DISTRIBUTED IMPLEMENTATIONS OF THE PARTICLE FILTER WITH PERFORMANCE BOUNDS

ARASH MOHAMMADI

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE (EECS)
YORK UNIVERSITY
TORONTO, ONTARIO
NOVEMBER 2013

Abstract

The focus of the thesis is on developing distributed estimation algorithms for systems with nonlinear dynamics. Of particular interest are the agent or sensor networks (AN/SN) consisting of a large number of local processing and observation agents/nodes, which can communicate and cooperate with each other to perform a predefined task. Examples of such AN/SNs are distributed camera networks, acoustic sensor networks, networks of unmanned aerial vehicles, social networks, and robotic networks.

Signal processing in the AN/SNs is traditionally centralized and developed for systems with linear dynamics. In the centralized architecture, the participating nodes communicate their observations (either directly or indirectly via a multi-hop relay) to a central processing unit, referred to as the fusion centre, which is responsible for performing the predefined task. For centralized systems with linear dynamics, the Kalman filter provides the optimal approach but suffers from several drawbacks, e.g., it is generally unscalable and also susceptible to failure in case the fusion centre breaks down. In general, no analytic solution can be determined for systems with nonlinear dynamics. Consequently, the conventional Kalman filter cannot be used and one has to rely on numerical approaches. In such cases, the sequential Monte Carlo approaches, also known as the particle filters, are widely used as approximates to the Bayesian estimators but mostly in the centralized configuration.

Recently there has been a growing interest in distributed signal processing algorithms where:

(i) There is no fusion centre; (ii) The local nodes do not have (require) global knowledge of the network topology, and; (iii) Each node exchanges data only within its local neighborhood. Distributed estimation have been widely explored for estimation/tracking problems in linear systems. Distributed particle filter implementations for nonlinear systems are still in their infancy and are the focus of this thesis.

In the first part of this thesis, four different consensus-based distributed particle filter implementations are proposed. First, a constrained sufficient statistic based distributed implementation of the particle filter (CSS/DPF) is proposed for bearing-only tracking (BOT) and joint bearing/range tracking problems encountered in a number of applications including radar target tracking and robot localization. Although the number of parallel consensus runs in the CSS/DPF is lower compared to the existing distributed implementations of the particle filter, the CSS/DPF still requires a large number of iterations for the consensus runs to converge. To further reduce the consensus overhead, the CSS/DPF is extended to distributed implementation of the unscented particle filter, referred to as the CSS/DUPF, which require a limited number of consensus iterations. Both CSS/DPF and CSS/DUPF are specific to BOT and joint bearing/range tracking problems. Next, the unscented, consensus-based, distributed implementation of the particle filter (UCD/DPF) is proposed which is generalizable to systems with any dynamics. In terms of contributions, the UCD/DPF makes two important improvements to the existing distributed particle filter framework: (i) Unlike existing distributed implementations of the particle filter, the UCD/DPF uses all available global observations including the most recent ones in deriving the proposal distribution based on the distributed UKF, and; (ii) Computation of the global estimates from local estimates during the consensus step is based on an optimal fusion rule. Finally, a multi-rate consensus/fusion based framework for distributed implementation of the particle filter, referred to as the CF/DPF, is proposed. Separate fusion filters are designed to consistently assimilate the local filtering distributions into the global posterior by compensating for the common past information between neighbouring nodes. The CF/DPF offers two distinct advantages over its counterparts. First, the CF/DPF framework is suitable for scenarios where network connectivity is intermittent and consensus can not be reached between two consecutive observations. Second, the CF/DPF is not limited to the Gaussian approximation for the global posterior density. Numerical simulations verify the near-optimal performance of the proposed distributed particle filter implementations.

The second half of the thesis focuses on the distributed computation of the posterior Cramér-Rao lower bounds (PCRLB). The current PCRLB approaches assume a centralized or hierarchical architecture. The exact expression for distributed computation of the PCRLB is not yet available and only an approximate expression has recently been derived. Motivated by the distributed adaptive resource management problems with the objective of dynamically activating a time-variant subset of observation nodes to optimize the network's performance, the thesis derives the exact expression, referred to as the dPCRLB, for computing the PCRLB for any AN/SN configured in a distributed fashion. The dPCRLB computational algorithms are derived for both the off-line conventional (non-conditional) PCRLB determined primarily from the state model, observation model, and prior knowledge of the initial state of the system, and the online conditional PCRLB expressed as a function of past history of the observations. Compared to the non-conditional dPCRLB, its conditional counterpart provides a more accurate representation of the estimator's performance and, consequently, a better criteria for sensor selection. The thesis then extends the dPCRLB algorithms to quantized observations. Particle filter realizations are used to compute these bounds numerically and quantify their performance for data fusion problems through Monte-Carlo simulations.

Acknowledgements

This thesis could have not been completed without the encouragement, collaboration, and support of a tremendous number of people.

First and foremost, I would like to extend my sincere thanks to my advisor, Professor Amir Asif for his encouragement, support, and patience throughout my Ph.D. studies. I would definitely not reach this point without his support and encouragements. It was a privilege for me to work with an extraordinary supervisor like him with unlimited patience.

I would like to extend my sincere appreciations to my supervisory committee members, Profs. Natalija Vlajic and Hui Jiang for their great personality, friendly advice and valuable guidance. I wish to thank my thesis defense committee members, Profs. Mark Coates, Spiros Pagiatakis and Suprekash Datta for accepting to be members of the committee.

A special thanks to my family. Words can not express how grateful I am to my mother, father, and my sister for all of the sacrifices that they have made for me.

My special thanks go to my dear wife, Farnoosh, for her unlimited love and endless support. It is really fortunate to have someone who believes in you more than yourself. Farnoosh has supported me constantly during all these years. Her true love provides me double motivation to work harder. I would like to say that the PhD degree is earned by her, not me.

Finally, I would like to thank my fellow colleagues, Mohammad Sajjadieh, Nariman Farsad, Mohammad Reza Faghani, and Alireza Moghadam for their help and support. Special thanks to my best friends, Ehsan Karamad, Alireza Katebi, and Vahid Roshanaei.

Table of Contents

List of Tables

List of Figures

Abbreviations

Abbreviation	**Description**
ACT	Anti-clockwise coordinated turn model
AN	Agent network
AVS	Acoustic vector sensor
BOT	Bearings-only tracking
CA	Constant acceleration model
CCT	Clockwise coordinated turn model
CDF	Cumulative distribution function
CF/DPF	Consensus/fusion distributed implementation of the particle filter
CRLB	Cramér-Rao Lower Bound
CV	Constant velocity model
CSS/DPF	Constrained sufficient statistic based distributed implementation of the particle filter
CSW	Cumulative sum of weights
dPCRLB	Distributed computation of the Posterior Cramér-Rao Lower Bound
DCN	Distributed camera network
DKF	Distributed Kalman filter
DOA	Direction of arrival

DPF	Distributed particle filter
EKF	Extended Kalman filter
FC	Fusion centre
FIM	Fisher information matrix
FR/DPF	Fusion based, reduced order, distributed implementation of the particle filter
GMM	Gaussian mixture model
GPS	Global positioning system
GSS	Global sufficient statistics
IID	Independent and identically distributed
GMM	Gaussian Mixture Model
LPN	Local Processing node
LSS	Local sufficient statistics
MC	Monte Carlo
MMSE	Minimum mean square error
MSE	Mean square error
PCRLB	Posterior Cramér-Rao Lower Bound
PDF	Probability density function
PDN	Power distribution system
PMU	Phasor measurement unit
RHS	Right hand side
RMS	Root mean square error
SIS	Sequential importance sampling
SIR	Sampling importance resampling

SMC	Sequential Monte Carlo
SN	Sensor Network
SNR	Signal to noise ratio
SVM	Support Vector Machines
UAV	Unmanned aerial vehicles
UCD/DPF	The unscented, consensus-based, distributed implementation of the particle filter
UKF	Unscented Kalman filter
UPF	Unscented particle filter
WSN	Wireless sensor network

Symbols

Symbols	Description (section of first occurrence)
N	Number of processing nodes/agents with processing and observation functionalities (2.1)
k	Time/iteration index (2.1)
$[\cdot]^T$	Index T is used as superscript to denote matrix transposition (2.1)
$\mathbf{x}(k)$	State vector at iteration k (2.1)
n_x	Number of state variables (2.1)
l	Index l is used as superscript to refer to one of the N nodes/agents (2.1)
$\aleph^{(l)}_{\text{fuse}}(k)$	The set of neighboring nodes for node l (2.1)
$\lvert \cdot \rvert$	The cardinality operator (2.1)
$\lvert\aleph^{(l)}_{\text{fuse}}(k)\rvert$	Number of neighboring nodes for node l (2.1)
$\mathbf{z}^{(l)}(k)$	Observations made at node l at iteration k (2.1)
$\mathbf{z}(k)$	Global observation vector at iteration k (2.1)
$\aleph^{(l)}_{\text{obs}}(k)$	Set of sensors connected to node l (2.1)
$\boldsymbol{\xi}(\cdot)$	Global uncertainties in the process model (2.1)
$\boldsymbol{\zeta}(\cdot)$	Global uncertainties in the observation model (2.1)

$f(\cdot)$	Global state function (2.1)
$g(\cdot)$	Global observation function (2.1)
$g^{(l)}(\cdot)$	local observation function (2.1)
$\mathbb{E}\{\cdot\}$	Expectation operator (2.1)
$\text{diag}(\cdot)$	A block diagonal matrix of its element (2.1)
$\boldsymbol{R}^{(l)}(k)$	Error covariance matrix of the observations made at node l (2.1)
$\boldsymbol{R}(k)$	Global observation error covariance matrix (2.1)
$\boldsymbol{R}^{ij}(k)$	Covariance matrix between the observation noises of node i and j (2.1)
$\mathcal{G}$	The agent/sensor network's communication graph (2.1)
$\boldsymbol{\nu}$	The node set of the communication graph (2.1)
$\mathcal{E}$	The edge set of the communication graph (2.1)
$\boldsymbol{L}$	Laplacian matrix for graph $\mathcal{G}$ (2.1)
$\Delta_{\mathcal{G}}$	Maximum degree for graph $\mathcal{G}$ (2.1)
$P\left(\mathbf{x}(k)\vert\mathbf{z}(1{:}k)\right)$	Global filtering distribution (2.1.1)
$P(\mathbf{x}(k)\vert\mathbf{x}(k-1))$	State transition model (2.1.1)
$P(\mathbf{z}(k)\vert\mathbf{x}(k))$	Global observation likelihood (2.1.1)
$P(\mathbf{z}^{(l)}(k)\vert\mathbf{x}(k))$	Local observation likelihood (2.1.1)
$P\left(\mathbf{x}(k)\vert\mathbf{z}(1{:}k-1)\right)$	Global prediction distribution (2.1.1)
$P\left(\mathbf{x}(k)\vert\mathbf{z}^{(l)}(1{:}k)\right)$	Local filtering distribution (2.1.1)
$P\left(\mathbf{x}(k)\vert\mathbf{z}^{(l)}(1{:}k-1)\right)$	Local prediction distribution (2.1.1)
$S^{(l)}$	Local sub-system l in reduced-order configuration (2.1.2.2)
$\mathbf{x}^{(l)}(k)$	Local state vector in reduced-order configuration (2.1.2.2)
$f^{(l)}(\cdot)$	Local state function in reduced-order configuration (2.1.2.2)

$n_{x^{(l)}}$	Number of states in the local state vector $\mathbf{x}^{(l)}(k)$ (2.1.2.2)
$\boldsymbol{d}^{(l)}(k)$	The coupling state vector (2.1.2.2)
$\boldsymbol{T}^{(l)}(k)$	The nodal transformation matrix (2.1.2.2)
$\boldsymbol{T}^{(l,j)}(k)$	Shared state transformation matrix (2.1.2.2)
$\hat{\mathbf{x}}(k)$	The global state estimate at iteration k (2.1.2.2)
$\hat{\mathbf{x}}^{(l)}(k)$	The local state estimate at node l (2.1.2.2)
$\hat{\boldsymbol{P}}(k)$	The global covariance matrix (2.1.2.2)
$\hat{\boldsymbol{P}}^{(l)}(k)$	The local covariance matrix at node l (2.1.2.2)
$[\cdot]^{+}$	Moore-Penrose generalized inverse (or the right pseudo inverse) (2.1.2.2)
$\hat{\boldsymbol{P}}^{(l,j)}(k)$	The covariance block corresponding for the shared states between subsystem $S^{(i)}$ and $S^{(j)}$ (2.1.2.2)
$\mathcal{N}(\mu, \Sigma)$	Normal distribution with mean μ and covariance Σ (2.2.1)
$\zeta_{\mathcal{N}}(k)$	White Gaussian state forcing terms (2.2.1)
$\xi_{\mathcal{N}}^{(l)}(k)$	White Gaussian observation noise at node l (2.2.1)
$\boldsymbol{F}(k)$	Linear state model (2.2.1)
$\boldsymbol{G}^{(l)}(k)$	Linear local observation model at node l (2.2.1)
$\hat{\mathbf{x}}(k\|k-1)$	The conditional mean of the state variables at iteration k given observations up to time $k-1$ (2.2.1)
$\boldsymbol{P}(k\|k-1)$	The covariance associated with the estimate $\hat{\mathbf{x}}(k\|k-1)$ (2.2.1)
$\hat{\mathbf{x}}(k\|k)$	The conditional mean of the state variables at iteration k given observations up to time k (2.2.1)
$\boldsymbol{P}(k\|k)$	The covariance associated with the estimate $\hat{\mathbf{x}}(k\|k)$ (2.2.1)
$\hat{\mathbf{x}}(k)$	The same as $\hat{\mathbf{x}}(k\|k)$ (2.2.1)

$\boldsymbol{P}(k)$	The same as $\boldsymbol{P}(k\|k)$ (2.2.1)
$\boldsymbol{\mathcal{K}}(k)$	Kalman gain matrix at iteration k (2.2.1)
$\boldsymbol{Q}(k)$	Covariance matrix of Gauss-Markov linear state model (2.2.1)
$\hat{\boldsymbol{y}}(k\|k)$	Information vector (2.2.1)
$\boldsymbol{Y}(k\|k)$	Information matrix (2.2.1)
$\boldsymbol{i}(k)$	Observation information vector (2.2.1)
$\boldsymbol{I}(k)$	Observation information matrix (2.2.1)
$\boldsymbol{i}^{(l)}(k)$	Local observation information vector at node l (2.2.1)
$\boldsymbol{I}^{(l)}(k)$	Local observation information matrix at node l (2.2.1)
κ	Scaling parameter in the unscented Kalman filter (2.2.1.1)
$\mathcal{W}_i, \boldsymbol{\chi}_i(k)$	Sigma points in the unscented Kalman filter (2.2.1.1)
$\boldsymbol{\chi}_i(k\|k-1)$	Predicted sigma points (2.2.1.1)
$\boldsymbol{\mathcal{Z}}_i(k\|k-1)$	Predicted observation sigma points (2.2.1.1)
$\hat{\mathbf{z}}(k\|k-1)$	Predicted observation estimate (2.2.1.1)
$\boldsymbol{P}_{zz}(k\|k-1)$	The autocovariance of predicted observations (2.2.1.1)
$\boldsymbol{P}_{xz}(k\|k-1)$	The cross-covariance between predicted observation and predicted state estimates (2.2.1.1)
$\mathbb{X}_i(k), W_i(k)$	Particle and its associated weight at iteration k (2.2.2)
N_s	Number of particles in the centralized architecture (2.2.2)
$\delta(\cdot)$	The Dirac delta function (2.2.2)
$q(\mathbf{x}(0:k)\|\mathbf{z}(1:k))$	Proposal distribution (2.2.2)
N_{eff}	Effective sample size (2.2.2)
$\hat{\boldsymbol{y}}^{(\text{fused},l)}(k)$	Fused local information vector at node l (2.3.2)
$\hat{\boldsymbol{Y}}^{(\text{fused},l)}(k)$	Fused local information matrix at node l (2.3.2)

$\boldsymbol{Y}^{(i\cap j)}(k\|k)$	The information matrix of the channel filter between node i and j (2.3.2)
$\boldsymbol{y}^{(i\cap j)}(k\|k)$	The information vector of the channel filter between node i and j (2.3.2)
$\hat{\boldsymbol{i}}^{(\text{fused},l)}(k)$	Fused local observation information vector at node l (2.3.2)
$\boldsymbol{I}^{(\text{fused},l)}(k)$	Fused local observation information matrix at node l (2.3.2)
$\hat{\boldsymbol{x}}^{(\text{fused},l)}(k\|k)$	Fused state estimate at node l (2.3.2)
$\boldsymbol{P}^{(\text{fused},l)}(k\|k)$	Covariance of the fused state estimate at node l (2.3.2)
t	Consensus iteration index (2.4.1)
$X_c^{(l)}(\cdot)$	Consensus variable at node l (2.4.1)
$\boldsymbol{U}$	Consensus matrix (2.4.1)
$\|\| \cdot \|\|_2$	Euclidean $\boldsymbol{L}^2$ distance (2.4.1)
$\rho(.)$	Spectral radius of a matrix (2.4.1)
$r_{\text{asym}}(\boldsymbol{U})$	Asymptotic convergence rate of an average consensus algorithm (2.4.1)
$\lambda_i(\boldsymbol{U})$	Eigenvalues of the consensus matrix (2.4.1)
$\bar{\mathbf{x}}^{(l)}(k)$	Local state estimate computed from the particle set (2.5.1)
$\bar{\boldsymbol{P}}^{(l)}(k)$	Local state estimate computed from the particle set (2.5.1)
$\mathcal{S}(\cdot)$	Global sufficient statistics (GSS) can be computed as a function $\mathcal{S}(\cdot)$ of the local sufficient statistics (LSS) (3.1.1)
$\mathcal{Y}^{(l)}(\cdot)$	Local sufficient statistic (LSS) at node l (3.1.1)
$G(\cdot)$	Global sufficient statistic (GSS) (3.1.1)
$\mathbb{Z}_\theta^{(l)}(\mathbf{x}(k))$	True bearing observation made at node l (3.1.2.1)
$Z_\theta^{(l)}(k)$	Noisy observation made at node l (3.1.2.1)

$G_{\theta,*}(k)$	GSS corresponding to the bearing observations (3.1.2.1)
$\lambda^{(l)}(k)$	Coordinates of observation node l (3.1.2.1)
$\mathbb{Z}_{\phi}^{(l)}(\mathbf{x}(k))$	True azimuth observation made at node l (3.1.2.2)
$Z_{\phi}^{(l)}(k)$	Noisy azimuth observation made at node l (3.1.2.2)
$G_{\phi,*}(k)$	GSS corresponding to the azimuth observations (3.1.2.2)
$\mathbb{Z}_{\mathrm{R}}^{(l)}(\mathbf{x}(k))$	True range observation made at node l (3.1.2.2)
$Z_{\mathrm{R}}^{(l)}(k)$	Noisy range observation made at node l (3.1.2.2)
$G_{\mathrm{R},*}(k)$	GSS corresponding to the range observations (3.1.2.3)
$\mathcal{Y}_{\phi,i}^{(l)}(k)$	LSS corresponding to the bearing observation function (3.1.2.3)
$\mathcal{Y}_{R,*}^{(l)}(k)$	LSS corresponding to the range observation function (3.1.2.3)
$\mathcal{Y}_{\phi,i}^{(l)}(k)$	LSS corresponding to the azimuth observation function (3.1.2.3)
n_{GSS}	Number of global sufficient statistics (3.1.2.5)
$N_c(\boldsymbol{U})$	Total number of consensus iterations required for convergence (3.1.2.5)
$\mathbf{x}_{\mathrm{UKF}}^{(l)}(k\|k-1)$	The local predicted state estimate in the CSS/DUPF (3.2)
$\boldsymbol{P}_{\mathrm{UKF}}^{(l,\mathrm{Fused})}(k)$	Covariance matrix of the proposal distribution in the CSS/DUPF (3.2)
$\hat{\mathbf{x}}_{\mathrm{UKF}}^{(l,\mathrm{Fused})}(k)$	Mean value of the proposal distribution in the CSS/DUPF (3.2)
$\mathbb{X}_i^{(l,\mathrm{LF})}(k), \mathbb{X}_i^{(l,\mathrm{LF})}$	Particles and associated weights at node l used in the local filter (4.1.1)
$\mathbb{X}_i^{(l,\mathrm{FF})}(k), W_i^{(l,\mathrm{FF})}(k)$	Particles and associated weights at node l used in the fusion filter (4.1.2)
N_{FF}	Number of particles used in the fusion filter (4.1.2)
$\boldsymbol{\mu}^{(l)}(k)$ and $\boldsymbol{P}^{(l)}(k)$	Mean and covariance of local particles at node l

	during the filtering step of iteration k (4.1.2)
$\boldsymbol{\nu}^{(l)}(k)$ and $\boldsymbol{R}^{(l)}(k)$	Mean and covariance of local particles at node l
	during the prediction step (4.1.2)
$\mathbf{x}_{c1}^{(l)}(\cdot)$ and $\mathbf{x}_{c2}^{(l)}(\cdot)$	Consensus variables in the CF/DPF algorithm (4.1.2)
ΔT	observations arrive at constant time intervals of ΔT (4.2)
T_c	Update cycle of the fusion filter (4.2)
$\mathbb{X}_i^{(l,\mathrm{MFF})}(k), \mathbb{X}_i^{(l,\mathrm{MFF})}$	Particles and associated weights at node l used in the
	modified fusion filter (4.2)
$\boldsymbol{J}(\mathbf{x}(0\colon k))$	Accumulated FIM associated with the estimate $\hat{\mathbf{x}}(0\colon k)$ (5.1.1)
$\boldsymbol{J}(\mathbf{x}(k))$	Instantaneous FIM associated with the estimate $\hat{\mathbf{x}}(k)$ (5.1.1)
$\boldsymbol{J}(\mathbf{x}(0\colon k+1\mid k))$	Accumulated predictive FIM (5.1.1)
$\boldsymbol{J}(\mathbf{x}(k+1\mid k))$	Instantaneous predictive FIM (5.1.1)
∇ and Δ	The operators for the first and second order partial derivatives (5.1.1)
$\boldsymbol{D}^{**}(k)$	Matrices for recursive computation of the centralized FIM (5.1.1)
$\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(0:k))$	Accumulated local FIM associated with the estimate $\hat{\mathbf{x}}^{(l)}(0\colon k)$ (5.2.1)
$\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$	Instantaneous local FIM associated with the estimate $\hat{\mathbf{x}}^{(l)}(k)$ (5.2.1)
$\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(0:k+1\mid k))$	Accumulated predictive local FIM (5.2.1)
$\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1\mid k))$	Instantaneous local FIM predictive local FIM (5.2.1)
$\tilde{\boldsymbol{J}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$	Alternative form of the instantaneous local FIM (5.2.1)
$\tilde{\boldsymbol{J}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1\mid k))$	Alternative form of the instantaneous predictive local FIM (5.2.1)
$\boldsymbol{C}^{**}(k)$	Matrices for recursive computation of the distributed FIM (5.2.1)
$[\boldsymbol{D}^{**}(k)]^{(l)}$	Local matrices for recursive computation of $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$ (5.2.2)
$\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(0\colon k))$	Accumulated auxiliary FIM (5.3.1)
$\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(k))$	Instantaneous auxiliary FIM (5.3.1)

$\boldsymbol{I}(\mathbf{x}(0\!:\!k))$	Accumulated conditional FIM (5.3.1)
$\boldsymbol{L}(\mathbf{x}(k))$	Instantaneous conditional FIM (5.3.1)
$\boldsymbol{I}(0 : k+1\|k)$	Accumulated conditional predictive FIM (5.3.1)
$\boldsymbol{L}(\mathbf{x}(k{+}1\|k))$	Instantaneous predictive conditional FIM (5.3.1)
$P_c(k)$	Conditional posterior distribution (5.3.1)
N_f	Number of local processing nodes (6.1.1)
$N_{ss}^{(l)}$	Sensors connected to processing node l (6.1.1)
$N_{\max}(k)$	The total number of active observation nodes (6.1.1)
$N_{\max}^{(l)}(k)$	Maximum number of sensors activated by processing node l (6.1.1)
$Z^{(l,m)}(k)$	Local observation at sensor node m connected to node l (6.1.1)
$g^{(l,m)}(\cdot)$	Local observation model at sensor node m connected to node l (6.1.1)
$\zeta^{(l,m)}(\cdot)$	Local uncertainty at sensor node m connected to fusion node l (6.1.1)

1 Thesis Overview

1.1 Introduction

Agent networks (AN) [1], commonly referred to as sensor networks (SN), are collections of individual processing nodes that observe a common phenomenon locally and combine the sensor data to derive some globally meaningful information. A possible configuration for AN/SNs is shown in Fig. 1.1, which uses the centralized topology. The blocks labeled 1 to N in Fig. 1.1 represent the sensing devices, referred in the following discussion as local nodes or simply nodes, and $\mathbf{z}^{(l)}$, for $(1 \leq l \leq N)$, denote the sensor observations transmitted to the fusion centre. Depending on the functionality of the AN/SN, the problem of combining information at the fusion center can be posed either as a detection problem [2], i.e., determining the current state from a finite number of known states, or an estimation problem [3], i.e., estimating the value of some quantity related to the observations. Because of the low cost of sensors and the robustness against network failure due to inherent redundancy in such systems, AN/SNs have attracted considerable attention in recent years. Although originally proposed mainly for military tracking and control devices, agent networks now span a wide array of applications in the scientific, industrial, health-care, agriculture and domestic domains. Owing to the commercial availability of low cost sensors with broadcasting capabilities, AN/SNs have moved over from the research arena into real world. Examples of the AN/SN systems are underwater sensor networks [4], networks of unmanned aerial

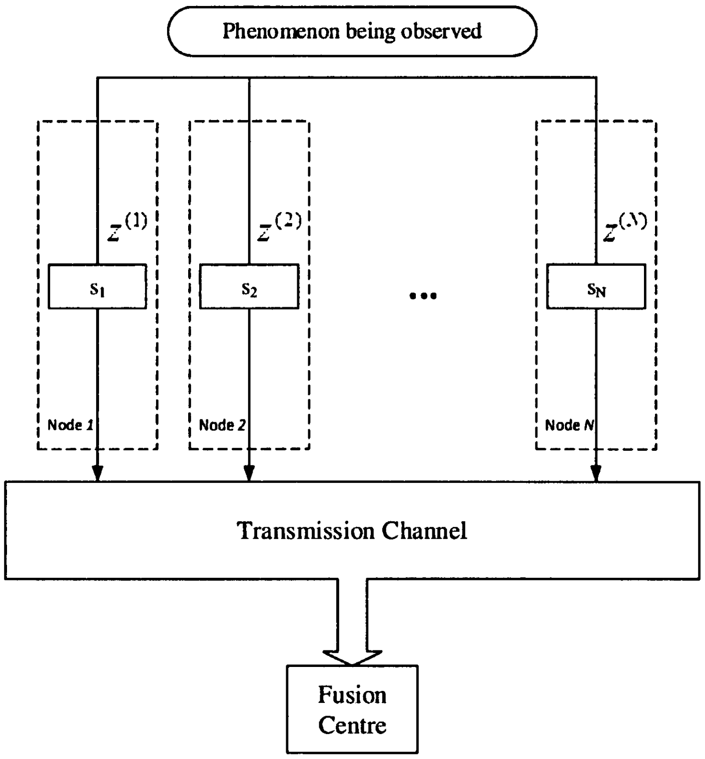


Figure 1.1: Centralized architecture.

vehicles (UAV) [5], robotic networks [6, 7], and camera networks [8]. Some common applications of the AN/SNs are listed below.

