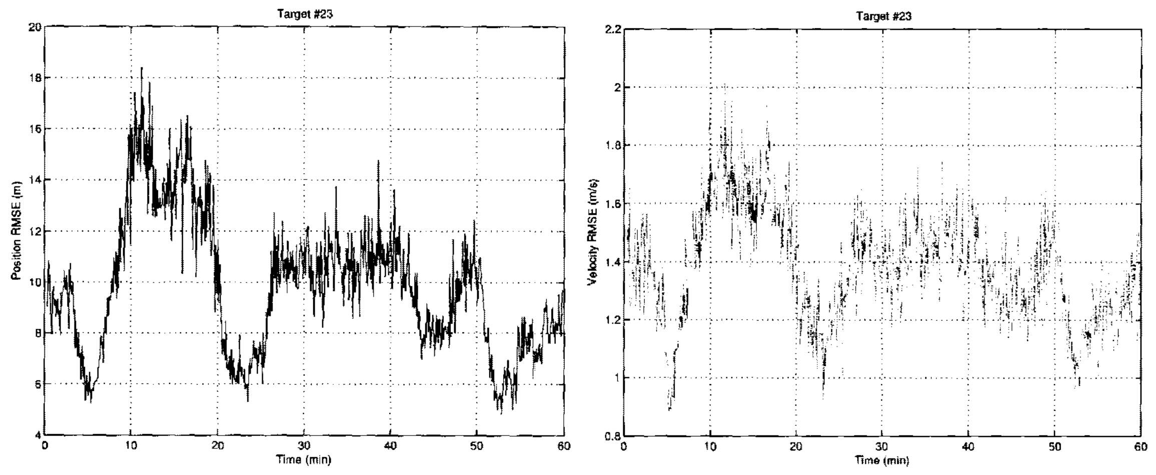


Figure 2.14: The position and velocity RMSE of target 23, group 6.

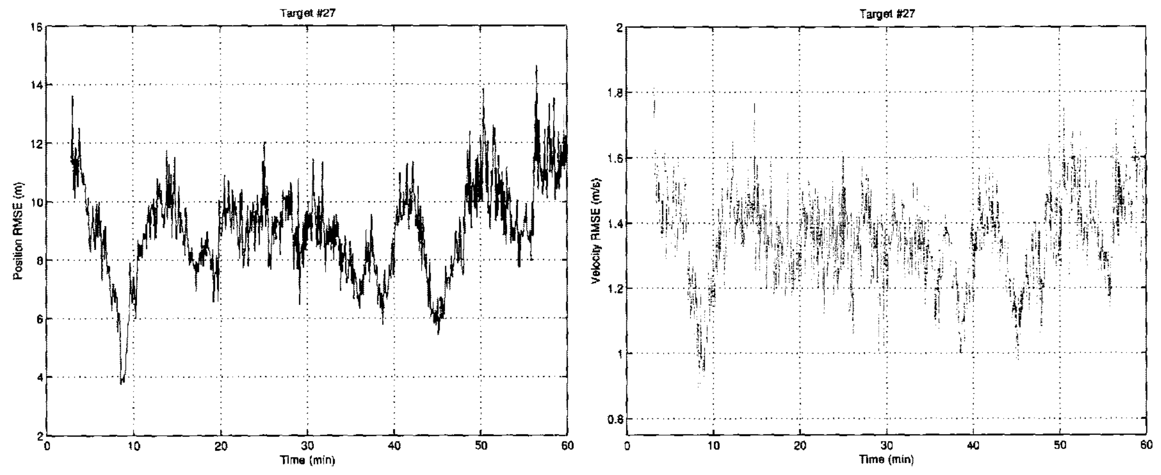


Figure 2.15: The position and velocity RMSE of target 27, group 7.

Ten sectors were chosen for scan during each cycle.

2.4 Multi-level hierarchy of MDPs for sensor management

In a single-level MDP approach the surveillance area is divided into a number of MDP sectors. Each MDP sector may contain a number of targets an informative value of which during one radar scan is represented collectively for that sector as follows:

$$J_z^{scan}(k,s) = \sum_{j=1}^{T(z)} \log |I_{s,j}(k|k)| - \log |I_j(k|k-1)| \tag{2.48}$$

where z is an MDP sector containing $T(z)$ targets at time k. Some MDP sectors may not contain any detected targets. For those sectors, undetected targets are the

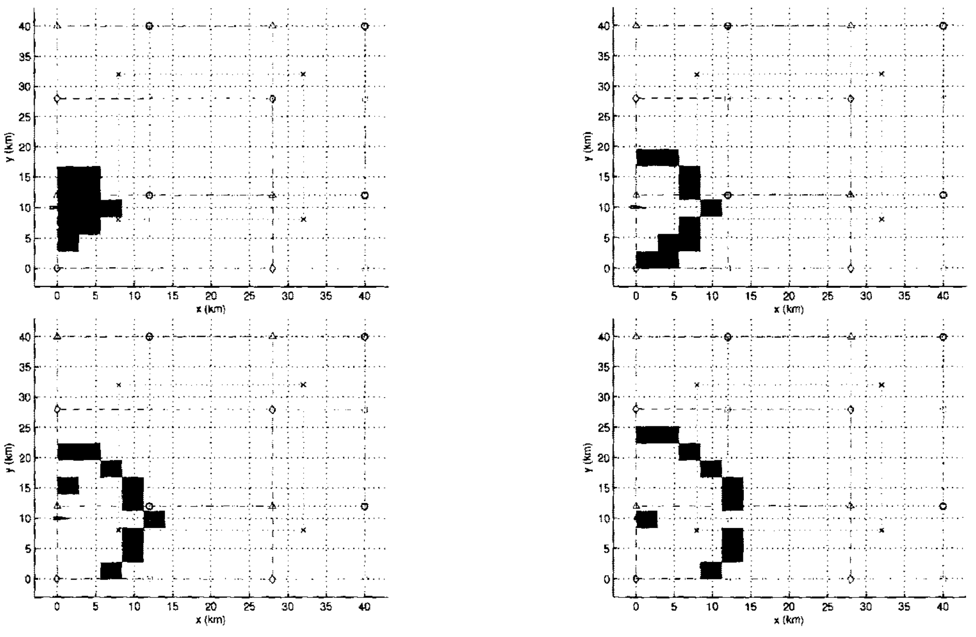


Figure 2.16: Scan decisions for one of the UAVs during four consecutive cycles. Ten sectors were chosen for scan during each cycle.

possible source of information. Both detected and expected targets in an MDP sector will define its reward function $R(s)$ where s a state of a UAV. Combination of the state of the populated MDP sectors as well as the current location of a UAV determines UAV's state. The solution of the Bellman equation describing the corresponding MDP process characterized by its set of states S, actions A, transition probabilities Pr as well as the reward function on states $R(s)$ will provide the UAV with the optimal action $a(s)$ for each of its possible states s. Fig. 2.4 shows a grid of MDP sectors designated on the surveillance area. When reaching a neighboring MDP sector after taking a previous action, according to the state s, a UAV chooses an action a which corresponds to one of the possible flight directions. Although s

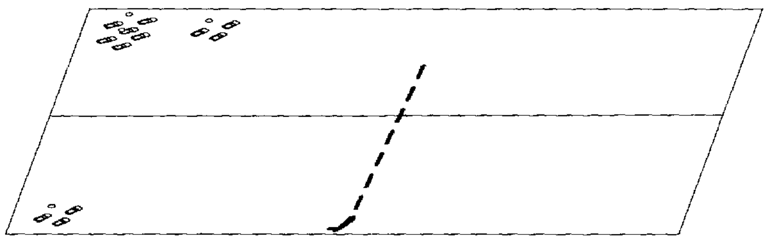


Figure 2.17: UAV trajectory in a single-level MDP process. The decision regarding a UAV trajectory may be suboptimal as some, if not the majority of the targets may be located away from the center of an MDP sector.

single-level MDP based approach presents an attractive approach to sensor management. The computational complexity of solving an MDP is P-complete thus the proposed method is computationally attractive. In this section, we would like to present existing drawbacks to a single-level MDP approach. As we mentioned, after dividing the area under surveillance into MDP sectors, all the targets located in the same MDP sector will be represented collectively. The approach above is certainly attractive in case of multiple targets as the optimization problem need not to be solved for each one of the targets individually; whereas the collective representation will provide adequate precision levels for solving the overall problem. On the other hand, the decision regarding a UAV trajectory may be suboptimal as may be seen in Fig. 2.17. Some, if not the majority of the targets may be located away from the center of an MDP sector. Despite that, as an MDP sector is the lowest discrete level to which the optimization problem is divided, the trajectory will connect centers of two adjacent MDP sectors ignoring the location of the targets within. One solution

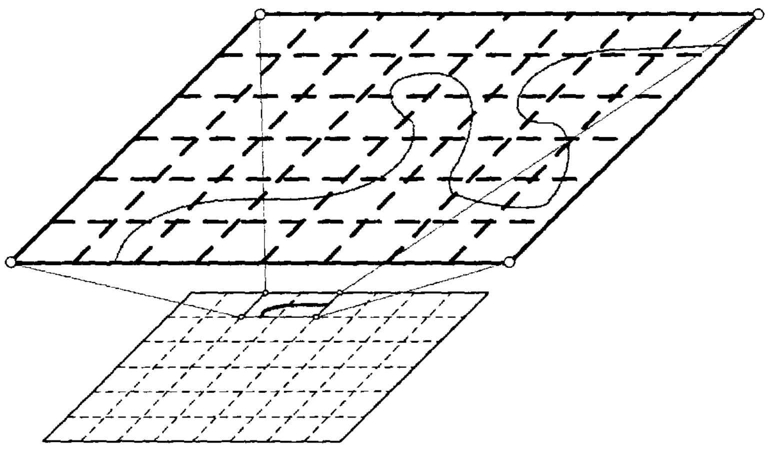


Figure 2.18: Multi-level hierarchy of MDPs. Each MDP sector of the upper level is represented by a number of MDP sectors of the underlying MDP process. Each level in the hierarchy solves a problem at a different level of abstraction.

could be increasing the number of MDP sectors, but that will unnecessary increase the complexity of the problem as many of the MDP sectors may not be so densely populated to justify increasing the number of MDP sectors.

The proposed solution is multi-level hierarchy of MDPs controlling each of the UAVs. Each level in the hierarchy solves a problem at a different level of abstraction. In this thesis, we present two levels of MDP based UAV management. The first (hight) level is exactly the same as is a single-level MDP approach. The second level comprise two adjacent MDP sectors of the higher level providing an additional level of MDP control decision making for UAVs.

Fig. 2.18 shows the MDP sectors division of the surveillance area for the two MDP processes and Figure 2.20 shows the pseudo code of the multi-level UAV control

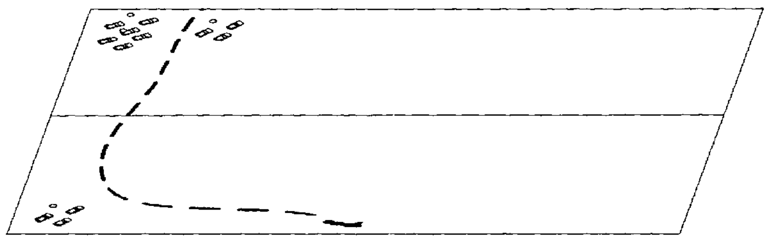


Figure 2.19: UAV trajectory in a multi-level MDP process. A UAV will navigate more precise in a populated MDP sectors of the higher level as they will be represented by a lower-level MDP process

algorithm. As a result of a multi-level MDP hierarchy, a UAV will navigate more precisely in a populated MDP sectors of the higher level as they will be represented by a lower-level MDP process. That is illustrated in Fig. 2.19.

2.4.1 Simulation results

Fig. 2.21 shows the path of the first UAV and the 13 targets when a single-level and a two-level MDP approaches respectively were employed. As can be seen from Fig. 2.21(left), UAVs do not always succeed to move exactly as planned, which is particularly visible in a single-level MDP approach. Thanks to the MDP approach any outcome will have the corresponding optimal action to take. When using multi-level MDP approach, fine-tuning that takes place at lower levels of MDP processes corrects navigation errors at higher levels.

Fig. 2.22 to Fig. 2.24 shows 50 Monte Carlo run position RMSE results of the

```
function MultiLevelUAVControlAlgorithm
    1. Initialization: Designate surveillance area. Set initial parameters
    2. UAV movement control.
        (a) If (NoPolicySetL1 Or ReachedPolicyUpdateL1)
            ChangePolicyL1:
        (b) ExecuteL1Maneuver
            • L1BoundaryCheck
              Until(ReachedNewL1Sector)
              If(L2 enabled And L1SectorIsPopulated)
              UAVControlAlgorithm L2
              Else
              ContinueL1Maneuver according to $a(s)$
            • NewL1SectorDecisionPoint
              If(ReachedPolicyUpdateL1)
              ChangePolicyL1
              Else
              DetermineNewActionL1
              ExecuteManeuverL1
    3. Scan decision control.
    4. Process obtained measurements.
```

Figure 2.20: Multi-level UAV control.

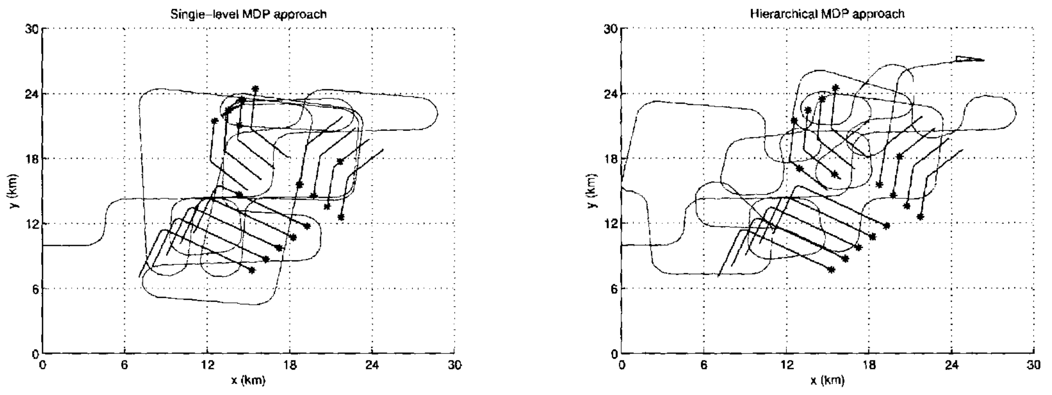


Figure 2.21: The trajectories of the first UAV and the 13 targets. Single-level MDP approach (left) and two-level hierarchical MDP approach (right).

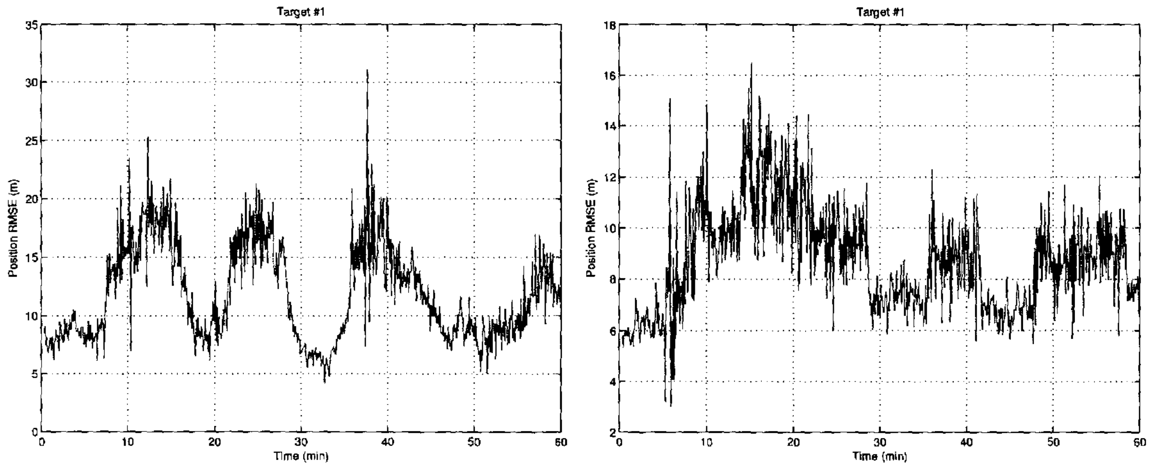


Figure 2.22: The position RMSE of target 1, group 1. The single-level MDP results are on the left and 2-level MDP results are on the right.

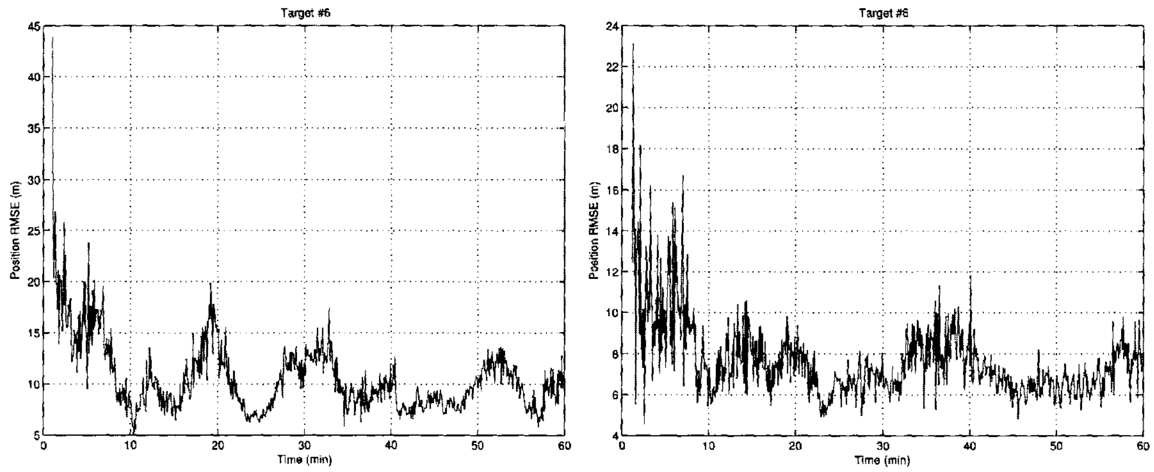


Figure 2.23: The position RMSE of target 6, group 2. The single-level MDP results are on the left and 2-level MDP results are on the right.

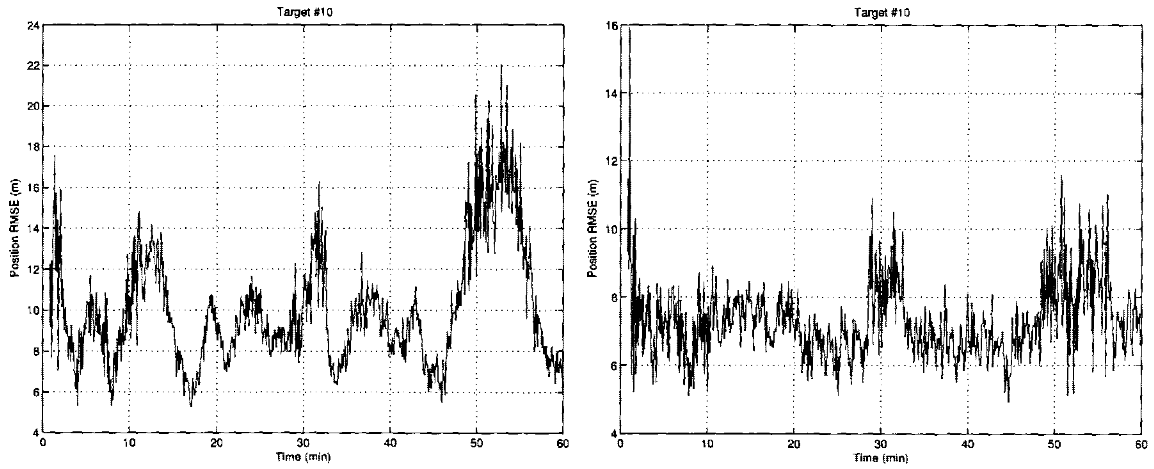


Figure 2.24: The position and velocity RMSE of target 10, group 3.

Tar	Position RMSE (m)	
no.	Single-layer.	Multi-layer.
1.	11.9	8.7
2.	10.6	7.9
3.	10.0	7.2
4.	9.4	7.1
5.	9.6	7.3
6.	10.6	7.6
7.	9.8	7.0
8.	10.2	7.1
9.	11.6	8.2
10.	10.1	7.2
11.	9.3	6.6
12.	9.6	6.9
13.	11.1	8.1
Average Improvement		27.6%

Table 2.2: Performance metrics comparison between single-layer and multi-layer MDP hierarchy.

tracks corresponding to the targets from groups one to three, respectively. The results of a single-layer controlled MDP are shown on the left, the results corresponding to a multiple-level hierarchy are shown on the right. Table. 2.2 shows 50 Monte Carlo run position RMSE results of the tracks corresponding to the targets 1 to 13 from the three groups for both methods. The results demonstrate the ability of the proposed approach to maintain an acceptable level of accuracy for different targets. The two-level MDP approach demonstrated higher level of accuracy than the single-level one.

Chapter 3

Dynamic Element Matching based MDP for Sensor Management

In this chapter, considering the problem of collaborative sensor management and data fusion for multitarget tracking, we propose an altered version of a classical Value Iteration algorithm, one of the most commonly used techniques to calculate the optimal policy for Markov Decision Processes (MDPs). Dynamic Element Matching (DEM) algorithms, widely used for reducing harmonic distortion in Digital-to-Analog converters, are used as a core element in the modified algorithm.

3.1 Dynamic Element Matching Based Modified Value Iteration Algorithm

3.1.1 Finding Optimal Policy of MDP - Drawbacks

As mentioned above, a policy π is a mapping from states to actions when to each state s_t there corresponds a single action a_t. The policy π is optimal only if the value of each of the states $V_{\pi}(s)$ is maximized and is equal to the optimal value of state $V^*(s)$ for all the states $s \in S$. The optimal V^* is defined to be unique and is the solution of the Bellman equation. Therefore it is implied that for any state s there could be just one optimal action $a(s)$.

This approach introduces a number of drawbacks to the practical problems solved using MDP framework. Obviously, choosing just one action for the given state out of several possible actions will expose the issue of choosing such an action based merely on its yielding the highest value of state and eliminating another possible actions which yield the same or similar yet lower values of state. Another aspect is that because of inevitable noise components present in the system, the action yielding the highest value of state, would not necessary be the one yielding the highest value of state had the noise be eliminated from the system. In addition, choosing a single action for the given state may cause the following problems in practical applications:

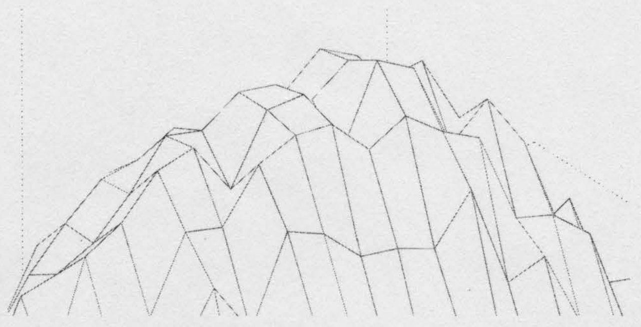


Figure 3.1: Actions corresponding to highest values of different states. In a classical approach only one action having a highest value will be chosen each time.

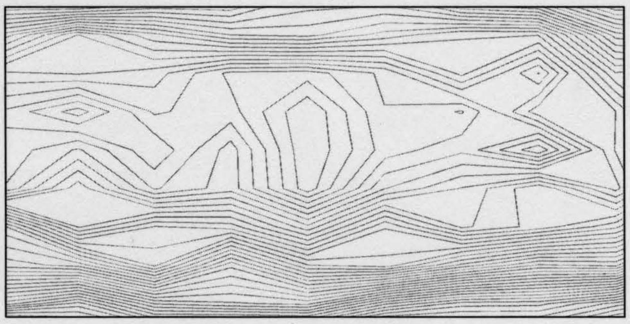


Figure 3.2: Contour plot of actions having highest values of state.

- Loop formation, especially taking into account loops mistakenly formed because of processing noisy observation data on the basis of which the policy is calculated.

- Decreasing the potential coverage thus reducing the likelihood of discovering new targets in the surveillance area. Obviously, by moving via different paths, a UAV can identify a larger number of potential targets.

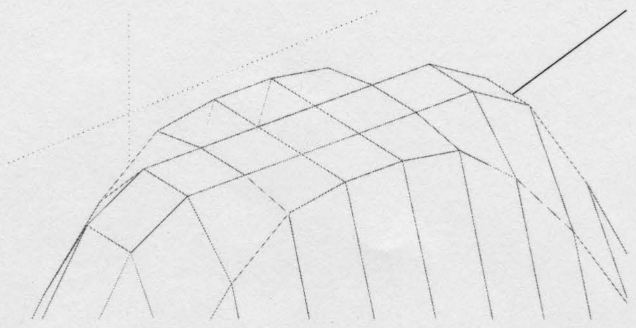


Figure 3.3: A set of actions having highest values of state that have been selected to form the set of potential candidate-actions.

3.1.2 Dynamic Element Matching

An early description of the Dynamic Element Matching principle was given in a patent by Van de Plassche [71] from 1976 as a mean of enhancing performance of current sources. His system is based on a circuit transferring to the output, according to a cyclic permutation, a number of currents, which differ in their values due to the mismatches in the manufacturing process. The resulting current has a value equal to the average value of the currents and a ripple that is formed by the differences between the currents and thus can be subsequently removed by a low pass filter. The proposed DEM algorithm reduced errors caused by mismatches in analog components of current sources by appropriate selection of different current sources every time it was required. Later, Van de Plassche described a high accuracy D/A converter, which employed the DEM principle [72]. Since then numerous patents and publications emerged [24],[60], describing use of the DEM algorithms in current generators design, as well as for achieving high integral linearity and low total harmonic distortion (THD)

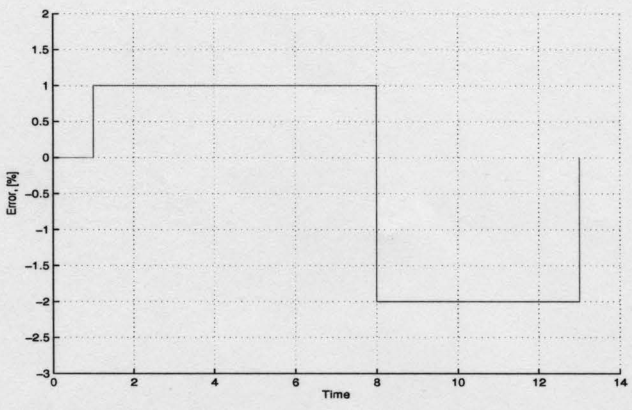


Figure 3.4: Regular sequence of element selection.

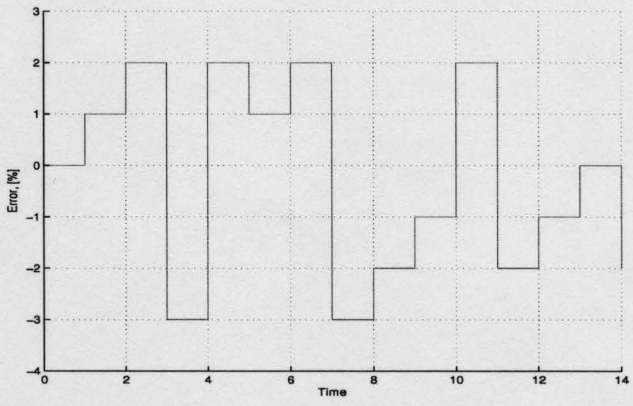


Figure 3.5: Randomized DEM element selection.

in D/A and A/D converters, without requiring precisely matched components. DEM techniques are most commonly used in multi-bit Delta-Sigma converters to achieve the required integral linearity without the use of precise component matching by dynamically rearranging the interconnections of mismatched components. In 2002 there was proposed an approach for incorporating a two-dimensional DEM technique for improving of a mixed signal WTA tracking [6].

There have been proposed many DEM techniques: Dynamic Element randomization [23], Dynamic element Rotation [79], Individual level averaging [60], among many others. All of those techniques similar in dynamically rearranging the interconnections of mismatched components, converting harmonic distortions resulting from elements mismatch into a wide-bandwidth noise. In the case of Dynamic Element randomization, the purpose is to remove the correlation between the mismatch error at one time and the mismatch error at any other time thus converting it into white noise. The relationship between the in-band rms noise divided by full scale and the percentage element mismatch is [67]

$$\sigma\left[\frac{n_{inband}}{V_0 M}\right] = \frac{1}{\sqrt{R}\sqrt{M}}\sigma\left[\frac{\Delta E_i}{E}\right] \tag{3.1}$$

where M is the number of elements (actions in our case), R is an oversampling ratio (number of occurrences of the same state in the policy) and $\frac{\Delta E_i}{E}$ is a fractional element mismatch (value of state mismatch in our case).

3.1.3 Modified Value Iteration Method

The proposed solution to the issues above is introducing a modified value (or policy) iteration algorithm utilizing DEM approach. The algorithm will identify several potential actions yielding highest value of state for a given state. After applying a certain threshold to the actions yielding highest values of state for each particular

```
function ModifiedValueIteration(MDP)
    returns the optimal policy $\pi^*(s)$,
    inputs:  MDP=$<S, A, P, R>$
    $V(s) = 0, \; s \in \mathcal{S}$
    $\Delta_s \leftarrow 0, \; s \in \mathcal{S}$
    loop
        for $s \in \mathcal{S}$
            $v \leftarrow V(s)$
            $V_{s_t} \leftarrow P(s_{t+1}|s_t, a_t)\,[R(s_t, a_t) + \gamma V(s_{t+1})]$
            $V_{1,2,\ldots,m}(s) \leftarrow \max_{a_{1,2,\ldots,m}} \sum_{s_t \in S} V_{s_t}$
            $\Delta \leftarrow \max_a(\Delta, |v - V(s)|)$
        return $\Delta_s$
    untill $max(\Delta_s) < \Theta, \; s \in \mathcal{S}$
    return $\pi^*(s) = DEM(arg \max_{a_{1,2,\ldots,m}}(\Delta, |v - V(s)|))$
```

Figure 3.6: Modified Value iteration algorithm. DEM algorithm is utilized to select dynamically the current action out of several equalized candidates.

state, a group of several candidate actions are identified for each state. Consequently, one of DEM algorithms is applied in order to choose the winner-action for a given state. Each time this state is reached, another winner-action will be chosen based on the DEM method used. The pseudo-code of the modified VI algorithm is shown in Figure 3.6.

To quantify the results and verify the advantage of the DEM-based MDP, we introduce the following metrics:

- Mean Opportunity Index (MOI) — MOI is defined as a mean candidate actions number per state in a policy. Higher MOI will increase a potential for a better performance while maintaining the same precision.

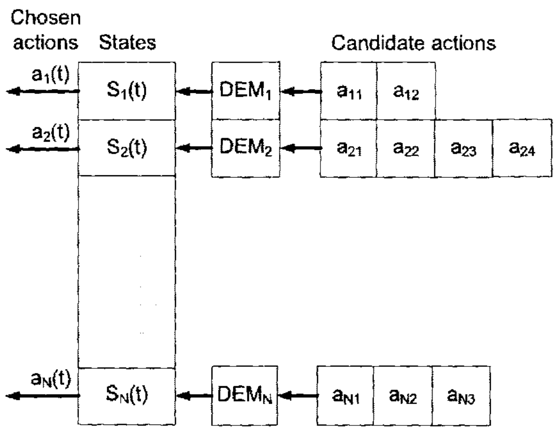


Figure 3.7: Modified MDP solution scheme. Dynamic Element Randomization or Dynamic Element Rotation can be incorporated to select action candidates winners in each of the DEM blocks.

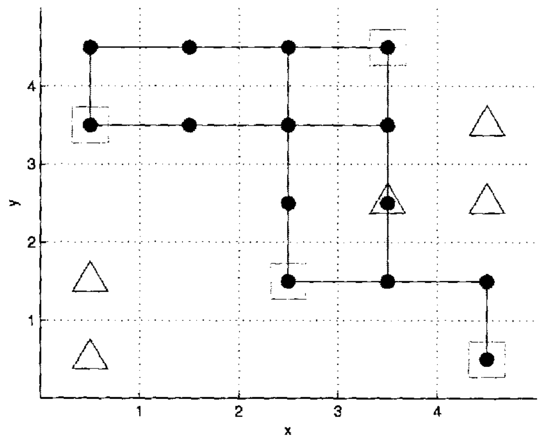


Figure 3.8: Regular MDP is used for path selection. Squares represent sectors populated by targets. Triangles represent sectors with undetected targets not taken into consideration in the current policy. Connected circles show the decision maker path.

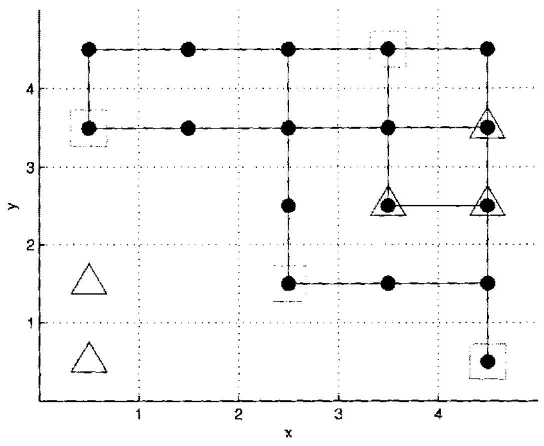


Figure 3.9: Modified DEM based MDP is used for path selection. Area coverage is enhanced thus covering part of previously undetected targets.

- Maximal Opportunity Index (MAX OI) — MAX OI is defined as a maximal candidate actions number per state in a policy.
- State Coverage Index (SCI) — SCI is defined as a mean ratio of a total number of different states reached during an execution of a policy to a total number of states.
- Area Coverage Index (ACI) — ACI is defined as a mean ratio of a total number of different locations reached during an execution of a policy to a total number of location. Higher ACI means that the resulting area coverage during a policy execution is higher. Higher area coverage will result in better detection of new targets.

Table 4.3 demonstrates the numerical results in terms of the metrics above while executing policies solved using a conventional VI algorithm and a randomized DEM

	original VI algorithm	DEM based VI algorithm	Improvement %
MOI	1	1.72	72%
MAX OI	1	4	400%
SCI	0.06	0.08	33.3%
ACI	0.56	0.68	21.4%

Table 3.1: Opportunity Index and State Coverage Index metrics for an original and randomized DEM-based VI algorithms.

based VI algorithm respectively. Figure 3.8 shows the decision maker (UAV) path where regularly solved MDP is used for path selection. Squares represent sectors populated by targets, triangles represent sectors with undetected targets not taken into consideration in the current policy. Figure 3.9 shows the path in a randomized DEM based VI algorithm simulation. Area coverage is increased covering part of previously undetected targets.

3.2 Simulation Results

In this section, we present simulation results obtained from a 60 minute simulation involving four UAVs and ten targets under surveillance. The surveillance region dimensions were 42 by 42 km. The state of the target j was of the form:

$$\mathbf{x}^j = [x^j \quad \dot{x}^j \quad y^j \quad \dot{y}^j]' \tag{3.2}$$

As in the previous simulations, we convert the original measurements obtained from the sensor s in the form

$$[r(k,s,j) \;\; \theta(k,s,j) \;\; \dot{r}(k,s,j)]' \tag{3.3}$$

to the measurement vector of the following form:

$$\mathbf{z}^j = [x^j \;\; y^j \;\; \dot{r}^j]' \tag{3.4}$$

The original measurement vector (4.16) is assumed to contain independent additive Gaussian noise with variances denoted as σ_r^2, σ_θ^2 and $\sigma_{\dot{r}}^2$ respectively. The overall surveillance region has been divided into four clusters, equal to the number of the UAVs deployed. The UAVs move at a constant speed of 40 m/s. The track maintenance is performed by measurement to track association, performed by the auction algorithm [15]. A white noise acceleration model is assumed for the targets with process noise standard deviation of 1.5 m/s^2. The measurement noise standard deviations are σ_r = 15m, $\sigma_\theta = 15^{-3}$ rad and $\sigma_{\dot{r}}$ = 1.5m/s. The scan time used in this simulation is 5 sec. Five sectors were chosen for scan during each cycle. Fig. 3.10 and Fig. 3.11 show the snapshots of the simulation at the times of 10 and 500 sec respectively. Table 3.2 and Table 3.3 show the summary for position and velocity RMSE respectively for all the targets. As can be seen from the results, the modified algorithm demonstrated an average accuracy improvement in position and velocity RMSE of the targets under surveillance increasing State Coverage Index by more than 30 percent resulting in better detection of new targets.

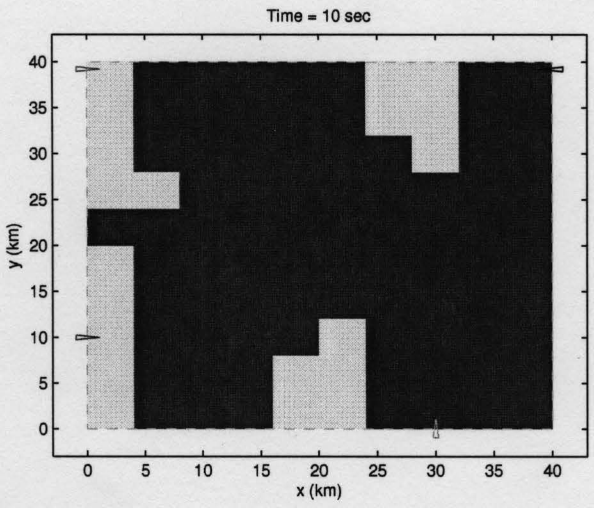


Figure 3.10: Snapshots depicting UAV locations and target tracks at the beginning of the policy.

Tar	Position RMSE (m)		Improvement
no.	Original alg.	DEM based alg.	%
1.	19.2	17.7	8.8%
2.	18.2	14.6	25.0%
3.	13.9	15.4	-9.7%
4.	16.8	17.5	-4.0%
5.	20.2	13.9	45.9%
6.	21.3	14.4	47.2%
7.	14.8	16.5	-10.7%
8.	18.4	11.6	58.1%
9.	17.4	13.0	34.4%
10.	15.9	14.7	8.2%
Average Improvement			20.3%

Table 3.2: Performance metrics summary for position RMSE.

Tar	Velocity RMSE (m)		Improvement
no.	Original alg.	DEM based alg.	%
1.	1.6	1.6	-3.9%
2.	1.7	1.7	0.6%
3.	1.8	1.6	12.9%
4.	1.8	1.5	14.8%
5.	1.3	1.0	26.6%
6.	1.6	1.2	26.2%
7.	1.5	1.3	20.9%
8.	1.8	1.5	18.3%
9.	1.7	1.6	9.2%
10.	1.5	1.7	-9.2%
Average Improvement			11.7%

Table 3.3: Performance metrics summary for velocity RMSE.

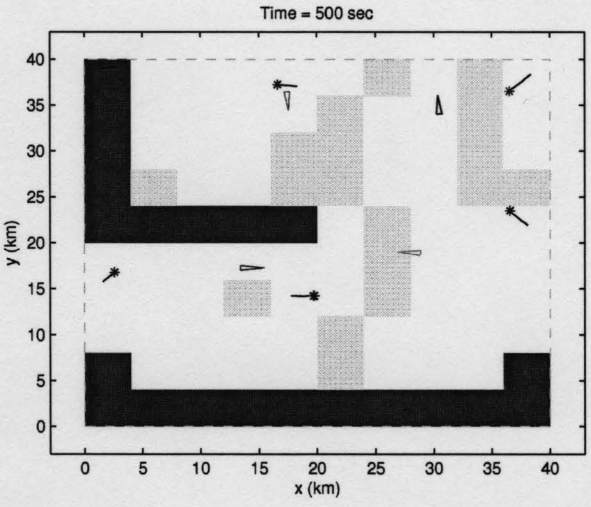


Figure 3.11: UAV locations and targets in the process of policy execution.

Chapter 4

Information Flow Control for Collaborative Distributed Data Fusion For Multisensor Multitarget Tracking

The current chapter presents a solution for controlling the information flow in distributed data fusion architectures of various types — distributed track fusion, track fusion using tracklets, or distributed composite tracking [65]. In these aforementioned approaches, local and remote sensor tracks, tracklets, or AMRs are passed between platforms to be processed in a distributed data fusion process at each platform.

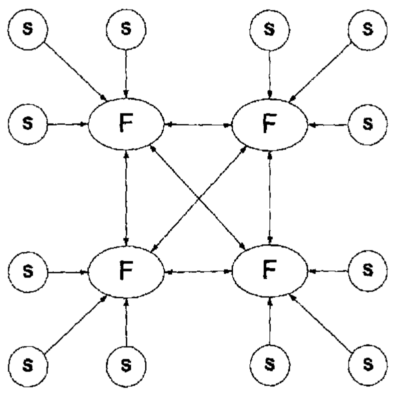


Figure 4.1: Distributed data fusion example. The data from each platform should be passed to all the other platforms.

We can distinguish between the two cases of platforms interchanging data over communication network. Figures 4.1 and 4.2 show two cases of such platforms interchanging data over communication network. In Figure 4.1, the data from each platform should be passed to all the other platforms. Obviously, the same data coming from a certain platform are requested by other platforms as well, which causes redundancy and overloads the communication channels. Moreover, the task of providing tracking information from each node (platform) to every other node may be just infeasible taking into account large number of the nodes in the network and significantly high data rates as opposed to the limitations of the communication channels capacity. Figure 4.2 demonstrates a different approach to the problem. We are considering a node that requests specific information originating from other nodes. The figure depicts a special case in which $node_3$ requests information originating from $node_1$. The most straightforward (and, obviously, not the most optimal) solution

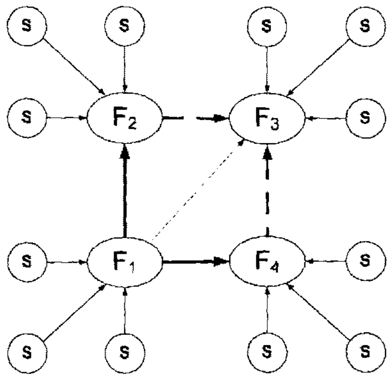


Figure 4.2: Data flow in a distributed data fusion system. Solid lines represent existing data flows. Dashed and dotted lines represent candidate data flow channels to be decided on.

would be requesting this information from the source, i.e., *node*$_1$. But as can be seen from the figure, the information originating from *node*$_1$ is also transmitted to nodes 2 and 4. It should be noted though that the information (e.g., tracks, tracklets or AMRs) transmitted from a node to other nodes may differ in its characteristics and quality. Therefore the required information may be obtained from neighboring nodes as well, as depicted in Figure 4.2, thus eliminating redundancy in the transmitted information, unnecessary load on the communication channels, time overhead in getting the information, higher refusal probability, etc. If we consider *node*$_3$'s request for data originating from *node*$_1$, the decision to be made in this case is which of the platforms — 1, 2 or 4 — should supply the requested information.

We consider two separate tasks — controlling the data fusion

4.1 Decision Mechanism

Choosing to get some specific information from one of the nearest nodes rather than from the information source itself introduces new decisions to be made, like, for example, which of the several available nodes the information should be requested from. For instance, $node_3$ may request $node_4$ to provide information originating from $node_1$, but this may not be the most optimal decision. The communication channel from $node_4$ to $node_3$ may not accommodate the required data rate that removes, at least temporarily, $node_4$ from the possible node candidate list. What is important to note is that in such a case, the optimal configuration is not achievable by a single decision or solution, but is actually a multi-stage process where to each state of the system corresponds the optimal (or more precisely, the most optimal taking into account the information available at the requesting node) action that can be taken in the given state.

4.2 Data Lookup

When discussing the decision making process, we assumed that the information regarding the availability of the required information among the nodes is known. In fact, one of the fundamental problems that has to be addressed in order to apply the aforementioned decision process is in identifying all the nodes that possess the required information. The distributed data fusion architecture does not include any centralized control and consists of the nodes that are decentralized, potentially unreliable and heterogenous. Locating content in such system becomes therefore a complex task. Nodes may join, leave or become inaccessible, but this should not affect the operation of the total distributed system. One solution would be to use one of the available distributed lookup protocols to efficiently identify the nodes that store the desired data. For this purpose we define a space of the keys K. To each node in the distributed system assigned is a key $n_i \in K$. We use a hash function h to map any particular source of information into the space K as well. For example, each sensor will be identified by a corresponding key $s_i \in K$. What is important to note is that the number of keys that a platform possesses equals the total number of sources of information it has, including the ones originating from the platform and the ones that a platform receives from others. The idea is then to assign each node to keep information about the identity of the information sources, values of the hash function of which lie in a certain proximity to the platform's key n_i. The resulting structure is called Distributed Hash Table (DHT). Several architectures of DHTs have been proposed: Chord [87], CAN [76], Kademlia [64], Tapestry [99] among others. Figure 4.3 demonstrates a node look-up process in the Chord DHT architecture [87]. The shortest available look-up time is $O(logN)$, where N is total number of the data sources in the distributed system.