- Target tracking [9]: A standard application of the AN/SNs is in surveillance applications where a noncooperative target, such as a vehicle, aircraft, person, or animal, is tracked within the range of the AN/SN system. In the case of passive tracking, the target itself emits a signal that is sensed by the local observation nodes (sensors), and the AN/SN estimates (tracks) time-varying properties of the target such as its position and velocity. The converse case is active tracking, where the probing signal is emitted by the sensor array and its reflection (backscatter) is used for estimating the target properties.

- Industrial control and monitoring [2]: AN/SNs are used to monitor physical and environmental conditions like temperature, pressure, sound, humidity, motion, or pollution. In control applications, the system being monitored shuts off as soon as one of the environmental controls exceeds a pre-determined threshold.
- Home surveillance and consumer electronics [8]: AN/SNs are also used in home surveillance to form a virtual perimeter around a property in order to monitor the progression of intruders by passing information from one node to another.
- Assess tracking and supply chain management [10]: AN/SNs are utilized by warehouses to track the distribution of provisions to different retailers.
- Intelligent agriculture and environmental sensing [11]: AN/SNs are deployed in agricultural farms to control, for example, the supply of water, pesticides, and fertilizers by monitoring the status of the crops.
- Health-care monitoring [12]: AN/SNs are used in health monitoring applications like tracking the posture or movements of a patient. By attaching sensors to the bodies of the patient, their movements can be observed.

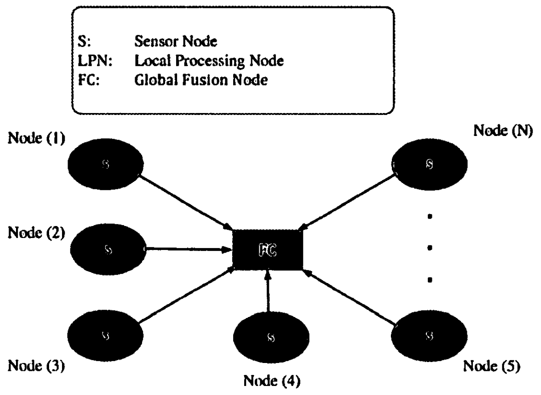
Other possible applications of the AN/SN systems are pollution source localization [13] and chemical plume tracking [14].

Traditional multisensor systems, where local sensors do not perform any preliminary processing of data and a central processor performs the specified operation completely on its own, are referred to as centralized AN/SNs. In Figure 1.1, each local node in the centralized multi-sensor network transfers its raw observation to the fusion node without any processing. A major hurdle faced while designing such centralized AN/SNs is the constraint in the communication bandwidth needed to transmit the observation from a local sensor to the fusion centre. One way of overcoming this hurdle is to perform some preliminary processing [2] of the data at each sensor and then transmit the compressed information to the fusion centre. Alternatively, the fusion centre can be completely eliminated provided that the local nodes cooperate with each other to reach a global solution. Referred to as the distributed or decentralized AN/SNs [1], such networks are said to have intelligence at each node and are the focus of our discussion in this thesis. In the application context considered in the thesis, the local nodes cooperatively estimate certain parameters (or states) of the surrounding environment based on local observations (measurements). They need to cooperate because their local observations are individually insufficient for obtaining reliable estimates. This is where distributed estimation algorithms proposed in the thesis come into play.

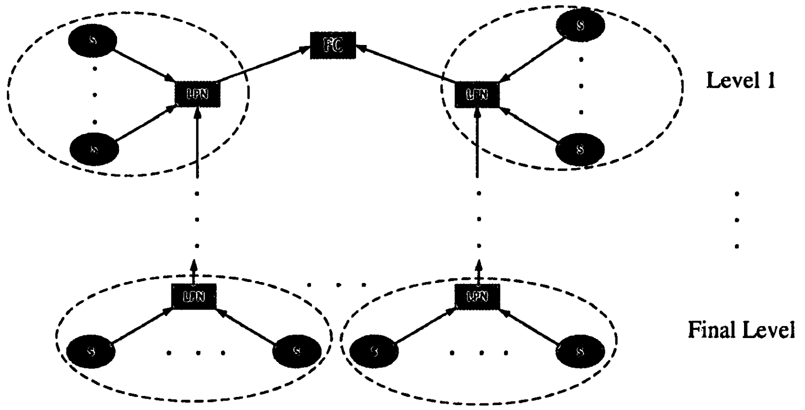
1.1.1 AN/SN Estimation Architectures

As shown in Fig. 1.2 and Fig. 1.3, an AN/SN system can be configured into three main architectures.

1. **Centralized Estimation Architecture** : Traditional state estimation approaches in AN/SNs are centralized (Fig 1.2(a)) where the participating nodes/agents communicate their raw observations (either directly or indirectly via a multi-hop relay) to a central processing unit, referred to as the fusion centre (FC), which is responsible for performing a predefined task. The centralized architecture is simple to implement but is generally unscalable to adding more sensor nodes to the system. It is also susceptible to failure in case the FC breaks down. Another issue is the short life expectation of the sensor nodes. In multi-hop relay communication networks, for example, nodes far away from the FC typically communicate their data to nodes closer to the FC till the FC receives their data. Nodes in the immediate neighbourhood of the FC relay more data which means more massage transfers compared to the nodes far from the FC. Energy consumption (energy required for transferring a massage times the number of massages) is unbalanced in the centralized



(a)



(b)

Figure 1.2: Estimation architectures. (a) Centralized; (b) Hierarchical. In a centralized architecture, all nodes forward their observations to the fusion centre, which estimates the overall state of the system. In a hierarchical architecture, observations are first forwarded to the local processing nodes. Local processing nodes then, transfer partial or fully processed data either to the fusion centre or to another local processing node in a lower level.

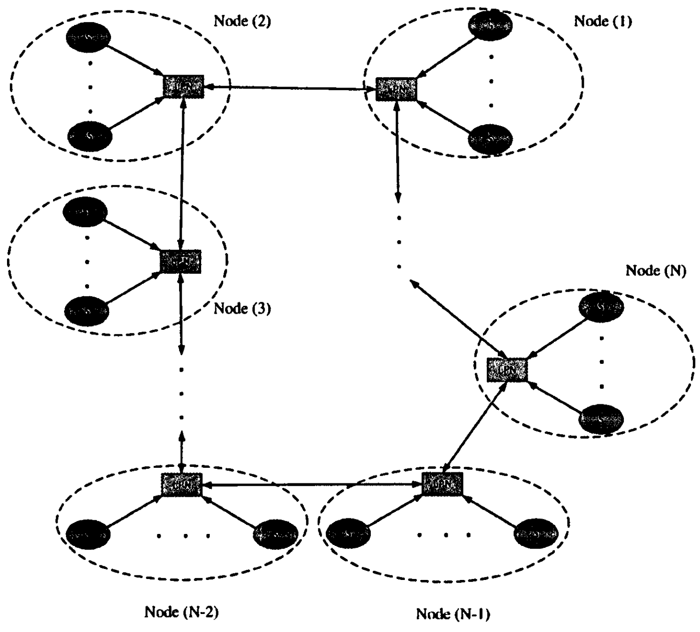


Figure 1.3: Distributed estimation architecture.

network, and mostly concentrated near the FC. Over time, such a mechanism depletes the nodes closer to the FC leading to a system failure. An additional complexity in centralized estimation arises with a change in the network topology requiring the routing tables to be redesigned adding to the complexity of the centralized architecture.

2. Hierarchical Estimation Architecture : In the hierarchical architecture (Fig 1.2(b)), a subset of sensor nodes is associated with a local processing node (local fusion centre) to which local observations from the associated sensor nodes are transferred. Instead of sending raw observations, local processing nodes first process the local observations and then communicate partial or fully processed local data to the FC. In other words, communication from the observation nodes to the FC takes place via the processing nodes. The overall performance of the system still depends on the FC to combine the local processed data into a global state estimate. Though the computation burden in the hierarchical estimation is shared by the FC and local processing nodes, the hierarchical architecture still faces several of the issues discussed for the centralized estimation including a single point of failure and scalability problems.

3. Distributed Estimation Architecture: Recently, there has been a growing interest in distributed estimation algorithms. Fig 1.3 shows an example of a distributed estimation architecture [15] which entail a scenario with two different type of local nodes: (i) *Observation nodes* (sensors) with limited power which only record data, and; (ii) *Local processing nodes* with higher power resources. Each local processing node computes its local track based only on the observations limited to the active sensors connected to it and then cooperates distributively with other local processing nodes in its neighbourhood to compute the global state estimate. Note that in such a distributed architecture there is no global FC, therefore, the sensors and the local processing nodes do not require global knowledge of the network topology. Further, each local processing node collects data from the sensors within its communication range and exchanges data only with other local processing nodes in its local neighbourhood. Such a distributed architecture offers three advantages over the centralized topology.

1. Fusion occurs locally and the successful operation of the network is not dependent on the global FC.

2. Global knowledge of the network topology is not needed locally. Instead, each node only establishes connections with its neighboring nodes.

3. Communication occurs on a node-to-node basis within local neighbourhoods.

The thesis focuses on developing distributed estimation/tracking algorithms for AN/SN based on the architecture presented in Fig. 1.3. Initially, I consider the limiting case where all nodes within a neighbourhood serve the dual task of sensing locally and processing the local collection

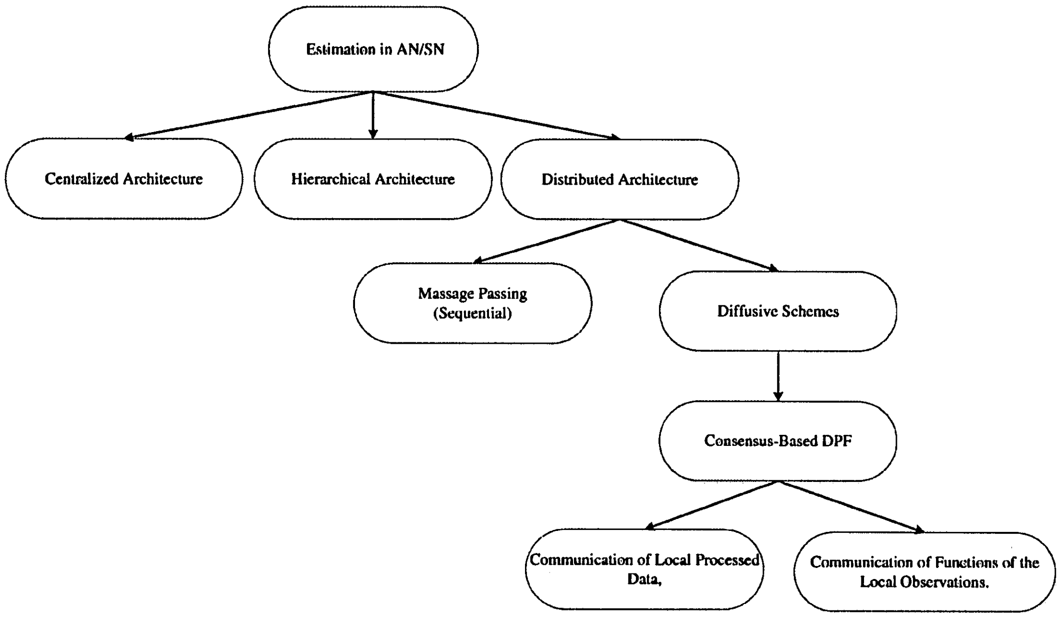

Estimation in AN/SN
Centralized Architecture
Hierarchical Architecture
Distributed Architecture
Massage Passing (Sequential)
Diffusive Schemes
Consensus-Based DPF
Communication of Local Processed Data,
Communication of Functions of the Local Observations.

Figure 1.4: Taxonomy of estimation algorithms in networked systems.

of observations. Such a setup is used to develop the distributed estimation approaches and establishing the performance bounds. Subsequently, I extend these results to the more specialized setup proposed in Fig. 1.3.

1.1.2 Classification of Distributed Estimation Algorithms in AN/SN

Fig. 1.4 presents a taxonomy of estimation algorithms in AN/SNs, namely centralized, hierarchical, and distributed, based on the network architecture. Distributed estimation approaches can be further classified into the following two categories based on the type of communication used in the underlying AN/SN.

1. **Message Passing Schemes** [16, 17]: where the information flows in a sequential, *pre-defined* manner from a node to one of its neighboring nodes via a cyclic path till the entire network is traversed.

2. **Diffusive Schemes** [18–30]: where each node communicates its local information by interacting only with its immediate neighbours. In dynamical environments where frequent changes in the underlying network are a common practice, diffusive approaches significantly improve the robustness of the system.

A promising member of the diffusive algorithms are the consensus approaches [31–35], which are simple distributed methods with minimal computation, communication and synchronization requirement used to fuse local quantities that are scattered across the network. Type of information (local quantities) communicated across the network varies from raw data such as local observations or some elementary function of the local observations [18–22] to processed data such as local likelihoods and state posterior/filtering estimates evaluated at individual nodes [23–27]. As described below, a further classification of the distributed estimation algorithms is based on the portion of the overall state vector estimated at each local node.

1. **Full-order algorithms:** where the entire state variables are estimated at each node. A drawback of such algorithms is that each node needs to maintain an estimate for all of the state variables.

2. **Reduced-order Algorithms:** where a subset of state variables in the global state-vector is estimated at each node based on the local measurements and the information transmitted from the neighboring nodes. Reduced-order algorithms are suitable for large-scale dynamical systems [37–40], where the dimension of the state vector is large and the observations are sparse with only a few state variables being measured at the local nodes. A drawback of such algorithms is that the estimate of the entire state vector is not available locally.

1.1.3 Distributed Particle Filters

Estimation and tracking techniques are usually based on probabilistic methods (Bayesian framework), a standard approach for distributed estimation and data fusion problems. For linear systems with Gaussian excitation and observation noise, the Kalman filter [41, 42] provides the optimal approach. In general no analytic solution can be determined for systems with non-linear dynamics and non-Gaussian forcing terms. Consequently, the direct Kalman filter cannot be used and one has to rely on numerical approaches. In such cases, the sequential Monte Carlo (SMC) approaches [43], also known as the bootstrap filtering, condensation algorithm, and particle filters, are used as approximates to the Bayesian estimators. The particle filters are the SMC (on-line) analogue of the extended/unscented Kalman filters [44] with the added advantage that they approach the optimal Bayesian estimators if sufficient samples of the posterior distribution are available. Since the seminal work by Gordon *et al.* [45], the particle filters have been widely used in the centralized configuration. Developing hierarchical [46–48] and distributed implementation of the particle filter is computationally demanding and requires large bandwidth for information transfers between the local nodes. Although distributed estimation have been widely explored for estimation/tracking problems in linear systems, distributing particle filters implementations for non-linear systems are still in their infancy. Recent developments in the hardware and advances in communication, however, have paved the way for the development of distributed implementations of the particle filter.

My thesis focus on consensus-based distributed implementations of the particle filters for AN/SN systems with non-linear dynamics and non-Gaussian forcing terms. Next, I briefly review the major contributions [49–69] of the thesis.

1.2 Thesis Contributions

I focus on the following research problems in the thesis.

1. **Consensus-Based Distributed Implementation of the Particle Filter** [49,50,57–61]: I propose three consensus-based, distributed implementations of the particle filter.

    - **The CSS/DPF and the CSS/DUPF** [49, 57]: I propose a constrained sufficient statistic based distributed implementation of the particle filter (CSS/DPF) for bearing-only tracking (BOT) and joint bearing/range tracking problems encountered in a number of applications including radar target tracking and robot localization. The CSS/DPF runs localized particle filters at each node to compute the global sufficient statistics of the overall likelihood as a function (summation) of the local sufficient statistics.

      Pros and cons: Existing distributed consensus-based particle filter implementations proposed in the literature [20, 22] require a large number of parallel consensus runs at each iteration of the particle filter which adds considerable consensus overhead to the distributed estimator. The CSS/DPF is proposed with the goal of developing a distributed particle filter that has reduced consensus overhead and affordable complexity. In the CSS/DPF, the number of parallel consensus runs is reduced to 6 for 2-D BOT, 16 for 3-D BOT, and 12 for joint bearing/range tracking. The proposed CSS/DPF still depends on the convergence of each of the consensus runs which itself requires a large number of consensus iterations. To further reduce the consensus overhead, the CSS/DPF is extended to a distributed implementation of the unscented particle filter, referred to as the CSS/DUPF, which require limited number of consensus iterations. Although computationally efficient, the CSS/DPF and CSS/DUPF are highly specialized and restricted to applications where the global sufficient statistics (GSS) can be expressed as a linear combination (summation) of the local sufficient statistics (LSS). The CSS/DPF and CSS/DUPF can not be generalized to any system.

- **The UCD/DPF** [50,59]: The unscented, consensus-based, distributed implementation of the particle filter (UCD/DPF) [59] couples the unscented Kalman filter (UKF) with the particle filter such that the UKF estimates the Gaussian approximation of the proposal distribution which is used to generate new particles for the next iteration of the particle filter.

  Pros and cons: In terms of contributions, the UCD/DPF makes two important improvements to the existing distributed particle filter framework: (i) Unlike existing distributed implementations [24,27] of the particle filter, the UCD/DPF uses all available global observations including the most recent ones in deriving the proposal distribution based on the distributed UKF, and; (ii) Computation of the global estimates from local estimates during the consensus step is based on an optimal fusion rule. Improvement (ii) replaces the commonly used local averaging approach and, along with (i), enhances the performance of the UCD/DPF. Further, the UCD/DPF paves the way for incorporating future developments in consensus-based distributed Kalman filters to the distributed particle filtering framework. However, the UCD/DPF approximates the global posterior with a Gaussian distribution. A second limitation of the UCD/DPF is the requirement on each node to wait until consensus is reached before running the next iteration of the particle filter. This is possible only in networks where communication is relatively inexpensive as compared to sensing, i.e., in rendezvous control or coordination of mobile sensors. I propose the CF/DPF framework, presented next, to address these issues.

2. **The CF/DPF Framework** [51, 52, 62, 63]: A major problem in distributed estimation networks is unreliable communication (especially in large and multi-hop networks), which results in communication delays and information loss. Referred to as intermittent network connectivity, this issue has been investigated broadly in the context of the Kalman filter. Such methods are, however, limited to linear systems and have not yet been extended to non-linear systems. The thesis addresses this gap. I propose a multi-rate consensus/fusion based framework for distributed implementation of the particle filter referred to as the CF/DPF. The CF/DPF framework is based on running localized particle filters to estimate the overall state vector at each observation node. Separate fusion filters are designed to consistently assimilate the local filtering distributions into the global posterior by compensating for the common past information between neighbouring nodes. The CF/DPF offers two distinct advantages over its counterparts. First, the CF/DPF framework is suitable for scenarios where network connectivity is intermittent and consensus can not be reached between two consecutive observations. Second, the CF/DPF is not limited to the Gaussian approximation for the global posterior density.

3. **Distributed Computation of the PCRLB** [53–55, 64]: In order to evaluate the performance of the proposed *distributed*, *non-linear* framework, I derive the posterior Cramér-Rao lower bounds (PCRLB), (also referred in literature as the Bayesian CRLB). The current PCRLB approaches assume a centralized or hierarchical architecture. The exact expression for distributed computation of the PCRLB is not yet available and only an approximate expression [15] has recently been derived. The thesis derives the exact expression, referred to as the dPCRLB, for computing the PCRLB for a AN/SN configured in a distributed fashion.

    - **Conditional dPCRLB**: Motivated by the distributed adaptive resource management problems, the thesis derives recursive expressions for online computation of the *conditional* dPCRLB [55]. Compared to the *non-conditional* PCRLB, the conditional PCRLB is a function of the past history of observations made and, therefore, a more accurate representation of the estimator's performance and, consequently, a better criteria for sensor selection. Previous algorithms to compute the conditional PCRLB are limited to centralized architectures, which involve a FC, thus making them unsuitable for distributed topologies. The thesis also addresses this gap.

4. **Distributed Sensor Selection** [56, 65–67]: I consider the problem of sensor resource management for distributed, nonlinear estimation applications with the objective of dynamically activating a time-variant subset of observation nodes to optimize the network's performance [67]. The PCRLB is a predictive benchmark of the tracker's achievable performance and has recently been proposed as a criteria for sensor selection. Existing PCRLB-based selection techniques are, however, primarily limited to centralized and hierarchical architectures, and when extended to distributed topologies use approximate expressions for computing the PCRLB. I propose a dPCRLB-based observation node selection procedure for distributed sensor networks. A combination of minimum and average consensus algorithms are used to select a subset of observation nodes.

1.3 Organization of the Thesis

Chapter 1 provided an overview and a summary of important contributions made in the thesis. The rest of the thesis is organized as follows.

- Chapter 2 presents an introduction to the problem of distributed state estimation. A classification of the existing distributed estimation algorithms is provided including their applications to some practical estimation problems.

- Chapter 3 considers the problem of consensus-based distributed implementation of the particle filter. Different consensus-based distributed implementations of the particle filter are proposed.

- Chapter 4 introduces the proposed CF/DPF framework. In the CF/DPF, the fusion filters can run at a rate different from that of the local filters. I further investigate this multi-rate nature of the proposed framework, recognize three different scenarios, and describe how the CF/DPF handles each of them. For the worse-case scenario with the fusion filters lagging the local filters exponentially, I derive a modified-fusion filter algorithm that limits the lag to an affordable delay.

- In Chapter 5, I derive distributed expressions for computing the PCRLB for an AN/SN configured in a distributed topology referred to as the dPCRLB. I consider both full-order and reduced-order distributed estimation problems and derive algorithms for computing the dPCRLB for each case.

- In Chapter 6, I consider distributed sensor selection problem where I propose dPCRLB-based algorithms for dynamically selecting a subset of sensors.

- Chapter 7 concludes the thesis and provides some directions for future work.

To maintain consistency in the thesis, each chapter includes numerical simulations related to the results presented in that chapter.

1.4 Publications

The following are the publications published or under revision from this dissertation research.

Chapter 3: Consensus-based Distributed Implementation of the Particle Filter

1 A. Mohammadi and A. Asif, "Application of Constraint Sufficient Statistics to Distributed Particle Filters: Bearing/Range Tracking," manuscript of 30 pages submitted to *IEEE Transactions on Signal Processing*, 2013.

2 X. Zhong, A. Mohammadi, A. B. Premkumar and A. Asif, "A Distributed Unscented Particle Filtering Approach for Multiple Acoustic Source Tracking Using an Acoustic Vector Sensor Network," manuscript of 30 pages submitted to *Elsevier Signal Processing*, 2013.

3 A. Mohammadi and A. Asif, "Consensus-based Particle Filter Implementations for Distributed Non-linear Systems", chapter 9 in *Nonlinear Est. & Applications to Industrial System Control*, Editor G. Rigatos, 2011.

4 A. Mohammadi and A. Asif, "A Constraint Sufficient Statistics based Distributed Particle Filter for Bearing Only Tracking", in Proceedings of *IEEE International Conference on Communication (ICC)*, pp. 3670-3675, 2012.

5 A. Mohammadi and A. Asif, "Consensus-Based Distributed Unscented Particle Filter," in Proceedings of *IEEE Statistical Signal Processing Workshop (SSP)*, pp. 237-240, 2011.

6 X. Zhong, A. Mohammadi, Wenwu Wang, A.B. Premkumar, and A. Asif, "Acoustic Source Tracking in a Reverberant Environment Using a Pairwise Synchronous Microphone Network," in Proceedings of *IEEE International Conference on Information Fusion*, 2013.

7 A. Mohammadi and A. Asif, "A Distributed Consensus Plus Innovation Particle Filter for Networks with Communication Constraints", manuscript of 4 pages submitted to *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

Chapter 4: Distributed Particle Filter with Intermittent/Irregular Consensus Convergence

8 A. Mohammadi and A. Asif, "Distributed Particle Filter Implementation with Intermittent/Irregular Consensus Convergence," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2572-2587, May 15, 2013.

9 A. Mohammadi and A. Asif, "Full Order Nonlinear Distributed Estimation in Intermittently Connected Networks," in Proceedings of *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

10 A. Mohammadi and A. Asif, "Full order distributed particle filters for intermittent connections: Feedback from fusion filters to local filters improves performance," in Proceedings of *IEEE Statistical Signal Processing Workshop (SSP)*, pp. 524-527, Aug. 2012.

11 A. Mohammadi and A. Asif, "A Consensus/Fusion based Distributed Implementation of the Particle Filter," in Proceedings of *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 285-288, Dec. 2011.

Chapter 5: Computation of the PCRLB for Distributed Architectures (dPCRLB)

12 A. Mohammadi and A. Asif, "Posterior Cramer-Rao Bounds for Full and Reduced-order Distributed Bayesian Estimation", manuscript of 39 pages accepted with minor revision in *IEEE Transactions on Aerospace & Electronic Systems*, 2013.

13 A. Mohammadi and A. Asif, "Decentralized Conditional Posterior Cramer-Rao Lower Bound for Nonlinear Distributed Estimation," *IEEE Signal Processing Letters*, vol. 20, no. 2, pp. 165-168, 2013.

14 A. Mohammadi and A. Asif, "Theoretical Performance Bounds for Reduced-order Linear and Nonlinear Distributed Estimation," in Proceedings of *IEEE Global Communiactions Conference (GLOBECOM)*, pp. 3905-3911, Dec. 2012.

15 A. Mohammadi and A. Asif, "Distributed Posterior Cramer-Rao Lower Bound for Nonlinear Sequential Bayesian Estimation," in Proceedings of *IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pp. 509-512, June 2012

Chapter 6: Sensor Selection in Distributed Networks

16 A. Mohammadi, A. Asif, X. Zhong, and A. B. Premkumar, "Distributed Computation of the Conditional PCRLB for Quantized Decentralized Particle Filters", manuscript of 4 pages submitted to *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, available online at: *arXiv:1307.5435*, 2014.

17 A. Mohammadi and A. Asif, "Decentralized Computation of the Conditional Posterior Cramer-Rao Lower Bound: Application to Adaptive Sensor Selection", in Proceedings of *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

18 A. Mohammadi and A. Asif, "Decentralized Sensor Selection based on the Distributed Posterior Cramér-Rao Lower Bound," in Proceedings of *IEEE International Conference on Information Fusion*, pp.1668-1675, July 2012.

19 A. Mohammadi, A. Asif, X. Zhong, and A.B. Premkumar, "Decentralized Bayesian Estimation with Quantized Observations: Theoretical Performance Bounds," in Proceedings of *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp.149-156, May 2013.

Reduced-order Distributed Estimation

20 A. Mohammadi, A. Asif, and S. Saxena, "Reduced Order Distributed Particle Filter Estimation in Nonlinear Electric Power Grid," manuscript of 29 pages submitted to *IEEE Journal of Selected Topics in Signal Processing,* Special Issue on *Signal Processing in Smart Electric Power Grid*, 2013.

21 A. Mohammadi and A. Asif, "Distributed State Estimation for Large-scale Nonlinear Systems:A Reduced Order Particle Filter Implementation," in Proceedings of *IEEE Statistical Signal Processing Workshop (SSP)*, pp. 249-252, Aug. 2012.

22 A. Mohammadi and A. Asif, "Distributed particle filtering for large scale dynamical systems," in Proceedings of *IEEE International Multitopic Conference*, pp.1–5, 2009.

23 A. Mohammadi and A. Asif, "Reduced Order Distributed Particle Filter for Electric Power Grids", manuscript of 4 pages submitted to *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

2 Literature Review of Distributed Estimation

Statistical estimation theory deals with situations where the values of unknown parameters need to be evaluated from observations made under a state of uncertainty. The goal is to provide a rational framework for dealing with such situations. The Bayesian approach, the main theme of this chapter, is a well known framework of formulating and dealing with such statistical estimation problems. The literature on Bayesian estimation is vast, therefore, in this chapter, I restrict myself to common approaches such as the Kalman filter [41], extended/unscented Kalman filter [44], or sequential Monte Carlo methods (the particle filter) [43]. Traditionally, these Bayesian approaches were developed for a centralized architecture with a fusion centre responsible for collecting observations from across the agent/sensor network (AN/SN) to compute the overall state estimates. In the mid 90's, research on distributed estimation [3,70–74] was initiated for systems with linear dynamics for which the Kalman filter is the optimal estimator. References [75–81] proposed several distributed implementations of the Kalman filter without requiring a fusion centre. Although distributed estimation has been widely explored for estimation/tracking problems in linear systems, distributed particle filter implementations for non-linear systems are somewhat limited because of their high computational complexity and considerable bandwidth overhead due to a large number of information transfers between neighbouring nodes. In the early 2000, one such attempt for hierarchical architectures [46,47] was considered for systems with non-linear dynamics using the particle filters. For distributed architectures, work on the implementations of the particle filter is still in its infancy. Recent developments in hardware and advances in communication have paved the way for practical distributed implementations of the particle filter for an arbitrary deployed nonlinear AN/SN.

In this chapter, I review the fundamentals of the centralized and distributed Bayesian filtering in Section 2.1. The Kalman filter and particle filter are introduced in Section 2.2. The state of the art distributed implementations of the Kalman filter and particle filter are presented in Sections 2.3 and 2.5, respectively, with Section 2.4 reviewing the consensus approaches used for fusing localized state estimates into the global estimate. Section 2.6 introduces several potential applications for the distributed particle filters proposed in the thesis.

2.1 Background

Consider an AN/SN comprising of N nodes[1] observing a set of n_x state variables

$$\mathbf{x}(k) = [X_1(k), X_2(k), \dots, X_{n_x}(k)]^T, \tag{2.1}$$

where $k \geq 0$ is the time/iteration index, n_x is the number of state variables, and T denotes matrix transposition. The set of neighboring nodes for node l for, $(1 \leq l \leq N)$, is denoted by $\aleph^{(l)}_{\text{fuse}}(k)$. In the case that node l, for example, is connected to all other nodes, $\aleph^{(l)}_{\text{fuse}}(k) = N-1$. Node l makes measurements at discrete time instants k, $(1 \leq k)$ as follows

$$\mathbf{z}^{(l)}(k) = [Z^{(l,1)}(k), \dots, Z^{(l,m)}(k), \dots, Z^{(l,|\aleph^{(l)}_{\text{obs}}(k)|)}(k)]^T, \tag{2.2}$$

[1]The term node here refers to a processing node or an agent with processing and observation functionalities.

where $\aleph_{\text{obs}}^{(l)}(k)$ is the set of sensors connected to node l and $|\cdot|$ is the cardinality operator. The overall state-space representation of the system is given by

$$\text{State Model:} \qquad \mathbf{x}(k) = \boldsymbol{f}\big(\mathbf{x}(k-1), \boldsymbol{\xi}(k)\big) \tag{2.3}$$

$$\text{Observation Model:} \quad \underbrace{\begin{bmatrix} \mathbf{z}^{(1)}(k) \\ \vdots \\ \mathbf{z}^{(N)}(k) \end{bmatrix}}_{\mathbf{z}(k)} = \underbrace{\begin{bmatrix} \boldsymbol{g}^{(1)}(\mathbf{x}(k)) \\ \vdots \\ \boldsymbol{g}^{(N)}(\mathbf{x}(k)) \end{bmatrix}}_{\boldsymbol{g}(\mathbf{x}(k))} + \underbrace{\begin{bmatrix} \boldsymbol{\zeta}^{(1)}(k) \\ \vdots \\ \boldsymbol{\zeta}^{(N)}(k) \end{bmatrix}}_{\boldsymbol{\zeta}(k)}, \tag{2.4}$$

where $\boldsymbol{\xi}(\cdot)$ and $\boldsymbol{\zeta}(\cdot)$ are, respectively, the global uncertainties in the process and observation models. In the Bayesian estimation framework, the objective is to determine the optimal value of the state vector $\mathbf{x}(k)$ given observations $\mathbf{z}(k)$, state dynamics $\boldsymbol{f}(\cdot)$, and statistics for the state and observation uncertainties $\{\boldsymbol{\xi}(k), \boldsymbol{\zeta}(k)\}$.

In this theses, the state and observation functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ can possibly be non-linear, and vectors $\boldsymbol{\xi}(\cdot)$ and $\boldsymbol{\zeta}(\cdot)$ are not necessarily restricted to white Gaussian noise. Examples of the state and observation models for several practical applications are provided later in Section 2.6. The agents/nodes of the network are modeled as vertices of the communication graph $\mathcal{G} = (\boldsymbol{\nu}, \boldsymbol{\mathcal{E}})$, namely as elements of the node set $\boldsymbol{\nu} = \{1, \ldots, N\}$. The edge set $\boldsymbol{\mathcal{E}} \subseteq \boldsymbol{\nu} \times \boldsymbol{\nu}$ represents the network's communication constraints, i.e., if node l can send information to node m then $(l, m) \in \boldsymbol{\mathcal{E}}$. For graph $\mathcal{G}$, the maximum degree $\Delta_{\mathcal{G}} = \max_l |\aleph_{\text{fuse}}^{(l)}(k)|$, where $|\aleph_{\text{fuse}}^{(l)}(k)|$ is the number of neighboring nodes for node l, and $|\cdot|$ denotes the cardinality operator. Also relevant is the Laplacian matrix $\boldsymbol{L}$ for graph $\mathcal{G}$, defined in terms of its elements $\{L_{ij}\}$ with $L_{ll} = |\aleph_{\text{fuse}}^{(l)}(k)|$, $L_{lm} = -1$ if $(l, m) \in \boldsymbol{\mathcal{E}}$, and $L_{lm} = 0$ otherwise.

Unless otherwise stated, the measurement noise at two different nodes is assumed to be uncorrelated, i.e.,

$$\boldsymbol{R}^{(ij)}(k) = \mathbb{E}\{\boldsymbol{\zeta}^{(i)}(k)\boldsymbol{\zeta}^{(j)^T}(k)\} = 0, \tag{2.5}$$

where $\mathbb{E}\{\cdot\}$ is the expectation operator and $\boldsymbol{R}^{(ij)}(k)$ is the covariance matrix between the observation noise of node i and j. Eq. (2.5) results in a block diagonal noise covariance matrix $\boldsymbol{R}(k)$ for the overall system as

$$\boldsymbol{R}(k) = \text{diag}\left[\boldsymbol{R}^{(1)}(k), \boldsymbol{R}^{(2)}(k), \ldots, \boldsymbol{R}^{(N)}(k)\right], \tag{2.6}$$

where diag[$\cdot$] represents a block diagonal matrix with the specified elements arranged along the diagonal, and $\boldsymbol{R}^{(l)}(k)$ is the error covariance matrix for observations made at node l. When sensors are deployed densely and close to each other, such an assumption may not hold anymore. In such scenarios, one can group the nearby sensors on the basis of a specified characteristic function to form sub-systems or cliques with the cliques assumed uncorrelated [17]. Sensors within each subsystem communicate their observations to the processing node associated with that clique. In such a case, $\boldsymbol{R}(k)$ will be block diagonal with each constituent block $\boldsymbol{R}^{(l)}(k)$, for $(1 \leq l \leq N)$, a full matrix.

2.1.1 Centralized Bayesian Estimation

In the following explanation for sequential Bayesian estimation, the evolution of the state variables is modeled as a first-order Markov process[2]. Because of the Markovian property, the value of the state $\mathbf{x}(k)$ in a first order Markov process depends only on the value of the immediately proceeding state $\mathbf{x}(k-1)$ and is independent of both the observations and states proceeding $(k-1)$, i.e.,

$$P\big(\mathbf{x}(k)|\mathbf{x}(0\!:\!k-1), \mathbf{z}(1\!:\!k-1)\big) = P\big(\mathbf{x}(k)|\mathbf{x}(k-1)\big). \tag{2.7}$$

[2]Although the discussion in this section considers a first-order Markov process for the state dynamics (a standard approach in the target tracking problems), the results presented here are generalizable to higher-order Markov processes.

Assuming conditional independence such that given the current state values $\mathbf{x}(k)$, the observation vector $\mathbf{z}(k)$ is conditionally independent of the prior states variables, i.e.,

$$P\big(\mathbf{z}(k)|\mathbf{x}(0\!:\!k)\big) = P\big(\mathbf{z}(k)|\mathbf{x}(k)\big), \tag{2.8}$$

the joint probability distribution of the state variables and the observations up to iteration k is given by

$$P\big(\mathbf{x}(0\!:\!k), \mathbf{z}(1\!:\!k)\big) = P\big(\mathbf{x}(0)\big) \prod_{j=1}^{k} P\big(\mathbf{z}(j)|\mathbf{x}(j)\big) P\big(\mathbf{x}(j)|\mathbf{x}(j-1)\big). \tag{2.9}$$

In the probabilistic form, the estimation problem in the Bayesian framework is equivalent to determining the conditional filtering density $P(\mathbf{x}(k)|\mathbf{z}(1 : k), \mathbf{x}(0))$, i.e., the probability of the state variables for all time instances $k > 0$ given the recorded observations and the knowledge of the initial state $\mathbf{x}(0)$. For simplicity, the initial condition is being omitted from the representation of the filtering density which results in the notation $P(\mathbf{x}(k)|\mathbf{z}(1 : k))$. Using the Bayes' rule the filtering density can be expressed in terms of the *sensor model* and the predicted probability density function as follows

$$P(\mathbf{x}(k)|\mathbf{z}(1 : k)) = \frac{\overbrace{P(\mathbf{z}(k)|\mathbf{x}(k))}^{\text{Likelihood}}\overbrace{P(\mathbf{x}(k)|\mathbf{z}(1 : k-1))}^{\text{Predicted Density}}}{\underbrace{P(\mathbf{z}(k)|\mathbf{z}(1 : k-1))}_{\text{Normalization}}}. \tag{2.10}$$

The denominator $P(\mathbf{z}(k)|\mathbf{z}(1 : k-1))$ in Eq. (2.10) is independent of the state variables and can be set as the normalizing constant, i.e., $P(\mathbf{z}(k)|\mathbf{z}(1 : k-1)) = \alpha$. The second term $P(\mathbf{x}(k)|\mathbf{z}(1 : k-1))$ in the numerator of Eq. (2.10) can be expanded in terms of the *state transition model* $P(\mathbf{x}(k)|\mathbf{x}(k-1))$ and the filtering density $P(\mathbf{x}(k-1)|\mathbf{z}(k-1))$ as follows

$$\begin{aligned} P\big(\mathbf{x}(k)|\mathbf{z}(1\!:\!k-1)\big) &= \int P(\mathbf{x}(k), \mathbf{x}(k-1)|\mathbf{z}(1\!:\!k-1))d\mathbf{x}(k-1) \\ &= \int P(\mathbf{x}(k)|\mathbf{x}(k-1), \mathbf{z}(1\!:\!k-1))P(\mathbf{x}(k-1)|\mathbf{z}(1\!:\!k-1))d\mathbf{x}(k-1). \end{aligned} \tag{2.11}$$

Using the Markovian property (Eq. (2.7)), the above equation reduces to

$$P\big(\mathbf{x}(k)|\mathbf{z}(1\!:\!k-1)\big) = \int P(\mathbf{x}(k)|\mathbf{x}(k-1)) \times P(\mathbf{x}(k-1)|\mathbf{z}(1\!:\!k-1))d\mathbf{x}(k-1). \tag{2.12}$$

Finally, the normalization term $P(\mathbf{z}(k)|\mathbf{z}(1 : k-1))$ in Eq. (2.10) can be expanded using the Chapman-Kolomogrov formula [41] as follows

$$P\big(\mathbf{z}(k)|\mathbf{z}(1:k-1)\big) = \int P\big(\mathbf{z}(k)|\mathbf{x}(k)\big)P\big(\mathbf{x}(k)|\mathbf{z}(1:k-1)\big)d\mathbf{x}(k). \tag{2.13}$$

Eq. (2.10) is referred to as the *observation update step*, and Eq. (2.11) is referred to as the *prediction update step.* In the Bayesian framework, Eqs. (2.10)-(2.13) define a recursive solution to compute the filtering density based on the following steps:

Step 1. *Prediction Update*: Given $P(\mathbf{x}(k-1)|\mathbf{z}(1\!:\!k-1))$ compute $P(\mathbf{x}(k)|\mathbf{z}(1\!:\!k-1))$.

Step 2. *Normalization Update* Compute the normalization factor $P(\mathbf{z}(k)|\mathbf{z}(1:k-1))$.

Step 3. *Observation Update*: Using the sensor model $P(\mathbf{z}(k)|\mathbf{x}(k))$ compute $P(\mathbf{x}(k)|\mathbf{z}(1:k))$.

One method, referred to as the maximum a posteriori (MAP) estimation, obtains the state estimate $\hat{\mathbf{x}}(k)$ by determining the value of $\mathbf{x}(k)$ that maximizes $P(\mathbf{x}(k)|\mathbf{z}(1\!:\!k))$. In multisensor Bayesian estimation, several nodes make their own observations $\mathbf{z}^{(l)}(k)$ based on model (2.4). The conditional probability $P(\mathbf{z}^{(l)}(k)|\mathbf{x}(k))$ then serves the role of a sensor model and can be utilized in the distributed implementation of the Bayesian estimation algorithms. The multisensor form of Bayes' rule requires conditional independence (Eq. (2.5)), which results in the following global likelihood function

$$P\big(\mathbf{z}(k)|\mathbf{x}(k)\big) = P\big(\mathbf{z}^{(1)}(k),\ldots,\mathbf{z}^{(N)}(k)|\mathbf{x}(k)\big) = \prod_{l=1}^{N} P\big(\mathbf{z}^{(l)}(k)|\mathbf{x}(k)\big). \tag{2.14}$$

From Eq. (2.10), we have

$$P\big(\mathbf{x}(k)|\mathbf{z}^{(1)}(k),\ldots,\mathbf{z}^{(N)}(k)\big) = \alpha P\big(\mathbf{x}(k)|\mathbf{z}(1\!:\!k-1)\big)\prod_{l=1}^{N} P\big(\mathbf{z}^{(l)}(k)|\mathbf{x}(k)\big), \tag{2.15}$$

where $\alpha \triangleq P(\mathbf{z}(k)|\mathbf{z}(1:k-1))$ is the normalizing constant. Eq. (2.15) is known as the *independent likelihood pool* [41]. This indicates that the filtering density of state variables $\mathbf{x}(k)$ based on the observation of individual nodes is proportional to the multiplication of the prior density $P(\mathbf{x}(k)|\mathbf{z}(1:k-1))$ with product of the individual likelihood functions $P(\mathbf{z}^{(l)}(k)|\mathbf{x}(k))$ for each sensor node.

2.1.2 Distributed Bayesian Estimation

In centralized estimation, the local observations are directly forwarded to the fusion centre for updating the state estimates. An alternative to the centralized approach is hierarchical estimation where instead of forwarding raw observations to the fusion centre, partially processed data are communicated by the local nodes to the fusion centre. In the hierarchical estimation, the computation burden at the fusion centre is, therefore, reduced. In the literature, the hierarchical estimation is sometimes referred to as decentralized estimation. Finally, distributed estimation is defined as the setup where all nodes perform local computations to derive local estimates. There is no central processing unit available and a fusion step is instead utilized to derive the global estimate from the local estimates. The distributed estimation approaches do not require prior global knowledge of the network topology. Instead, each local node has local network knowledge confined to its immediate neighborhood within which it establishes a direct communication link. The main challenge here is to guarantee that all nodes reach a common reliable estimate of the state variables. In the distributed estimation framework, the global estimate could potentially be sub-optimal due to the localized nature of fusion process. In addition, communication overhead is increased due to the introduction of the fusion step.

The distributed implementations can themselves be classified into two main categories: (i) *Full-order estimation*, which replicate an n_x-order filter at each node estimating all n_x states of the system, and; (ii) *Reduced-order estimation* [84–87], which decomposes the large-scale system into smaller subsystems with only a subset of n_x state variables estimated within each subsystem. For a large-scale dynamical system [37–40], the reduced-order methods are generally more efficient than the full-order implementations both in terms of the computational complexity and the number of transmissions (information transfers) between neighbouring nodes. Next, I review the full-order and reduced-order configurations in the context of the sequential Bayesian estimation.

2.1.2.1 Distributed Full-Order Configuration

In full-order distributed estimation, the distributed full-order estimation model at node l, $(1 \leq l \leq N)$, is given by

$$\begin{aligned} \mathbf{x}(k) &= \boldsymbol{f}\big(\mathbf{x}(k-1), \boldsymbol{\xi}(k)\big) & (2.16) \\ \text{and} \quad \mathbf{z}^{(l)}(k) &= \boldsymbol{g}^{(l)}\big(\mathbf{x}(k), \boldsymbol{\zeta}^{(l)}(k)\big), & (2.17) \end{aligned}$$

where the entire state vector $\mathbf{x}(k)$ is estimated at node l based only on its local observations. After computing the state estimates locally, the local state estimates are fused through interactions between local neighbourhoods in a distributed fashion to form the global estimate. In this thesis, I assume that the global observation model is observable though the local observation model at each node may become unobservable for certain iterations.

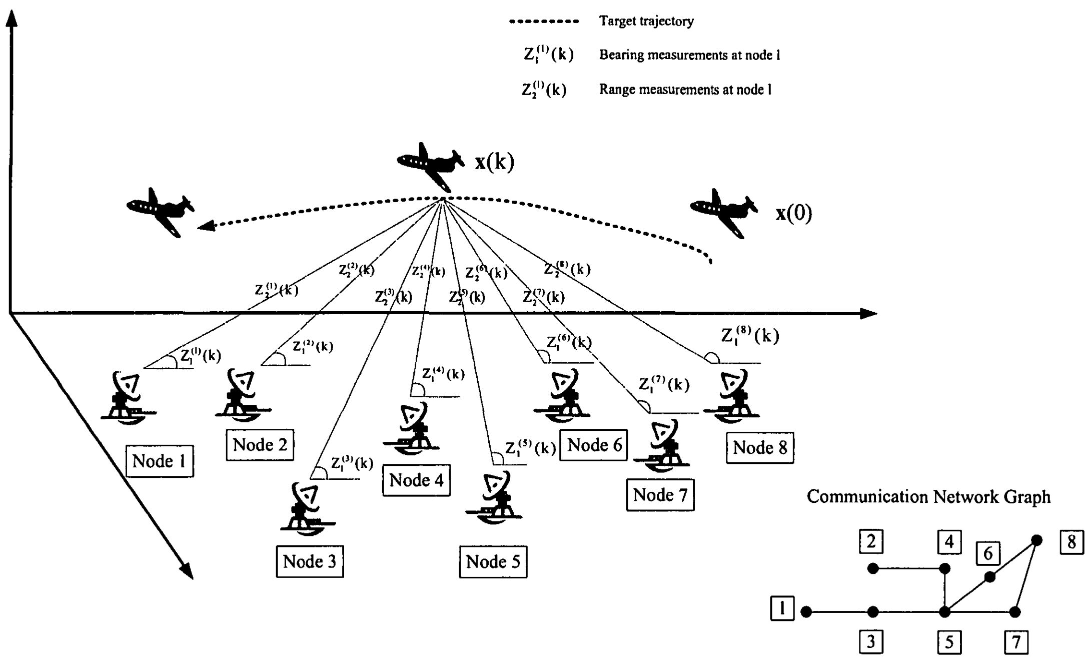
An example of a full-order distributed estimator is the estimation of the 2-D or 3-D spatial location of a moving object over time, e.g., to track an animal in wildlife monitoring, to track an aeroplane or missile in defence applications or to track an object in video surveillance sequences. Figs. 2.1 and 2.2 provide two illustrative examples. Fig 2.1 shows a distributed full-order target tracking application of an aeroplane with eight processing nodes. The state vector $\mathbf{x}(k)$ comprises of the 3D coordinates $\{X(k), Y(k), Z(k)\}$ of the plane and its speed $\{\dot{X}(k), \dot{Y}(k), \dot{Z}(k)\}$ along the three coordinates, i.e., $\mathbf{x}(k) = [X(k), Y(k), Z(k), \dot{X}(k), \dot{Y}(k), \dot{Z}(k)]^T$. Node l, for $(1 \leq l \leq 8)$, makes two measurements $[Z_1^{(l)}(k), Z_2^{(l)}(k)]$ at time k: (i) The bearing/angle $Z_1^{(l)}(k)$ between the node's platform and plane, and; (ii) The range $Z_2^{(l)}(k)$ between the node and plane. Fig. 2.1 depicts the neighbourhood of each node on the sub-graph included on the bottom left of the figure which shows a direct communication link between each pair of neighbouring nodes.

A second illustrative example considered in Fig. 2.2 is the video tracking application, where a distributed camera network with five local nodes (cameras) estimates the 2-D coordinates $\{X_i(k), Y_i(k)\}$ and speed $\{\dot{X}_i(k), \dot{Y}_i(k)\}$ of all five persons over time with the overall state vector $\mathbf{x}_i(k) = [X_i(k), Y_i(k), \dot{X}_i(k), \dot{Y}_i(k)]^T$, for $1 \leq i \leq 5$. As is shown, each camera has a limited field of view and at each time instant k may not be able to observe all five persons. By cooperating with its neighbouring nodes, however, each camera can obtain a reliable estimate of all targets over time assuming that the overall system is observable, i.e., each person is observed by at least one camera at all times.

Generally, two different scenarios are considered for the distributed full-order estimation:

1. **Scenario 1.** (Estimation based only on local measurements): Node l, $1 \leq l \leq N$, updates its local estimates based on its individual measurement $\mathbf{z}^{(l)}(1\!:\!k)$. Local filtering distributions $P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k))$ are then fused into the global posterior $P(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k))$ in a distributed fashion using, for example, a gossip type algorithm.

2. **Scenario 2.** (Estimation based on local measurements and previous global estimate): Same as Scenario 1 except local estimates are based on both the local measurements as well as the previous *global state estimates* (which themselves are based on the collective observations made previously across the network). This leads to local $P(\mathbf{x}(k)|\mathbf{z}(1\!:\!k-1), \mathbf{z}^{(l)}(k))$ being computed at node l. As in Scenario 1, the local filtering estimates $P(\mathbf{x}(k)|\mathbf{z}(1\!:\!k-1), \mathbf{z}^{(l)}(k))$ are then fused into the global posterior $P(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k))$ distributively.



Target trajectory
$Z_1^{(l)}(k)$ Bearing measurements at node l
$Z_2^{(l)}(k)$ Range measurements at node l
x(k)
x(0)
Node 1
Node 2
Node 3
Node 4
Node 5
Node 6
Node 7
Node 8
Communication Network Graph

Figure 2.1: Illustrative Example 1: A distributed target tracking application with 8 nodes. The target is an aeroplane with state vector $\mathbf{x}(k) = [X(k), Y(k), Z(k), \dot{X}(k), \dot{Y}(k), \dot{Z}(k)]^T$, i.e., the plane's 3D location $\{X(k), Y(k), Z(k)\}$ and its speed $\{\dot{X}(k), \dot{Y}(k), \dot{Z}(k)\}$. The local observation $\mathbf{z}^{(l)}(k) = [Z_1^{(l)}(k), Z_2^{(l)}(k)]^T$ at node l, for $(1 \le l \le 8)$, consists of the bearing measurement $Z_1^{(l)}(k)$ and the range measurement $Z_2^{(l)}(k)$. The communication graph corresponding to the processing nodes is included on the bottom left of the figure which shows the communication links between neighbouring nodes.