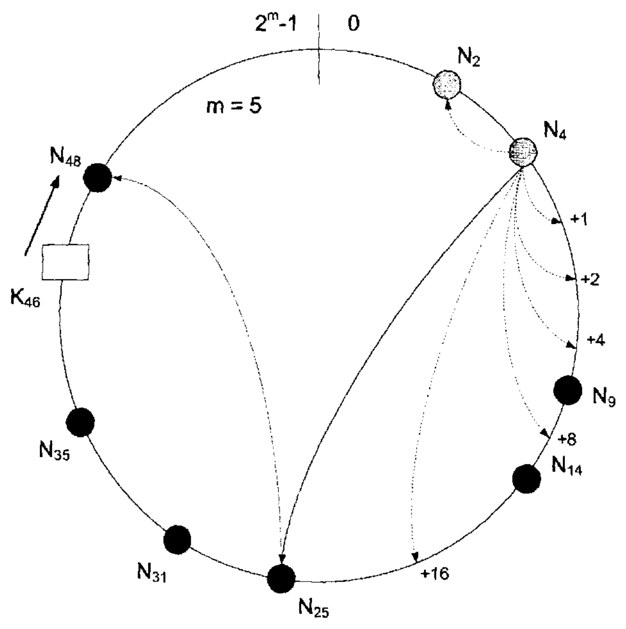


Figure 4.3: Key lookup in Chord.

4.3 MDP Based Multisensor Fusion for Multitarget Tracking

The current section presents a solution for controlling the information flow between the platforms. The tracking data exchanged among those platforms can be of the following types: tracks, tracklets, or Associated Measurement Reports (AMRs). Subsequently, the tracking data that are passed between platforms are processed in a distributed data fusion process. We would like to engage an MDP based decision mechanism, similar to the one used to control UAVs' movement, to provide each platform with the required data for the distributed data fusion process while reducing redundancy in the information flow in the overall system. We will express below

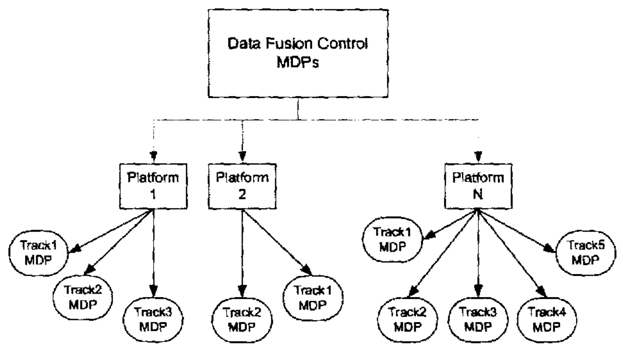


Figure 4.4: Fusion control MDP scheme.

the components of the MDP for information flow control in terms of the parameters of the optimization problem that we are facing. Here, the multisensor fusion problem for multitarget tracking is mapped into a collection of corresponding MDP problems, each one being solved by the corresponding node. Each node will have a corresponding set of MDP parameters S, A, P, R reflecting the optimization problem this node needs to solve. Figure 4.4 shows the fusion control scheme in which a separate MDP is designated to each active track. A pseudo code for a fusion control algorithm is shown in Figure 4.5.

4.3.1 Set of States: S

The state of a node has the general structure depicted in Table 4.1. Each field takes one or more bits of digital information which are enumerated in the table. Below we describe the elements of the table:

- *Original node*: This field specifies the node from which the sought information

```
function FusionControl
  1. Initialization: Set initial parameters
  2. Policy Calculation:
     (a) Update list of initial and confirm tracks - $T$
     (b) For $tr \subset T$
         If (NoPolicySet($tr$) Or ReachedPolicyUpdate($tr$))
         ChangePolicy:
           i. Calculate updated TransitionProbabilityMatrix $P$
          ii. Calculate updated RewardFunction $R$
         iii. Calculate MDP policy $\pi_{tr}$
  3. Data fusion control:
     For each track $tr$ request data from sources defined by $a_{tr}(s)$
```

Figure 4.5: Fusion Control algorithm.

originates.

- *Supplying node*: This field specifies all the nodes that currently receive the information originating from the *Original node.*

- *Data available*: This field indicates whether the requested data from one of the *Supplying nodes* is currently arriving and available.

- *Refusals*: This field contains the total number of refusals from the corresponding *Supplying node.*

Table 4.1: General structure of the node states

Original node	Supplying node	Status	Data bit	Bit #
$platform_{o1}$	$platform_{n1}$	Data avail.	0	0
		Refusals	1	1
			0	2
	$platform_{n2}$	Data avail.	1	3
		Refusals	1	4
			0	5
	$platform_{n3}$	Data avail.	1	6
		Refusals	0	7
			0	8
$platform_{o2}$	$platform_{n4}$	Data avail.	1	9
		Refusals	0	10
			1	11
	$platform_{n5}$	Data avail.	1	12
		Refusals	0	13
			0	14
	$platform_{n6}$	Data avail.	0	15
		Refusals	1	16
			0	17

4.3.2 Set of Actions: A

The set A contains all the possible actions that a node can take in order to specify the requested information sources in its next step of decision making. In our case, we have $n+1$ different actions in the set, expressing a request for information from any of the n nodes posessing the required information and an additional action of not requesting information at all. It should be noted that one of the existing trade-offs is requesting information from more than one source. That increases the probability of a positive outcome (the requested source providing the requested information) but at the same time increases the overall network load which may have a negative impact on the requesting platform itself. Generally, it is acceptable in certain cases to request information from more than one source taking into account the relative unreliability of available sources. Obviously, when there is a large number of information sources, the policy of requesting information from multiple sources may be potentially harmful to both the whole distributed data fusion network and the requesting node itself.

4.3.3 Transition Probabilities: P

The transition probability matrix specifies the probability $P(s_{t+1}|s_t, a_t)$ of transition to a specific state s_{t+1}, provided the transition is done from another state s_t while performing a certain action a_t. The platform requesting data from other platforms has information regarding the holders of the required information as well as the knowledge of other circumstances that may influence successful reception of this information — for example, distributed network channels capacity, current load of the mentioned channels, the load of the nodes that have to supply information. Also, a specific node requesting information may have a priority rating index that may be different for various nodes.

For example, under the assumption that all other parameters such as network channels load, etc., are equal, $platform_i$ will have a higher chance to receive the required information from a certain node only because that node has higher priority rating index for $platform_i$. Other factors may also influence the probability P — for example, weather conditions at a certain node that may influence its chances to successfully transmit the requested information, hardware reliability, survival probability among many others. Also, a node may be able to learn empirically the characteristics of other nodes and, thereby, adjust transition probabilities [86, 85]. All such information is eventually translated into the transition probability matrix. We can see this process as a mapping of all the relevant features of the external world into the transition probability matrix described above.

4.3.4 Real-valued Reward Function on States: R

The vector R contains the values of the immediate rewards associated to being in a certain state. Similar to the reward function for the UAV movement control MDP, we use information based objective function based on the Fisher information measure:

$$J = \sum_j \log |I_j(k|k)| = \sum_j \log |P_j(k|k)^{-1}| \tag{4.1}$$

where $P_j(k|k)$ is the posterior covariance matrix of the state vector corresponding to target j at time k. It is expressed as follows:

$$P_j(k|k)^{-1} = P_j(k|k-1)^{-1} + Y_j(k) \tag{4.2}$$

where $P_j(k|k-1)^{-1}$ is the predicted state information (inverse of the state prediction covariance matrix) and $Y_j(k)$ is the new information that is given by:

$$Y_j(k) = H(k,s,j)'R(k,s,j)^{-1}H(k,s,j) \tag{4.3}$$

where $H(k,s,j)$ is the measurement matrix and $R(k,s,j)$ is the measurement covariance matrix at the time step k corresponding to the sensor s from which the incoming measurement has originated and target j. Then, the expected updated information $I_j(k|k)$ can be expressed as follows:

$$I_j(k|k) = I_j(k-1|k) + H(k,s,j)'R(k,s,j)^{-1}H(k,s,j) \tag{4.4}$$

The reward associated with a measurement arriving from a remote sensor s after being successfully associated with one of the tracks of the platform is therefore expressed as the expected information gain corresponding to sensor from which the measurements originated. If there are N_s AMR's arriving from the same remote sensor s which are associated with the tracks of the receiving platform, then the expected information gain is given by

$$J_{N_s}(k, s) = \sum_{j=1}^{N_s(k)} \log |I_{s,j}(k|k)| - \log |I_j(k|k-1)| \tag{4.5}$$

where $I_j(k|k-1)$ is the predicted information matrix and $I_{s,j}(k|k)$ is the updated information matrix corresponding to target j.

4.4 Distributed Tracking Algorithms Implementing MDP-Based Data Fusion System

In this section, we present three distributed data fusion algorithms — associated measurement fusion, tracklet fusion and track-to-track fusion — to be applied with the MDP-based data fusion system.

4.4.1 Associated Measurements Fusion

In this architecture, the measurements, which are associated with the local tracks, are transmitted. Although the sensors may generate a large number of measurements, particularly in a dense clutter environment, only a few of them are associated with tracks and transmitted. Also, if the positions, measurement covariance and measurement matrix of all sensors connected to the distributed fusion network are available to each platform, then the corresponding quanti- ties are not required to be communicated. Since the measurements are considered to be independent of the state dynamics, this architecture requires much less computation. When using associated measurements fusion each platform performs the following steps: associating (either local or received from another platform) measurements with the current tracks and updating the tracks using the associated measurements. If the associated measurements originated from the same platform they are transmitted to other platforms using the communication policy described above.

4.4.2 Track-to-track Fusion

In this algorithm, we assume that the trackers on all the platforms start with the same information about tracks at time t_0. The tracks are updated by local trackers on each platform using the measurements received from the sensors on that platform up to time t_1. At this time the local tracks are broadcasted and each local tracker updates its tracks using the information received from the other local trackers. Since the tracks in the local trackers are started with the same initial information and the targets go through the same noisy transformation process the tracks obtained by different local trackers are correlated. The computation of the exact cross-covariance matrices in a track-to- track fusion system would hugely increase either computation or communication load [84]. This work uses the approximation proposed in [9, 29]. It assumes that at the time of computation of the cross-covariances between local tracks the covariances of all of the local tracks have already reached the steady state. The terms of the the cross-covariance matrix $P^{\times}$ for a one dimensional tracking problem with state consisting of position and velocity $[x\ \dot{x}]$, are found as

$$p_{ij}^{\times} = \rho_{ij}\sqrt{p_{ii}^{l} p_{jj}^{k}} \quad i, j \in \{1, 2\} \tag{4.6}$$

where p_{ii}^{l} and p_{jj}^{k} denote elements i,j of covariance matrices P^{l} and P^{k} belonging to a certain track originating from platforms l and k; ρ_{ij} are unknown cross-correlation coefficients. There were proposed a number of ways to calculate cross-correlation coefficients [29, 9]. We used the following values of cross-correlation coefficients proposed in [9]: $\rho_{11} = 0.15$,$\rho_{12} = 0.25$,$\rho_{22} = 0.70$. For a 2-D tracking problem with the target state given by $[x\ \dot{x}\ y\ \dot{y}]$, the approximate cross-covariance matrix used in this work is given by

$$P^{\times} = \begin{bmatrix} p^{\times}_{11,x} & p^{\times}_{12,x} & 0 & 0 \\ p^{\times}_{21,x} & p^{\times}_{22,x} & 0 & 0 \\ 0 & 0 & p^{\times}_{11,y} & p^{\times}_{12,y} \\ 0 & 0 & p^{\times}_{21,y} & p^{\times}_{22,y} \end{bmatrix} \tag{4.7}$$

When using track-to-track fusion each platform performs the following steps: associating new measurements to the local tracks and updating them, periodically communicating the tracks to the other local trackers, computing the approximate cross-covariance matrices between tracks, associating the received tracks to local tracks, fusing the tracks received from the other trackers with the local tracks.

Denoting the mth track from platform i by T^i_m and the corresponding state and covariances by $\hat{x}^i_m$ and P^i_m, respectively, for a track set $\{T^1_{k1}, T^2_{k2}, \ldots, T^n_{k_n}\}$ the maximum likelihood estimate of the fused track states is given by

$$x^{fused}_S = (E'\bar{P}_S^{-1}E)^{-1}E'\bar{P}_S^{-1}\bar{x}_S \tag{4.8}$$

where S is the track set $\{T^1_{k1}, T^2_{k1}, \ldots, T^n_{k1}\}$, $E = [I_{n_x}\ I_{n_x}, \ldots, I_{n_x}]$ is $nn_x \times n_x$ matrix and n_x is the dimention of the state vector. Matrices $\bar{x}_S$ and $\bar{P}_S$ are given by (4.9) and (4.10), respectively,

$$\bar{x}_S = \begin{bmatrix} x^1_{k1} \\ \vdots \\ x^n_{k_n} \end{bmatrix} \tag{4.9}$$

$$\bar{P}_S = \begin{bmatrix} P^1_{k_1} & P^{1,2}_{k_1,k_2} & \dots & P^{1,n}_{k_1,k_n} \\ P^{2,1}_{k_2,k_1} & P^2_{k_2} & \dots & P^{2,n}_{k_2,k_n} \\ \vdots & \vdots & \ddots & \vdots \\ P^{n,1}_{k_n,k_1} & P^{n,2}_{k_n,k_2} & \dots & P^n_{k_n} \end{bmatrix} \tag{4.10}$$

The covariance matrix of the fused track is given by

$$P^{fused}_S = (E'\bar{P}^{-1}_S E)^{-1} \tag{4.11}$$

4.4.3 Tracklet Fusion

As tracklets are track data not cross-correlated with the common information among the platforms, for the data fusion purposes they can be treated like measurements and be associated to tracks of other platforms. That solves the problem of data synchronization typical to track-to-track fusion considered above. Since not all the correlation among the versions of the same track maintained by several platforms can be removed, such an approach is only reliable when dealing with targets having small maneuvering index. There were proposed a number of methods for calculating tracklets [41, 42]. In this thesis, we use an algorithm in which a tracklet is the state of a track decorrelated with the state of the same target at the time when the last tracklet was transmitted. For this decorrelation operation the older state requires to be predicted to the time of the more recent one. For tracklet computation, the state of the corresponding track must be observable from the measurements received after the last communication of a tracklet corresponding to the track. That is, at least two measurements are required if the target state vector contains the position and the velocity. Assuming that the last tracklet was transmitted at the time k, the track was last updated at time step $k+i$, and there are enough measurements between time step k and time step $k+n$, the tracklet and the corresponding covariance matrix at time step $k+n$ are given according to [35] by (4.12) and (4.13), respectively. That is,

$$\begin{aligned} x_l(k+n) &= \hat{x}(k+n|k) + P(k+n|k)[P(k+n|k) \\ &- P(k+n|k+i)]^{-1}(\hat{x}(k+n|k+i) - \hat{x}(k+n|k)) \end{aligned} \tag{4.12}$$

$$
\begin{aligned}
P_l(k+n) &= P(k+n|k)[P(k+n|k) - P(k+n|k+i)]^{-1}P(k+n|k) \\
&- P(k+n|k)
\end{aligned}
\tag{4.13}
$$

When using tracklet fusion, each platform performs the following steps: associating new measurements to the local tracks and updating latter, periodically computing the tracklets and communicating them to other platforms, associating the received tracklets to local tracks using 2-D association ([15]) and finally fusing the received tracklets with the local tracks.

4.5 Simulation Results

In this section, we present and discuss the simulated scenarios and the results achieved using two test cases. The first one will apply the suggested approach on three different data fusion methods, namely, associated measurements fusion, track-to-track fusion and tracklets fusion comparing their performance, computational and communication load. The second test-case will present an internal analysis of the decision mechanism.

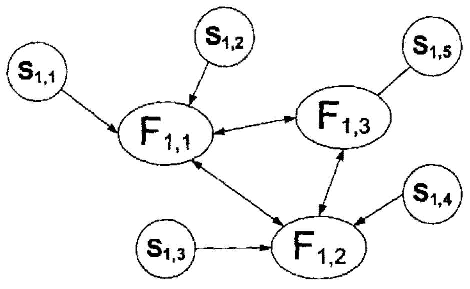


Figure 4.6: The sensors and platforms connectivity scheme in a cluster.

4.5.1 Simulation Results: Test-case A

The following test-case will demonstrate MDP based distributed data fusion approach on an example of associated measurement distributed data fusion system. The simulated system consists of a cluster, which contains 3 platforms and 5 sensors. Each sensor belongs to a specific platform associated with a corresponding tracker. Sensors 1 and 2 are connected to $tracker_1$, sensors 3 and 4 to $tracker_2$ and sensor 5 is connected to $tracker_3$. The sensors and platforms connectivity scheme of the cluster is shown in Figure 4.6. The solid lines designate local communication channels between each of the sensors and the corresponding tracker. These communication channels are assumed dedicated and reliable due to relatively short distance between the sensors and the corresponding tracker. The dashed lines designate communication channels connecting different platforms (trackers) of the cluster. A platform requesting data from another platform is not guaranteed to receive them due to various reasons such as overloaded communication channels, low priority of the requesting platform, etc. The sensors are located at $x_s^i = [65, 155]'$, $[80, 140]'$, $[10, -65]'$, $[30, -40]'$, $[-80, 20]'$ km, respectively. The original measurements are obtained from the sensor s in the form

$$z = [r(k,s,j) \;\; \theta(k,s,j)]' \tag{4.14}$$

where $r(k,s,j)$ and $\theta(k,s,j)$ are the range and the azimuth angle of the target j supplied by sensor s at time t_k, respectively. The measurement vector is assumed to contain independent additive Gaussian noise. The sensor s range r and bearing Θ standard deviations are $[\sigma_r^s,\ \sigma_\Theta^s] = [40,\ 2.5]$, $[35,\ 2]$, $[52.5,\ 3.5]$, $[30,\ 2.5]$, $[45,\ 4.5]$ (in m, mrad, respectively). The sampling intervals of the sensors are 2.5, 3.5, 2, 3, 1.5 s, respectively. The false alarms are uniformly distributed in the coverage areas of the sensors with the number of false alarms Poisson distributed with means of 40,40,100,50,50, respectively. The simulation included two closely spaced targets. The scenario of the two target movements includes several constant velocity stages interleaved with coordinated turn maneuvers performed at rates $|\omega| = 4\ ^\circ/\mathrm{s}$. The initial positions of the targets are $[\xi_j, \eta_j] = [5, 68.6]$ km, $[5, 69.1]$ km, respectively, and the initial targets velocity is 300 m/s. The Figure 4.7 shows the coverage areas of the sensors and the target trajectories. In the simulation, measurements associated with existing tracks, or AMRs, are transmitted between the platforms controlled by internal MDP processes. The transmitted measurements are naturally independent of the tracks of the corresponding platforms. The track maintenance is performed by a 2-D measurement-to-track association, performed by the Auction algorithm [15], and track update is

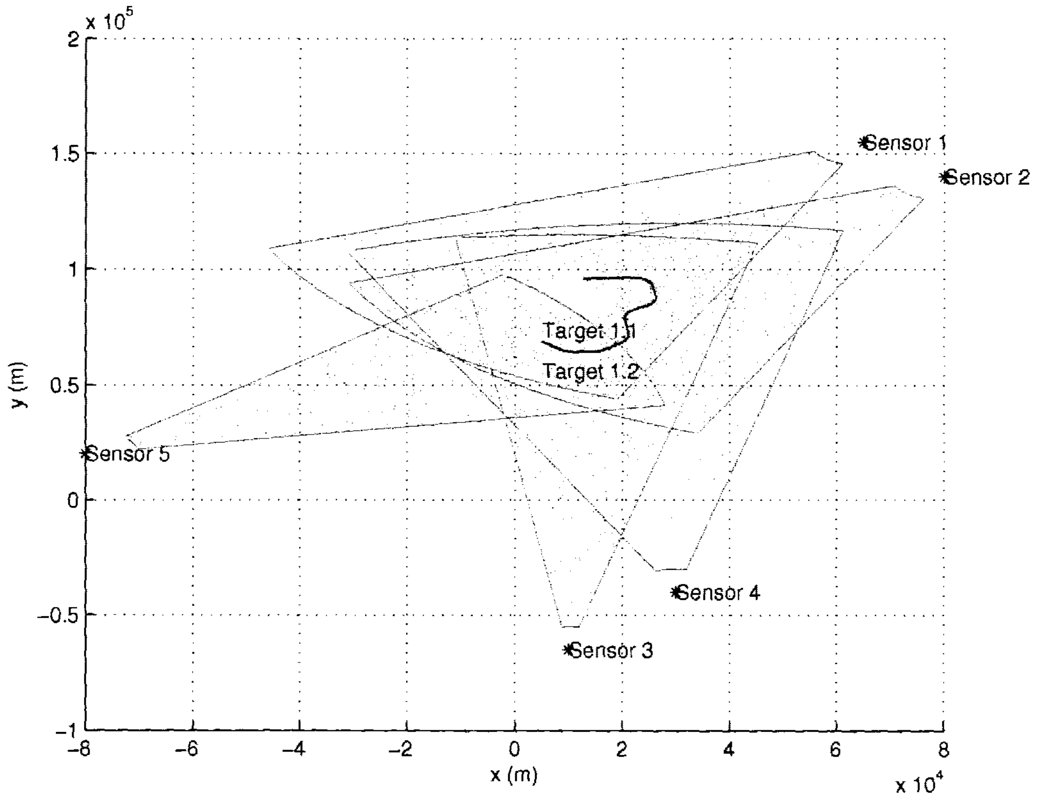


Figure 4.7: Test-case A. Coverage area of the sensors and targets trajectory.

performed using a Kalman filter. A white noise acceleration model is assumed for the targets with process noise standard deviation σ_v of 15 m/s^2. The state of the target j is of the form

$$\mathbf{x}^j = [x^j \quad \dot{x}^j \quad y^j \quad \dot{y}^j]' \tag{4.15}$$

where x^j and y^j are the x and y Cartesian coordinates of the target j and $\dot{x}^j$ and $\dot{y}^j$ its x and y velocity components, respectively. We convert the original measurements obtained from the sensor s in the form

$$z = [r(k,s,j) \quad \theta(k,s,j)]' \tag{4.16}$$

where $r(k,s,j)$, $\theta(k,s,j)$ and $\dot{r}(k,s,j)$ are the range and the azimuth angle of the target j supplied by sensor s at time t_k, respectively, to the measurement vector of the form

$$\mathbf{z}^j = [x^j \quad y^j]' \tag{4.17}$$

using the standard coordinate convertion [7]:

$$\mathbf{z}^j = [r^j \cos\Theta^j \quad r^j \sin\Theta^j]' \tag{4.18}$$

The measurement covariance matrix R corresponding to the converted measurement is given by [7]

$$R_L = \begin{bmatrix} R_L^{11} & R_L^{12} \\ R_L^{12} & R_L^{22} \end{bmatrix} \tag{4.19}$$

The elements of R_L are

$$R_L^{11} = r^2\sigma_\Theta^2 \sin^2\Theta + \sigma_r^2 \cos^2\Theta \tag{4.20}$$

$$R_L^{12} = (\sigma_r^2 - r^2\sigma_\theta^2)\sin\Theta\cos\Theta \tag{4.21}$$

$$R_L^{22} = r^2\sigma_\Theta^2 \cos^2\Theta + \sigma_r^2 \sin^2\Theta \tag{4.22}$$

In the simulation two-point track initialization is applied [7].