Scenario 1 is useful for networks with intermittent connectivity where consensus[3] on the local state estimates may not be reached between two consecutive observations. In such cases, two filters are implemented for state estimation. The local filter updates the local states while the global filter derives the overall state estimate from its local counterparts. The local filters continue to assimilate local observations independent of the global filter. Once the global filter has converged,

[3]Consensus in distributed estimation is the process of establishing a consistent value for some statistics of the state vector across the network by interchanging relevant information between the connected neighboring nodes.

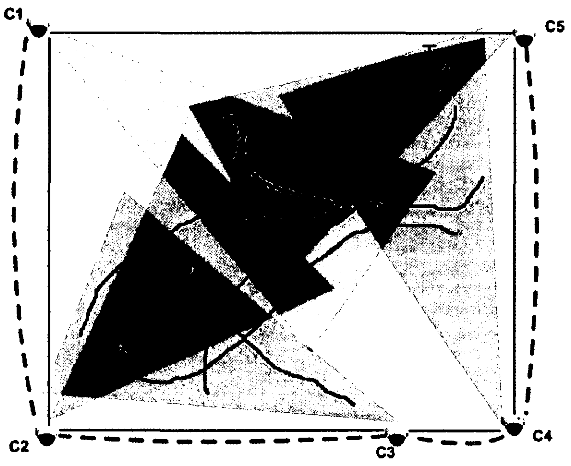


Figure 2.2: Illustrative Example 2: A distributed camera network with five local nodes (cameras) with partially overlapped field of view where each node estimates the 2-D locations of all five persons over time. Each person's track over time is depicted with a different color. The field of view of each camera is also shown with triangles. A communication link (communication link is symmetric) between two neighbouring cameras is shown with the dotted blue line.

it incorporates the recent local states estimates to form the global state estimate. Scenario 2 is useful in applications where communication is relatively inexpensive as compared to sensing, e.g., in rendezvous control or coordination of mobile sensors. Consensus on the state estimates is reached between two consecutive observations. With the availability of the global state estimate, local state estimates are discarded and the next iteration is continued based on the global estimates. Unlike Scenario 1, where the local state estimates at iteration k is computed using the local estimates at iteration $k-1$, Scenario 2 updates the local state estimates at iteration k from the global state estimate at iteration $k-1$.

2.1.2.2 Distributed Reduced-Order Configuration

In large-scale physical systems arising, for example, in meteorology, physical oceanography, or resulting from discretization of partial differential equations, the discretized dynamical models are sparse and localized. The observation $\mathbf{z}^{(l)}(k)$ made at node l, for $1 \le l \le N$, is also localized such that a subset of state variables $\mathbf{x}^{(l)}(k) \subset \mathbf{x}(k)$ (referred to as the local state vector) is observed at each node [84]. For such reduced-order systems $S^{(l)}$, the observation model (Eq. (2.4)) for node l reduces to

$$S^{(l)}: \qquad \mathbf{z}^{(l)}(k) = \boldsymbol{g}^{(l)}\left(\mathbf{x}^{(l)}(k)\right) + \boldsymbol{\zeta}^{(l)}(k). \tag{2.18}$$

The local state vectors in the above equation may have shared states, i.e., $|\mathbf{x}^{(l)}(k) \cap \mathbf{x}^{(j)}(k)| \ge 0$, for $1 \le l, j \le N$, where $|\cdot|$ is cardinality of a set. The reduced-order state-space model is obtained by spatially decomposing the overall system based on the observable states at each node. Other states, if present, are treated as forcing terms. The reduced-order state model at node l (derived from Eq. (2.3) by partitioning) is then given by

$$S^{(l)}: x^{(l)}(k) = \boldsymbol{f}^{(l)}\left(\mathbf{x}^{(l)}(k-1), \boldsymbol{d}^{(l)}(k-1)\right) + \boldsymbol{\xi}^{(l)}(k). \tag{2.19}$$

where $\boldsymbol{d}^{(l)}(k)$ is the coupling state vector. When the overall system is partitioned into subsystems, the dynamical model for a subsystem may contain states that are directly observed by the subsystem and additional states that are not observed but are part of the global state model. The coupling state vector $\boldsymbol{d}^{(l)}(k)$ includes such states which are not directly observed but are part of the subsystem's model. Let $n_{x^{(l)}}$ denote the number of states in the local state vector $\mathbf{x}^{(l)}(k)$. The relationship between the local state vector $\mathbf{x}^{(l)}(k)$ and global vector $\mathbf{x}(k)$ can be expressed as

$$\mathbf{x}^{(l)}(k) = \boldsymbol{T}^{(l)}(k)\mathbf{x}(k), \tag{2.20}$$

with $\boldsymbol{T}^{(l)}(k)$ denoting the $(n_{x^{(l)}} \times n_x)$ nodal transformation matrix [87]. The local process functions are constructed using a similar nodal transformation, i.e., $\boldsymbol{f}^{(l)}(\mathbf{x}^{(l)}(k), \boldsymbol{d}^{(l)}(k)) = \boldsymbol{T}^{(l)}(k)\boldsymbol{f}(\mathbf{x}(k))$. The local state estimate at node l has the same relation to the global state estimate, i.e., $\hat{\mathbf{x}}^{(l)}(k) = \boldsymbol{T}^{(l)}(k)\hat{\mathbf{x}}(k)$. Further, the relationship between the global covariance $\hat{\boldsymbol{P}}(k)$ and local covariance matrix $\hat{\boldsymbol{P}}^{(l)}(k)$ is

$$\hat{\boldsymbol{P}}^{(l)}(k) = \boldsymbol{T}^{(l)}(k)\hat{\boldsymbol{P}}(k)\left[\boldsymbol{T}^{(l)}(k)\right]^T. \tag{2.21}$$

To arrange node l's information $\hat{\boldsymbol{P}}^{(l)}(k)$ in the global state-space, we use the covariance transformation

$$\hat{\boldsymbol{P}}_G^{(l)}(k) = \left[\boldsymbol{T}^{(l)}(k)\right]^+\hat{\boldsymbol{P}}^{(l)}(k)\left[\boldsymbol{T}^{(l)}(k)\right]^{+^T}. \tag{2.22}$$

where $[\boldsymbol{T}^{(l)}(k)]^+$ refers to the Moore-Penrose generalized inverse (or the right pseudo inverse) of $\boldsymbol{T}^{(l)}(k)$, i.e., $[\boldsymbol{T}^{(l)}(k)]^+ = \boldsymbol{T}^{(l)^T}(k)\left[\boldsymbol{T}^{(l)}(k)\boldsymbol{T}^{(l)^T}(k)\right]^{-1}$. Subsystems $S^{(l)}$ and $S^{(j)}$ may have shared states. The shared state transformation matrix $\boldsymbol{T}^{(l,j)}(k)$ is a $(n_{|x^{(l)} \cup x^{(j)}| \times n_x})$ matrix where $|x^{(l)} \cup x^{(j)}|$ is the number of shared states between subsystems $S^{(l)}$ and $S^{(j)}$. Each row of $T^{(l,j)}(k)$ has only one non-zero entry at the location of the shared states. The shared state transformation matrix $\boldsymbol{T}^{(l,j)}(k)$ is used to extract the covariance block

$$\hat{\boldsymbol{P}}^{(l,j)}(k)=\left[\boldsymbol{T}^{(l,j)}(k)\boldsymbol{T}^{(l)}(k)^T\right]\hat{\boldsymbol{P}}^{(l)}(k)\left[\boldsymbol{T}^{(l,j)}(k)\boldsymbol{T}^{(l)}(k)^T\right]^T \tag{2.23}$$

corresponding to the shared states. To arrange the covariance block $\hat{\boldsymbol{P}}^{(l,j)}(k)$ corresponding for the shared states in the global state space $\hat{\boldsymbol{P}}_G^{(l,j)}(k)$, the following covariance transformation (similar to (2.22)) is used

$$\hat{\boldsymbol{P}}_G^{(l,j)}(k) = \left[\boldsymbol{T}^{(l,j)}(k)\right]^+\hat{\boldsymbol{P}}^{(l,j)}(k)\left[\boldsymbol{T}^{(l,j)}(k)\right]^{+^T}. \tag{2.24}$$

To recap, the process model (2.19) and observation model (2.18) collectively provide the nonlinear, localized reduced-order representation for the dynamical system.

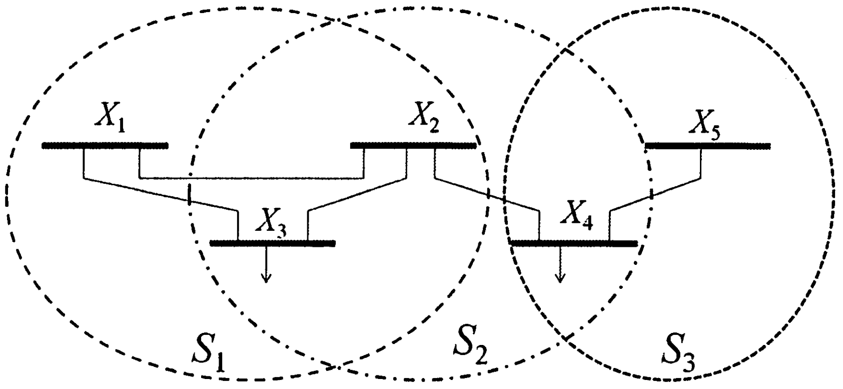


Figure 2.3: Illustrative Example: Spatial decomposition of a nonlinear system with five states into three subsystems S_1, S_2 and S_3.

Illustrative Example:

I illustrate the spatial partitioning procedure with an illustrative example based on a system shown in Fig. 2.3 with five state variables X_1, X_2, X_3, X_4, and X_5 which are partially observed by three distributed nodes $N = 3$. The ranges of the three observation nodes are shown using dotted circles. The overall state model is given by

$$\begin{bmatrix} X_1(k) \\ X_2(k) \\ X_3(k) \\ X_4(k) \\ X_5(k) \end{bmatrix} = \begin{bmatrix} f_1(X_1(k-1), X_2(k-1)) \\ f_2(X_1(k-1), X_2(k-1), X_4(k-1)) \\ f_3(X_1(k-1), X_3(k-1)) \\ f_4(X_3(k-1), X_5(k-1)) \\ f_5(X_4(k-1), X_5(k-1)) \end{bmatrix} + \begin{bmatrix} \xi_1(k) \\ \xi_2(k) \\ \xi_3(k) \\ \xi_4(k) \\ \xi_5(k) \end{bmatrix}, \qquad (2.25)$$

where $f_i(\cdot)$, for $(1 \leq i \leq 5)$, are nonlinear functions. The observation $\mathbf{z}^{(l)}(k)$ at node l, for $(1 \leq l \leq 3)$, is sparse such that only a subset of state variables $\mathbf{x}^{(l)}(k) \subset \mathbf{x}(k)$ (referred to as local state vector) is observed at each sensor node. The local state vectors $\mathbf{x}^{(l)}(k)$ may have shared states i.e., $|\mathbf{x}^{(i)}(k) \cap \mathbf{x}^{(j)}(k)| \geq 0$, for $(1 \leq i, j \leq N)$, where $|\cdot|$ denotes the cardinality of a set.

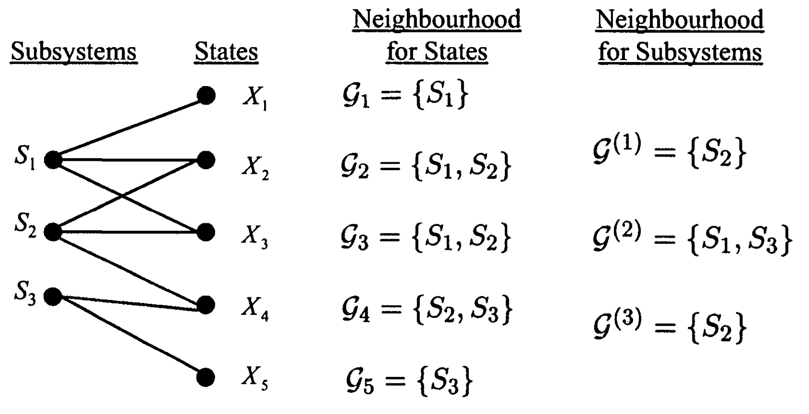

Subsystems
States
Neighbourhood for States
Neighbourhood for Subsystems
S_1
S_2
S_3
X_1
X_2
X_3
X_4
X_5
$\mathcal{G}_1 = \{S_1\}$
$\mathcal{G}_2 = \{S_1, S_2\}$
$\mathcal{G}_3 = \{S_1, S_2\}$
$\mathcal{G}_4 = \{S_2, S_3\}$
$\mathcal{G}_5 = \{S_3\}$
$\mathcal{G}^{(1)} = \{S_2\}$
$\mathcal{G}^{(2)} = \{S_1, S_3\}$
$\mathcal{G}^{(3)} = \{S_2\}$

Figure 2.4: The bipartite graph representing the illustrative system. Notation $\mathcal{G}^{(l)}$ corresponds to the neighbourhood of Subsystem S_l while $\mathcal{G}_n$ corresponds to a set of subsystems which include state $X_n(\cdot)$ in their local state vector.

An example of the localized observation model illustrated in Fig. 2.3 is given by

$$S_1:\quad \mathbf{z}^{(1)}(k) = g^{(1)}(\underbrace{X_1(k), X_2(k), X_3(k)}_{\mathbf{x}^{(1)}(k)=[X_1(k)X_2(k)X_3(k)]^T}) + \zeta^{(1)}(k) \tag{2.26}$$

$$S_2:\quad \mathbf{z}^{(2)}(k) = g^{(2)}(\underbrace{X_2(k), X_3(k), X_4(k)}_{\mathbf{x}^{(2)}(k)=[X_2(k)X_3(k)X_4(k)]^T}) + \zeta^{(2)}(k) \tag{2.27}$$

$$S_3:\quad \mathbf{z}^{(3)}(k) = g^{(3)}(\underbrace{X_4(k), X_5(k)}_{\mathbf{x}^{(3)}(k)=[X_4(k)X_5(k)]^T}) + \zeta^{(3)}(k). \tag{2.28}$$

The localized states $\{\mathbf{x}^{(1)}(k), \mathbf{x}^{(2)}(k), \mathbf{x}^{(3)}(k)\}$ defined as subscripts in Eqs. (2.26)-(2.28) extracted from the overall state vector $\mathbf{x}(k)$ overlap. In our example, $(\mathbf{x}^{(1)} \cap \mathbf{x}^{(2)}) = \{X_2(k), X_3(k)\}$. It is also possible that no shared state exists between distant subsystems, for example, $\{\mathbf{x}^{(1)} \cap \mathbf{x}^{(3)}\} = \{\}$. The aforementioned decomposition is achieved by implementing a subsystem around each observation node. Thus, the total number of subsystems in our example is equal to the number of observation nodes. Alternatively, a combination of observation nodes may be coupled to limit the total number of subsystems, if desired.

In the reduced-order configuration, the state model is also partitioned. The reduced order state model for each subsystem is obtained by decomposing the overall dynamics (Eq. 2.25) based on the observable states within each subsystem. Other states, if present, in the reduced order process models are treated as forcing terms. In our illustrative example, the reduced-order process models for the three Subsystems S_1, S_2, and S_3 are given by

$$S_1\colon \mathbf{x}^{(1)}(k) = \boldsymbol{f}^{(1)}\left(\mathbf{x}^{(1)}(k-1), \boldsymbol{d}^{(1)}(k-1)\right) + \boldsymbol{\xi}^{(1)}(k) \tag{2.29}$$

$$S_2\colon \mathbf{x}^{(2)}(k) = \boldsymbol{f}^{(2)}\left(\mathbf{x}^{(2)}(k-1), \boldsymbol{d}^{(2)}(k-1)\right) + \boldsymbol{\xi}^{(2)}(k) \tag{2.30}$$

$$S_3\colon \mathbf{x}^{(3)}(k) = \boldsymbol{f}^{(3)}\left(\mathbf{x}^{(3)}(k-1), \boldsymbol{d}^{(3)}(k-1)\right) + \boldsymbol{\xi}^{(3)}(k) \tag{2.31}$$

where $\boldsymbol{d}^{(1)}(k) = \{X_4(k)\}$, $\boldsymbol{d}^{(2)}(k) = \{X_1(k), X_5(k)\}$, and $\boldsymbol{d}^{(3)}(k) = \{X_3(k)\}$ are the forcing terms. Finally, I note that a state variable may be estimated in more than one subsystem. For example, X_2 and X_3 in Fig. 2.3 are both shared between S_1 and S_2 with different local estimates. For each state variable X_n, $(1 \leq n \leq n_x)$, I define a different state-based neighbourhood $\mathcal{G}_n$ which includes subsystems having X_n in their local state vector. If $\mathcal{G}_n$ contains more than one subsystem, there are multiple estimates of X_n available. Fig. 2.4 lists state neighbourhood $\mathcal{G}_n$ and subsystem neighbourhood $\mathcal{G}^{(l)}$ for system shown in Fig. 2.3. Next, I briefly review key state-of-the-art centralized and distributed estimation approaches.

2.2 Centralized Estimation

The Kalman filter [41] and particle filter [43] are implementations of the general Bayesian filtering equations. While the Kalman filter is generally used for estimation in linear systems with additive Gaussian forcing terms, the particle filter is more general encompassing nonlinear systems with colored forcing terms.

2.2.1 The Kalman Filter

In the Kalman filter framework, the state and observation functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ (Eqs. (2.3) and (2.4)) are linear as follows

$$\begin{aligned} \text{State Model:} \qquad \mathbf{x}(k) &= \boldsymbol{F}(k)\mathbf{x}(k-1) + \boldsymbol{\xi}_{\mathcal{N}}(k) && (2.32) \\ \text{Observation Model at Node } l\text{:} \qquad \mathbf{z}^{(l)}(k) &= \boldsymbol{G}^{(l)}(k)\mathbf{x}(k) + \boldsymbol{\zeta}_{\mathcal{N}}^{(l)}(k), && (2.33) \end{aligned}$$

where vectors $\boldsymbol{\xi}_{\mathcal{N}}^{(l)}(\cdot)$, for $(1 \leq l \leq N)$, and $\boldsymbol{\zeta}_{\mathcal{N}}(\cdot)$ are restricted to white Gaussian noise. Compared to Eqs. (2.16) and (2.17), $\boldsymbol{f}(\mathbf{x}(k-1)) = \boldsymbol{F}(k)\mathbf{x}(k-1)$ and $\boldsymbol{g}^{(l)}(\mathbf{x}(k)) = \boldsymbol{G}^{(l)}(k)\mathbf{x}(k)$ in the above model. The Kalman filter is a minimum mean square error (MMSE) estimator with the following notation used for the conditional mean of the state variables during the prediction step

$$\hat{\mathbf{x}}(k|k-1) \triangleq \mathbb{E}\{\mathbf{x}(k)|\mathbf{z}(1\!:\!k-1)\}, \tag{2.34}$$

at iteration k given observations up to time $k-1$. The conditional covariance matrix of $\mathbf{x}(k)$ given the observations $\mathbf{z}(1:k-1)$, i.e., the covariance associated with the estimate $\hat{\mathbf{x}}(k|k-1)$, is defined as follows

$$\boldsymbol{P}(k|k-1) \triangleq \mathbb{E}\{(\mathbf{x}(k) - \hat{\mathbf{x}}(k|k-1))(\mathbf{x}(k) - \hat{\mathbf{x}}(k|k-1))^T|\mathbf{z}(1:k-1)\}. \tag{2.35}$$

Conventional Kalman Filter: For a single sensor scenario ($N = 1$), one can drop index l in Eq. (2.33), and the Kalman filter equations are

Prediction Step:

$$\begin{aligned} \boldsymbol{P}(k|k-1) &= \boldsymbol{F}(k)\boldsymbol{P}(k-1|k-1)[\boldsymbol{F}(k)]^T + \boldsymbol{Q}(k) && (2.36) \\ \hat{\mathbf{x}}(k|k-1) &= \boldsymbol{F}(k)\hat{\mathbf{x}}(k-1|k-1) && (2.37) \\ \boldsymbol{S}(k|k-1) &= [\boldsymbol{G}(k)]^T\boldsymbol{P}(k|k-1)\boldsymbol{G}(k) + \boldsymbol{R}(k) && (2.38) \\ \boldsymbol{\mathcal{K}}(k) &= \boldsymbol{P}(k|k-1)\boldsymbol{G}(k)\boldsymbol{S}(k|k-1)^{-1} && (2.39) \end{aligned}$$

Observation Update Step:

$$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + \boldsymbol{\mathcal{K}}(k)(\mathbf{z}(k) - [\boldsymbol{G}(k)]^T\hat{\mathbf{x}}(k|k-1)) \tag{2.40}$$

$$\begin{aligned}\boldsymbol{P}(k|k) &= \boldsymbol{P}(k|k-1) - \boldsymbol{P}(k|k-1)\boldsymbol{G}(k)\boldsymbol{S}(k|k-1)^{-1}[\boldsymbol{G}(k)]^T\boldsymbol{P}(k|k-1) \\ &= [I - \boldsymbol{\mathcal{K}}(k)[\boldsymbol{G}(k)]^T]\boldsymbol{P}(k|k-1).\end{aligned} \tag{2.41}$$

Matrix $\boldsymbol{R}(k)$ denotes the error covariance matrix for the global observation $\boldsymbol{\zeta}_{\mathcal{N}}(k)$ and $\boldsymbol{Q}(k)$ denotes the covariance matrix associated with the forcing terms $\boldsymbol{\xi}_{\mathcal{N}}(k)$ in the state model. As for the prediction step, the following notation is associated with the conditional mean and covariance of the estimated state variables

$$\hat{\mathbf{x}}(k|k) \triangleq \mathbb{E}\{\mathbf{x}(k)|\mathbf{z}(1:k)\} \tag{2.42}$$

$$\text{and} \quad \boldsymbol{P}(k|k) \triangleq \mathbb{E}\{\big(\mathbf{x}(k) - \hat{\mathbf{x}}(k|k)\big)\big(\mathbf{x}(k) - \hat{\mathbf{x}}(k|k)\big)^T\}, \tag{2.43}$$

Information Filter: In the centralized implementation, all observations are forwarded to the fusion centre where Eq. (2.36)-(2.41) are used to compute the state estimates. To reduce the computational complexity of the Kalman filter, an implementation of the Kalman filter called the information filter [41] is derived using the matrix inversion lemma [41]. The following definitions are used in developing the information filter implementation. The information state is defined as $\hat{\boldsymbol{y}}(k|k) \triangleq \boldsymbol{P}(k|k)^{-1}\hat{\mathbf{x}}(k|k)$, and the information matrix is defined as $\boldsymbol{Y}(k|k) = \boldsymbol{P}(k|k)^{-1}$. The update equations (Eqs. (2.40) and (2.41)) for the information filter are given by

$$\hat{\boldsymbol{y}}(k|k) = \hat{\boldsymbol{y}}(k|k-1) + \underbrace{[\boldsymbol{G}(k)]^T\boldsymbol{R}(k)^{-1}\mathbf{z}(k)}_{\boldsymbol{i}(k)} \tag{2.44}$$

$$\boldsymbol{Y}(k|k) = \boldsymbol{Y}(k|k-1) + \underbrace{\boldsymbol{G}(k)\boldsymbol{R}^{-1}(k)[\boldsymbol{G}(k)]^T}_{\boldsymbol{I}(k)}, \tag{2.45}$$

where the prediction equations (Eq. (2.36)-(2.39)) are expressed in terms of $\hat{\boldsymbol{y}}(k|k-1)$ and $\boldsymbol{Y}(k|k-$

1) as follows

$$\hat{\boldsymbol{y}}(k|k-1) = (\boldsymbol{I} - \boldsymbol{\Omega}(k))\boldsymbol{F}^{-T}(k)\hat{\boldsymbol{y}}(k-1|k-1) \tag{2.46}$$

$$\boldsymbol{Y}(k|k-1) = \boldsymbol{M}(k) - \boldsymbol{\Omega}(k)\boldsymbol{\Sigma}(k)[\boldsymbol{\Omega}(k)]^T, \tag{2.47}$$

with $\boldsymbol{I}$ being the identity matrix of appropriate dimensions,

$$\boldsymbol{M}(k) = \boldsymbol{F}^{-T}(k)\boldsymbol{Y}(k-1|k-1)\boldsymbol{F}^{-1}(k), \tag{2.48}$$

$$\boldsymbol{\Sigma}(k) = \boldsymbol{M}(k) + \boldsymbol{Q}^{-1}, \tag{2.49}$$

$$\text{and} \quad \boldsymbol{\Omega}(k) = \boldsymbol{M}(k)[\boldsymbol{\Sigma}(k)]^{-1}. \tag{2.50}$$

The derivation of the information filter is given in [41]. The main advantage of the information filter over the Kalman filter is the relative simplicity of its update stage for centralized architectures. However, the simple observation update step of the information filter comes at the price of more complicated predication equations for computing $\hat{\boldsymbol{y}}(k|k-1)$ and $\boldsymbol{Y}(k|k-1)$. The information filter is also suitable for networks with hierarchical architecture. For an N-sensor network, the centralized information vector $\boldsymbol{i}(k)$ and its associated information matrix $\boldsymbol{I}(k)$ can be expressed in terms of their localized counterparts as $\boldsymbol{i}(k) \triangleq \sum_{l=1}^{N} \boldsymbol{i}^{(l)}(k)$ and $\boldsymbol{I}(k) \triangleq \sum_{l=1}^{N} \boldsymbol{I}^{(l)}(k)$. Then Eqs. (2.44)-(2.45) are reduced to

$$\hat{\boldsymbol{y}}(k|k) = \boldsymbol{y}(k|k-1) + \sum_{l=1}^{N} \underbrace{[\boldsymbol{G}^{(l)}(k)]^T \boldsymbol{R}^{(l)^{-1}}(k)\mathbf{z}^{(l)}(k)}_{\boldsymbol{i}^{(l)}(k)}, \tag{2.51}$$

$$\text{and} \quad \boldsymbol{Y}(k|k) = \boldsymbol{Y}(k|k-1) + \sum_{l=1}^{N} \underbrace{\boldsymbol{G}^{(l)}(k)\boldsymbol{R}^{(l)^{-1}}(k)[\boldsymbol{G}^{(l)}(k)]^T}_{\boldsymbol{I}^{(l)}(k)}. \tag{2.52}$$

For a hierarchical AN/SN, $\boldsymbol{I}(k)$ and $\boldsymbol{i}(k)$ are obtained from their local counterparts, i.e., $\boldsymbol{I}(k) \triangleq \sum_{l=1}^{N} \boldsymbol{I}^{(l)}(k)$ and $\boldsymbol{i}(k) \triangleq \sum_{l=1}^{N} \boldsymbol{i}^{(l)}(k)$. These terms are computed locally and forwarded to the fusion centre.

Combination of the Kalman and the Information Filters: A third form of the Kalman filter is derived below which is a combination of the conventional Kalman filter and the information filter with simple update and prediction steps. By rearranging Eq. (2.45), I have

$$\boldsymbol{I} - \boldsymbol{P}(k|k)\boldsymbol{G}(k)\boldsymbol{R}^{-1}(k)[\boldsymbol{G}(k)]^T = \boldsymbol{P}(k|k)\boldsymbol{P}^{-1}(k|k-1), \tag{2.53}$$

where $\boldsymbol{I}$ is an identity matrix with appropriate dimension. Eq. (2.53) results in the following Kalman filter equations

Prediction Step:

$$\begin{aligned} \boldsymbol{P}(k|k-1) &= \boldsymbol{F}(k)\boldsymbol{P}(k-1|k-1)[\boldsymbol{F}(k)]^T + \boldsymbol{Q}(k) && (2.54) \\ \hat{\mathbf{x}}(k|k-1) &= \boldsymbol{F}(k)\hat{\mathbf{x}}(k-1|k-1) && (2.55) \end{aligned}$$

Observation Update Step:

$$\begin{aligned} \boldsymbol{P}(k|k)^{-1} &= \boldsymbol{P}(k|k-1)^{-1} + \boldsymbol{G}(k)\boldsymbol{R}^{-1}(k)[\boldsymbol{G}(k)]^T && (2.56) \\ \hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \boldsymbol{P}(k|k)\boldsymbol{G}(k)\boldsymbol{R}^{-1}(k)\Big(\mathbf{z}(k) - \boldsymbol{G}(k)\hat{\mathbf{x}}(k|k-1)\Big) \\ &= \hat{\mathbf{x}}(k|k-1) + \boldsymbol{P}(k|k)\Big(\boldsymbol{i}(k) - \boldsymbol{I}(k)\hat{\mathbf{x}}(k|k-1)\Big). && (2.57) \end{aligned}$$

Finally, I note that for a linear dynamical system with normally distributed forcing terms and observation noise, the Kalman filter is optimal. In many practical applications, however, the state-space model is non-linear and the forcing terms are non-Gaussian.

2.2.1.1 Kalman Filter for Nonlinear Systems

A well known approximation of the Kalman filter for non-Gaussian, nonlinear Bayesian estimation is the extended Kalman filter (EKF) [43]. The EKF filter is based on the principle of linearizing the state and observation models using Taylor series expansions for the observation update step (Eqs. (2.40)-(2.41)). The series approximations in the EKF algorithm can, however, lead to poor representations of the nonlinear functions and probability distributions of interest. As a as result, the EKF filter can diverge from the optimal solution. Another form of the Kalman filter for nonlinear systems is referred to as the unscented Kalman filter (UKF) [44]. The UKF is based on the intuition that it is easier to approximate a Gaussian distribution than it is to approximate nonlinear functions. Generally, the UKF leads to more accurate results than the EKF for nonlinear systems. Below I review the UKF which is incorporated later in the Thesis to develop distributed nonlinear estimation implementations of the particle filter.

Unscented Kalman Filter: In the UKF, the statistics (estimate $\hat{\mathbf{x}}(k|k)$ and error covariance matrix $\boldsymbol{P}(k|k)$) of the state variables are updated using the unscented[4] transform. In principle, the UKF uses the true nonlinear state and observation models and, instead, approximates the distribution of the state variable with a Gaussian distribution. In other words, the filtering density $P(\mathbf{x}(k-1)|\mathbf{z}(1\!:k-1))$ in the UKF is represented with a Gaussian distribution which is specified using a set of deterministically selected sample points, referred to as sigma points. These sigma points completely capture the mean and covariance of the filtering density at time $k-1$. When propagated through the nonlinear functions, the sigma points capture the posterior mean and covariance of the filtering density $P(\mathbf{x}(k)|\mathbf{z}(1\!:k))$ at time k. Given the state estimate $\hat{\mathbf{x}}(k-1|k-1)$ and its error covariance matrix $\boldsymbol{P}(k-1|k-1)$, the UKF involves the following steps for iteration (k).

1. A set of $(2n_x+1)$ deterministic samples (referred to as the sigma points) $\{\mathcal{W}_i, \chi_i(k-1)\}_{i=0}^{2n_x}$ are calculated based on the following equation

$$\chi_i(k-1) = \hat{\mathbf{x}}(k-1|k-1) \pm \left\{\sqrt{(n_x+\kappa)\boldsymbol{P}(k-1|k-1)}\right\}_i, \qquad \text{for} \qquad 1 \le i \le 2n_x, \quad (2.58)$$

where term $\{\sqrt{(n_x+\kappa)\boldsymbol{P}(k-1|k-1)}\}_i$ corresponds to the i^{th} column of the square root of

[4]Unscented transform is a method for evaluating the statistics of a random variable after a non-linear transformation as is described in the context of the UPF in this section [44].

matrix $(n_x+\kappa)\boldsymbol{P}(k-1|k-1)$ and the initial condition is given by $\chi_0(k) = \hat{\mathbf{x}}(k-1|k-1)$. The corresponding weights for the Sigma points $\{\mathcal{W}_i\}_{i=1}^{2n_x}$ are given by $\mathcal{W}_i = 1/(2(n_x+\kappa))$, where κ is a scaling parameter and the initial condition for the sigma points is $\mathcal{W}_0 = \kappa/(n_x+\kappa)$.

2. The sigma points computed in Step 1 are propagated through the state equation (Eq. (2.3)) to generate the predicted sigma points

$$\chi_i(k|k-1) = \boldsymbol{f}(\chi_i(k-1)), \quad \text{for } i = 0, \ldots, 2n_x. \tag{2.59}$$

3. The predicted sigma points $\chi_i(k|k-1)$ are then propagated through the observation equation (Eq. (2.4)) to generate the predicted observation sigma points

$$\boldsymbol{\mathcal{Z}}_i(k|k-1) = \boldsymbol{g}(\chi_i(k|k-1)), \quad \text{for } i = 0, \ldots, 2n_x. \tag{2.60}$$

4. The predicted state estimate $\hat{\mathbf{x}}(k|k-1)$, its error covariance matrix $\boldsymbol{P}(k|k-1)$, and the predicted observation estimate $\hat{\mathbf{z}}(k|k-1)$ are computed from the following expressions

$$\hat{\mathbf{x}}(k|k-1) = \sum_{i=0}^{2n_x} \mathcal{W}_i \chi_i(k|k-1), \tag{2.61}$$

$$\boldsymbol{P}(k|k-1) = \sum_{i=0}^{2n_x} \mathcal{W}_i \Big(\chi_i(k|k-1) - \hat{\mathbf{x}}(k|k-1)\Big)\Big(\chi_i(k|k-1) - \hat{\mathbf{x}}(k|k-1)\Big)^T, \tag{2.62}$$

$$\hat{\mathbf{z}}(k|k-1) = \sum_{i=0}^{2n_x} \mathcal{W}_i \boldsymbol{\mathcal{Z}}_i(k|k-1). \tag{2.63}$$

5. The autocovariance $\boldsymbol{P}_{zz}(k|k-1)$ of predicted observations, the cross-covariance $\boldsymbol{P}_{xz}(k|k-1)$ between predicted observation and predicted state estimates are computed as follows

$$\boldsymbol{P}_{zz}(k|k-1) = \sum_{i=0}^{2n_x} \mathcal{W}_i \Big(\boldsymbol{\mathcal{Z}}_i(k|k-1) - \hat{\mathbf{z}}(k|k-1)\Big)\Big(\boldsymbol{\mathcal{Z}}_i(k|k-1) - \hat{\mathbf{z}}(k|k-1)\Big)^T, \tag{2.64}$$

$$\boldsymbol{P}_{xz}(k|k-1) = \sum_{i=0}^{2n_x} \mathcal{W}_i \Big(\chi_i(k|k-1) - \hat{\mathbf{x}}(k|k-1)\Big)\Big(\boldsymbol{\mathcal{Z}}_i(k|k-1) - \hat{\mathbf{z}}(k|k-1)\Big)^T. \tag{2.65}$$

6. The final step is to compute the updated statistics as follows

$$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + \mathcal{K}(k)\Big(\mathbf{z}(k) - \mathbf{z}(k|k-1)\Big) \tag{2.66}$$

$$\boldsymbol{P}(k|k) = \boldsymbol{P}(k|k-1) - \mathcal{K}(k)\boldsymbol{P}_{zz}(k|k-1)\mathcal{K}^T(k), \tag{2.67}$$

where the Kalman gain is given by $\mathcal{K}(k) = \boldsymbol{P}_{xz}(k|k-1)\boldsymbol{P}_{zz}(k|k-1)^{-1}$.

Note that in the UKF algorithm, Steps 1-5 can be performed off-line and the new measurements are only involved in Step 6. The UKF has, however, the limitation that it approximates the filtering density $P(\mathbf{x}(k)|\mathbf{z}(1:k))$ as a Gaussian distribution. The particle filter presented next does not impose any such restriction.

2.2.2 The Particle Filter

For nonlinear systems with non-Gaussian excitation, in general, no analytic solution can be determined. Consequently, the direct Kalman filter cannot be used and one has to rely on numerical Sequential Monte Carlo (SMC) approaches, also known as the bootstrap filtering, condensation algorithm, and particle filters [45], as approximates to the Bayesian estimators. The particle filter does not impose any restrictions on the filtering density. The particle filter is based on the principle of sequential importance sampling [43], a suboptimal technique for implementing Bayesian estimator recursively (Eqs. (2.10)-(2.13)) through Monte Carlo simulations. Below, I describe the principle of sequential importance sampling (SIS) [44], a subcategory of the SMC approach.

2.2.2.1 Importance Sampling

Importance sampling is an approach to evaluate an integral, e.g.,

$$\mathbb{E}_{P(\mathbf{x}|\mathbf{z})}\{\boldsymbol{h}(\mathbf{x})\} = \int \boldsymbol{h}(\mathbf{x})P(\mathbf{x}|\mathbf{z})d\mathbf{x} \tag{2.68}$$

where $\mathbb{E}\{.\}$ denotes expectation. A numeric way to compute $\mathbb{E}\{h(\mathbf{x})\}$ is to draw N_s random samples $\mathbb{X}_i$, for $(1 \leq i \leq N_s)$, from the probability distribution $P(\mathbf{x}|\mathbf{z})$, evaluate the function $h(\mathbf{x})$ at these samples, and then compute their statistical mean as follows

$$\mathbb{E}\{\boldsymbol{h}(\mathbf{x})\} \approx \sum_{i=1}^{N_s} \boldsymbol{h}(\mathbb{X}_i) P(\mathbb{X}_i|\mathbf{z}). \tag{2.69}$$

In practice, however, the distribution $P(\mathbf{x}|\mathbf{z})$ is either unavailable, or, it is difficult to obtain particles from this distribution. Therefore, the particles are instead derived from a proposal distribution $q(\mathbf{x}|\mathbf{z})$. Eq. (2.68) can then be written as a function of the proposal distribution as follows

$$\mathbb{E}\{\boldsymbol{h}(\mathbf{x})\} = \int \boldsymbol{h}(\mathbf{x}) \underbrace{\frac{P(\mathbf{x}|\mathbf{z})}{q(\mathbf{x}|\mathbf{z})}}_{W} q(\mathbf{x}|\mathbf{z}) d\mathbf{x}, \tag{2.70}$$

where W is called the weight function. Eq. (2.69), therefore, changes to

$$\mathbb{E}\{\boldsymbol{h}(\mathbf{x})\} \approx \sum_{i=1}^{N_s} \boldsymbol{h}(\mathbb{X}_i) W_i P(\mathbb{X}_i|\mathbf{z}) \tag{2.71}$$

with weights $W_i = P(\mathbb{X}_i|\mathbf{z})/q(\mathbb{X}_i|\mathbf{z})$, for $(1 \leq i \leq N_s)$, associated to the vector particles $\mathbb{X}_i$.

2.2.2.2 Centralized Particle Filter

With relation to the state model, Eq. (2.3), the particle filter iteratively estimates the state vector $\mathbf{x}(k)$, for $(k \geq 1)$, based on the overall observations $\mathbf{z}(1\!:\!k)$ and the given value of the previous state $\mathbf{x}(k-1)$. The centralized particle filter uses a set of samples (or 'particles') $\{\mathbb{X}_i(k)\}_{i=1}^{N_s}$ and associated weights $\{W_i(k)\}_{i=1}^{N_s}$ to estimate the system state $\mathbf{x}(k)$. At the end of iteration $k-1$ in steady state, let

$$\mathbb{X}_i(k-1) = [X_{1,i}(k-1), X_{2,i}(k-1), \ldots, X_{n_x,i}(k-1)], \tag{2.72}$$

denote an n_x-dimensional vector sample (referred to as a vector particle). Based on a statistical distribution, a combination of N_s vector particles are used to represent the true posterior distribution of the state vector $\mathbf{x}(k-1)$. Subscript i, for $(1 \leq i \leq N_s)$, therefore, indicates that N_s number of n_x-dimensional particles are available to represent the state vector $\mathbf{x}(k-1)$ at time instant $k-1$. To represent the time evolution of the particles, I use the notation

$$\mathbb{X}_i(0:k-1) = [X_{1,i}(0:k-1), X_{2,i}(0:k-1), \ldots, X_{n_x,i}(0:k-1)], \tag{2.73}$$

for $(1 \leq i \leq N_s)$. Time index $(0:k-1)$ implies that all n_x-dimensional vector particles from time iteration 0 to $k-1$ are available. Associated with each vector particle $\mathbb{X}_i(k-1)$ is its corresponding weight $W_i(k-1)$, for $(1 \leq i \leq N_s)$. The weights are normalized such that $\sum_{i=1}^{N_s} W_i(k-1) = 1$ at iteration $k-1$. As for the state particles, notation $W_i(0:k-1)$ represents the evolution of the weights over time. If required, the overall filtering distribution of the state vector at iteration $k-1$ can be expressed in terms of the particles and their associated weights as

$$P(\mathbf{x}(k-1)|\mathbf{z}(1:k-1)) \approx \sum_{i=1}^{N_s} W_i(k-1)\delta\big(\mathbf{x}(k-1) - \mathbb{X}_i(k-1)\big), \tag{2.74}$$

where $\delta(\cdot)$ denotes the Dirac delta function.

Given particles $\mathbb{X}_i(k-1)$, the values of the particles $\mathbb{X}_i(k)$ at time instant k are updated by generating random particles from the proposal distribution $q(\mathbf{x}(0:k)|\mathbf{z}(1:k))$. For SIS, the proposal distribution is chosen such that it satisfies the following factorization

$$q\big(\mathbf{x}(0:k)|\mathbf{z}(1:k)\big) = q\big(\mathbf{x}(0:k-1)|\mathbf{z}(1:k-1)\big)q\big(\mathbf{x}(k)|\mathbf{x}(1:k-1), \mathbf{z}(1:k)\big), \tag{2.75}$$

then one can obtain particles $\mathbb{X}_i(0:k) \sim q\big(\mathbf{x}(0:k)|\mathbf{z}(1:k)\big)$ by augmenting each of the existing samples $\mathbb{X}_i(0:k-1) \sim q\big(\mathbf{x}(0:k-1)|\mathbf{z}(1:k-1)\big)$ with the new particles generated as follows

$$\text{Prediction Step:} \qquad \mathbb{X}_i(k) \sim q\big(\mathbf{x}(k)|\mathbf{x}(0:k-1), \mathbf{z}(1:k)\big). \tag{2.76}$$

The next step is to update the weights as follows

$$\text{Observation Update Step:} \qquad W_i(k) \propto W_i(k-1)\frac{P\big(\mathbf{z}(k)|\mathbb{X}_i(k)\big)P\big(\mathbb{X}_i(k)|\mathbb{X}_i(k-1)\big)}{q\big(\mathbb{X}_i(k)|\mathbb{X}_i(0:k-1), \mathbf{z}(1:k)\big)}, \tag{2.77}$$

where notation $\propto$ stands for the proportional sign, which changes to an equality with the introduction of a constant. The accuracy of this importance sampling approximation depends on how close the proposal distribution is to the true posterior distribution. The optimal choice [44] for the proposal distribution that minimizes the variance of importance weights is the filtering density conditioned upon $\mathbf{x}(0:k-1)$ and $\mathbf{z}(1:k)$, i.e.,

$$q\big(\mathbf{x}(k)|\mathbf{x}(0:k-1),\mathbf{z}(1:k)\big) = P\big(\mathbf{x}(k)|\mathbf{x}(0:k-1),\mathbf{z}(1:k)\big). \tag{2.78}$$

Because of the difficulty in sampling Eq. (2.78), a common choice [44] for the proposal distribution is the transition density, $P(\mathbf{x}(k)|\mathbf{x}(k-1))$, referred to as the sampling importance resampling (SIR) filter, where the weights are pointwise evaluation of the likelihood function at the particle values, i.e.,

$$W_i(k) \propto W_i(k-1)P\big(\mathbf{z}(k)|\mathbb{X}_i(k)\big). \tag{2.79}$$

If the weights $W_i(k)$ are all equal from the previous iteration, then $W_i(k) \propto P(\mathbf{z}(k)|\mathbb{X}_i(k))$. The likelihood function $P(\mathbf{z}(k)|\mathbb{X}_i(k))$ is derived from the observation equation (Eq. (2.4)). Algorithm 1 highlights the main steps in the SIR filter.

Fig. 2.5 shows a graphical representation of the SIR algorithm for iteration k. In the top plot, the particles $\mathbb{X}_i(k)$ are generated from the transitional density $P(\mathbf{x}(k)|\mathbf{x}(k-1))$ which is a Gaussian distribution in this example. In the middle plot, the weights are computed from the likelihood function $P(\mathbf{z}(k)|\mathbf{x}(k))$ which results in the weighted particle $\{\mathbb{X}_i(k), W_i(k)\}_{i=1}^{N_s}$ as shown in the third plot.

The SIR filter has two drawbacks. First, it does not use the newly acquired observations. Second, it leads to degeneracy in the particle filter with a few samples having relatively higher weights, i.e., after a few iterations, most of the vector particles have negligible weights. A measure of degeneracy is the effective sample size $N_{\text{eff}}(k) = 1/(\sum_{i=1}^{N_s} W_i^2(k))$. A typical approach to

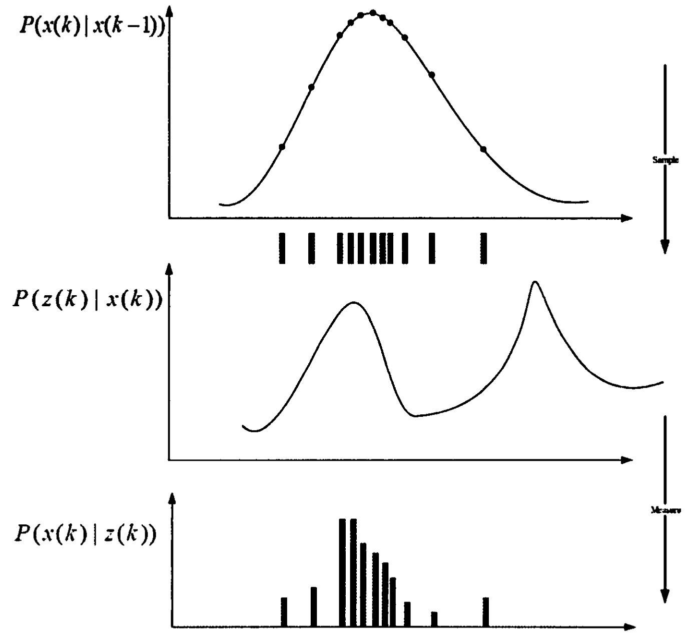
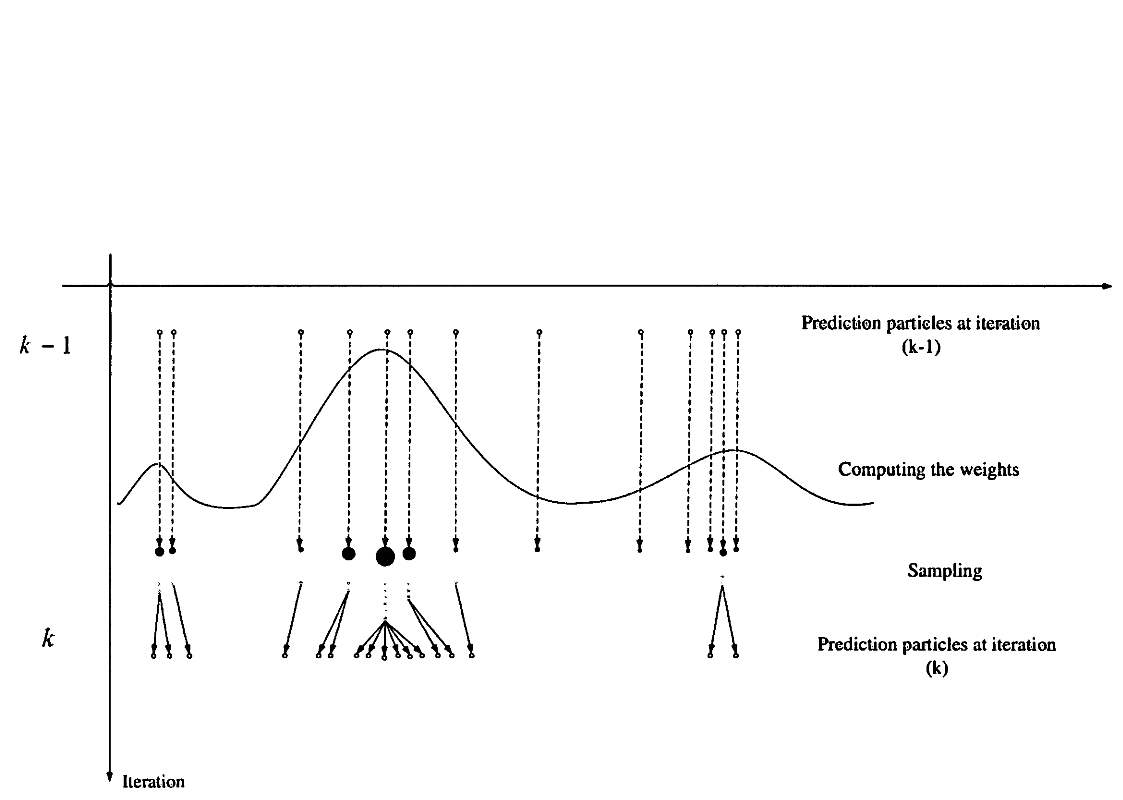


Figure 2.5: The SIR filter for estimating the posterior conditional probability (represented by blue bars).

avoid the degeneracy problem is to introduce the re-sampling [30] whenever $N_{\text{eff}}(k)$ falls below a threshold. Algorithm 2 highlights the main steps in the systematic resampling algorithm, where $U(.)$ stands for uniform distribution. The re-sampling algorithm maps particles $\mathbb{X}_i(k)$ and their weights $W_i(k)$ to resampled particles $\{\mathbb{X}_{i*}(k)\}_{i=1}^{N_s}$ such that $P\{\mathbb{X}_{i*}(k) = \mathbb{X}_j(k)\} = W_j(k)$. The resulting sample sequence is independent, identically distributed (IID) and, hence, the new weights are uniform (same).

Fig. 2.6 depicts the basic concept of the particle filter in the form of a graphical representation. In this example, a standard particle filter starts at time $k-1$ with a set of uniformly weighted particles $\{\mathbb{X}_i(k\text{–}1), 1/N_s\}_{i=1}^{N_s}$ (the top yellow dots), which yields an approximation of the prediction density $P(\mathbf{x}(k\text{–}1)|\mathbf{z}(1\colon k\text{–}2))$. Each particle $\mathbb{X}_i(k-1)$ is updated to $\mathbb{X}_i(k)$ by generating random samples from the proposal distribution. In the filtering step, the importance weight $W_i(k)$ is



$k-1$
Prediction particles at iteration (k-1)
Computing the weights
Sampling
k
Prediction particles at iteration (k)
Iteration

Figure 2.6: A pictorial description of the particle filter [44].

updated using the observation $\mathbf{z}(k)$ made at time k (the top red line). This results in the weighted particles $\{\mathbb{X}_i(k), W_i(k)\}_{i=1}^{N_s}$, which provide an approximation of $P(\mathbf{x}(k)|\mathbf{z}(1\colon k))$ (the top purple dots). Next, the resampling step selects only the particles with significant weights and resamples to obtain the new particles with uniformly weighted particles $\{\mathbb{X}_i(k), 1/N_s\}_{i=1}^{N_s}$ which still is an approximation of $P(\mathbf{x}(k)|\mathbf{z}(1\colon k))$. This process is executed recursively.

As noted previously, the particle filter implementation presented above is referred to as the SIR filter. Later in the Thesis, other forms of the particle filters are discussed, e.g., the unscented particle filter. Having presented a review of the centralized Kalman filter and particle filter, Section 2.3 presents distributed Kalman filters which serves as a precursor to distributed estimation for non-linear systems.