Below we show the results obtained from applying the following three communication policies:

1. All the AMRs are shared among all the platforms. In this case, the associated measurements obtained at one platform are transmitted to all other platforms within the same cluster.

2. No information is shared among the platforms.

3. The process of information (AMRs) sharing at each platform is controlled by

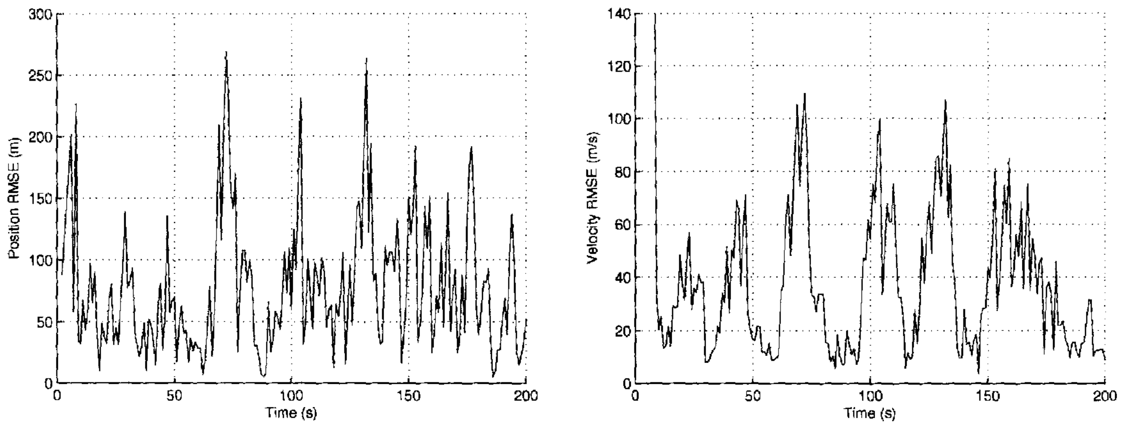
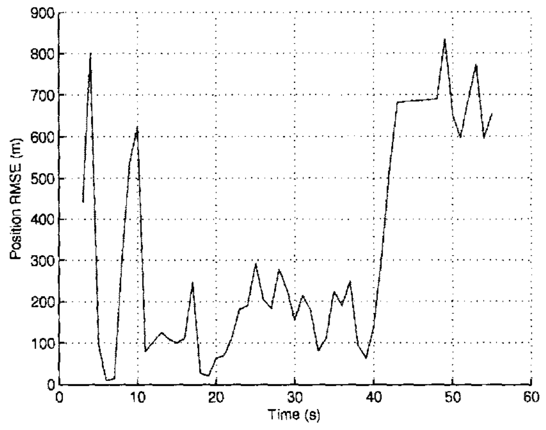


Figure 4.8: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, all AMRs shared.

the dedicated MDP. To each state of the platform corresponds an action of requesting AMR from one or more sensors of the platform or not requesting information at all.

Figure 4.8 shows the position and velocity RMSE of $target_1$ state estimation, respectively, for the first policy when all the AMRs shared among the platforms. Figure 4.9 shows the position and velocity RMSE of $target_1$ state estimation, respectively, for the second policy when no information is shared among the platforms. Tracking data is available only for $t < 55$ s after which the targets come out of the coverage area of the sensors of $platform$ 3. Figure 4.10 shows the position and velocity RMSE of $target_1$ state estimation, respectively, for the 3rd policy when the data fusion is controlled by MDP.



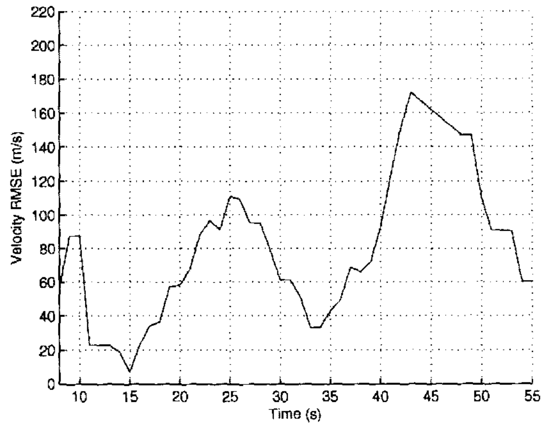
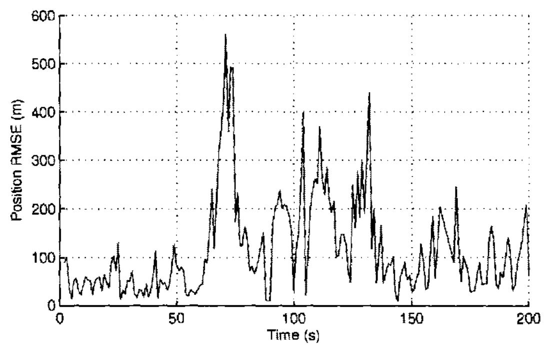


Figure 4.9: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, no information shared. Tracking data is available only for $t < 55$ s after which the targets come out of the coverage area of the sensors of the platform.



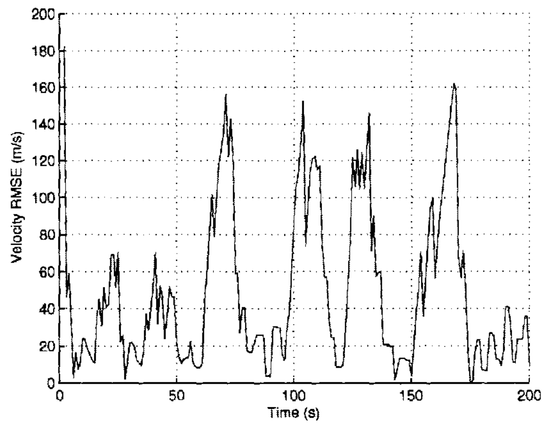


Figure 4.10: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, MDP controlled fusion.

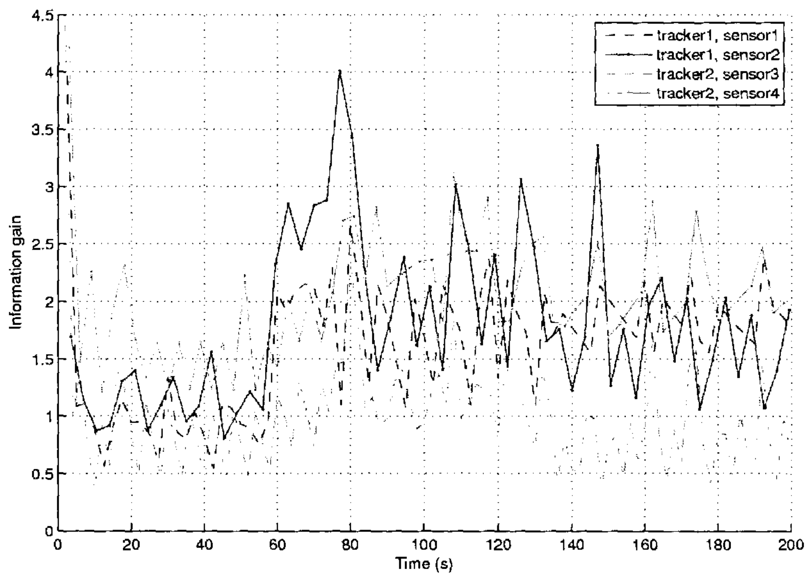


Figure 4.11: Information gain of $platform_3$.

Figure 4.11 shows the information gain of $platform_3$ for all the AMRs arriving from the sensors of $platforms_1$ and $platform_2$. Figure 4.12 shows the integrated information gain values obtained by integrating the values of the information gain during the policy re-calculation stages during the following time intervals: $[5, 15]$, $[60, 75]$, and $[120, 135]$ s. The MDP policy of $platform_3$ was updated several times during the simulation at the end of the policy re-calculation stages above.

Table 4.2 shows the time-averaged position and velocity RMSE in the targets state estimation of the $platform_3$ after 100 Monte Carlo runs. The performance with no information sharing mode is the worst. The performance of the mode in which all the AMRs are transmitted among all the sensors is the best. We can see that the performance of the MDP controlled mode is much better that that of the first

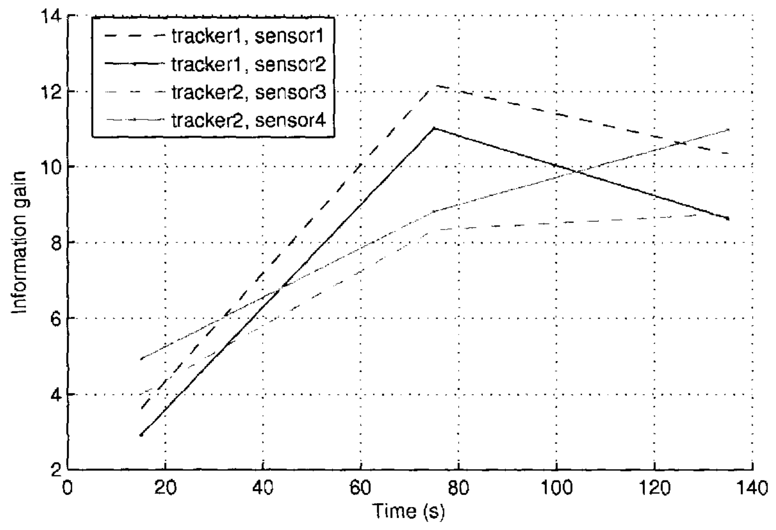


Figure 4.12: Information gain of $platform_3$, integrated during policy re-calculation stages.

Table 4.2: Performance metrics summary, test-case A

Communication mode	Position RMSE (m)		Velocity RMSE (m/s)	
	$Target_1$	$Target_2$	$Target_1$	$Target_2$
All AMRs shared	73.2	79.6	38.4	39.5
No AMRs shared	382.1	300.3	90.8	79.0
MDP controlled	103.8	109.7	44.4	45.8

mode but still worse that that of the second mode. In MDP controlled mode the maximal number of sensors that could transmit AMRs to $platform_3$ was restricted to two sensors. We can see that the information flow in the system was reduced by half, which in many cases may justify the reduction in performance. In many cases the situation in which all the platforms transmit AMRs to all the other platforms is infeasible. The performance may be increased though by increasing the maximal allowed number of sensors transmitting remote AMRs to any given platform.

4.5.2 Simulation Results: Test-case B

The following test-case will demonstrate internal analysis of the decision mechanism. The simulation assumes a platform (node) that fuses tracking information originating from both its local sensors and the sensors located on the remote node.

In the simulated test case, the platform designated as $platform_7$ is in need of tracking information originating from $platform_1$. This information is requested by additional platforms besides $platform_7$, so it is available from the following nodes: $platform_1$ (naturally), $platform_3$, and $platform_4$. It is the data lookup approach described above that is responsible for locating all the nodes in the distributed system possessing the required information. It should be noted that the information that $platform_3$ and $platform_4$ receive from $platform_1$ is downsampled at different ratios, which means that though these nodes are closer to $platform_7$, the quality of information available from them is lower. Note that $platform_7$ range is also sufficient for covering the region in question but at a relatively poor quality. Therefore, $platform_7$ is facing a decision problem regarding the choice it has to make, which will be modeled by a Markov decision process.

Table 4.3 shows the sequence of states and actions resulted from calculating the MDP policy and then executing it.

A single row of a table contains a step number, an MDP state and the action that have been taken according to the optimal policy so found. The latter two are

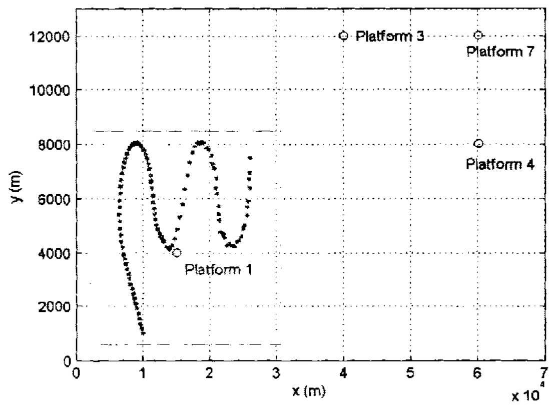


Figure 4.13: The target trajectory and the estimated track. $platform_7$ is in need of tracking information originating from $platform_1$ which is re-transmitted and available from $platform_3$ and $platform_4$ as well.

Table 4.3: The sequence of states and actions

Frame number	Current state	Act.	#	Current state	Act.
1	000 000 000	100	12	010 100 000	010
2	100 000 000	100	13	010 001 000	010
3	100 000 000	100	14	010 101 000	010
4	100 000 000	100	15	010 010 000	001
5	100 000 000	100	16	010 010 100	001
6	100 000 000	100	17	010 010 100	001
7	001 000 000	100	18	010 010 100	001
8	010 000 000	100	19	010 010 001	001
9	010 100 000	010	20	010 010 010	010
10	010 100 000	010	21	010 100 010	010
11	010 100 000	010	22	....	...

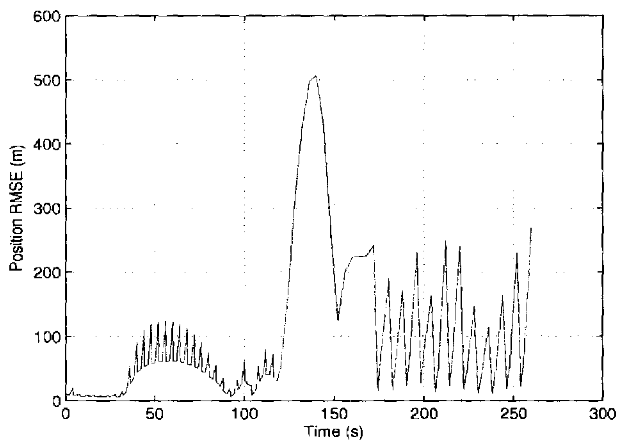


Figure 4.14: RMSE metrics of data fusion at $platform_7$.

specified in a binary format. The optimal policy for the node being in any feasible state indicates the exact action it should take. The policy has been calculated with discount factor $\gamma = 0.9$ and error bound of value of state $\varepsilon = 0.1$. Below we comment lines of Table 4.3.

- *Step 1:* This state is the initial one. We can see that no information is being received and no information is being requested at this state by the node. According to the policy, the action corresponding to this state $\pi^*(s) = 100$ that tells that $platform_1$ is requested to supply the information.

- *Steps 2–6:* $platform_1$ supplies the required information and the resulting action is to continue receiving information from it.

- *Step 7:* The resulting state indicates that $platform_1$ either denied to supply the information or the data have not arrived at the destination. Resulting action did not change and the current node still requests information from $platform_1$.

- *Step 8:* The requested information from $platform_1$ is denied for the second time. This time, according to the policy, the action corresponding to the current state is to request information from $platform_3$.
- *Steps 9-12:* $platform_3$ supplies the required information and the resulting action is to continue receiving information from this node.
- *Step 13:* For the first time $platform_3$ denies the information. The resulting action is the same.
- *Step 14:* Information from $platform_3$ resumes and the action is the same.
- *Step 15:* $platform_3$ refuses for the second time, though not in sequence. Then $platform_4$ is requested.
- *Steps 16–18:* Requested information from $platform_4$ arrives, the action is the same.
- *Step 19:* No data arrived from $platform_4$, action is the same.
- *Step 20:* No data arrived from $platform_4$ for the second time, the action is to request information from $platform_3$.
- *Step 21:* Data from $platform_3$ successfully arrived, refusal history of $platform_3$ is cleared, the action is to continue requesting information from $platform_3$.

- ...

Figure 4.13 shows the target trajectory and the estimated track resulting from the data fusion process corresponding to steps 2–12 described above. The scenario of the target movement includes several constant velocity stages interleaved with coordinated turn maneuvers performed at rates $|\omega| = 1$ °/s. The initial target position $[\xi, \eta]$ is $[6200, 0]$ m and the initial target velocity is 200 m/s. It is assumed that the sensor measures the target range, bearing and range rate. As mentioned, $platform_7$ is in need of tracking information originating from $platform_1$, which covers the marked region in Figure 4.13. Both $platform_3$ and $platform_4$ receive AMRs from $platform_1$ though downsampled with different ratios: 4 for $platform_3$ and 5 for $platform_4$. $platform_7$ range is also sufficient for covering the region in question albeit at a relatively poor quality. Its range and bearing standard deviations are $\sigma_r = 450$ m, $\sigma_\Theta = 0.02$ °/s, respectively. $platform_1$ has similar characteristics with $\sigma_r = 400$ m, $\sigma_\Theta = 0.02$ °/s, but because of its proximity to the target, the estimate resulting from it is more precise.

Figure 4.14 shows the RMSE metrics out of the data fusion process after 300 Monte Carlo runs. During the first 120 scans corresponding to steps 2–6, $platform_7$ obtained the AMRs at the original sampling rate from $platform_1$ and combined this information with its own measurements. In the period between scans 120 and 170 (steps 7–8) no information arrived from $platform_1$, which resulted in steep precision drop because the estimation was done using measurements of $platform_7$ itself only. After scan 170 (steps 9–12) downsampled AMRs originating from $platform_1$ began to arrive from $platform_3$, which resulted in improved performance though worse than the one corresponding to steps 2–6.

The decision process that has been simulated contained probability and reward matrices of considerable dimensions. In the demonstrated example the dimensions of P are $512 \times 512 \times 3$. Dimensions of the reward matrix are 512×3. Using sparse matrix presentation is therefore very effective, especially justified by the fact that a considerable part of the transition probability and reward matrices were not populated.

4.5.3 Simulation Results: Test-case C

The following test-case will compare the performance of the approach among three different data fusion methods, namely, associated measurements fusion, track-to-track fusion and tracklets fusion.

The simulated cluster contains 5 platforms and 8 sensors. Each sensor belongs to a specific platform associated with a corresponding tracker. Sensors 1 and 2 are connected to $tracker_1$, sensors 3 and 4 to $tracker_2$, sensor 5 to $tracker_3$, sensors 6 and 7 to $tracker_4$ and sensor 8 is connected to $tracker_5$. The sensors are located at $x_s^i =$ $[65, 155]'$, $[80, 140]'$, $[10, -65]'$, $[30, -40]'$, $[-80, 20]'$, $[-70, 130]'$, $[-40, 150]'$, $[20, 150]'$ km, respectively. As in the previous test-case, the original measurements are obtained from the sensor s in the form $[r(k,s,j) \quad \theta(k,s,j)]'$ where $r(k,s,j)$ and $\theta(k,s,j)$ are the range and the azimuth angle of the target j supplied by sensor s at time t_k, respectively. The sensor s range r and bearing Θ standard deviations are $[\sigma_r^s,\ \sigma_\Theta^s] =$ [40, 2.5], [35, 2], [52.5, 3.5], [30, 2.5], [45, 4.5], [52.5, 3.5], [52.5, 3.5], [45, 4.5] (in m, mrad, respectively). The sampling intervals of the sensors are 2.5, 3.5, 2, 3, 1.5, 2, 2, 1.5 s, respectively. The false alarms are uniformly distributed in the coverage areas of the sensors with the number of false alarms Poisson distributed with means of 40, 40, 100, 50, 50, 40, respectively. The simulation included two closely spaced targets with the same scenario of the target movements as in the previous test-case. Figure 4.15 shows the coverage areas of the sensors and the targets trajectory.

4.5.3.1 Associated Measurements Fusion

In this sub-section, we demonstrate the simulation results of distributed data fusion system based on associated measurements fusion. Figure 4.16 shows the position and velocity RMSE of $target_1$ state estimation, respectively, when all the AMRs are shared among the platforms. Figure 4.17 shows the position and velocity RMSE of $target_1$ state estimation, respectively, for the case when no information is shared among the platforms. Figure 4.18 shows the position and velocity RMSE of $target_1$ state estimation, respectively, for the case when the data fusion is controlled by MDP.

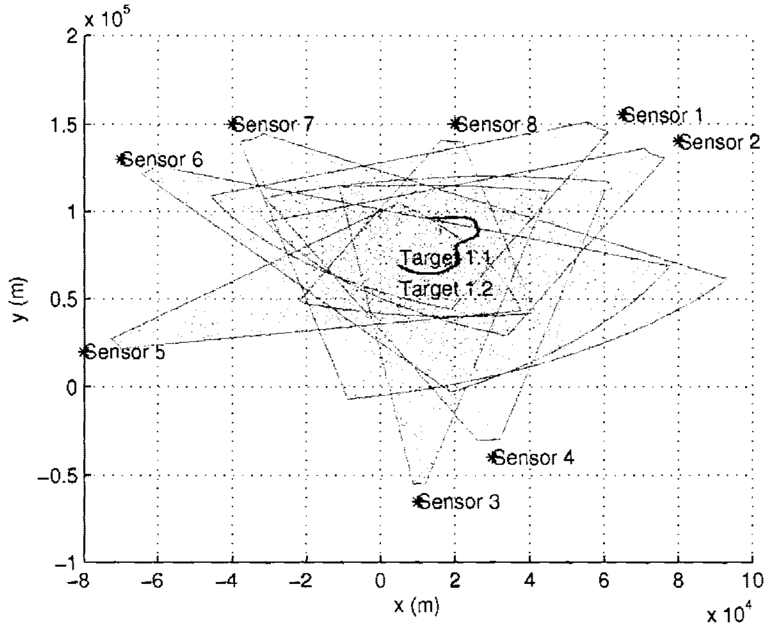


Figure 4.15: Test-case C. Coverage area of the sensors and targets trajectory. Eight sensors belonging to five different platforms are present.