Algorithm 1 SIR([*in*] $\{\mathbb{X}_i(k-1)\}_{i=1}^{N_s}$, $\mathbf{z}(k)$, [*out*] $\{\mathbb{X}_i(k)\}_{i=1}^{N_s}$, $\{W_i(k)\}_{i=1}^{N_s}$)

```
Input: (i) $\{\mathbb{X}_i(k-1)\}_{i=1}^{N_s}$ – State particles, and; (ii) $\mathbf{z}(k)$ – Observation.
Output: (i) $\{\mathbb{X}_i(k)\}_{i=1}^{N_s}$ – Updated state particles, and; (ii) $\{W_i(k)\}_{i=1}^{N_s}$ – weights for updated
    state particles.
1: for $i = 1 : N_s$, do
       • Update particles by sampling $P(\mathbf{x}(k)|\mathbb{X}_i(k-1))$.
       • Compute weights based on $W_i(k) \propto P(\mathbf{z}(k)|\mathbb{X}_i(k))$.
2: end for
3: Determine the normalization factor $s = \sum_{i=1}^{N_s} W_i(k)$.
4: for $i = 1 : N_s$, do
       • Normalize $W_i(k) = W_i(k)/s$.
5: end for
6: Resample based on Algorithm 2.
```

2.3 Distributed Kalman Filters

The Kalman filter has a simple recursive structure which makes it suitable for distributed estimation problems. Several, distributed Kalman filter approaches [77–81] have been proposed for both full-order and reduced-order estimations. I describe two general frameworks (estimate-then-fuse (state estimation fusion) and fuse-then-estimate (observation fusion)) that are common to all approaches.

2.3.1 State Estimation Fusion (Estimate-Then-Fuse)

In the estimate-then-fuse framework for the Kalman filter [42], the local state estimates are first computed and then fused together to form the global state estimate. Node l, for $(1 \leq l \leq$

Algorithm 2 RESAMPLE([*in*] $\{\mathbb{X}_i(k), W_i(k)\}_{i=1}^{N_s}$, [*out*] $\{\mathbb{X}_{j*}(k), W_j(k), i_j\}_{j=1}^{N_s}$,)

```
Input: (i) $\{\mathbb{X}_i(k), W_i(k)\}_{i=1}^{N_s}$ – State particles and their associated weights.
Output: (i) $\{\mathbb{X}_{j*}(k), W_i(k)\}_{j=1}^{N_s}$ – Resampled state particles and their associated weights, and;
    (ii) $\{i_j\}_{j=1}^{N_s}$ – The index of the parent for each resampled particle.
1: Initialize the cumulative sum of weights (CSW): $C_1 = W_1(k)$
2: for $i = 2 : N_s$, do
       • Construct CSW: $C_i = C_{i-1} + W_i(k)$
3: end for
4: Start at the bottom of the CSW: $i = 1$
5: Draw a starting point: $u_1 \sim U[0, N_s^{-1}]$
6: for $j = 1 : N_s$, do
       • Move along the CSW: $u_j = u_1 + N_s^{-1}(j-1)$
7:     while $u_j > c_i$ do
       • i=i+1
8:     end while
       • Assign sample: $\mathbb{X}_{j*}(k) = \mathbb{X}_i(k)$
       • Assign weight: $W_j(k) = N_s^{-1}$
       • Assign parent: $i_j = i$.
9: end for
```

N), maintains its own estimated version $\hat{\boldsymbol{y}}^{(l)}(k|k) = [\boldsymbol{P}^{(l)}(k|k)]^{-1}\hat{\mathbf{x}}^{(l)}(k|k)$ of the information vector and the corresponding information matrix $\boldsymbol{Y}^{(l)}(k|k) = [\boldsymbol{P}^{(l)}(k|k)]^{-1}$. Since the prediction equations only depend on the state model (Eq. (2.3)), they can be computed locally without requiring any cooperation from the neighbouring nodes. The local prediction step at node l is

Local Prediction Step:

$$\hat{\boldsymbol{y}}^{(l)}(k|k-1) = (\boldsymbol{I} - \boldsymbol{\Omega}^{(l)}(k))\boldsymbol{F}^{-T}(k)\hat{\boldsymbol{y}}^{(l)}(k-1|k-1) \tag{2.80}$$

$$\boldsymbol{Y}^{(l)}(k|k-1) = \boldsymbol{M}^{(l)}(k) - \boldsymbol{\Omega}^{(l)}(k)\boldsymbol{\Sigma}^{(l)}(k)[\boldsymbol{\Omega}^{(l)}(k)]^T, \tag{2.81}$$

where $\boldsymbol{I}$ is an identity matrix of proper dimension and

$$\boldsymbol{M}^{(l)}(k) = \boldsymbol{F}^{-T}(k)\boldsymbol{Y}^{(l)}(k-1|k-1)\boldsymbol{F}^{-1}(k), \tag{2.82}$$

$$\boldsymbol{\Sigma}^{(l)}(k) = \boldsymbol{M}^{(l)}(k) + \boldsymbol{Q}^{-1}, \tag{2.83}$$

$$\text{and} \quad \boldsymbol{\Omega}^{(l)}(k) = \boldsymbol{M}^{(l)}(k)[\boldsymbol{\Sigma}^{(l)}(k)]^{-1}. \tag{2.84}$$

The local observation update equations for node l are then given by

Local Observation Update Step:

$$\hat{\boldsymbol{y}}^{(l)}(k|k) = \hat{\boldsymbol{y}}^{(l)}(k|k-1) + \boldsymbol{G}^{(l)}(k)^T \boldsymbol{R}^{(l)^{-1}}(k)\mathbf{z}^{(l)}(k) \tag{2.85}$$

$$\boldsymbol{Y}^{(l)}(k|k) = \boldsymbol{Y}^{(l)}(k|k-1) + \boldsymbol{G}^{(l)}(k)\boldsymbol{R}^{(l)^{-1}}(k)\boldsymbol{G}^{(l)^T}(k), \tag{2.86}$$

The global state estimate is then computed at each node by fusing its local state estimates with the communicated state estimates of its neighbouring nodes. A problem with estimate-then-track is the correlation between the local state estimates. The local state estimates across the neighbouring nodes are correlated due to the following two reasons: (i) The same forcing/excitation term is used in the localized state models for the neighbouring nodes, and; (ii) Some past observations incorporated in the local estimates may also be common between the local nodes [42], e.g., two nodes may have both received observation from a common third node during a previous iteration, or, they may have directly communicated to each other and incorporated the other nodes observation in updating their local estimates.

Next, I will review the channel filter approach [42] which associates an additional filter for each communication link to track the common information between a pair of neighbouring nodes.

Using channel filters, one can implement the optimal distributed Kalman filter for linear systems observed with An/SN configured using the tree connected network topologies.

2.3.1.1 Channel Filters

The channel filter framework was proposed in [42] to ensure consistency of the fused estimate by removing common information between own local estimate and the received estimate from the neighbouring node. In the context of distributed Kalman filter for tree connected networks the channel filter framework associates a channel filter for each communication link connecting a pair of local nodes. Using the channel filter, the local information vector $\hat{\boldsymbol{y}}^{(i)}(k|k)$ at node i and the local information vector $\hat{\boldsymbol{y}}^{(j)}(k|k)$ at node j are combined to form the fused information vector $\hat{\boldsymbol{y}}^{(ij)}(k|k)$ as follows

$$\hat{\boldsymbol{y}}^{(ij)}(k|k) = \hat{\boldsymbol{y}}^{(i)}(k|k) + \hat{\boldsymbol{y}}^{(j)}(k|k) - \hat{\boldsymbol{y}}^{(i\cap j)}(k|k), \tag{2.87}$$

where $\boldsymbol{y}^{(i\cap j)}(k|k)$ is the channel filter's information vector as explained below. Similarly, the fused information matrix is computed as follows

$$\boldsymbol{Y}^{(ij)}(k|k) = \boldsymbol{Y}^{(i)}(k|k) + \boldsymbol{Y}^{(j)}(k|k) - \boldsymbol{Y}^{(i\cap j)}(k|k), \tag{2.88}$$

where $\boldsymbol{Y}^{(i\cap j)}(k|k)$ is the channel filter's information matrix as explained bellow. Eqs. (2.87) and (2.88) have a number of important implications: (i) When the common information set is empty, the joint estimate can be computed by summing local estimates in their information form; (ii) There is no need for a fusion center to provide the global predictions which simplifies the computation and reduces the communication, and; (iii) Once the common information is decided, the rest of distributed estimation is straightforward. The problem, however, is how to determine such common information. Based on Eqs. (2.44) and (2.45), the channel filter extracts the common information using the following equations

$$\begin{aligned}\boldsymbol{y}^{(i\cap j)}(k|k) &= \boldsymbol{y}^{(i\cap j)}(k|k-1) + \Big[\boldsymbol{y}^{(i)}(k|k) - \boldsymbol{y}^{(i\cap j)}(k|k-1)\Big] + \Big[\boldsymbol{y}^{(j)}(k|k) - \boldsymbol{y}^{(i\cap j)}(k|k-1)\Big] \\ &= \boldsymbol{y}^{(i)}(k|k) + \boldsymbol{y}^{(j)}(k|k) - \boldsymbol{y}^{(i\cap j)}(k|k-1), \qquad (2.89)\end{aligned}$$

$$\text{and} \quad \boldsymbol{Y}^{(i\cap j)}(k|k) = \boldsymbol{Y}^{(i)}(k|k) + \boldsymbol{Y}^{(j)}(k|k) - \boldsymbol{Y}^{(i\cap j)}(k|k-1). \qquad (2.90)$$

where the predictive channel filter equations are obtained in a similar fashion as the prediction step of the the information filter (Eqs. (2.91)-(2.55)), i.e.,

$$\hat{\boldsymbol{y}}^{(i\cap j)}(k|k-1) = (\boldsymbol{I} - \boldsymbol{\Omega}^{(i\cap j)}(k))\boldsymbol{F}^{-T}(k)\hat{\boldsymbol{y}}^{(i\cap j)}(k-1|k-1), \qquad (2.91)$$

$$\text{and} \quad \boldsymbol{Y}^{(i\cap j)}(k|k-1) = \boldsymbol{M}^{(i\cap j)}(k) - \boldsymbol{\Omega}^{(i\cap j)}(k)\boldsymbol{\Sigma}^{(i\cap j)}(k)[\boldsymbol{\Omega}^{(i\cap j)}(k)]^T, \qquad (2.92)$$

where $\boldsymbol{I}$ is an identity matrix of proper dimension,

$$\boldsymbol{M}^{(i\cap j)}(k) = \boldsymbol{F}^{-T}(k)\boldsymbol{Y}^{(i\cap j)}(k-1|k-1)\boldsymbol{F}^{-1}(k), \qquad (2.93)$$

$$\boldsymbol{\Sigma}^{(i\cap j)}(k) = \boldsymbol{M}^{(i\cap j)}(k) + \boldsymbol{Q}^{-1}, \qquad (2.94)$$

$$\text{and} \quad \boldsymbol{\Omega}^{(i\cap j)}(k) = \boldsymbol{M}^{(i\cap j)}(k)[\boldsymbol{\Sigma}^{(i\cap j)}(k)]^{-1}. \qquad (2.95)$$

By using the estimate of the common information (provided by channel filters), node l, for $(1 \leq l \leq N)$, uses the following fusion rules

$$\hat{\boldsymbol{y}}^{(\text{fused},l)}(k|k) = \hat{\boldsymbol{y}}^{(l)}(k|k-1) + \sum_{i\in\aleph^{(l)}_{\text{fuse}}(k)} \Big(\hat{\boldsymbol{y}}^{(i)}(k|k) - \hat{\boldsymbol{y}}^{(l\cap i)}(k|k-1)\Big) \qquad (2.96)$$

$$\text{and} \quad \boldsymbol{Y}^{(\text{fused},l)}(k|k) = \boldsymbol{Y}^{(l)}(k|k-1) + \sum_{i\in\aleph^{(l)}_{\text{fuse}}(k)} \Big(\boldsymbol{Y}^{(i)}(k|k) - \boldsymbol{Y}^{(l\cap i)}(k|k-1)\Big), \qquad (2.97)$$

where $\aleph^{(l)}_{\text{fuse}}(k)$ is set of the neighbouring nodes for node l. The channel filters only provide the consistent estimate when the network is tree-connected.

2.3.1.2 Distributed Unscented Kalman Filter

Distributed unscented Kalman filter (DUKF) [7, 82, 83] is another example of the Kalman filter based distributed state estimation fusion algorithms. The centralized UKF was described in Section 2.2.1.1. Below, the distributed implementation of the UKF based on [7] is presented. Please refer to [82, 83] for alternative DUKF implementations. The DUKF for iteration k is based on the following two steps:

1. Each node runs a local UKF based on its local observation $\mathbf{z}^{(l)}(k)$, the fused global state estimate $\hat{\mathbf{x}}^{(\text{fused},l)}(k-1|k-1)$ and its corresponding error covariance matrix $\boldsymbol{P}^{(\text{fused},l)}(k-1|k-1)$ from the previous iteration $(k-1)$ of the DUKF. The localized version of the UKF is based the six steps outlined in Section 2.2.1.1. In Step 1, the global statistics from the previous iteration ($\hat{\mathbf{x}}^{(\text{fused},l)}(k-1|k-1)$ and $\boldsymbol{P}^{(\text{fused},l)}(k-1|k-1)$) are used to calculate the local sigma points $\{\mathcal{W}_i^{(l)}, \boldsymbol{\chi}_i^{(l)}(k-1)\}_{i=0}^{2n_x}$. Steps 2-5 remain the same in nature and compute localized statistics (superscript (l) is added to different terms computed in Steps 2-5 to show their localized nature). In Step 6, the local observation $\mathbf{z}^{(l)}(k)$ is used instead of the global observation vector $\mathbf{z}(k)$ to compute the following updated local statistics

$$\hat{\mathbf{x}}^{(l)}(k|k) = \hat{\mathbf{x}}^{(l)}(k|k-1) + \mathcal{K}^{(l)}(k)\Big(\mathbf{z}^{(l)}(k) - \mathbf{z}^{(l)}(k|k-1)\Big) \tag{2.98}$$

$$\boldsymbol{P}^{(l)}(k|k) = \boldsymbol{P}^{(l)}(k|k-1) - \mathcal{K}^{(l)}(k)\boldsymbol{P}_{zz}^{(l)}(k|k-1)[\mathcal{K}^{(l)}(k)]^T, \tag{2.99}$$

   where $\mathcal{K}^{(l)}(k) = \boldsymbol{P}_{xz}^{(l)}(k|k-1)\boldsymbol{P}_{zz}^{(l)}(k|k-1)^{-1}$.

2. The global statistics are then computed distributively based on the following fusion rules [7]

$$\boldsymbol{P}^{(\text{fused},l)}(k|k) = \sum_{l=1}^{N}\left[\boldsymbol{P}^{(l)}(k|k)\right]^{-1} \tag{2.100}$$

$$\hat{\mathbf{x}}^{(\text{fused},l)}(k|k) = \left[\boldsymbol{P}^{(\text{fused},l)}(k|k)\right]^{-1} \times \sum_{l=1}^{N}\left[\boldsymbol{P}^{(l)}(k|k)\right]^{-1}\hat{\mathbf{x}}^{(l)}(k|k). \tag{2.101}$$

The two summation terms in Eqs. (2.100) and (2.101) are computed distributively by running two vector consensus runs (one for the global mean and one for the global covariance matrix).

2.3.2 Likelihood/Observation Fusion (Fuse-Then-Estimate)

As stated in the previous section, care should be taken to compensate for the common information present in the local state estimates in the estimate-then-fuse framework. An alternative approach is based on the fuse-then-estimate framework, which leads to the fusion of the weighted observations and associated covariances. The issue of the common information in the state estimates is, therefore, automatically resolved. Based on the combined KF/IF implementation (Eqs. (2.54)-(2.57)), iteration k of the fuse-then-track framework consists of the following four steps:

Step 1. Given the fused local state estimate $\mathbf{x}^{(\text{fused},l)}(k-1|k-1)$ for iteration $k-1$ and its corresponding error covariance matrix $\boldsymbol{P}^{(\text{fused},l)}(k-1|k-1)$, node l, for $(1 \leq l \leq N)$, performs the prediction step as follows

$$\begin{aligned}
\mathbf{x}^{(l)}(k|k-1) &= \boldsymbol{F}(k)\mathbf{x}^{(\text{fused},l)}(k-1|k-1) && (2.102)\\
\boldsymbol{P}^{(l)}(k|k-1) &= \boldsymbol{F}(k)\boldsymbol{P}^{(\text{fused},l)}(k-1|k-1)[\boldsymbol{F}(k)]^T + \boldsymbol{Q}(k). && (2.103)
\end{aligned}$$

Step 2. Node l computes its local information vector $\boldsymbol{i}^{(l)}(k)$ and the local information matrix $\boldsymbol{I}^{(l)}(k))$ as follows

$$\begin{aligned}
\boldsymbol{i}^{(l)}(k) &= [\boldsymbol{G}^{(l)}(k)]^T\boldsymbol{R}^{(l)^{-1}}(k)\mathbf{z}^{(l)}(k), && (2.104)\\
\boldsymbol{I}^{(l)}(k) &= \boldsymbol{G}^{(l)}(k)\boldsymbol{R}^{(l)^{-1}}(k)[\boldsymbol{G}^{(l)}(k)]^T, && (2.105)
\end{aligned}$$

and communicates them to its immediate neighbouring nodes.

Step 3. Once node l has received data from all its neighbouring nodes, it fuses them as follows

$$\boldsymbol{i}^{(\text{fused},l)}(k) = \sum_{i \in \aleph^{(l)}_{\text{fuse}}(k)} [\boldsymbol{G}^{(i)}(k)]^T \boldsymbol{R}^{(i)^{-1}}(k)\mathbf{z}^{(i)}(k) \tag{2.106}$$

$$\boldsymbol{I}^{(\text{fused},l)}(k) = \sum_{i \in \aleph^{(l)}_{\text{fuse}}(k)} [\boldsymbol{G}^{(i)}(k)]^T \boldsymbol{R}^{(i)^{-1}}(k)\boldsymbol{G}^{(i)}(k). \tag{2.107}$$

Step 4. The observation update state of the Kalman Filter (Eqs. (2.56)-(2.57)) is then performed locally as

$$\boldsymbol{P}^{(\text{fused},l)}(k|k) = \Big([\boldsymbol{P}^{(l)}(k|k-1)]^{-1} + \boldsymbol{I}^{(\text{fused},l)}(k)\Big)^{-1}, \tag{2.108}$$

$$\mathbf{x}^{(\text{fused},l)}(k|k) = \mathbf{x}^{(l)}(k|k-1) + \boldsymbol{P}^{(\text{fused},l)}(k|k)\Big(\boldsymbol{i}^{(\text{fused},l)}(k) - \boldsymbol{I}^{(\text{fused},l)}(k)\mathbf{x}^{(l)}(k|k-1)\Big). \tag{2.109}$$

In an all-to-all communication network, i.e. when there exists a direct link between node l and all other nodes in the network, Eqs. (2.108) and (2.109) result in the centralized estimates at each node. In other words, the local estimates at each node are the same as the centralized estimate. Having an all-to-all communication network is, however, a limiting constraint. Consensus-based[5] distributed implementation of the Kalman filter is developed based on this framework to extend distributed estimation to arbitrary network topologies. Such methods compute the summation terms in Eq. (2.106) and Eq. (2.107) over the entire network instead of limiting the summation terms to local neighbourhoods, i.e.,

$$\boldsymbol{i}^{(\text{fused},l)}(k) = \sum_{i=1}^{N} [\boldsymbol{G}^{(i)}(k)]^T \boldsymbol{R}^{(i)^{-1}}(k)\mathbf{z}^{(i)}(k) \tag{2.110}$$

$$\boldsymbol{I}^{(\text{fused},l)}(k) = \sum_{i=1}^{N} [\boldsymbol{G}^{(i)}(k)]^T \boldsymbol{R}^{(i)^{-1}}(k)\boldsymbol{G}^{(i)}(k). \tag{2.111}$$

Two average consensus algorithms (as explained below in Section 2.4) can be used to compute Eq. (2.110) and Eq. (2.111) in a distributed fashion.

[5]Consensus in distributed filtering is the process of establishing a consistent value for some statistics of the state vector across the network by interchanging relevant information between the connected neighboring nodes.

In the next section, I will present the aforementioned consensus algorithms in more detail.

2.4 Average Consensus Algorithms

Consensus algorithms and their randomized counterparts, the gossip algorithms [88], form the foundation of distributed computing [89] with a long history in distributed processing and decision making [90], information processing in sensor networks [32,33], multi-agent collaboration [91], vehicle formation [92], tracking and data fusion [79,93], and distributed inference [94]. Consensus algorithms are generally iterative in nature, where each node begins with a set of local information. At each iteration, data is exchanged between a subset of nodes, which assimilates new information to update the local parameters. A recent review on the average consensus algorithms can be found in [32] or [33]. These consensus algorithms do not require specialized routing [33] and perform reasonably well even in imperfect scenarios such as sensor networks with error-prone communications, node/link failures, and channel noise [95–97]. Further, average consensus algorithms have been extended in many directions, e.g., continuous time average consensus algorithms as described in [32] and non-linear average consensus algorithms [98, 99]. The design of fast consensus algorithms has been investigated in [100], the concept of consensus likelihood described in [21] and the concept of Kalman-consensus which considers the problem of consensus seeking with relative uncertainty in distributed systems presented in [101]. In this chapter, I limit the discussion to the discrete time linear average consensus algorithms, a sub-class of the classical average-consensus algorithms.

2.4.1 Discrete Time Linear Consensus Algorithms

Suppose there are N-nodes with inconsistent information denoted by $X_c^{(l)}(t)$, $(1 \leq l \leq N)$, where t is the consensus time index that is different from the filtering time index k. With reference to my previous discussion in Section 2.3.2, at iteration k, node l, for $(1 \leq l \leq N)$, initializes its local consensus state $X_c^{(l)}(0)$ as follows

$$X_c^{(l)}(0) = \boldsymbol{i}^{(l)}(k) = [\boldsymbol{G}^{(l)}(k)]^T \left[\boldsymbol{R}^{(l)}(k)\right]^{-1} \mathbf{z}^{(l)}(k). \tag{2.112}$$

The objective of the consensus algorithm is to communicate relevant information amongst the neighbouring nodes to iteratively update the consensus state $X_c^{(l)}(t)$ at node l such that it eventually converges to its centralized counterpart $\boldsymbol{i}^{(\text{fused},l)}(k)$ given by Eq. (2.110). Mathematically, the updated value at node l is

$$X_c^{(l)}(t+1) = \Upsilon\Big(X_c^{(l)}(t), X_c^{(i)}(t)\Big), \quad i \in \aleph_{\text{fuse}}^{(l)}(k), \tag{2.113}$$

where $\aleph_{\text{fuse}}^{(l)}(k)$ represents the set of neighbouring nodes for node l in graph $\mathcal{G}$. Eq. (2.113) represents a distributed algorithm because each node only receives/communicates information from/to its neighbouring nodes via communication links permitted by graph $\mathcal{G}$.

Definition: A distributed algorithm for graph $\mathcal{G}$ can achieve consensus asymptotically if:

1. There exists a time instant T_c such that $X_c^{(l)}(T_c) = \alpha$, for $(1 \leq l \leq N)$, i.e.,

$$X_c^{(l)}(t) = \alpha, \quad \forall t \geq T_c. \tag{2.114}$$

2. All nodes reach a common value asymptotically

$$\lim_{t \to \infty} X_c^{(l)}(t) = \alpha, \tag{2.115}$$

where $\alpha \in \Re$ is the collective decision of the sensor nodes in the network and is referred to as the group decision, stationary, converged, or equilibrium value.

Moreover, if this common value is the average of the initial values of the consensus states, i.e., $\alpha = 1/N \sum_{l=1}^{N} X_c^{(l)}(0)$, then the algorithm is said to achieve average consensus. In other words, reaching a consensus implies an asymptotic convergence to a one-dimensional agreement space defined as $X_c^{(1)}(t) = X_c^{(2)}(t) = \ldots = X_c^{(N)}(t)$, for $(t \geq T_c)$. Collecting all $X_c^{(l)}(t)$'s in a vector $\mathbf{x}_c(t)$, the agreement space can be expressed as $\mathbf{x}_c(t) = \alpha\mathbf{1}$, for $(t \geq T_c)$, where $\mathbf{1} = [1, 1, \ldots, 1]^T$ is a unit column vector with 1 as its entries. A further distinction is made based on whether the consensus is constrained or unconstrained.

1. **Unconstrained consensus** is simply an *alignment problem* where the agreement value is not important and it only suffices that the consensus states of all nodes asymptotically converge to the same value.

2. **Constraint consensus**, referred to as the χ-consensus in this chapter, requires the consensus state to asymptotically converge to a function $\chi(\mathbf{x}_c(0))$ of initial values.

An average consensus algorithm is a χ-consensus algorithm with $\chi(\mathbf{x}_c(0)) = 1/N \sum_{l=1}^{N} X_c^{(l)}(0)$, which is often used in distributed signal processing applications. The goal of an average consensus algorithm is to guarantee the convergence of the algorithm to the mean value for any choice of initial conditions.