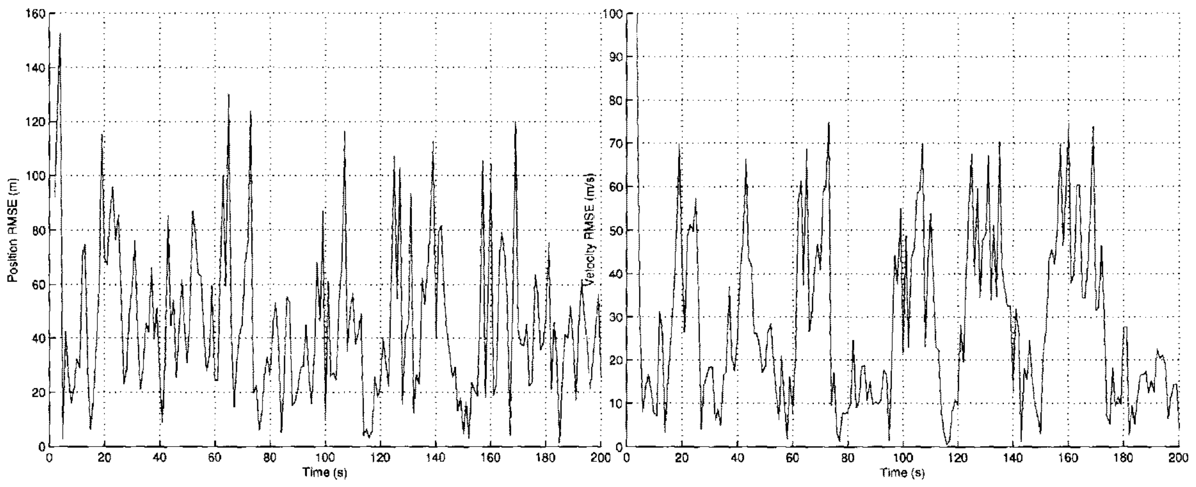


Figure 4.16: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, all AMRs shared, associated measurements fusion.

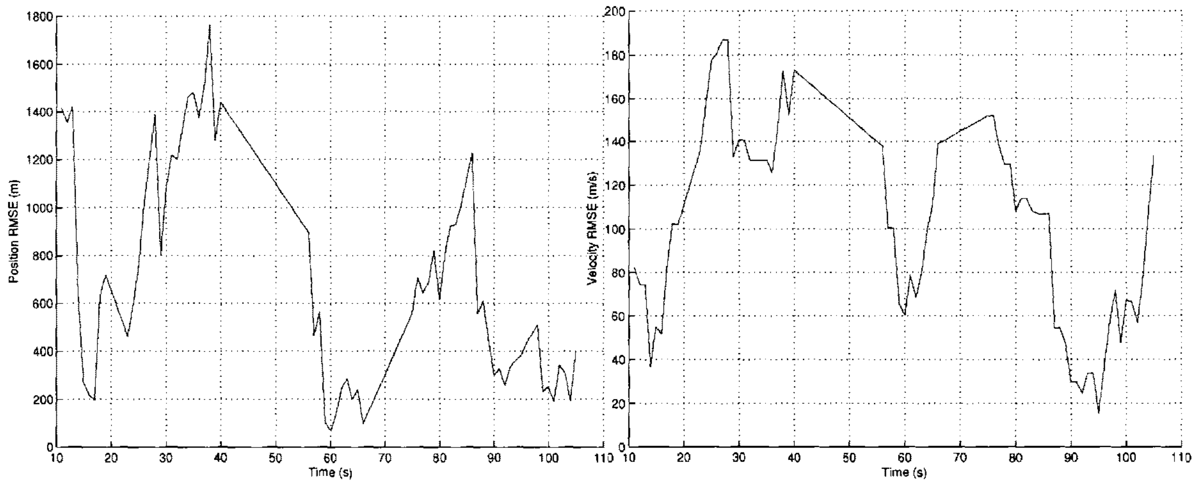


Figure 4.17: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, no information shared, associated measurements fusion. Tracking data is available only for $t < 105$ s after which the targets come out of the coverage area of the sensors of the platform.

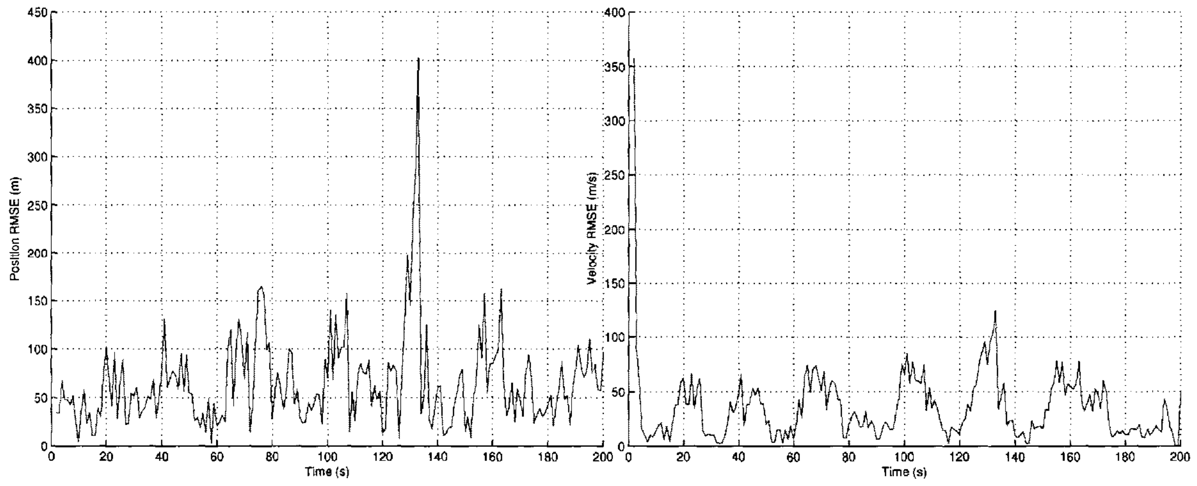


Figure 4.18: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, MDP controlled, associated measurements fusion.

Table 4.4: Performance metrics summary, associated measurements fusion

Communication mode	Position RMSE (m)		Velocity RMSE (m/s)	
	$Target_1$	$Target_2$	$Target_1$	$Target_2$
All AMRs shared	45.5	45.1	32.4	27.8
No AMRs shared	703.3	664.8	100.7	70.4
MDP controlled, 4 sources	65.5	62.1	35.1	34.9
MDP controlled, 2 sources	72.6	71.1	36.5	36.9
2 best sources	94.4	98.6	45.9	43.3

Table 4.4 shows the time-averaged position and velocity RMSE in the targets state estimation of the after 100 Monte Carlo runs. The performance of the mode with no information sharing is the worst. The performance of the mode in which all the AMRs are transmitted among all the sensors is the best. We can see that the performance of the MDP controlled mode is much better that that of the first mode but still worse that that of the second mode. In MDP controlled mode the maximal number of sensors that could transmit AMRs to $platform_3$ was first restricted to two sensors and then to four sensors. The information flow in the system was reduced, which in many cases may justify the reduction in performance. In many cases the situation in which all the platforms transmit AMRs to all the other platforms is infeasible. The performance may be increased though by increasing the maximal allowed number of sensors transmitting remote AMRs to any given platform.

Table 4.4 also includes the results of an additional simulation in which there were always picked two sources having the best characteristics — sources number two and six. The results were inferior to those of an MDP-based approach.

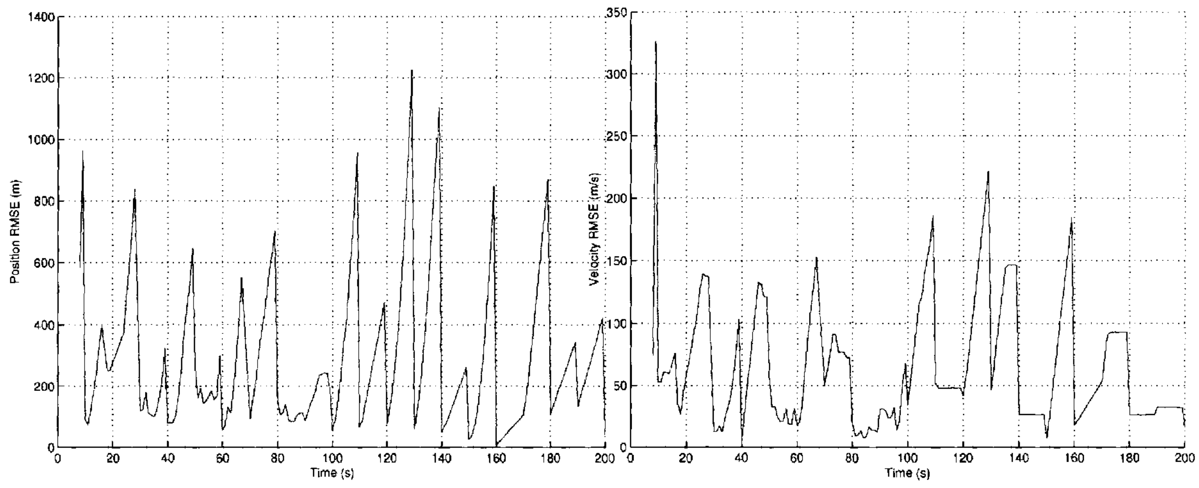


Figure 4.19: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, all tracks shared, track-to-track fusion.

4.5.3.2 Track-to-track Fusion

In this sub-section, we demonstrate the simulation results of distributed data fusion system based on track-to-track fusion. Figure 4.19 shows the position and velocity RMSE of $target_1$ state estimation, respectively, for the first policy when all the tracks are shared among the platforms. Figure 4.20 shows the position and velocity RMSE of $target_1$ state estimation, respectively, for the second policy when no information is shared among the platforms. Figure 4.21 shows the position and velocity RMSE of $target_1$ state estimation, respectively, for the third policy when the data fusion is controlled by MDP.

Table 4.5 shows the time-averaged position and velocity RMSE in the targets state estimation after 100 Monte Carlo runs.

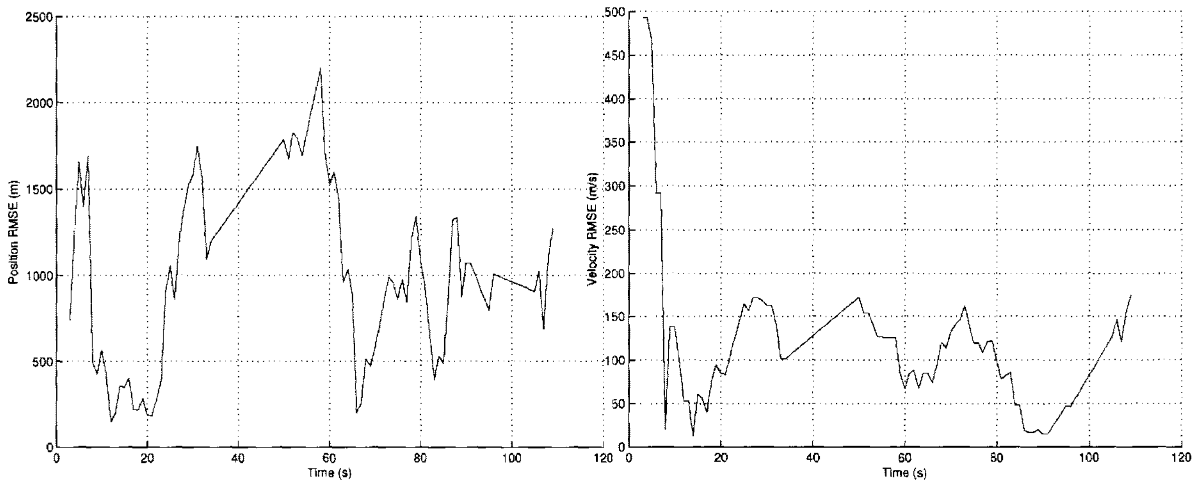


Figure 4.20: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, no information shared, track-to-track fusion. Tracking data is available only for $t <$ 110 s after which the targets come out of the coverage area of the sensors of the platform.

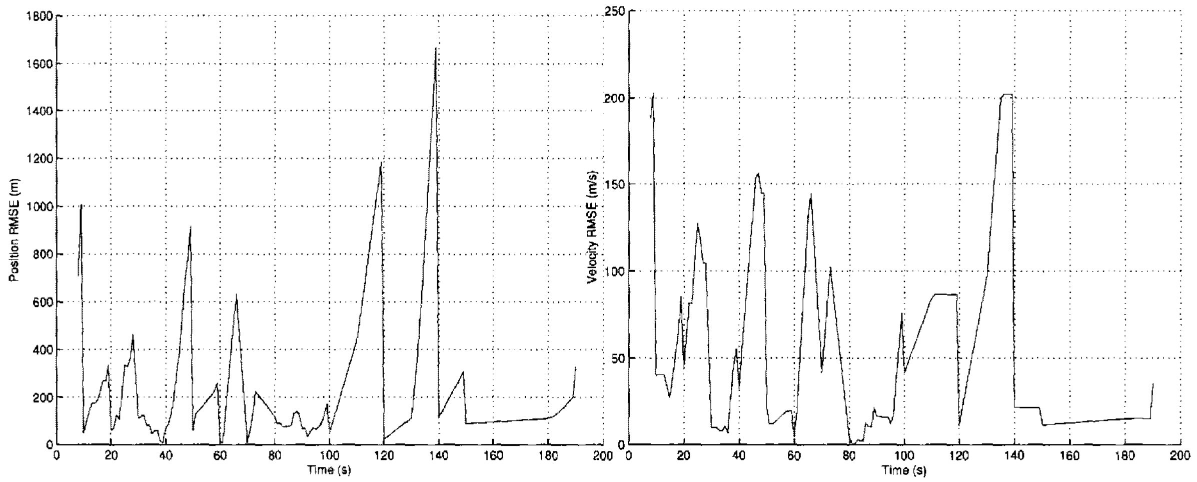


Figure 4.21: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, MDP controlled, track-to-track fusion.

Table 4.5: Performance metrics summary, track-to-track fusion

Communication mode	Position RMSE (m)		Velocity RMSE (m/s)	
	$Target_1$	$Target_2$	$Target_1$	$Target_2$
All tracks shared	268.2	272.6	60.8	65.9
No tracks shared	993.2	814.5	121.7	103.0
MDP controlled	297.3	301.6	65.9	71.3

Table 4.6: Performance metrics summary, tracklet fusion

Communication mode	Position RMSE (m)		Velocity RMSE (m/s)	
	$Target_1$	$Target_2$	$Target_1$	$Target_2$
All tracks shared	215.1	204.7	58.4	59.8
No tracks shared	749.4	750.4	104.2	104.6
MDP controlled	268.8	250.8	58.8	65.4

4.5.3.3 Tracklet Fusion

In this sub-section, we demonstrate the simulation results of distributed data fusion system based on tracklet fusion. Figure 4.22 shows the position and velocity RMSE of $target_1$ state estimation, respectively, for the first policy when all the tracklets are shared among the platforms. Figure 4.23 shows the position and velocity RMSE of $target_1$ state estimation, respectively, for the second policy when no information is shared among the platforms. Figure 4.24 shows the position and velocity RMSE of $target_1$ state estimation, respectively, for the third policy when the data fusion is controlled by MDP.

Table 4.6 shows the time-averaged position and velocity RMSE in the targets state estimation of the $platform_3$ after 100 Monte Carlo runs.

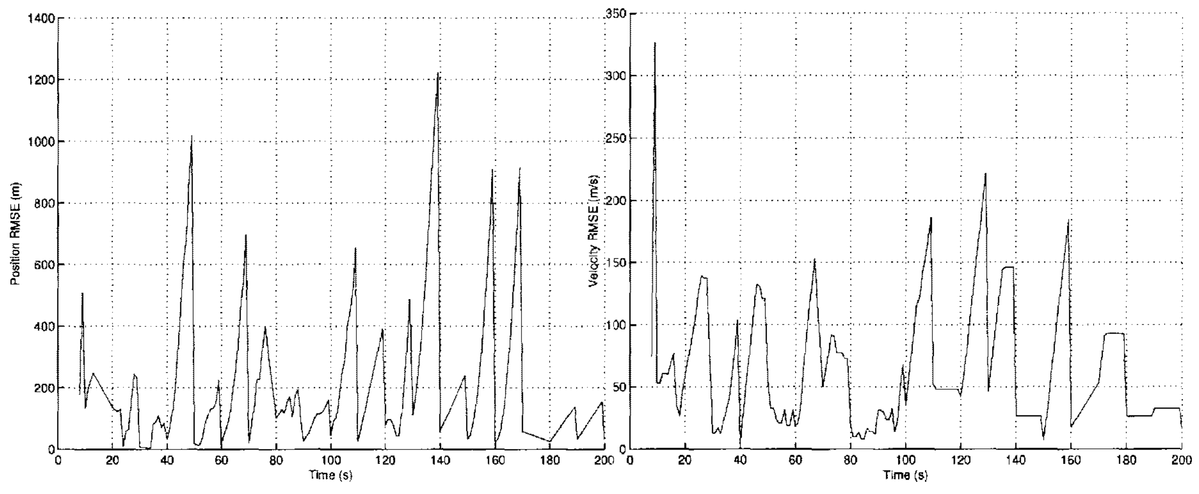


Figure 4.22: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, all tracklets shared, tracklet fusion.

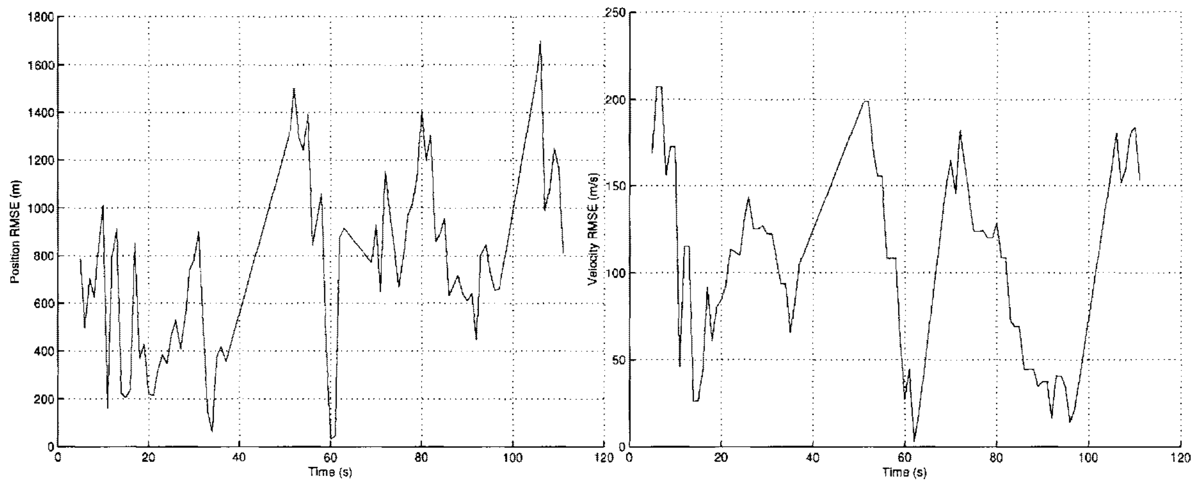


Figure 4.23: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, no information shared, tracklet fusion. Tracking data is available only for $t < 110$ s after which the targets come out of the coverage area of the sensors of the platform.

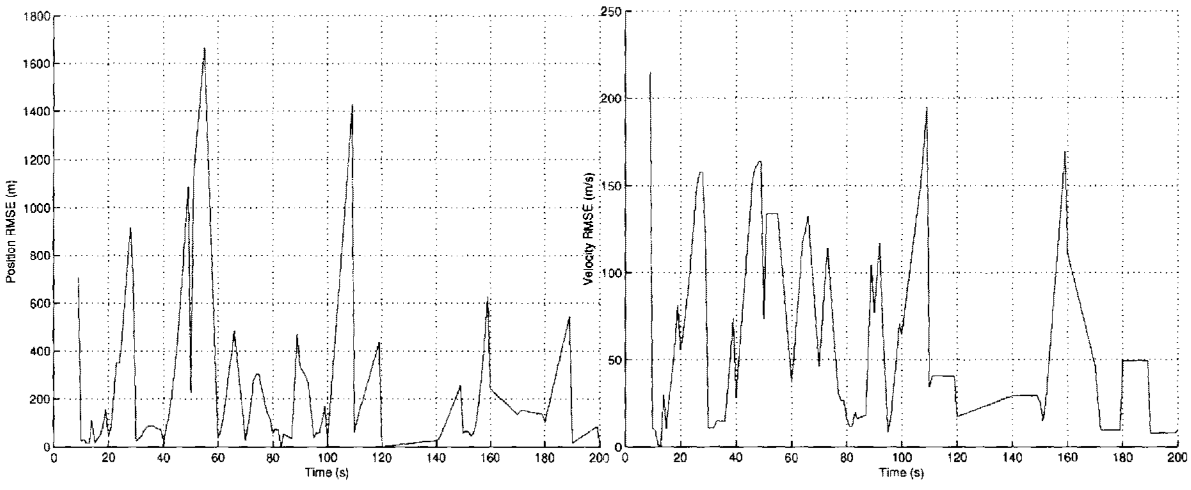


Figure 4.24: Position RMSE (left) and velocity RMSE (right) of $target_1$, $platform_3$, MDP controlled, tracklet fusion.

4.6 Communication Data Rate and Computational Load in Distributed Tracking Algorithms

The focus of this sub-section will be on the communication cost of the aforementioned data fusion types. One of the main advantages of network-centric approaches, where the sensors may be located at significant distances from one another, versus platform-centric one, where the sensors are located in close vicinity from one another, is in achieving a higher precision and robustness by the separation of sensors located on distinct platforms [8]. The sharing of tracking data results in tracking results that are in general more complete, more accurate, and more timely than those obtained in the case of a single sensor or a single platform. In many cases, sharing the data provides tracking information that is unavailable for many of the participating platforms or may be obtained at significantly lower quality. One of the main barriers in network-centric tracking is the transfer of tracking data from one platform to another via a communication link capable of transferring the required volumes of information. This communication load should be taken into consideration in the design of the tracking system and cannot be ignored. As mentioned before, the task of providing tracking information from one platform to anther node may be infeasible because of the limitations of the communication channel capacity. The decision mechanism, presented in this thesis, provides each platform with the required data for the distributed data fusion process subject to the available channel capacities and reducing redundancy in the information flow in the overall system. Below we will address the communication requirements for each of the distributed data fusion approaches we are using, namely, associated measurement fusion, tracklet fusion and track-to-track fusion [65].

When associated measurement reports (AMR) arriving from other platforms as well as from the local sensors are used to form a common tracking picture, the data rate Ψ for a platform participating in the data exchange process is given by

$$\Psi_{AMR} = \frac{N_{meas/scan}}{\delta_{scan}} n_{data} \tag{4.23}$$

where δ_{scan} is the tracking interval, $N_{meas/scan}$ is the number of associated measurements per scan and n_{data} is the allocated number of bits used by a system to transmit the data between the platforms:

$$n_{data} = n_{meas} + n_{meas\_acc} + n_{number} + n_{time} \tag{4.24}$$

where n_{meas}, n_{meas_acc}, n_{number},and n_{time} are the bit allocation numbers for a transmitted measurement, its related accuracy information (unique components of a covariance matrix), track number, and the measurement's time, respectively. The bit allocation for AMR-based data consisting of range, bearing and elevation is

$$n_{meas} = n_{range} + n_{bearing} + n_{elevation} \tag{4.25}$$

$$n_{meas\_acc} = n_{r\_acc} + n_{b\_acc} + n_{e\_acc} \tag{4.26}$$

The resulting data rate is therefore

$$\begin{aligned} \Psi_{AMR} &= \frac{N_{meas/scan}}{\delta_{scan}}(n_{range} + n_{bearing} + n_{elevation} + n_{r\_acc} \\ &+ n_{b\_acc} + n_{e\_acc} + n_{number} + n_{time}) \end{aligned} \tag{4.27}$$

In the case of track-to-track fusion, the data rate is given by

$$\Psi_{tracklet} = \frac{N_{meas/scan}}{\delta_{scan}} n_{data} \tag{4.28}$$

For the state of the target of the form

$$\mathbf{x} = [x \;\; \dot{x} \;\; y \;\; \dot{y} \;\; z \;\; \dot{z}]' \tag{4.29}$$

where x, y and z are the Cartesian coordinates of the target and $\dot{x}$, $\dot{y}$ and $\dot{z}$ are its x, y and z velocity components, respectively, n_{data} is given by

$$\begin{aligned} n_{data} &= 3n_{pos} + 3n_{vel} + n_{number} \\ &+ n_{time} + 21n_{acc} \end{aligned} \tag{4.30}$$

where n_{acc} is the number of bits allocated for the 21 unique elements of the state covariance matrix. The resulting data rate is given by

$$\Psi_{track2track} = \frac{N_{meas/scan}}{\delta_{scan}}(3\ n_{pos} + 3\ n_{vel} + n_{number} + n_{time} + 21\ n_{acc}) \tag{4.31}$$

The data rate in the case of track-to-track fusion is similar to the track-to-track

Table 4.7: Communication and computational load

Fusion method/Metrics	Ψ (bit/s)	t_{comp} (s)	$\epsilon_{\mathbf{x}\ RMSE}$ (m)
AMR fusion, all tracks shared	587.8	14.2	45.3
AMR fusion, no tracks shared	0	9.7	684.0
AMR fusion, MDP controlled	310.4	13.6	63.8
Track fusion, all tracks shared	645.6	10.9	270.4
Track fusion, no tracks shared	0	9.3	903.8
Track fusion, MDP controlled	338.9	13.8	299.4
Tracklet fusion, all tracks shared	645.6	9.2	209.9
Tracklet fusion, no tracks shared	0	8.5	749.9
Tracklet fusion, MDP controlled	338.9	13.1	259.8

one and is given by

$$\Psi_{tracklet} = \frac{N_{meas/scan}}{\delta_{scan} N_{meas/tracklet}} n_{data} = \frac{\Psi_{track2track}}{N_{meas/tracklet}} \tag{4.32}$$

where $N_{meas/tracklet}$ is the measurements number per one tracklet. The resulting data rate is given by

$$\Psi_{tracklet} = \frac{N_{meas/scan}}{\delta_{scan} N_{meas/tracklet}} (3\ n_{pos} + 3\ n_{vel} + n_{number} + n_{time} + 21\ n_{acc}) \tag{4.33}$$

4.6.0.4 Communication and Computational Load Results

Table 4.7 and Table 4.8 show the resulting communication and computational load results for each of the distributed data fusion approaches.

The tables features different metrics for the data fusion methods used — data rate Ψ for a platform participating in the data exchange process, computation time

Table 4.8: Communication and computational load derivatives

Fusion method/Metrics	$\Psi\epsilon_{\mathbf{x}\ RMSE}$	Ψt_{comp}	$t_{comp}\epsilon_{\mathbf{x}\ RMSE}$	$\Psi\epsilon_{\mathbf{x}\ RMSE}t_{comp}$
AMR fusion, all tracks shared	$2.6 \cdot 10^4$	$8.3 \cdot 10^3$	$6.4 \cdot 10^2$	$3.7 \cdot 10^5$
AMR fusion, no tracks shared	0	0	$6.6 \cdot 10^3$	0
AMR fusion, MDP controlled	$1.9 \cdot 10^4$	$4.2 \cdot 10^3$	$8.7 \cdot 10^2$	$2.7 \cdot 10^5$
Track fusion, all tracks shared	$1.7 \cdot 10^5$	$7.0 \cdot 10^3$	$2.9 \cdot 10^3$	$1.9 \cdot 10^6$
Track fusion, no tracks shared	0	0	$8.4 \cdot 10^3$	0
Track fusion, MDP controlled	$1.0 \cdot 10^5$	$4.6 \cdot 10^3$	$4.1 \cdot 10^3$	$1.4 \cdot 10^6$
Tracklet fusion, all tracks shared	$1.3 \cdot 10^5$	$5.9 \cdot 10^3$	$1.9 \cdot 10^3$	$1.2 \cdot 10^6$
Tracklet fusion, no tracks shared	0	0	$6.4 \cdot 10^3$	0
Tracklet fusion, MDP controlled	$8.8 \cdot 10^4$	$4.4 \cdot 10^3$	$3.4 \cdot 10^3$	$1.1 \cdot 10^6$

t_{comp} for the overall simulated time of 200 s, and averaged position RMSE $\epsilon_{\mathbf{x}\ RMSE}$. Also, the table contains the combinations of the metrics above, for example the product of required computation time and required data rate — Ψt_{comp}. For all of the metrics and their combinations the higher range means less favorable results, which makes the comparison easier. The lowest data rate was achieved when using MDP based associated measurements data fusion, followed by MDP-based track-to-track or tracklet data fusion simulations. The highest one was registered during track-to-track or tracklet based data fusion simulations when all the data were shared, both of which used the same time interval between data transmissions — 10 s. In terms of computation time, the best results were achieved when using tracklet based data fusion method when no data were shared among the platforms. The highest computation time resulted in the track-to-track MDP-based data fusion simulation. The best accuracy was achieved when using AMR data fusion sharing all the information, closely followed by MDP-based AMR data fusion results. The worst accuracy was achieved in track-to-track data fusion simulation when no data were shared among the platforms. The best $\Psi\epsilon_{\mathbf{x}\ RMSE}$ metrics was achieved when using MDP based associated measurements data fusion and the worst when using track-to-track data fusion when all the data were shared among the platforms. In summary, associated measurements with MDP-based data fusion yielded the best results in the majority of the metrics categories and track-to-track data fusion with all tracks shared gave the worst results in most of them. That could be explained in part by the need to transmit the elements of covariance matrix combined with the fact that as the computation of the exact cross-covariance matrices in a track-to-track fusion system would hugely increase either computation load, approximation techniques are used used to compute them. Similar reasons can be stated for the tracklet data fusion as both covariance matrices need to be transmitted. Also, due to the common process noise between the tracks, not all correlations between the various versions of the same track, maintained by different local trackers, can be removed by using the tracklets. Therefore, this algorithm is only an approximation and it is reliable only for targets with small maneuvering index.

Chapter 5

A Distributed Multisensor-Multitarget Tracking Testbed for Sensor Management and Data Fusion

5.1 SYSTEM DESCRIPTION

Being a distributed Data Fusion System (DFS) emulator, the testbed is composed of one or more of Data Fusion Centers (DFC) which may reside on distinct computer systems, together forming a common emulating system interconnected via a network.

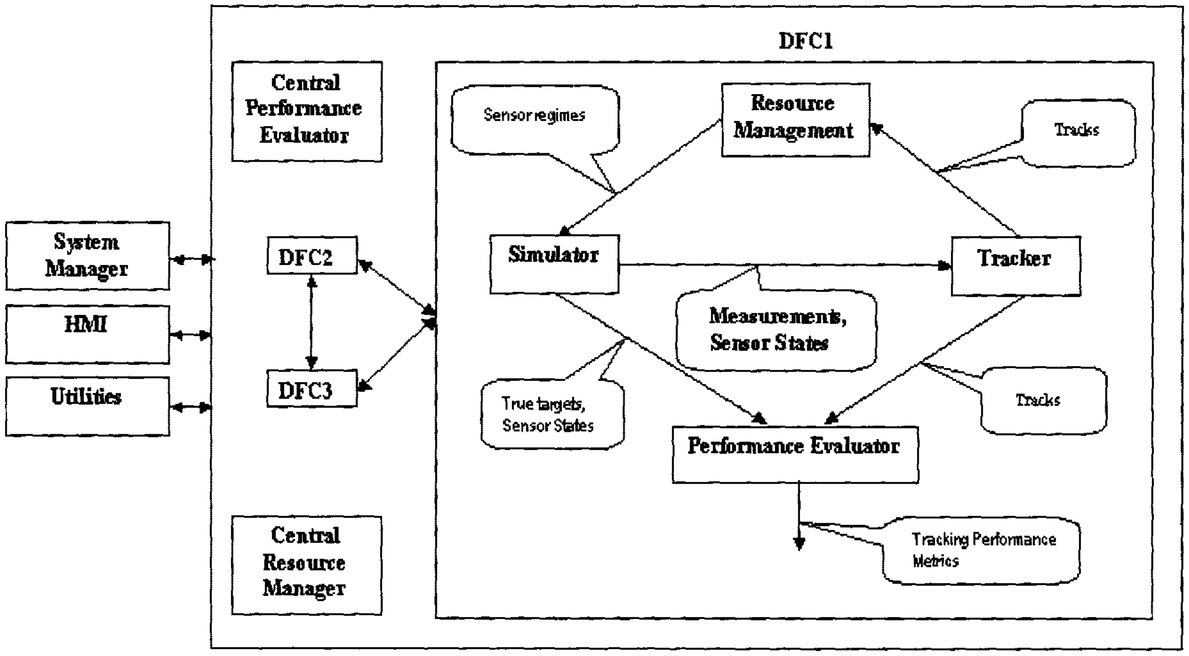


Figure 5.1: Testbed block diagram.

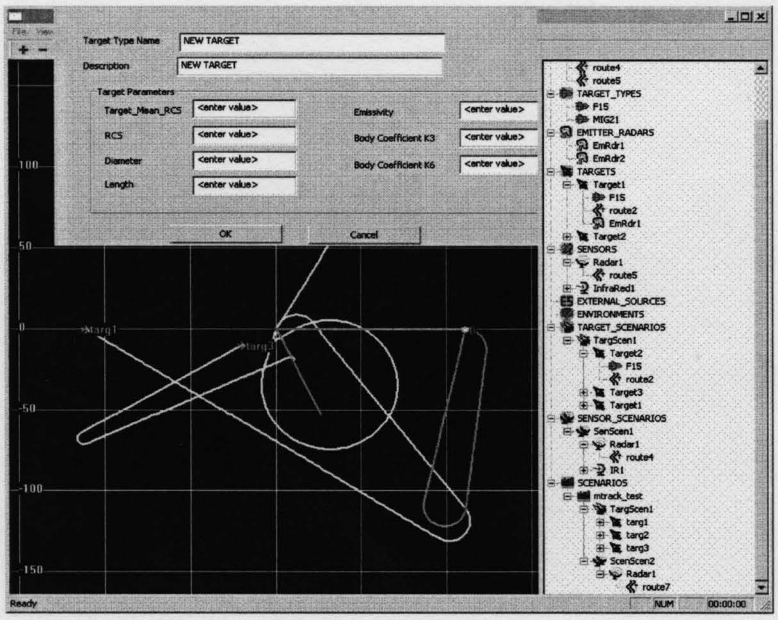
A DFC thus presents one of the main elements which compose the emulated system and interconnection which define its final architecture. Fig. 5.1 shows the block diagram of the testbed as well as interaction between its main modules. A DFC consists of the four main operational modules: Simulator, Tracker, Resource Manager, Tracking Performance Evaluator, and three auxiliary ones: System Manager, Human-Machine Interface (HMI), and Utilities module.

5.1.1 System design principles

The system is compounded in Object Oriented (OO) form. It consists of separated blocks with maximal independent functionality and minimal data flow between these blocks. The system is flexible in reference to further changes of its components at later stages without substantial system reorganization. Many of the system components are simulated in either simplified or a full form, in reference to the fullness of physical phenomena simulation. For example, probability of target detection may be defined as a sensor parameter or calculated taking into account target size, environment specific, sensor operational regimes, etc. Target parameters and trajectories are also defined by different ways: with fixed route points or leg kinematic model. The system supports joint work of different Data Fusion Centers, processing output data of various sensors, with arbitrary sensor to DFC and DFC-to-DFC inter-connections.

5.1.2 Human-Machine Interface (HMI) Module

The description of the testbed is started from auxiliary modules as they are the interface between the user and the testbed itself. The HMI module is responsible for definition of system work scenario and results visualization. It includes Scenario Generator (SG) and DFC Connection Module.



Target Type Name: NEW TARGET

Description: NEW TARGET

Target Parameters

Target_Mean_RCS	<enter value>	Emissivity	<enter value>
RCS	<enter value>	Body Coefficient K3	<enter value>
Diameter	<enter value>	Body Coefficient K6	<enter value>
Length	<enter value>		

Figure 5.2: Scenario View Mode.

5.1.2.1 Scenario Generator

The Scenario Generator is built by using the "Lego" principle. This user-friendly interface allows creating scenarios from different elements, in a way similar to how it is done in the popular game. Elements of each type (routes, radar-emitters, sensors, targets, environments, etc.) are created first, then the scenarios which consist of these elements are created. A desired item is added into a created component (for example, to insert a route description to a certain target) by simply dragging the item using the mouse. Those operations cause the corresponding changes in Simulative Scenario Components Tree storage, which is a database reflecting the current user settings regarding various scenarios and all their components which have been defined by a user so far. Fig. 5.2 shows Scenario View Mode, where on the right side all the various elements of the defined scenarios and their components are defined, whereas on the left side we can see the graphical representation of the chosen scenarios including the trajectories of all the targets and sensors.

5.2 Peer-to-Peer Implementation of Data Flow for Network-Centric Architecture Simulation

As shown in Fig. 5.1, the testbed implements a decentralized multisensor tracking and fusion architecture. Each DFC thus represents one of the nodes which can interchange information with other DFC nodes. The information can be measurements or tracks. Each DFC acts independently and has its tracker to process the measurement data information which is obtained from the sensors pertaining to the specific DFC. Also, each DFC is actually an independent standalone instance of a testbed application running on the same or remote computer. One of the DFC nodes, depicted in Fig. 5.1, is defined as a main DFC node which designates that it is the node belonging to the current testbed application instance. There can be created an arbitrary configuration of DFC connections using graphic interface of the DFC Connection Module. In a specific testbed application, a specific desired data fusion configuration should be first defined. Fig. 5.3 shows an example of a possible data flow configuration between

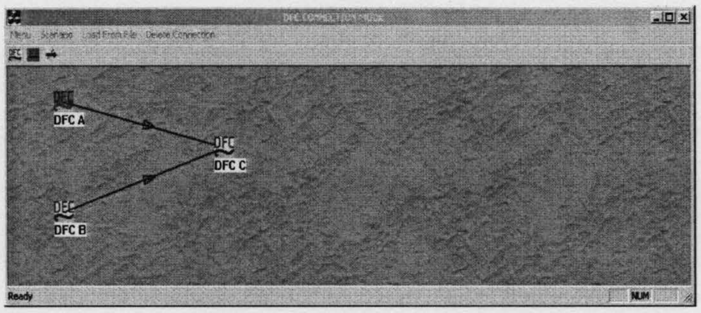


Figure 5.3: DFC Connections View.

different data fusion centers. DFC "C", which is the DFC of the current node (main DFC center), should receive track data from DFC "A" and DFC "B". Both DFC "A" and DFC "B" designate remote DFCs running as standalone testbed applications on the same or remote locations. A specific tracker type as well as the scenario to be run on the main DFC should be selected. After that, a connection with remote stations, where DFC "A" and DFC "B" are running, should be established.

The communication scheme of testbed instances can be classified as peer-to-peer type to distinguish it from the client-server model, where one can identify a fixed division into clients and servers. In our case, every testbed instance can communicate with any other testbed. When considering a specific one-direction communication channel between two testbeds, then it is convenient to see it as a client-server type connection. Each testbed contains both server and client mechanisms in order to be able to receive information from the multiple clients (server side) and to send information to multiple servers (client side).

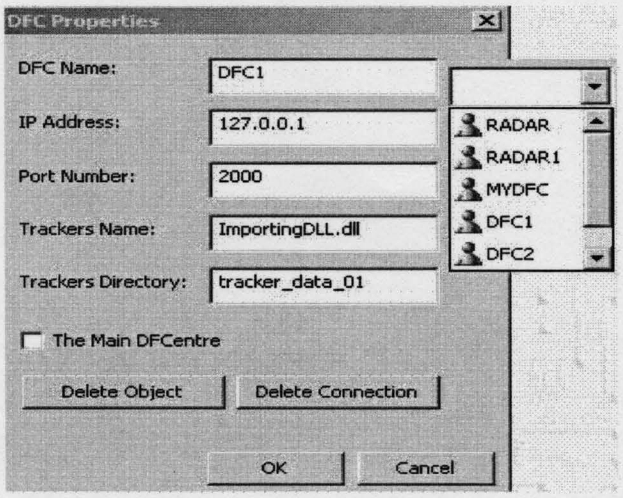
5.2.0.2 Server side

As mentioned, though the overall communication scheme is peer-to-peer, as micro level we consider each instance of data transfer between two testbeds as client-server connection. The server waits for the incoming client calls. When the incoming client call is received and accepted, the server thread will not start the communication with this client. Instead, it will create a new thread and will pass the client connection to this thread letting the communication between the server and this new client become the full-time job for this newly created thread. Then the server will wait for another potential incoming communication. By doing so, the server can handle multiple clients. Each DFC has the following properties, as shown in Fig. 5.4:

- Name.
- IP address.
- Port number.

5.2.0.3 Client side

The part of a particular testbed mechanism that is responsible to transmitting the relevant tracking data to remote testbed or testbeds is called client. Each client contains two tables - "New connections" and "Established connections". "New connections" table contains the parameters (IP address and the port number) of the


DFC Properties

DFC Name: DFC1

IP Address: 127.0.0.1

Port Number: 2000

Trackers Name: ImportingDLL.dll

Trackers Directory: tracker_data_01

The Main DFCentre

Figure 5.4: DFC connection properties.

remote DFC connection to which should be established. After the successful connection with the remote DFC, the responsibility for maintaining the connection is transferred to the "Established connections" table, and the corresponding record in the "New connections" table is cleaned.

5.2.0.4 Time synchronization

Each testbed has its own local time which is to be synchronized with the rest of the testbeds participating in the peer-to-peer communication. The example of the synchronization scheme between two testbed instances is as follows:

- Both testbeds equalize the simulation speed.
- Round-trip-time (RTT) of data communication is measured.

- If the local time difference between two testbeds is within a pre-defined threshold, taking into account the measured RTT, data transfer is initiated. Otherwise the following steps are taken:
  - The testbed with more advanced local time will pause.
  - The testbed where the local time lags behind will run and notify the other one ahead of time before both local times are equalized.

In case of more than two testbeds, all the testbeds are paused and continue running after testbeds that lag behind in time equalize their local time with them according to the scheme above.

5.2.1 Simulator module

The Simulator module provides simulation of defined scenarios of a given DFC, generating measurements based on simulative sensors, targets, target emitters, and other elements of the scenario. The module also simulates different physical phenomena of environments in which the simulation takes place, affecting signal propagation. The scenario simulation consists of the following components:

- *Target Scenario* simulation
- *Sensor Scenario* simulation
- *Environment Scenario* simulation

The simulation process is organized as follows. Initially, upon receiving the simulation scenario, the following actions are performed: sensors and targets are generated, corresponding routes are transformed into target and sensor trajectories, defined environment regions are transformed into the internal format. Further, each simulation step, the following actions are performed:

- For each target a current target location and emissions are calculated.
- Each sensor's current state is evaluated and the lists of targets, which can be detected by this sensor, are obtained.
- For each target, which can be detected by this sensor, the signal on the sensor input (including noise and interference accompanying the signal) is evaluated, taking into account the current sensor and emitter parameters, and current environment conditions on the signal path.
- For each signal received by this sensor, the detection occurrence is randomly generated depending on the signal characteristics, noise level and sensor parameters. If the detection occurs, corresponding measurements are generated.
- For each sensor, the false alarms, arising due to clutter, internal sensor noise, as well as other reasons, are generated.