An important class of a discrete time, linear average consensus algorithm is given by

$$X_c^{(l)}(t+1) = U_{ll}(t)X_c^{(l)}(t) + \sum_{j \in \aleph_{\text{fuse}}^{(l)}(k)} U_{lj}(t)X_c^{(j)}(t), \tag{2.116}$$

which can alternatively be expressed as $\mathbf{x}_c(t+1) = \boldsymbol{U}(t)\mathbf{x}_c(t)$ in the matrix-vector format, where $\boldsymbol{U}(t) \triangleq \{U_{ij}\} \in \Re^{(N \times N)}$ is referred to as the consensus matrix representing the configuration of graph $\mathcal{G}$. In other words, the sparsity pattern of the consensus matrix models the communication network over which the neighbouring nodes can communicate. A possible choice for $\boldsymbol{U}(t)$ is described later. A third form for Eq. (2.116) is given by

$$X_c^{(l)}(t+1) = X_c^{(l)}(t) + \sum_{j \in \aleph_{\text{fuse}}^{(l)}(k)} U_{lj}(t)\left(X_c^{(l)}(t) - X_c^{(j)}(t)\right), \tag{2.117}$$

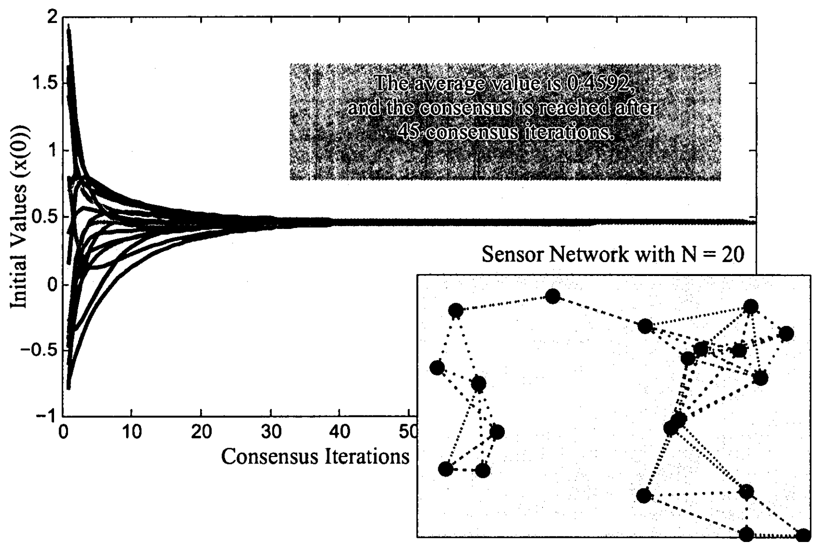


Figure 2.7: An example of average consensus algorithm with 20 sensor nodes.

derived by exploiting the stochastic matrix property $\boldsymbol{U}(t)\mathbf{1} = \mathbf{1}$. Note that this property implies that the sum of the entries of any row of matrix $\boldsymbol{U}$ is always 1, i.e., $U_{ll}(t) = 1 - \sum_{j \in \mathcal{N}^{(l)}_{\text{fuse}}(k)} U_{lj}(t)$, which when substituted in Eq. (2.116) results in the new expression. Eq. (2.117) provides an intuitive interpretation for average consensus as a control action to the old consensus value that corrects for the difference from the consensus state.

Fig. 2.7 shows an example of an average consensus algorithm in a network with 20 nodes. Connections between neighbouring nodes are shown with dotted lines in the small block on the lower right of Fig. 2.7. Node l, for $(1 \leq l \leq 20)$, initializes its consensus state $X^{(l)}_c(0)$ with the value shown in Fig. 2.7 and uses Eq. (2.116) to update its consensus state. After 45 iterations the consensus converges, i.e., $X^{(l)}_c(t) = \sum_{l=1}^{N} X^{(l)}_c(0) = 0.4592$, for $t > 45$.

There are two scenarios that may arise in the context of specific signal processing applications: (i) Deterministic consensus where the consensus matrix $\boldsymbol{U}$ is given and remains fixed, i.e., $\boldsymbol{U}(t) = \boldsymbol{U}$, and; (ii) Randomized consensus, where $\boldsymbol{U}(t)$ is drawn from some distributions on a set of stochastic matrices defined as $\mathscr{U}$. For fixed communication, Eq. (2.117) implies that $\mathbf{x}_c(t) = \boldsymbol{U}^t\mathbf{x}_c(0)$. In addition, from Eq. (2.115) we have

$$\lim_{t\to\infty} \mathbf{x}_c(t) = \lim_{t\to\infty} \boldsymbol{U}^t\mathbf{x}_c(0) = \frac{1}{N}(\sum_{i=1}^{N} X_c(0))\mathbf{1} = \frac{1}{N}(\mathbf{1}^T\mathbf{x}_c(0))\mathbf{1} = (\mathbf{1}\mathbf{1}^T/N)\mathbf{x}_c(0), \tag{2.118}$$

which is equivalent to the matrix equation $\lim_{t\to\infty}\boldsymbol{U}^t = \mathbf{1}\mathbf{1}^T/N$. Linear consensus algorithms (Eq. (2.116) or (2.117)) converges to the average for any initial vector $\mathbf{x}_c(0) \in \Re^N$ if and only if the identity $\lim_{t\to\infty}\boldsymbol{U}^t = \mathbf{1}\mathbf{1}^T/N$ holds.

Finally, the asymptotic convergence rate of a consensus algorithm is defined as follows

$$r_{\text{asym}}(\boldsymbol{U}) = \sup_{\lim_{t\to\infty}} \left[\frac{\| \mathbf{x}_c(t) - \bar{\mathbf{x}}_c \|_2}{\| \mathbf{x}_c(0) - \bar{\mathbf{x}}_c \|_2}\right]^{1/t}, \tag{2.119}$$

where $\| \cdot \|_2$ is the Euclidean $\boldsymbol{L}^2$ norm, i.e., $\| \mathbf{x}_c \| \triangleq \sqrt{\mathbf{x}_c^T\mathbf{x}_c}$. The following theorem [100] provides the necessary and sufficient conditions for convergence of a consensus algorithm.

Theorem 1. *An average consensus algorithm (e.g., Eq. (2.117)) converges, i.e.,* $\lim_{t\to\infty}\boldsymbol{U}^t = \mathbf{1}\mathbf{1}^T/N$ *holds if and only if*

$$\mathbf{1}^T\boldsymbol{U} = \mathbf{1}^T \tag{2.120}$$

$$\boldsymbol{U}\mathbf{1} = \mathbf{1} \tag{2.121}$$

$$\rho(\boldsymbol{U} - \mathbf{1}\mathbf{1}^T/N) < 1, \tag{2.122}$$

where $\rho(.)$ *denotes the spectral radius of a matrix, i.e., the largest eigenvalue of a matrix in the absolute values. Moreover, the asymptotic convergence rate can be expressed as*

$$r_{asym}(\boldsymbol{U}) = \rho(\boldsymbol{U} - \mathbf{1}\mathbf{1}^T/N). \tag{2.123}$$

The following results are observed from Theorem 1. First, Eq. (2.120) states that $\mathbf{1}$ is the left eigenvector of $\boldsymbol{U}$ associated with the eigenvalue of 1. For this case, we have

$$\sum_{i=1}^{N} X_c^{(i)}(t) = \mathbf{1}^T\mathbf{x}_c(t) = \mathbf{1}^T\boldsymbol{U}\mathbf{x}_c(t-1) = \mathbf{1}^T\mathbf{x}_c(t-1) = \sum_{i=1}^{N} X_c^{(i)}(t-1). \tag{2.124}$$

Eq. (2.124) is referred to as the preserving property i.e., the average of the consensus states is preserved at each iteration of consensus algorithm. Second, Eq. (2.121) illustrates that $\mathbf{1}$ is also the right eigenvector of $\boldsymbol{U}$ associated with the unitary eigenvalue. This condition sates that once consensus is reached, the value of the consensus variables remains unchanged, i.e., $\mathbf{1}$ is a fixed point of the linear iteration. Together with the first two conditions, Eq. (2.122) implies that 1 is a simple eigenvalue of $\boldsymbol{U}$ on the unit disk and its algebraic multiplicity is 1, i.e. it is a simple root of the characteristic polynomial of $\boldsymbol{U}$. Eq. (2.122) also implies that all other eigenvalues are strictly less than one in magnitude, i.e., $|\lambda_i(\boldsymbol{U})| < 1 \quad \forall i = \{2, \ldots, N\}$. For the subclass of consensus algorithms considered in here, a result from [31] shows that

$$r_{\text{asym}}(\boldsymbol{U}) = \max\{|\lambda_2(\boldsymbol{U})|, |\lambda_N(\boldsymbol{U})|\}, \tag{2.125}$$

i.e., the convergence rate of a discrete time linear consensus algorithm (Eq. (2.123)) is dependent on the second largest eigenvalue of the consensus matrix. To study the convergence rate, one must develop techniques to bound the eigenvalues of the consensus matrix. Fast linear consensus algorithms [100] are designed by minimizing the second largest eigenvalue of the consensus matrix. For continuous time consensus algorithm, the graph Laplacian $\boldsymbol{L}$ matrix and its spectral properties [32] are important graph related parameters which play a crucial role in the convergence analysis [32]. Necessary and sufficient conditions to guarantee convergence of average consensus algorithms in different scenarios, e.g., in presence of communication time-delays, packet drops, channel noises, link failures and quantization errors have been studied by many researchers [31–33, 95–97]. For a more detailed review of the convergence properties of the consensus algorithms, please refer to [31].

The question of how to assign the weight matrix $\boldsymbol{U}$ in Eq. (2.116) arises naturally at this point. A common choice is $\boldsymbol{U} = \boldsymbol{I} - \epsilon \boldsymbol{L}$ where $\epsilon \in (0, \frac{1}{\Delta_G}]$ and $\boldsymbol{U}$ satisfies [32] the conditions expressed in Eqs. (2.120)-(2.122). For other possible forms of the consensus matrix $\boldsymbol{U}$, please refer to [31–33].

For example, the Kalman-consensus method proposed in [101] designs the consensus matrix $\boldsymbol{U}$ by allocating proper weights to individual nodes with greater certainty in their performed estimation.

Finally, I note that alternative approaches to the consensus algorithms are the gossip algorithms which are generally randomized counterparts of the consensus algorithms. The difference in consensus and gossip algorithms lie in the selection of the neighbouring nodes to which the information is shared at each iteration. While consensus algorithms communicate with all neighbouring nodes, gossip algorithms randomly select a subset of neighbouring nodes and communicate only with that subset. Generally, the subset with which each node communicates varies from one gossip iteration to another. Another alternative to reach consensus on predefined statistical parameters is to use spanning trees [36] where the topology is specifically designed and known at each node.

2.5 Distributed Particle Filters

The distributed particle filter implementations considered in this section use the following state dynamics and observation model at node l, for $(1 \leq l \leq N)$

$$\mathbf{x}(k) = \boldsymbol{f}(\mathbf{x}(k-1)) + \boldsymbol{\xi}(k) \tag{2.126}$$

$$\mathbf{z}^{(l)}(k) = \boldsymbol{g}^{(l)}(\mathbf{x}(k)) + \boldsymbol{\zeta}^{(l)}(k), \tag{2.127}$$

with the entire state vector $\mathbf{x}(k)$ is estimated by running a localized particle filter at each node. So the following overview of the existing distributed implementations of the particle filter is mainly focused on full-order distributed configuration for nonlinear systems.

Since the seminal work by Gordon *et al.* [45], the particle filters have been widely used for statistical estimation but mostly in the centralized configuration. Developing distributed implementations of the particle filter is computationally demanding and places considerable bandwidth overhead for information transfer between the local processing nodes. Following the classification taxonomy shown in Fig. 1.4, distributed particle filter implementations can be organized into two main categories: Message passing schemes [16, 17] where information flows in a pre-defined, sequential manner from a node to one of its neighboring nodes via a cyclic path till the entire network is traversed, and; Diffusive schemes [18–27, 29, 30] where each node communicates its local information across the network by interacting with its immediate neighbors. In dynamical environments, where frequent changes in the underlying network topology due to mobility, node failure, and intermittent connectivity are a common practice, diffusive schemes significantly improve the robustness at the cost of certain communication overhead.

Consensus-based approaches are a special subcategory of diffusive schemes applicable to arbitrary network topologies [32, 33]. The basic idea behind the consensus-based distributed implementations is to express the fusion problem in a way such that it only involves average quantities. Although the consensus-based distributed Kalman filter implementations [32, 33, 77, 79–81] have been widely explored for estimation and tracking problems in linear systems, there is much room for developing distributed particle filter implementations for nonlinear systems. Further refinement of the consensus-based distributed particle filter implementations is based on the nature of the information transfers between the processing nodes. Examples of the information communicated within the network include the raw observations, local likelihoods, functions of the local observations [18–22], local state posterior, and local state estimates [23–27]. Coates *et al.* [16] use a parametric model of the partial likelihood function commonly referred to as the DPF via observation/likelihood fusion. Sheng *et al.* [17] approximate the partial local posteriors with a Gaussian mixture model (GMM) and communicate the parameters of the local GMM models between the neighboring nodes using a message passing setup. Sheng's implementation is commonly referred to as DPF via state estimation fusion. The DPF approaches based on state estimation fusion and observation/likelihood fusion are considered next.

2.5.1 DPF via State Estimation Fusion (Estimate-then-Fuse)

The state estimation fusion based DPF implementation is explained in terms of the SIR form of the particle filter (Section 2.2.2, Eqs. (2.76) and (2.77)). It consists of two steps: the local particle filtering step implemented at each node to evaluate the local particles $\mathbb{X}_i^{(l)}(k)$ and their corresponding weights $W_i^{(l)}(k)$ and the fusion step to combine local estimates into the global estimate. Based only on the local observations made at node l, the local observation update and the following fusion step are described below.

1. *Local Particle Filters*: At node l, the local particle filter first updates its particles as follows

$$\text{Local Prediction Step :} \qquad \mathbb{X}_i^{(l)}(k) \sim P\big(\mathbf{x}(k)|\mathbb{X}_i^{(l)}(k-1)\big). \tag{2.128}$$

The weights are pointwise evaluation of the local likelihood function at the particle values computed as

$$\text{Local Observation Update Step :} \qquad W_i^{(l)}(k) \propto W_i^{(l)}(k-1)P\big(\mathbf{z}^{(l)}(k)|\mathbb{X}_i^{(l)}(k)\big). \tag{2.129}$$

The local particle filter at node l approximates the local filtering density $P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k))$ as a Dirac mixture with a set of particles and their associated weights $\{\mathbb{X}_i^{(l)}(k), W_i^{(l)}(k)\}$ as

$$P\big(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k)\big) \approx \sum_{i=1}^{N_s} W_i^{(l)}(k)\delta\big(\mathbf{x}(k) - \mathbb{X}_i^{(l)}(k)\big), \tag{2.130}$$

where $\delta(\cdot)$ denotes the Dirac delta function.

2. *Fusion of Local Particles*: The global state estimate is computed by fusing the local filtering densities $P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k))$ represented via local particle sets $\{\mathbb{X}_i^{(l)}(k), W_i^{(l)}(k)\}$ across all nodes. To highlight the issues involved in the fusion step, the fusion problem between two

nodes i and j is first considered. For node i and j, the joint filtering density is given by

$$P(\mathbf{x}(k)|\mathbf{z}^{(i)}(1\!:\!k) \cup \mathbf{z}^{(j)}(1\!:\!k)) = \frac{P(\mathbf{x}(k)|\mathbf{z}^{(i)}(1\!:\!k)) \times P(\mathbf{x}(k)|\mathbf{z}^{(j)}(1\!:\!k))}{P(\mathbf{x}(k)|\mathbf{z}^{(i)}(1\!:\!k) \cap \mathbf{z}^{(j)}(1\!:\!k))}, \tag{2.131}$$

where $P(\mathbf{x}(k)|\mathbf{z}^{(i)}(1:k) \cup \mathbf{z}^{(j)}(1:k))$ is the fused filtering density based on observations at node i and j, and $P(\mathbf{x}(k)|\mathbf{z}^{(i)}(1:k) \cap \mathbf{z}^{(j)}(1:k))$ is the filtering density corresponding to the common information between nodes i and j. Computing Eq. (2.131) based on the local particles is challenging due to the following two main problems: (i) Transferring the whole particle set requires extensive communication resources, and; (ii) Even if the particles can be communicated, two separate dirac mixtures (e.g., $\{\mathbb{X}_i^{(l)}(k), W_i^{(l)}(k)\}$ and $\{\mathbb{X}_i^{(u)}(k), W_i^{(u)}(k)\}$) may not have the same region of support and their multiplication/division could be zero everywhere. To tackle these issues, a transformation is required on the particle representations ($\{\mathbb{X}_i^{(l)}(k), W_i^{(l)}(k)\}$) prior to communication. Gaussian distributions [24], grid-based techniques [47], GMMs [17] and Parzen representations [25,27] are different parametric continuous distributions used in the DPF implementations. Next, I consider the transformation approach based on Gaussian distribution for the local particles [24].

Instead of communicating the particles for fusion, node l approximates its local filtering density with a Gaussian distribution whose statistics (mean and covariance) are computed from the local particles. The statistics of the global filtering density are then calculated across the network from the local statistics by using average consensus algorithms on the local means and covariances. More specifically, the global filtering density (Eq. (2.15)) given by

$$P\big(\mathbf{x}(k), |\mathbf{z}(1\!:\!k)\big) \propto P(\mathbf{x}(k)|\mathbf{z}(1\!:\!k-1)) \prod_{i=1}^{N} P\big(\mathbf{z}^{(l)}(k)|\mathbf{x}(k)\big), \tag{2.132}$$

is factorized in terms of geometric mean of the modified local filtering densities as

$$P\big(\mathbf{x}(k)|\mathbf{z}(1\!:\!k)\big) \propto \sqrt[N]{\prod_{l=1}^{N} \tilde{P}\big(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k)\big)}, \tag{2.133}$$

where the modified local filtering density at node l, for $(1 \leq l \leq N)$, is given by

$$\tilde{P}\big(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k)\big) = P\big(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k-1)\big) \times P^{N}\big(\mathbf{z}^{(l)}(k)|\mathbf{x}(k)\big). \tag{2.134}$$

Reference [24] has proposed to approximate $\tilde{P}\big(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k)\big)$ as a Gaussian distribution $\mathcal{N}(\bar{\mathbf{x}}^{(l)}(k), \bar{\boldsymbol{P}}^{(l)}(k))$ where its statistics are computed from the local particles as follows

$$\bar{\mathbf{x}}^{(l)}(k) = \sum_{i=1}^{N_s} W_i^{(l)}(k)\mathbb{X}_i^{(l)}(k), \tag{2.135}$$

$$\bar{\boldsymbol{P}}^{(l)}(k) = \sum_{i=1}^{N_s} W_i^{(l)}(k)\Big(\mathbb{X}_i^{(l)}(k) - \bar{\mathbf{x}}^{(l)}(k)\Big)\Big(\mathbb{X}_i^{(l)}(k) - \bar{\mathbf{x}}^{(l)}(k)\Big)^T. \tag{2.136}$$

Since the product of Gaussians is itself a Gaussian, it can be shown [129] that

$$\begin{aligned} P\big(\mathbf{x}(k)|\mathbf{z}(1:k)\big) &\propto \sqrt[N]{\prod_{l=1}^{N}\mathcal{N}(\bar{\mathbf{x}}^{(l)}(k), \bar{\boldsymbol{P}}^{(l)}(k))} \\ &= \sqrt[N]{\mathcal{N}(\bar{\mu}^{(\text{fused})}(k), \bar{\boldsymbol{P}}^{(\text{fused})}(k))} = \mathcal{N}(\bar{\mu}^{(\text{fused})}(k), N \times \bar{\boldsymbol{P}}^{(\text{fused})}(k)) \end{aligned} \tag{2.137}$$

where

$$\big[\bar{\boldsymbol{P}}^{(\text{fused})}(k)\big]^{-1} = \sum_{i=1}^{N} \big[\bar{\boldsymbol{P}}^{(i)}(k)\big]^{-1}, \tag{2.138}$$

$$\text{and} \quad \bar{\mu}^{(\text{fused})}(k) = \bar{\boldsymbol{P}}^{(\text{fused})}(k)\sum_{i=1}^{N} \big[\bar{\boldsymbol{P}}^{(i)}(k)\big]^{-1}\bar{\mathbf{x}}^{(i)}(k). \tag{2.139}$$

Note that in computing the local weights $W_i^{(l)}(k)$, the weight update equation (Eq. (2.77)) changes as follows

$$W_i^{(l)}(k) \propto W_i^{(l)}(k-1)\frac{\Big[P\Big(\mathbf{z}^{(l)}(k)|\mathbb{X}_i^{(l)}(k)\Big)\Big]^N P\Big(\mathbb{X}_i^{(l)}(k)|\mathbb{X}_i^{(l)}(k-1)\Big)}{q\Big(\mathbb{X}_i^{(l)}(k)|\mathbb{X}_i^{(l)}(k-1), \mathbf{z}^{(l)}(k), \mathbf{z}(1:k-1)\Big)}. \tag{2.140}$$

If the proposal distribution is selected to be equal to the transitional density then Eq. (2.140) reduces to

$$W_i^{(l)}(k) \propto W_i^{(l)}(k-1)\Big[P\Big(\mathbf{z}^{(l)}(k)|\mathbb{X}_i^{(l)}(k)\Big)\Big]^N. \tag{2.141}$$

The statistics of the global filtering density given by Eq. (2.138)-(2.139) are obtained from the local statistics using several average consensus algorithms.

2.5.2 DPF via Likelihood/Observation Fusion (Fuse-then-Estimate)

The DPF via observation/likelihood fusion differs from the centralized particle filter mainly in the observation update step, i.e., in computing the weights (Eq. (2.77)). It consists of the following two steps.

1. *Local Prediction* is more or less similar to Eq. (2.128). However, algorithms of this category commonly implement synchronized local particle filters [16], i.e., a set of parallel particle filters where their random number generators have been initialized at the same point and, therefore, generate the same set of particles at each iteration. In other words, $\mathbb{X}_i(k) = \mathbb{X}_i^{(l)}(k)$ for, $(1 \leq l \leq N)$, i.e., particles at different nodes are the same, therefore, the index l is dropped for the notation used to denote the particle sets. The particles at node l are updated as follows

   $$\mathbb{X}_i(k) \;\sim\; P\big(\mathbf{x}(k)|\mathbb{X}_i(k-1)\big), \tag{2.142}$$

   resulting in the same set of local predictive particles at each node.

2. *Global Observation Update*: Considering the conditional independence of the observations made at neighbouring nodes (Eq. (2.5)) and using the global likelihood representation form Eq. (2.14), the weight update equation (Eq. (2.77)) is given by

   $$W_i(k) \propto W_i(k-1)P\big(\mathbf{z}(k)|\mathbb{X}_i(k)\big) = W_i(k-1)\prod_{l=1}^{N} P\big(\mathbf{z}^{(l)}(k)|\mathbb{X}_i(k)\big). \tag{2.143}$$

   In the centralized implementation, all observations are available at the fusion centre and Eq. (2.143) could potentially be used to evaluate the global likelihood function and to update the weights. In the distributed implementation, node l has restricted access limited to its local observation $\mathbf{z}^{(l)}(k)$ and can, therefore, only evaluate its local likelihood $P(\mathbf{z}^{(l)}(k)|\mathbb{X}_i(k))$ based on its vector particle $\mathbb{X}_i(k)$. The likelihoods $P(\mathbf{z}^{(m)}(k)|\mathbb{X}_i(k))$, $m \neq l$, are not available at node l and need to be communicated for updating the weights. A brute force distributed approach is to, first, express the weight update equation (Eq. (2.143)) as

$$\log W_i(k) \propto \log W_i(k-1) + \sum_{l=1}^{N} \log\left(P(\mathbf{z}^{(l)}(k)|\mathbb{X}_i(k))\right), \tag{2.144}$$

and then run average consensus algorithms across the network to compute the value of the summation term for each particle $\mathbb{X}_i(k)$. A total of N_s synchronous consensus runs are used to compute the summation terms for each particle (where N_s is the number of particles). There are two main issues with the DPF implementations using the likelihood/observation fusion. First, using synchronized local particle filters is somewhat restrictive. Second, requiring a total of N_s synchronous consensus algorithms introduces extensive communication overhead. Next, an alternative algorithm (namely DPF via set membership [20]) is proposed to address these issues.

2.5.2.1 DPF via Set Membership

The distributed implementation of the particle filter via set membership is a 4-step set-theoretic approach proposed in [20] to reduce the number of the particles communicated in the fusion step. In principle, the DPF via set membership reduces the communication overhead by computing the weight update equation (Eq. (2.144)) only for a small subset of particles selected using a set-theoretic approach as explained below.

1. *Local set selection*: Node l, for $(1 \leq l \leq N)$, implements a local particle filter and performs local set selection as follows

   (a) Oversample the particles and weights $\{\mathbb{X}_i(k-1), W_i(k-1)\}_{i=1}^{N_s}$ to extend the number of particles and obtain $\{\mathbb{X}_{i'}(k-1), W_{i'}(k-1)\}_{i'=1}^{L\times N_s}$ where $L \in \mathbb{N}$, and $\mathbb{N}$ denotes the set of natural numbers.

(b) Sample the transitional density $P(\mathbf{x}(k)|\mathbb{X}_{i'}(k-1))$ and compute the corresponding weights based on the local likelihood $P(\mathbf{z}^{(l)}(k)|\mathbf{x}(k))$ to obtain $\{\tilde{\mathbb{X}}_{i'}(k), \tilde{W}_{i'}^{(l)}(k)\}_{i'=1}^{L\times N_s}$. Assuming local set selection was successful in the previous iteration, all nodes had the same extended set of particles and associated weights. After sampling of the transitional density, the weights would be different at the local nodes.

(c) Resample N_s particles from $N_s \times L$ particles to obtain the local set of particles and weights $\{\mathbb{X}_i^{(l)}(k), W_i^{(l)}(k)\}_{i=1}^{N_s}$. After this step, local nodes would have different particles which explains why the superscript l reappears. The resampled particles are from local posterior $P(\mathbf{x}(k)|\mathbf{x}(k-1), \mathbf{z}^{(l)}(k))$.

(d) Node l, for $(1 \leq l \leq N)$, computes the coordinates of a box $\mathcal{E}^{(l)}(k)$ containing its particles. Term $\mathcal{E}^{(l)}(k)$ represents the region where $P(\mathbf{x}(k)|\mathbf{x}(k-1), \mathbf{z}^{(l)}(k))$ contains the majority of its mass.

2. *Global Set Determination*: All nodes cooperatively compute the intersection of their local boxes, i.e., the global box $\mathcal{E}(k)$ which contains samples corresponding to the region with the highest likelihood. Note that this can be implemented by running a combination of maximum and minimum consensus algorithms on $\mathcal{E}^{(l)}(k)$.

3. *Distributed Importance Density Sampling*: Once the global box is determined, it is used to form an approximate of the optimal proposal distribution $P(\mathbf{x}(k)|\mathbf{x}(k-1), \mathbf{z}(k))$ as follows

$$U(\mathbf{x}(k)|\mathbf{x}(k-1), \mathcal{E}(k)) = \frac{\alpha I(\mathbf{x}(k) \in \mathcal{E}(k)) + \beta I(\mathbf{x}(k) \notin \mathcal{E}(k))}{\gamma} P(\mathbf{x}(k)|\mathbf{x}(k-1)), \quad (2.145)$$

where γ is the normalizing constant to make $U(\cdot)$ a proper density, $I(\cdot)$ is an indicator function, and $\beta \ll \alpha$. Node l generates the predicted particles form the proposal distribution given by Eq. (2.145). Specifically, each node first draws particle from the transitional density

$P(\mathbf{x}(k)|\mathbf{x}(k-1))$. If the sample belongs to the global box $\mathcal{E}(k)$, it is accepted with high probability. Otherwise, the sample is discarded with high probability.