The simulator has tracking information extrapolation capability which is automatically provided for the cases when the time which it takes to a sensor or a group of sensors to provide a new portion of measurements is greater than the maximal time after which either true tracking information or extrapolated results should be obtained in order to update the system's database and output the tracking data to the screen or other output channels. Trackers which may be used in a system might already have that capability but in case they have not, the testbed provides it.

5.2.1.1 Target Scenario generation

Target Scenario generation includes calculation of the location and kinematic parameters for each target as well as emission and reflection parameters. It consists of a target list and of a list of *Target Scenarios*. Each of the targets includes *Target Type*, *Route* (or trajectory) and optionally the list of *Emitters*, corresponding to the particular target. *Target Scenario* generation is also responsible for determining emitters' instant states according to their operational scenario. *Target Scenario* generation includes route generation, which produces a route according to its parameters and calculates the target kinematic parameters (velocity, acceleration, climb rate, etc.) for the given time instant. It includes generation of the leg list. A route thus consists of a sequence of items composed in such a way that the end of the previous leg serves as a start for the current leg. Three different types of routes may be defined:

- Fixed-point (FP) route, consisting of FP legs
- Fixed-kinematic (FK) route, consisting of FK legs.

- Stationary route, designating a non-moving object.

FP Leg calculation

FP leg kinematic parameters at a given time instant are calculated based on its given parameters as well as the kinematic parameters at the end of the previous leg. (X_0, Y_0, Z_0) and (X_1, Y_1, Z_1) denote the leg start and end point coordinates respectively, and V_0 and V_1 denote the corresponding target speed values at the start and end of the leg respectively. At the first stage, the trajectory calculation is decomposed into 3 sequential parts:

- Calculating a curved line, a part of the total horizontal trajectory
- Straight-line segment with constant target acceleration
- Straight-line segment with constant target velocity

At the second stage, the target trajectory in z axis is evaluated, which is divided into 4 sequential segments:

- Vertical acceleration
- Constant climb rate
- Vertical deceleration
- Zero climb rate

First stage

The first segment end point coordinates are calculated as

$$X_M = [(X_C - X_1)\sqrt{1-\gamma^2} - (Y_C - Y_1)\gamma]\sqrt{1-\gamma^2} + X_1 \tag{5.1}$$

$$Y_M = [(X_C - X_1)\gamma - (Y_C - Y_1)\sqrt{1-\gamma^2}]\sqrt{1-\gamma^2} + Y_1 \tag{5.2}$$

where (X_C, Y_C) are the turn center coordinates, $\gamma = -\frac{R\delta}{\rho}$. R and ρ are the turn radius and the distance between the turn center and the current leg end point respectively. δ is equal to 1 for the right turn and -1 for the left one. $R = \frac{V_0^2}{a_n}$ where a_n is the target normal acceleration defined for x-y plane. Based on the R value and the coordinates of the left and right turn centers, a decision is made whether the turn is the right or left one. The target direction angle at the leg end point is given by

$$\varphi_1 = \arctan\frac{Y_M - Y_C}{X_M - X_C} - \frac{\pi}{2}\delta \tag{5.3}$$

The time of the target arrival to the first segment's end point is given by

$$T_{first} = \frac{[-(\varphi_1 - \varphi_0)\delta + 2\pi]mod2\pi}{a_n}V_0 \tag{5.4}$$

where φ_0 is the target direction angle at the leg start point defined for x-y plane. φ_0 is defined as the angle between X axis and the velocity vector projection to x-y plane.

The length of the second and the third segments (when the target is moving on a straight line) is given by

$$S_{line} = \sqrt{(X_M - X_1)^2 + (Y_M - Y_1)^2} \tag{5.5}$$

The second segment length is calculated as

$$S_{second} = V_0 T_{second} + \frac{aT_{second}^2}{2} \tag{5.6}$$

where $a = a_t sgn(V_1 - V_0)$ is the target tangential acceleration, and $T_{second} = \frac{V_1 - V_0}{a}$. sgn() is the sign function which returns 1 for positive numbers, -1 for negative, and zero for zero.

The third segment duration is calculated as $T_{third} = \frac{S_{line} - S_{second}}{V_1}$ and the total leg duration is given by $T_{leg} = T_{first} + T_{second} + T_{third}$

Second stage

Let us denote by H the required altitude change. The vertical acceleration is given by

$$a_z = min(kV_z, a_{zmax})sgn(H) \tag{5.7}$$

where k is the vertical acceleration coefficient, a_{zmax} is the maximal vertical acceleration, and V_z is the target climb rate. Then the maximal target climb rate is calculated as

$$V_{zmax} = min(V_z, \frac{a_z T_{leg}}{2})sgn(H) \tag{5.8}$$

The segments of the vertical target trajectory are obtained as follows. In the first segment (vertical acceleration segment), vertical acceleration is equal to a_z, thus the segment duration is equal to

$$T_{zaccel} = \frac{min(V_z, \sqrt{a_z H})}{|a_z|} \tag{5.9}$$

In the second segment (constant climb rate segment) the climb rate is equal to V_{zmax}, and the segment duration is given by

$$T_{zaccel} = \frac{|H|}{min(V_z, \sqrt{a_z H})} - T_{zaccel} \tag{5.10}$$

In the third segment (vertical deceleration segment), the vertical acceleration is equal to $-a_z$ and naturally, the segment duration is equal to the first segment duration T_{zaccel}.

In the fourth segment (zero climb rate segment) the climb rate and the vertical acceleration are both equal to zero, the segment duration is equal

$$T_{zerorate} = T_{leg} - 2T_{zaccel} - T_{climbconst} \tag{5.11}$$

FK Leg calculation

The kinematic model of the target j in FK leg modeling is given by

$$X^j_{k+1} = F^j_k X^j_k + \nu^j_k \tag{5.12}$$

F^j_k is the state transition matrix, ν^j_k is the process noise of target j and Q^t_k is its covariance matrix. The target motion can follow different kinematic models, among those white noise acceleration model, Wiener process acceleration model, and coordinated turn model. We assume motion in x-y plane and in z axis to be independent. A choice is given between a number of models for both x-y plane and z axis. There exist a set of the following models for x-y plane movement which have been implemented:

1. Discrete white noise acceleration model:

2. Discrete Wiener process acceleration model:

3. Coordinated turn model

Discrete white noise acceleration model and discrete Wiener process acceleration model were implemented for the motion along the z axis.

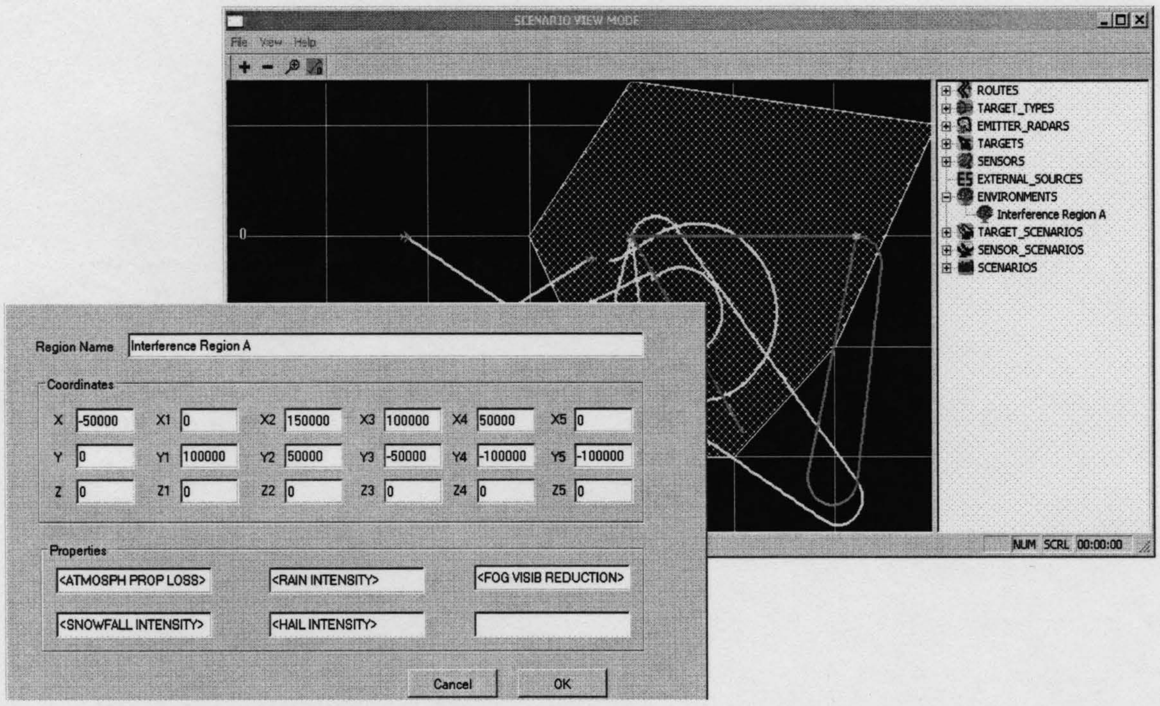
5.2.1.2 Sensor Scenario

The *Sensor Scenario* contains a list of sensors, which have different physical principles and performance regimes. Based on the *Target Scenario*, environment conditions and the Resource Management module commands, the sensors generate simulative measurements, which are the input for the further *Tracker* processing. The measurement generation is performed based on the following factors:

- Parameters of the signal, received by a sensor
- The sensor type and parameters, characterizing the sensor operational ability

5.2.1.3 Environment Scenario Simulation

Environment Scenario simulates physical parameters of the area under surveillance. Complex of physical phenomena affect initiation and propagation of signals, which could be detected by sensors of different constructions and types. Environment parameters are defined for the whole scenario and for particular areas. The *Environment Scenario* contains a list of special areas with particular properties. A proper definition of environment areas and their parameters allows simulating an impact of different



Region Name: Interference Region A

Coordinates

X	-50000	X1	0	X2	150000	X3	100000	X4	50000	X5	0
Y	0	Y1	100000	Y2	50000	Y3	-50000	Y4	-100000	Y5	-100000
Z	0	Z1	0	Z2	0	Z3	0	Z4	0	Z5	0

Properties

<ATMOSPH PROP LOSS>	<RAIN INTENSITY>	<FOG VISIB REDUCTION>
<SNOWFALL INTENSITY>	<HAIL INTENSITY>	

Figure 5.5: Environmental Regions Definition.

artificial and nature sources on overall system noise or propagation losses like built areas, roads, seacoast etc. *Environment Scenario* calculates a signal emission, which is received by a given sensor from a given target body or target equipment taking into consideration environment properties, which affect the signal propagation. The following features can be simulated:

- Weather conditions
- Sea conditions
- Zones with specific propagation losses for radar, acoustic and heating emission
- Clutter zones

Proper definition of environment parameters and areas allows simulating effects of different artificial and nature sources on additional noise or propagation losses. Fig. 5.5 demonstrates defining *Environmental Region*. *Environmental Region* is in its turn a part of the *Environmental Scenario* which together with *Target Scenario* and *Sensor Scenario* form a *Simulation Scenario.*

5.2.2 Tracker Module

The *Tracker Module* analyzes sensor measurements and presents the results graphically in *Real Time Mode*, reflecting the dynamics of observed targets. The main purpose of this module is to create and support the dynamic picture of moving and

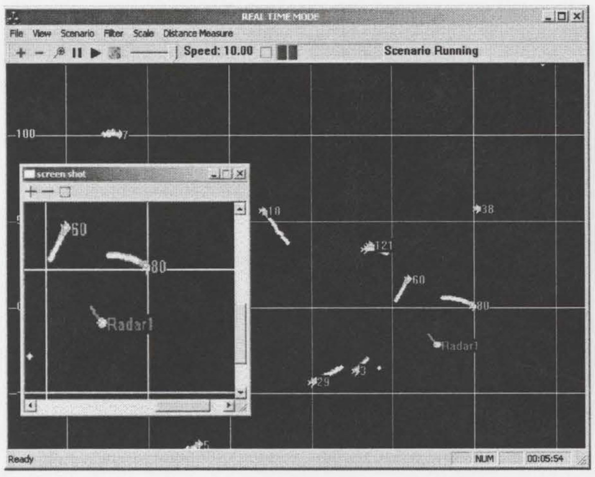


Figure 5.6: IMM/Assignment tracker simulation, one sensor and nine targets

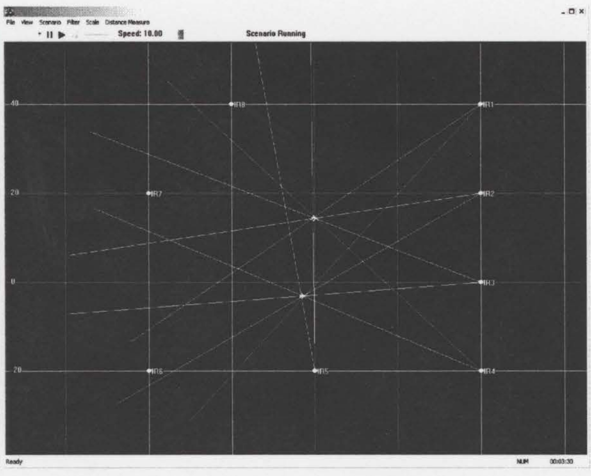


Figure 5.7: Particle Filter tracker simulation, 8 IR sensors and two targets

stationary targets based on the measurements, arriving from the different sensors and external information sources. The tracker module is realized as a dynamic library, which may be selected from the available tracker realizations and connected to the testbed "on-the-fly". Thus a library of trackers is maintained, all of them having a unified interface with the testbed and ability to be dynamically connected to it. Currently there are two implemented trackers in the library - the IMM/Assignment tracker, and the Particle Filter (PF) tracker. Fig. 5.6 shows simulation, involving IMM/Assignment tracker, where there is one sensor (a radar) and nine targets. The small window in the middle of the main simulation screen gives an opportunity to independently view an arbitrary part of the monitored area. Fig. 5.7 shows simulation, involving the PF tracker, with 8 Infra Red (IR) sensors and two targets. Fig. 5.8

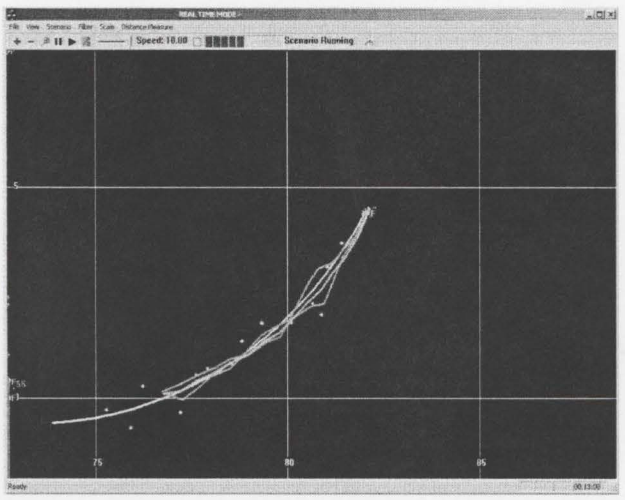


Figure 5.8: Data fusion screenshot. Pink dots designate measurements. Yellow line designates true target trajectory. Purple lines designate tracks from two different sensors. Cyan line designates the fused track.

shows data fusion screenshot. Pink dots designate measurements. Yellow line designates true target trajectory. Purple lines designate tracks from two different sensors. Cyan line designates the fused track. $2-D$ track-to-track association using auction algorithm is employed.

5.2.3 Tracking Performance Metrics Module

This module receives the actual data regarding all the target and sensor locations as well as the results from the *Tracker module* after a pre-defined number of Monte-Carlo runs of a given scenario. It analyzes the provided information and evaluates the quality of the *Tracker module* performance. The required performance evaluation method can be chosen from a list of available ones and its properties can be defined. The following metrics are currently implemented [43]:

- Track Accuracy: At each time step, a 2D assignment is performed between true targets and estimated tracks. Root mean sum squared error (RMSE) history in position and velocity are calculated for the associated targets.
- Completeness History: This is the proportion of the targets that should be tracked which are declared as tracks at each time in the scenario.
- Timeliness: Time at which a particular object that should be tracked has a valid declared track.
- Ambiguity:
  - Redundant track mean ratio: the number of declared tracks that are assignable to real objects that should be tracked, divided by the number of declared valid tracks.
  - Spurious track mean ration: the number of declared tracks that are unassignable to real objects that should be tracked, divided by the number of declared valid tracks.
- Track continuity:
  - Mean cumulative swap of tracks: the cumulative number of swaps of tracks

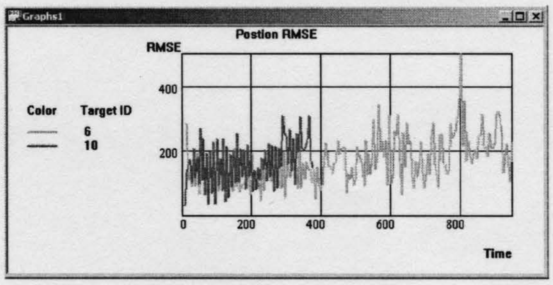


Figure 5.9: RMSE for two particular targets..

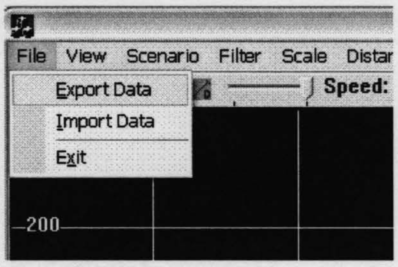
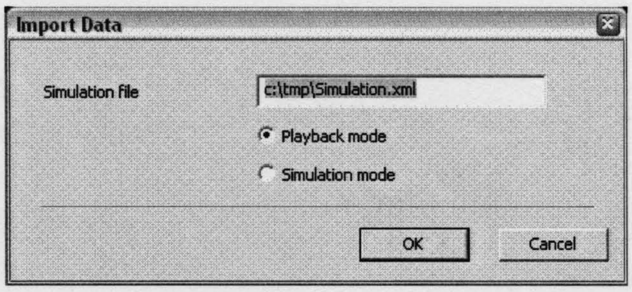


Figure 5.10: Simulation import/export menu options.

for particular objects and averaged across all objects by time t into the scenario.

– Mean cumulative broken tracks: the cumulative number of breaks of tracks for particular objects and averaged across all objects by time t into the scenario.

Fig. 5.9 shows position RMSE for two particular targets.


Import Data

Simulation file: c:\tmp\Simulation.xml

[✓ Playback mode]

Simulation mode

Figure 5.11: Simulation import dialog options: (a) Playback mode (b) Simulation mode

5.2.4 External data support

The testbed employs an Extensible Markup Language (XML) standard for storing its internal data and information exchange between different instances of the testbed. The testbed uses XML for two different tasks. The first one is storing configuration information, created objects and their relations in a database. The second task, is a feature of importing and exporting simulation data of specific scenarios with ability to import them to any particular testbed installations. Two modes are provided for importing and exporting simulation data:

- Playback mode: in this mode all the data related to a specific simulation that is stored as an external XML file is imported to allow exact repeat of this simulation with consequent analysis of its performance.

- Simulation mode: in this mode tracking/measurement information of the recorded simulation is fused in the current simulation, similar to how the track/measurement information arriving from another instance of the testbed is fused.

Chapter 6

Conclusions

In this thesis, we presented a solution for one of the main problems in network-centric tracking — decentralized information sharing among the platforms participating in the distributed data fusion. We proposed a Markov Decision Process based algorithm for controlling the flow of information in a network-centric data fusion architecture. The proposed decision process based algorithm for controlling the information flow in a network-centric data fusion architecture utilizes the approach of using information based objective function as one of the components in the formulated optimization problem. We demonstrated that the approach led to a substantial reduction in data flow volumes, which also incurred a certain price in terms of reduction in performance. In order to minimize communication among the nodes, node lookup is performed using a decentralized lookup substrate. A proposed decision mechanism provides the platforms, which are decentralized, heterogenous and potentially unreliable, with the required data for the distributed data fusion process while reducing redundancy in the information flow in the overall system. The node lookup is performed in $O(\log(N))$ and the computational complexity of solving the MDP is P-complete, which shows that the proposed method is computationally attractive for distributed data fusion applications.

The work applied the suggested approach to three different data fusion methods, namely, associated measurements fusion, track-to-track fusion and tracklet fusion. As one of the main barriers in network-centric tracking is the transfer of tracking data from one platform to another, the analysis included communication load of the three data fusion methods. This communication load should be taken into consideration in the design of the tracking system and cannot be ignored. As mentioned, the task of providing tracking information from one platform to anther node may be infeasible because of the limitations of the communication channel capacity. The lowest data rate was achieved when using MDP based associated measurements data fusion, followed by MDP-based track-to-track or tracklet data fusion simulations. The highest one was registered during track-to-track or tracklet based data fusion simulations when all the data were shared, both of which used the same time interval between data transmissions. In terms of computation time, the best results were achieved when using tracklet based data fusion method when no data were shared among the platforms. The highest computation time resulted in the track-to-track MDP-based data fusion simulation. The best accuracy was achieved when using AMR data fusion sharing all the information, closely followed by MDP-based AMR data fusion results. The worst accuracy was achieved in track-to-track data fusion simulation when no data were shared among the platforms. The best $\Psi\epsilon_{\mathbf{x}\ RMSE}$ metrics was achieved when using MDP based associated measurements data fusion and the worst when using track-to-track data fusion when all the data were shared among the platforms. In summary, associated measurements with MDP-based data fusion yielded the best results in the majority of the metrics categories and track-to-track data fusion with all tracks shared gave the worst results in most of them. That could be explained in part by the need to transmit the elements of covariance matrix combined with the fact that as the computation of the exact cross-covariance matrices in a track-to-track fusion system would hugely increase either computation load, approximation techniques are used used to compute them. Similar reasons can be stated for the tracklet data fusion as both covariance matrices need to be transmitted. Also, due to the common process noise between the tracks, not all correlations between the various versions of the same track, maintained by different local trackers, can be removed by using the tracklets. Therefore, this algorithm is only an approximation and it is reliable only for targets with small maneuvering index.

In this thesis, we also provided an efficient solution the problem of collaborative sensor management using Markov Decision Processes. We presented an altered version of a classical Value Iteration algorithm to calculate the optimal policy for Markov Decision Processes used to address the problem of collaborative sensor management and data fusion for multitarget tracking. Dynamic Element Matching algorithms were successfully used as a core element in the modified algorithm. A number of new introduced performance metrics, such as Mean Opportunity Index, Maximal Opportunity Index and Area Coverage Index, verify the effectiveness of a policy, especially for quantifying the impact of the modified DEM-based Value Iteration algorithm on an MDP policy. The modified algorithm, applied to control a group of UAVs carrying out surveillance over a region that included a number of moving targets, demonstrated an average accuracy improvement expressed in approximately 20 percent reduction in position RMSE of the targets under surveillance. State Coverage Index (SCI) was increased by more than 30 percent. The proposed method demonstrated robust performance while guaranteeing polynomial computational complexity resulting in better detection of new targets. The authors also present multi-level hierarchy of MDPs controlling each of the UAVs. Each level in the hierarchy solves a problem at a different level of abstraction providing UAVs with ability to navigate with a higher precision in a densely populated regions.

Also, we have presented a multisensor-multitarget tracking testbed for large-scale distributed scenarios. This tool gives an opportunity to research into various aspects of distributed tracking systems. Possible application areas are tracking algorithms, performance evaluation methods, target movement models, sensor models, environmental influence on tracking process, noise models pertaining to all the testbed elements, distributed data fusion architectures, security, actual tracking systems implementation, and many others.

6.1 Future Work

In this thesis, we use Markov Decision Processes as one of the tools for determining the optimal policy of actions for sensor management and data fusion control. In some cases, it may be desirable to recognize certain patterns (e.g. in target or sensor formation or communication routing patterns) during decision process which may have an impact on the decision process itself. In the future work we plan on incorporating recent advances in pattern recognition techniques, classification and co-operative training for sensor management and information flow control.

Also, in part of the cases, some of the targets under surveillance are of higher priority than the others. Target recognition and/or classification techniques, using methods such as Observable operator models, Bayesian network, etc, may aid in finding the optimal decision in that case. We also plan on incorporating adaptive and decision level fusion in the decision process.

Using image sensors together with GMTI radars for target surveillance is a possible direction in future work which can enhance target classification and recognition capabilities.

Bibliography

[1] D. Akselrod, C. V. Goldman, A. Sinha, and T. Kirubarajan, "Collaborative Sensor Management for Multitarget Tracking Using Decentralized Markov Decision Processes", *SPIE Defense and Security Symposium, Orlando*, FL, Apr. 17-21, 2006

[2] D. Akselrod, A. Sinha, C. V. Goldman, and T. Kirubarajan, "Efficient control of information flow for distributed multisensor fusion using markov decision processes", *Proc. 9th Int. Conf. on Information Fusion*, Florence, Italy, July 2006.

[3] D. Akselrod, A. Sinha, and T. Kirubarajan, "Collaborative distributed data fusion architecture using multi-level markov decision processes", *Proc. 10th Int. Conf. on Information Fusion*, Quebec, Canada, Jul. 9-12, 2007

[4] D. Akselrod, A. Sinha, and T. Kirubarajan, "Collaborative hierarchical markov decision processes based distributed data fusion and collaborative sensor management for multitarget multisensor tracking applications", *Proc. IEEE International*

Conference on Systems, Man, and Cybernetics (SMC 2007), Montreal, Canada, Oct. 7-10, 2007

[5] D. Akselrod, A. Sinha, and T. Kirubarajan, "Collaborative distributed sensor management for multitarget tracking using hierarchical Markov decision processes", *SPIE Optics & Photonics Symposium*, San Diego, CA, Aug.. 26-30, 2007

[6] D. Akselrod, A. Fish, and O. Yadid-Pecht, "A Mixed Signal Enhanced WTA Tracking System via 2-D Dynamic Element Matching", Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS'2002), Phoenix, AZ, May 26-29, 2002.

[7] Y. Bar-Shalom and X. R. Li, *Multitarget-multisensor tracking: principles and techniques*, YBS Publishing, 1995.

[8] Y. Bar-Shalom and W. D. Blair, Multitarget-Multisensor Tracking: Applications and Advances, Volume III, Artech House, 2000.

[9] Y. Bar-Shalom and H. Chen, "Multisensor track-to-track association for tracks with dependent errors", *Proc. of IEEE CDC, Bahamas*, Dec. 2004.

[10] Y. Bar-Shalom, X. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, Wiley, 2001.

[11] Y. BarShalom, X. R. Li and T. Kirubarajan, Estimation, Tracking and Navigation: Theory, Algorithms and Software, John Wiley & Sons, New York, June 2001.

[12] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson, "Coordinated Target Assignment and Intercept for Unmanned Air Vehicles", *IEEE Trans on Robotics and Automation*, Vol. 18, No. 6, pp. 911-922, Dec. 2002.

[13] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[14] K. Benameur, "Optimal Receiver Location for Emitter Tracking", Proceedings of the American Control Conference, Vol. 6, pp. 4276-4281, Arlington, VA, June 2001.

[15] D. P. Bertsekas, *Linear Network Optimization: Algorithms and Codes*, MIT Press, Cambridge, MA USA, 1991.

[16] D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[17] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, vol. 1,2, 1995.

[18] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of

decentralized control of Markov decision processes", *Mathematics of Operations Research*, 27 (4), 819-840, 2002.

[19] Y. Bin and K. Sycara, "Learning the quality of sensor data in distributed decision fusion", *Proc. 9th Int. Conf. on Information Fusion*, Florence, Italy, July 2006.

[20] S. Blackman, and R. Popoli, Design and Analysis of Modern Tracking Systems, Dedham, MA: Artech House, 1999.

[21] W. D Blair, G. A. Watson, T. Kirubarajan and Y. Bar-Shalom, "Benchmark for radar resource allocation and tracking in the presence of ECM", IEEE Trans. Aerospace and Electronic Systems, Vol. 34, No. 3, pp. 1015-1022, October 1998.

[22] A. Capponi, C. Pilotto, G. Golino, A. Farina, and L. Kaplan, "Algorithms for the selection of the active sensors in distributed tracking: comparison between Frisbee and GNS methods", *Proc. 9th Int. Conf. on Information Fusion*, Florence, Italy, July 2006.

[23] L. R. Carley and J. Kenney, "A 16-bit 4'th order noise-shaping D/A converter", Proceedings of the IEEE 1988 Custom Integrated Circuits Conference, pp. 21.7.1-21.7.4, Rochester, NY, May 1988.

[24] L. R. Carley,"A noise-shaping coder topology for 15+ bit converters", IEEE J.Solid-State Circuits, vol. SC-24, pp.267-273, Apr.1989.

[25] K. C. Chang, C. Y. Chong and Y. Bar-Shalom, "Joint Probabilistic Data Association in Distributed Sensor Networks", IEEE Trans. Automatic Control, Vol. 33(10), pp. 889-897, October 1986.

[26] K. C. Chang, R. K. Saha, and Y. Bar-Shalom, "On optimal track-to-track fusion", *IEEE Trans. on Aerospace and Electronic Systems*, Vol. 33, No. 4, pp. 1271-1276, Oct., 1997.

[27] H. Chen, T. Kirubarajan and Y. Bar-Shalom, "Centralized vs. Distributed Tracking Algorithms for Air to Air Scenarios", Proc. of SPIE Conf. on Signal and Data Processing of Small Targets, Vol. 4048, April 2000.

[28] L. Chen, M. Cetin, and A. Willsky, "Distributed data association for multi-target tacking in sensor networks", *Proc. 8th Int. Conf. on Information Fusion*, PA, July 2005.

[29] H. Chen and Y. Bar-Shalom, "Performance limits of track-to-track fusion versus centralized estimation: theory and application", *IEEE Trans. on Aerospace and Electronic Systems*, Vol. 39, No. 2, pp. 386-400, April, 2003.

[30] C. Y. Chong, S. Mori, and K. C. Chang, "Distributed multitarget multisensor tracking", in *Multitarget-Multisensor Tracking: Advanced Applications*, Chapter 8, pp.247-295, (Y. Bar-Shalom, editor), Artech House, 1990.

[31] C. Y. Chong, "Distributed architectures for data fusion", *Proc. 1st Int. Conf. on Information Fusion*, Las Vegas, NV, July 1998.

[32] C. Y. Chong and S. Mori, "Graphical models for nonlinear distributed estimation", *Proc. 7th Int. Conf. on Information Fusion*, Stockholm, Sweden, June 2004.

[33] C. Y. Chong, F. Zhao, S. Mori, and S. Kumar, "Distributed tracking in wireless ad hoc sensor networks", *Proc. of 6th Int. Conf. on Information Fusion*, Queensland, Australia, July 2003.

[34] C. Y. Chong and S. Mori, "Distributed fusion and communication management for target identification", *Proc. 8th Int. Conf. on Information Fusion*, PA, July 2005.

[35] C. Y. Chong, S. Mori, W. H. Barker, and K. C. Chang, "Architectures and algorithms for track association and fusion", *IEEE Aerospace and Electronic Systems Magazine*, Vol. 15, Issue 1, Jan. 2000.

[36] F. Dambreville and J.P LeCadre, "Detection with Spatial and Temporal Optimization of Search Efforts Involving Multiple Modes and Multiple Resources Management", Proceedings of the 4th Annual Conference on Information Fusion, 2001.

[37] D. Danu, A. Sinha, T. Kirubarajan, M. F. Farooq, and D. Peters "Performance

evaluation of multi-platform distributed data fusion methods for multi-target tracking", *Proc. of IEEE Aerospace Conf.*, Big Sky, MT USA, March 2007.

[38] S. Deb, M. Yeddanapudi, K. R. Pattipati, and Y. Bar-Shalom, "A generalized S-D algorithm for multisensor-multitarget state estimation", *IEEE Trans. on Aero. and Elec. Sys.*, vol. 33, no. 2, pp. 523-538, Apr. 1997.

[39] Z. Ding and L. Hong, "Static/Dynamic Distributed Interacting Multiple Model Fusion Algorithms for Multiplatform Multisensor Tracking", Optical Engineering, pp. 708-715, 1997.

[40] H. Durrant-Whyte and B. Grocholsky, "Management and Control in Decentralised Networks", *Proc. of International Conference on Information Fusion*, pp. 560-565, Cairns, Queensland, Australia, July 2003.

[41] O. E. Drummond, "A hybrid sensor fusion algorithm architecture and tracklets", *Proc. of SPIE Signal and Data Processing of Small Targets*, Vol. 3163, 1997.

[42] O. E. Drummond, "Tracklets and a hybrid fusion with process noise", *Proc. of SPIE Signal and Data Processing of Small Targets*, Vol. 3163, 1997.

[43] R. L. Rothrock and O. E. Drummond, "Performance Metrics for Multiple-Sensor, Multiple-Target Tracking," Signal and Data Processing of Small Targets 2000, Proceedings SPIE, Vol. 4048, pp. 521-531, Jul. 2000.

[44] M. Flint, M. Polycarpou, and E. Fernandez-Gaucherand, "Cooperative Control for Multiple Autonomous UAVs Searching for Targets", *Proc. of IEEE Conference on Decision and Control*, pp. 2823-2828, Las Vegas, Nevada, Dec. 2002.

[45] T. Furukawa, F. Bourgault, H. F. Durrant-Whyte, and G. Dissanayake, "Dynamic Allocation and Control of Coordinated UAVs to Engage Multiple Targets in a Time-Optimal Manner", *Proc. of IEEE Intl. Conf. on Robotics & Automation*, pp. 2353-2358, New Orleans, LA USA, April 2004.

[46] C. V. Goldman, S. Zilberstein, "Goal-Oriented Dec-MDPs with Direct Communication", *University of Massachusetts Computer Science Technical Report #04-44*, 2004.

[47] C. V. Goldman and S. Zilberstein, "Decentralized control of cooperative systems: Categorization and complexity analysis", *Journal of Artificial Intelligence Research*, vol. 22 pp. 143-174, 2004.

[48] M. L. Hernandez, T. Kirubarajan, and Y. Bar-Shalom, "Multisensor resource deployment using posterior Cramér-Rao bounds", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 2, April 2004.

[49] M. L. Hernandez, A. D. Marrs, N. J. Gordon, S. Maskell, and C. M. Reed,

"Cramér-Rao bounds for non-linear filtering with measurement origin uncertainty", *Proceedings of the 5th International Conference on Information Fusion*, Annapolis, Maryland, July 2002.

[50] T. J. Ho and M. Farooq, "Centralized and Decentralized IMM Algorithms for Multisensor Track Fusion", Proc. ONR/NPS Workshop on Estimation, Tracking and Fusion, Monterey, CA, May 2001.

[51] R. A. Howard, *Dynamic Programming and Markov Processes.* The MIT Press, Cambridge, MA, 1960.

[52] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey", *Journal of Artifcial Intelligence Research*, 4:237-285, 1996.

[53] M. Kalandros and L. Y. Pao, "Sensor Management for Tracking Interacting Targets", Proc. The ONR/NPS Workshop on Estimation, Tracking and Fusion, Monterey, CA, May 2001.

[54] T. Kirubarajan, Bar-Shalom, W. D. Blair and G. A. Watson, "IMMPDA Solution to Benchmark for Radar Resource Allocation and Tracking in the Presence of ECM", IEEE Trans. Aerospace and Electronic Systems, Vol. 34, No. 3, pp. 1023-1036, October 1998.

[55] T. Kirubarajan, Y. Bar-Shalom, K. R. Pattipati and I. Kadar, "Ground Target Tracking With Topography-Based Variable Structure IMM Estimator", IEEE Trans. Aerospace and Electronic Systems, AES36(1):2646, Jan. 2000.

[56] T. Kirubarajan, Y. BarShalom and K.R. Pattipati, "Multiassignment for Tracking a Large Number of Overlapping Objects", IEEE Transactions on Aerospace and Electronic Systems, AES37(1), pp. 221, January 2001.

[57] T. Kirubarajan, H. Wang and Y. BarShalom, "Efficient Multisensor Fusion Using Multidimensional Data Association", IEEE Transactions on Aerospace and Electronic Systems, AES37(2), pp. 386400, April 2001.

[58] T. Kirubarajan, H. Wang, and Y. Bar-Shalom, "Efficient multisensor fusion using multidimensional data association", *IEEE Trans. Aerospace and Electronic Systems*, AES-37(2), pp. 386-400, April 2001.

[59] A. Kumar, Y. Bar-Shalom and E. Oron, "Image Segmentation Based on Optimal Layering for Precision Tracking", Partitioning Data Sets, Series in Discrete Mathematics and Theoretical Computer Science, Vol. 19, pp. 155-168, AMS, 1995.

[60] B. H. Leung and S. Sutarja, "Multibit Sigma - Delta A/D converter incorporating a novel class of dynamic element matching techniques", IEEE Trans. Circuits Syst. II, vol.39, pp. 35-51 Jan. 1992.

[61] Li, X. R., "Optimal linear estimation fusionpart VII: dynamic systems", *Proc. of the Sixth International Conference of Information Fusion*, pp. 455 - 462, 2003.

[62] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the complexity of solving Markov decision problems", *Proc. UAI Conf.*, Montreal, QC, Canada, August 1995.

[63] T. W. McLain, P. R. Chandler, and M. Pachter, "A Decomposition Strategy for Optimal Coordination of Unmanned Air Vehicles", *Proc. of American Control Conference*, pp. 369-373, Chicago, IL, June 2000.

[64] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric", *Proc. 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, March 2002.

[65] J. R. Moore and W. D. Blair, "Practical aspects of multisensor tracking", in *Multitarget-Multisensor Tracking: Applications and Advances III* (Y. Bar-Shalom and W. D. Blair, editors), Norwood, Massachusetts: Artech House, 2000.

[66] R. Niu, K. Varshney, M. Moore, and D. Klamer, "Decision fusion in a wireless sensor network with a large number of sensors", *Proc. 7th Int. Conf. on Information Fusion*, Stockholm, Sweden, June 2004.

[67] S. R. Norsworthy, R. Schreier and G. C. Temes, Delta-Sigma Data Converters, Theory, Design, and Simulation. New York: IEEE Press, 1997

[68] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environmen", *IEEE Trans. on Automatic Control*, Vol. 49, No.8, pp. 1292-1302, Aug. 2004.

[69] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov chain decision processes", *Mathematics of Operations Research*, 12(3), 1987.

[70] D. Perugini, D. Lambert, L. Sterling, and A. Pearce, "Distributed information fusion agents", *Proc. 6th Int. Conf. on Information Fusion*, Queensland, Australia, July 2003.

[71] R. J. van de Plassche, "Precision current-source arrangement", U.S. Patent 3 982 172, Sept. 21, 1976.

[72] R. J. van de Plassche, "Dynamic element matching for high-accuracy monolithic D/A converters", IEEE J. Solid-State Circuits, vol. SC-11, pp. 795-800, Dec. 1976.

[73] A. B. Poore, and A. J. Robertson III, "A New Class of Lagrangian Relaxation Based Algorithms for a Class of Multidimensional Assignment Problems", Computational Optimization and Applications, Vol. 8, No. 2, pp. 129-150, Sept. 1997.

[74] M. L. Puterman, *Markov Decision Processes - Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Inc., New York, NY, 1994.

[75] Z. Rabinovich, C. V. Goldman, and J. S. Rosenschein, "The complexity of multiagent systems: The price of silence", *Proc. of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp. 1102-1103, Melbourne, Australia, 2003.

[76] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network", *Proc. ACM SIGCOMM 2001 Conf.*, San Diego, CA, August 2001.

[77] B. Ristic, S. Zollo, and S. Arulampalam, "Performance bounds for maneuvering target tracking using asynchronous multi-platform angle-only measurements", *Proceedings of the 4th International Conference on Information Fusion*, Montreal, Quebec, Aug. 2001.

[78] A. Rogers, R. K. Dash, N. R. Jennings, S. Reece, and S. Roberts, "Computational mechanism design for information fusion within sensor networks", *Proc. 9th Int. Conf. on Information Fusion*, Florence, Italy, July 2006.

[79] Y. Sakina, "Multi-bit $\Sigma\Delta$ Analog-to-Digital Converters with Nonlinearity Correction using Dynamic Barrel Shifting", Electronics research Laboratory, College of Engineering, University of California, Berkley CA, Memorandum No. UCB/ ERL M93/63, 1993.

[80] L. Sevgi, A. Ponsford and H. C. Chan, "An Integrated Maritime Surveillance

System Based on High-Frequency Surface-Wave Radars", IEEE Antennas and Propagation Magazine, Vol. 43(4), pp. 28 - 43, August 2001.

[81] A. Sinha, T. Kirubarajan, and Y. Bar-Shalom, "Optimal cooperative placement of UAVs for ground target tracking with Doppler radar", *Proc. of SPIE Signal Processing, Sensor Fusion, and Target Recognition*, Orlando, FL USA, April 2004.

[82] A. Sinha, T. Kirubarajan, and Y. Bar-Shalom, "Autonomous ground target tracking by multiple cooperative UAVs", *Proc. of IEEE Aerospace Conf.*, Big Sky, MT USA, March 2005.

[83] A. Sinha, H. Chen, D. G. Danu, T. Kirubarajan, and M. Farooq, "Estimation and decision fusion: A survey", *Proc. IEEE Int. Conf. on Engineering of Intelligent Systems*, Islamabad, Pakistan, April 2006.

[84] A. Sinha, T. Kirubarajan, and M. Farooq, "Performance test of a multi-platform distributed data fusion system", Technical report 2006/51, Estimation, Tracking and Fusion Laboratory (ETFLab), Electrical & Computer Engineering Department, McMaster University, Hamilton, Ontario, Canada, June, 2006.

[85] A. N. Steinberg, "Stimulative intelligence", *Proc. National Symp. on Sensor and Data Fusion*, McLean, VA, June 2006.

[86] A. N. Steinberg, "Open networks: generalized multi-sensor characterization", *Proc. of 9th Int. Conf. on Information Fusion*, Florence, Italy, July 2006.

[87] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications", *Proc. ACM SIGCOMM 2001 Conf.*, San Diego, CA, August 2001.

[88] R. S. Sutton and A. G. Barto, *Reinforcement Learning: an Introduction*, MIT Press, Cambridge, MA, 1998

[89] R. Tharmarasa, T. Kirubarajan, and M. L. Hernandez, "Large-scale sensor array management for multitarget tracking", *Accepted in IEEE Trans. Systems, Man and Cybernetics*, 2004

[90] R. Tharmarasa, T. Kirubarajan, M. L. Hernandez, and A. Sinha, "PCRLB based multisensor array management for multitarget tracking", *Accepted for IEEE Transaction on Aerospace and Electronic Systems*, 2005.

[91] H. Van Trees, *Detection, Estimation and Modulation Theory*, vol. I, Wiley, New York, 1968.

[92] P. Tichavsky, C. H. Muravchik, and A. Nehorai, "Posterior Cramér-Rao bounds for discrete-time nonlinear filtering", *IEEE Transactions on Signal Processing*, vol. 46, no. 5, pp. 1386–1396, May 1998.

[93] H. Wang, T. Kirubarajan, and Y. Bar-Shalom, "Large Scale Air Traffic Surveillance Using Imm Estimators With Assignment", IEEE Trans. Aerospace and Electronic Systems, Vol. 35, No. 1, pp. 255-266, Jan. 1999.

[94] G. A. Watson, W. D. Blair, and T. R. Rice, "Enhanced Electronically Scanned Array Resource Management through Multisensor Integration", Proc. SPIE Conf. Signal Processing of Small Targets, Orlando, FL, 3163, pp. 329-340, 1997.

[95] G. A. Watson and G. H. McCabe, "Benchmark Problem with a Multisensor Framework for Radar Resource Allocation and Tracking of Highly Maneuvering Targets, Closely-Spaced Targets, and Targets in the Presence of Sea-Surface-Induced Multipath", Technical Report NSWCDD, Dahlgren, VA, 1999.

[96] R. J. Williams and L. C. Baird III, "Tight performance bounds on greedy policies based on imperfect value functions", *Tech. Rep. NU-CCS-93-14*, Northeastern University, College of Computer Science, Boston, MA, 1993.

[97] E. J. Wright and K. B. Laskey, "Credibility models for multi-source fusion", *Proc. 9th Int. Conf. on Information Fusion*, Florence, Italy, July 2006.

[98] S. W. Yeom, T. Kirubarajan, and Y. Bar-Shalom, Y., "Track Segment Association, Fine-step IMM and Initialization with Doppler for Improved Track Performance", *IEEE Trans. on Aerospace and Electronic Systems*, Vol. 40, No. 1, pp. 293-309, Jan. 2004.

[99] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment", *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 1, pp. 41-53, January 2004.