4. *Weight Update*: For each accepted vector particle, a distributed average consensus algorithm computes its corresponding weights using Eq. (2.144).

5. *Resampling*: Finally, resampling is performed to generate N_s uniformly weighted particles.

The DPF via set membership [20] is an example of algorithms belonging to the fuse-then-estimate category. Alternative algorithms belonging to this category are [19,21,22]. Algorithms proposed in [21,22] are applicable when the global likelihood is exponentially distributed. In such scenarios, References [21,22] approximate the global likelihood as a function $g'(\cdot)$ of the summation of some other function $g''(\cdot)$ of the local observations, i.e., $P(\mathbf{z}(k)|\mathbf{x}(k)) = g'(\sum_{l=1}^{N} g''(\mathbf{z}^{(l)}(k)))$, which can be computed distributively using average consensus algorithms. Reference [19] constructs a distributed auxiliary particle filter algorithm such that every node has a copy of the same filter (the same weights and particles). To do this, local nodes execute a synchronization routine so that their random number generators have the same seeds; in this way, they always sample the same values. The algorithm proposed in [19] is similar in concept to the DPF via set membership [20]. A subset of effective particles are selected first by distributively computing preliminary weights for all the particles using gossip algorithms (randomized counterpart of consensus algorithms). The effective particles are the ones with the highest preliminary global weights. Once the effective particle set is selected, another runs of gossip algorithms are used to computed the updated weights.

Finally, in the context of distributed implementation of the Kalman filter, communicating state posteriors (fuse-then-estimate category (Section 2.3.1)) is advantageous over communicating functions of local observations or local likelihoods (fuse-then-estimate category (Section 2.3.2)) because the later would result loss of information in case packets are lost. If instead, information on the state posteriors is communicated, lost information can be recovered since it is implicitly present in the future state posteriors. In the context of DPF implementations, however in terms of the accuracy, the algorithms belonging to fuse-then-estimate category (Section 2.5.2) are less sensitive to the algorithms belonging to estimate-then-fuse category (Section 2.5.1). This is mainly due to the role of the proposal distribution. The algorithms belonging to the former category (Section 2.5.2), typically, use a distributively computed proposal distribution while algorithms belonging to the latter category (Section 2.5.1) usually incorporate a locally designed proposal distribution. Intuitively speaking, a combination of two categories will be able to both recover lost information (which is a property of algorithms belonging to estimate-then-fuse category) and at the same time implement a reasonable proposal distribution and reduce the sensitivity of the DPF implementation (which is a property of algorithms belonging to fuse-then-estimate category).

In summary, the existing distributed implementations of the particle filter suffer from some of the following drawbacks:

1. A large number of iterative parallel consensus runs is required to reach consensus on a selected set of global parameters between two consecutive iterations of the local particle filters. Algorithms belonging to the fuse-then-estimate category, such as the DPF via set membership, are more sensitive to this problem because they require a significant number of consensus runs.

2. Most of the existing distributed particle filter implementations are based on the SIR filter and use the transitional $P(\mathbf{x}(k)|\mathbf{x}(k-1))$ (Eq. (2.78)) as the proposal distribution. Such a selection is not optimal. Choosing the transitional distribution is a major challenge and a bottleneck to the performance of the distributed particle filters.

3. Some form of the Gaussian approximation is commonly used in the DPF implementations. For example, the global likelihood is approximated with a Gaussian distribution in [18]. Likewise, the global filtering/posterior distribution is assumed to be Gaussian in [23,24]. The advantage of the particle filter is lost by approximating the global posterior by a Gaussian distribution.

4. The consensus step used by the local particle filters is assumed to converge within the time interval available between two successive observations. The performance of the distributed approaches degrades substantially if consensus is not reached within two consecutive iterations of the local particle filters. A major problem in distributed estimation networks is unreliable communication (especially in large and multi-hop networks), which results in communication delays, information loss and, therefore, delays in convergence of the consensus step. Referred to as intermittent network connectivity [123,124], this issue has not been investigated in the context of the distributed particle filter implementations.

5. Computation of the global estimates from local estimates during the consensus step is based on an sub-optimal fusion rules (e.g., local averaging) which ignores the problem of common information between the local state estimates and results in the degradation of the overall performance.

In summary, drawback 4 is common to all existing DPF implementations. In addition, the DPF implementations suffer either from Drawback 1 (extremely high communication overhead [19,20]) or combination of Drawbacks 2, 3, and 5 (strong approximations and suboptimal fusion [21–24]). In the subsequent chapter, I develop distributed implementations of the particle filter to address the aforementioned issues.

2.6 Applications

In this section, I review potential applications of the distributed implementation of the particle filer considered in the thesis.

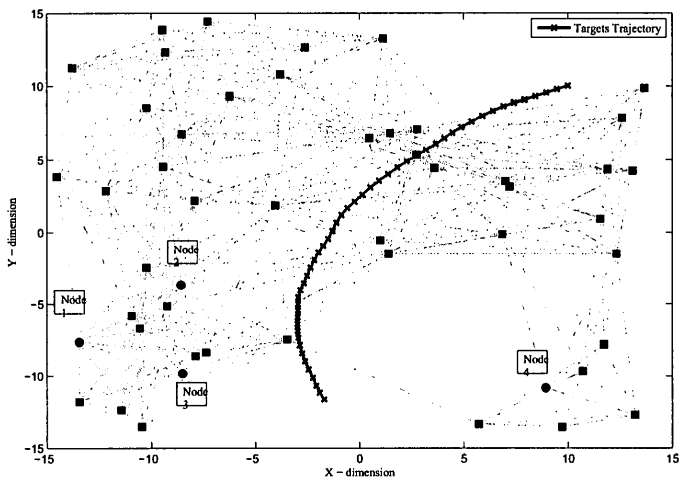
2.6.1 Bearing Only Tracking

The problem of bearings-only tracking (BOT), also referred to as target motion analysis, arises in a variety of non-linear signal processing applications including radar surveillance, underwater submarine tracking in sonar, and robotics [102, 103]. In terms of our state model (Eq. (2.3)), the state vector is given by $\mathbf{x}(k) = [X(k), Y(k), \dot{X}(k), \dot{Y}(k)]^T$. The trajectory of the target is described using different state models such as [103]: (i) Constant velocity model; (ii) Clockwise coordinated turn model; (iii) Anticlockwise model; (iv) Constant acceleration model, or; (v) some combination of (i)-(iv). For example, the clockwise coordinated turn state model is given by Eq. (2.3) with the state function

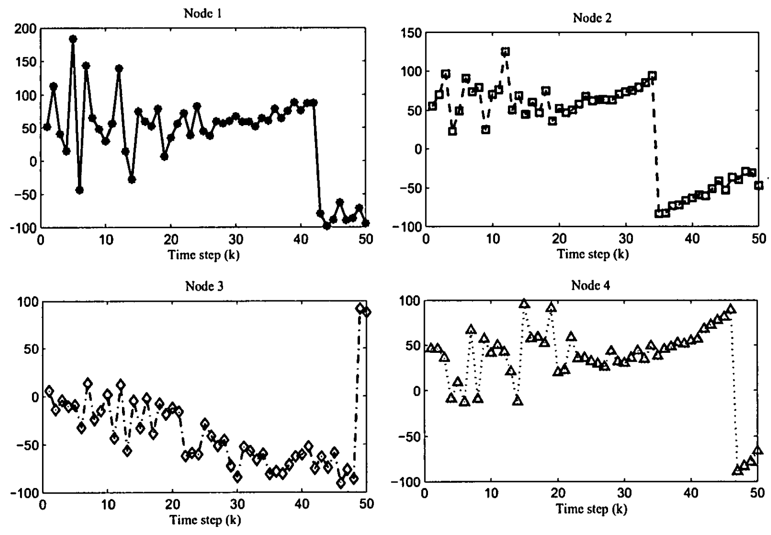
$$f(\mathbf{x}(k)) = \begin{bmatrix} 1 & 0 & \frac{\sin(\Omega(k)\Delta T)}{\Omega(k)} & -\frac{1-\cos(\Omega(k)\Delta T)}{\Omega(k} \\ 0 & 1 & \frac{1-\cos(\Omega(k)\Delta T)}{\Omega(k)} & \frac{\sin(\Omega(k)\Delta T)}{\Omega(k)} \\ 0 & 0 & \cos(\Omega(k)\Delta T) & -\sin(\Omega(k)\Delta T) \\ 0 & 0 & \sin(\Omega(k)\Delta T) & \cos(\Omega(k)\Delta T) \end{bmatrix}, \quad \text{with} \quad \Omega(k) = \frac{A_m}{\sqrt{(\dot{X}(k))^2 + (\dot{Y}(k))^2}}, \tag{2.146}$$

where ΔT is the sampling time and A_m is the manoeuvre acceleration parameter. Measurements are the target's bearings with respect to the platform of each node referenced (clockwise positive) to the y-axis, i.e.,

$$Z^{(l,m)}(k) \quad = \quad \text{atan}\left(\frac{X(k) - X^{(l,m)}(k)}{Y(k) - Y^{(l,m)}(k)}\right) + \varsigma^{(l,m)}(k), \tag{2.147}$$



(a)



(b)

Figure 2.8: The configuration and bearing measurements. (a) Initial sensor locations and one realization of the target's trajectory. (b) Bearing measurements at four randomly selected nodes.

$(X^{(l,m)}(k), Y^{(l,m)}(k))$ are the coordinates of node sensor m connected to processing node l. The overall observation vector is a combination of the local observations $Z^{(l,m)}(k)$ as given by Eq. (2.147). As shown in Eqs. (2.146) and (2.147), BOT is inherently a non-linear application with its non-linearity incorporated in the state dynamics and/or in the measurement model depending on the choice of the coordinate system used to formulate the problem. Fig. 2.8(a) plots the target's track as modeled by Eq. (2.146) along with the locations of the processing nodes. Fig. 2.8(b) shows the bearing measurements (in degree) obtained from four randomly selected nodes. The objective is to design a practical filter capable of estimating the kinematics (position $[X, Y]$ and velocity $[\dot{X}, \dot{Y}]$) of the target from the bearing angle measurements and prior knowledge of the target's motion.

Since each node has a limited communication range, local nodes configured using the centralized architecture have to send their local observations indirectly via multihop relay to the fusion centre. The fusion centre in the centralized particle filter needs to wait for all observations and then perform the estimation update which results in significant latency in computing the state estimates. In dynamic networks where the network size and connections can change due to node failure and/or communication link failure, observations may not reach the fusion centre at times. Further, any lost observation not reaching the fusion centre can not be recovered since estimation is limited to the fusion centre. Last but not the least, nodes in the immediate neighbourhood of the fusion centre relay more data which means that the energy consumption (energy required for transferring a massage times the number of massages) is unbalanced in the centralized architecture, and mostly concentrated near the fusion centre. Distributed estimation, on the other hand, overcomes these issues by maintaining local state estimates across the network and limiting the communication to local neighbourhoods. Most significantly, the latency issue can be resolved in the distributed estimation approaches with appropriate control of the consensus overhead.

2.6.2 Range Only Tracking

As the second application, I consider a distributed unicycle mobile robot localization problem using range only measurements [6, 7]. This is a good benchmark since the underlying dynamics is non-linear with non-additive forcing terms resulting in a non-Gaussian transitional state model. The state vector of the unicycle robot is defined by $\mathbf{x}(k) = [X(k), Y(k), \theta(k)]^T$, where $(X(k), Y(k))$ is the 2D coordinate of the robot and $\theta(k)$ is its orientation. The velocity and angular velocity are denoted by $\tilde{V}(k)$ and $\tilde{W}(k)$, respectively. The following discrete-time non-linear unicycle model [6] represents the state dynamics of the robot

$$\begin{aligned}
X(k) &= X(k-1) + \frac{\tilde{V}(k-1)}{\tilde{W}(k-1)}\Big(\sin\Big(\theta(k-1) + \tilde{W}(k-1)\Delta T\Big) - \sin\left(\theta(k-1)\right)\Big), && (2.148)\\
Y(k) &= Y(k-1) + \frac{\tilde{V}(k-1)}{\tilde{W}(k-1)}\Big(\cos\Big(\theta(k-1) + \tilde{W}(k-1)\Delta T\Big) - \cos\left(\theta(k-1)\right)\Big), && (2.149)\\
\text{and } \theta(k) &= \theta(k-1) + \tilde{W}(k-1)\Delta T + \xi_\theta \Delta T, && (2.150)
\end{aligned}$$

where ΔT is the sampling time and ξ_θ is the orientation noise term. The observations are range-only measurements given by

$$Z^{(l,m)}(k) = \sqrt{\left(X(k) - X^{(l,m)}(k)\right)^2 + \left(Y(k) - Y^{(l,m)}(k)\right)^2} + \zeta^{(l,m)}(k), \qquad (2.151)$$

where $(X^{(l,m)}(k), Y^{(l,m)}(k))$ are the coordinates of node sensor m connected to processing node l. Since the state is locally unobservable, the sensors have to cooperate with each other to estimate the robot's location. Distributed localization via range-only measurements is another application of distributed estimation algorithms where the state model is non-linear and is locally unobservable at individual sensor nodes.

2.6.3 Acoustic Source Localization

Another application of distributed particle filter approaches is in acoustic source localisation using an acoustic vector sensor (AVS) network. The AVS [105] employs a co-located sensor structure capable of providing 2-D (azimuth and elevation) direction of arrival (DOA) information. Recently, advances in distributed AN/SN systems have motivated the deployment of AVS networks for acoustic source localization. To track N_T acoustic sources located at $\mathbf{x}_m(k) = [X_m(k), Y_m(k), Z_m(k)]^T \in \mathbb{R}^{3\times 1}$, for $(1 \le m \le N_\mathrm{T})$, at time instant k, assume N AVS nodes at fixed locations $\mathbf{x}^{(l)} = [X^{(l)}, Y^{(l)}, Z^{(l)}]^T \in \mathbb{R}^{3\times 1}$, for $(1 \le l \le N)$, are arbitrarily deployed. The DOA of the acoustic signal associated with the mth source at the lth AVS node is given by

$$\begin{aligned} \phi_m^{(l)}(k) &= \tan^{-1}\left(\frac{X_m(k) - X^{(l)}}{Y_m(k) - Y^{(l)}}\right); \\ \psi_m^{(l)}(k) &= \tan^{-1}\left(\frac{Z_m(k) - Z^{(l)}}{\sqrt{(X_m(k) - X^{(l)})^2 + (Y_m(k) - Y^{(l)})^2}}\right), \end{aligned} \tag{2.152}$$

where $\phi_m^{(l)}(k) \in [-\pi,\ \pi]$ and $\psi_m^{(l)}(k) \in [-\pi/2,\ \pi/2]$ represent the azimuth angle and the elevation angle respectively, and superscript T denotes the transpose. Let

$$\boldsymbol{u}_m^{(l)}(k) = \left[\cos\psi_m^{(l)}(k)\cos\phi_m^{(l)}(k),\quad \cos\psi_m^{(l)}(k)\sin\phi_m^{(l)}(k),\quad \sin\psi_m^{(l)}(k)\right]^T \tag{2.153}$$

be the unit direction vector pointing out from the lth AVS sensor towards the mth source. Assuming that at time step k, T_0 number of snapshots are considered, the collection of acoustic source signals $\boldsymbol{s}_m(k)$, $(1 \le m \le N_\mathrm{T})$, is given by

$$\boldsymbol{S}(k) = [\mathrm{s}_1(k), \ldots, \mathrm{s}_{N_\mathrm{T}}(k)]^T \in \mathbb{C}^{N_\mathrm{T}\times T_0}. \tag{2.154}$$

The received signal model for the lth AVS node is as follows

$$\boldsymbol{y}^{(l)}(k) = \boldsymbol{g}^{(l)}(\boldsymbol{X}(k))\boldsymbol{S}(k) + \boldsymbol{\epsilon}^{(l)}(k), \tag{2.155}$$

where $\boldsymbol{X}(k) = [\mathbf{x}_1^T(k), \ldots, \mathbf{x}_{N_\mathrm{T}}^T(k)]^T$ is the source state, $\boldsymbol{g}^{(l)}(\boldsymbol{X}(k)) = [\boldsymbol{a}_n^1(k), \ldots, \boldsymbol{a}_n^{N_\mathrm{T}}(k)]$ with $\boldsymbol{a}_n^m(k) = [1, \boldsymbol{u}_m^{(l)}(k)]^T$ is the steering vector, and $\boldsymbol{\epsilon}^{(l)}(k) \in \mathbb{C}^{4\times T_0}$ represent the channel noise including the pressure and velocity noise terms. Note that the particle velocity terms are normalized by multiplying by a constant term $-\rho_0 c_0$, where ρ_0 and c_0 represent the ambient density and the propagation speed of the acoustic wave in the medium respectively. The noise process $\epsilon^{(l)}(k)$ is a sequence of complex-valued Independent and identically distributed (IID) circular Gaussian random variables with zero mean and covariance matrix $\mathbf{\Gamma}$.

Since dynamic sources are considered, the source state $\mathbf{x}_m(k)$ is constructed by cascading the original position component $\mathbf{x}_m^{\mathrm{p}}(k)$ with a velocity component $\mathbf{x}_m^{\mathrm{v}}(k)$. Constant velocity model is employed here to model the source dynamics as follows

$$\boldsymbol{X}(k) = \boldsymbol{F}\boldsymbol{X}(k-1) + \boldsymbol{G}(\boldsymbol{v}(k)), \tag{2.156}$$

where $\boldsymbol{v}(k)$ is the global uncertainties in the state process. The coefficient matrix $\boldsymbol{F}$ and $\boldsymbol{G}$ are defined respectively as

$$\mathbf{F} = \mathbf{I}_{N_{\mathrm{T}}} \otimes \begin{bmatrix} \mathbf{I}_3 & \Delta T \mathbf{I}_3 \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix}; \quad \mathbf{G} = \mathbf{I}_{N_{\mathrm{T}}} \otimes \begin{bmatrix} \frac{\Delta T^2}{2}\mathbf{I}_3 \\ \Delta T \mathbf{I}_3 \end{bmatrix}, \tag{2.157}$$

where $\mathbf{I}_q$ denotes the qth order identity matrix, ΔT represents the time period in seconds between the previous and current time step, and $\otimes$ denotes the Kronecker product. Eqs. (2.155) and (2.156) present the state-space model for the AVS network based tracking problem.

2.6.4 State Estimation in Power Grids

State estimation [106–109] in electrical power grids is used to monitor the state of the grid, enable energy management, optimize power flows, and perform reliability/security assessment. State forecasts are also used to analyze contingencies and determine necessary corrective actions against possible failures in the power systems. In the electric power distribution networks, the underlying state and observation models are highly nonlinear. The observations are geographically distributed across the entire distribution grid. The large dimensionality of the estimation problem precludes the direct application of the centralized particle filter primarily due to its high computational complexity. In other words, although the centralized approach is optimal, it is neither robust nor scalable to such large-scale dynamical systems with geographical distributed observation nodes primarily because of two reasons. First, extensive computations are required at the fusion node due to the high dimensionality of the dynamical systems. Second, the centralized implementation requires a large number of information transfers to the fusion center thus adding considerable latency (a major drawback for real-time applications) to the estimation mechanism.

The state estimation approaches in complex electric power distribution networks, typically consider the overall system as a union of several low-dimensional subsystems. Each subsystem is a combination of multiple, geographically distributed nodes representing a variety of power devices such as generating stations, compensators, or loads. Within each subsystem, the voltage and power supplied to a feeder at the substation are usually the only real time measurements available to the system operator at the distribution control centre. More extensive real time monitoring and control are required for effective operation of the system and for good quality of service to the customer coupled with the need to prevent wide-spread power blackouts. As outlined below, there are at lease three major aspects in the power grids that directly impact state estimation approaches and motivate development of distributed estimation implementations.

1. Monitoring the power grid over large geographical areas calls for distributed control, and hence, distributed state estimation to facilitate coordinated monitoring.

2. More advanced measurement technologies like phasor measurement units (PMUs) have offered hope for near real-time monitoring of the power grid. However, the latency introduced by the centralized estimation architecture is a major barrier toward achieving this goal.

3. To facilitate smart grid features such as demand response and two-way power flow, timely and accurate models and estimation approaches are required which calls for distributed on-line state estimation at the distribution level.

2.6.5 State Estimation in Distributed Camera Networks

Over the past decade, large-scale camera networks [110] have become increasingly popular in a wide range of applications, including: (i) Sports analysis; (ii) Security and surveillance; (iii) disaster response, and; (iv) Environmental modeling, where the objective is to follow the trajectory of a key target, e.g., a star player in a soccer game or a suspect in a surveillance environment. In many applications, bandwidth constraints, security concerns, and difficulty in storing and analyzing large amounts of image data centrally at a single location necessitate the development of distributed camera network (DCN) architectures [111]. In distributed tracking via camera network each camera acts as a local agent and estimates certain parameters of the target using a signal processing algorithm based upon its own set of video sequences. The local estimates are then shared with the neighboring cameras in an iterative, decentralized, gossip-type fashion, and a final estimate is computed across the network using consensus algorithms.

Most of the recent focus on distributed tracking algorithms for DCN is devoted to developing distributed implementation of the Kalman filters [111]. Although particle filters are popular for visual tracking [112, 113] in a centralized architecture, their distributed implementations are less explored for tracking in DCNs. Distributed particle filter approaches proposed in the Thesis can be applied (with proper modifications) for tracking problems in DCN.

2.7 Summary

In this Chapter, the Bayesian estimation approaches were reviewed as background material. The centralized and distributed Bayesian estimation framework were introduced in Section 2.1. Starting with linear systems, three implementations of the Kalman filter were presented in the Section 2.2.1. In many signal processing applications, the underlying processes are non-Gaussian and the state-space models are nonlinear. Direct implementation of the Kalman filter is, therefore, not practical. The particle filter was described in Section 2.2.2 as an alternative estimation approach for nonlinear systems. After presenting an overview of centralized estimation approaches, common distributed implementations of the Kalman filter were discussed in Section 2.3 for linear systems. Distributed implementation of the particle filter (DPF) were considered in Section 2.5 as an alternative to distributed Kalman filters for systems with nonlinear dynamics. The DPF were classified into 2 main categories: (i) Estimate-then-Fuse where local state estimates are first computed and then fused to compute the global estimate, and; (ii) Fuse-then-Estimate where the observation/likelihood information is communicated within local neighbourhoods in order to construct distributed implementation of the particle filter.

In summary, the following issues were identified with the existing distributed particle filter implementations: (i) A large number of parallel consensus runs is required by the local particle filters adding considerable overhead to the system; (ii) Selection of the proposal distribution is not optimal; (iii) Some form of the Gaussian approximation of the global posterior density and/or global likelihood is used in the DPF implementations, which affects the overall accuracy of the estimation mechanism; (iv) Requiring the consensus step to converge within the duration between two successive observations is a strict condition that may not be satisfied in networks with intermittent connectivity, and; (v) A sub-optimal fusion rule is used to derive the global estimate.

3 Consensus-based Distributed Implementation of the Particle Filter

Chapter 2 provided an overview of some of the existing distributed particle filter implementations developed for systems with nonlinear dynamics and non-Gaussian forcing and observation noise terms. A number of issues such as large communication overhead for the consensus step, sub-optimal selection of the proposal distribution, and requirement for the consensus step to converge between two consecutive observations were identified. Chapter 3 proposes three consensus-based distributed implementation of the particle filter to address some of these issues. The first approach is referred to as the constrained sufficient statistic based distributed implementation of the particle filter (CSS/DPF). The CSS/DPF belongs to the DPF via likelihood/observation fusion category (Section 2.5.2) and is proposed for distributed bearing-only tracking (BOT) and joint bearing/range tracking applications. The CSS/DPF runs localized particle filters at each sensor node and computes the global sufficient statistics of the overall system as a constraint function (summation) of the local sufficient statistics. The CSS/DPF is, therefore, a two stage procedure: (i) First, the average of the local sufficient statistics are computed distributively by running average consensus algorithms to derive the global sufficient statistics, and; (ii) Each node then updates its localized particle filter using the global sufficient statistics. The number of parallel average consensus runs in the CSS/DPF is lower in comparison to the state-of-the-art distributed particle filter implementations, thereby, reducing the communication complexity and bandwidth requirement. The second approach presented in this chapter is referred to as the CSS based unscented distributed particle filter (CSS/DUPF) which is a combination of the CSS/DPF and UCD/DPF for BOT and joint bearing/range tracking applications. The CSS/DUPF improves upon the CSS/DPF by introducing the UKF as the proposal distribution which is a better approximation of the optimal proposal distribution as compared to the transitional density.

The third proposed DPF approach is referred to as the unscented, consensus-based, distributed implementation of the particle filter (UCD/DPF). The UCD/DPF couples the unscented Kalman filter (UKF) with the localized particle filter at each node such that the UKF estimates a Gaussian approximation of the posterior distribution, which is then used as the proposal distribution in the particle filter. The UCD/DPF belongs to the DPF via state estimation fusion category (Section 2.5.1). Compared to the existing distributed implementations of the particle filter, the UCD/DPF offers two advantages. First, it uses all available local observations including the most recent ones in deriving the proposal distribution. Second, computation of the global estimate from local estimates during the consensus step is based on an optimal fusion rule.

Table 3.1 compares the proposed full-order distributed particle filter implementations. A range of characteristics for each implementation are compared in the table. Characteristics 1 and 2 define the type of fusion used in the distributed implementation. Characteristics 3 to 9 define important properties useful in selecting the implementation appropriate for the application at hand. Going from left to right, the CSS/DPF has the lowest computation and communication complexity but has a specialized implementation structure limited to specific applications. The CSS/DUPF is relatively more accurate than the CSS/DPF but has a higher computational complexity and still specifically designed for BOT and joint bearing/range tracking applications. The UCD/DPF has less communication complexity than the CSS/DUPF and generalizable to most applications.

Table 3.1: Comparison of different full-order DPF implementations.

Characteristics	**CSS/DPF**	**CSS/DUPF**	**UCD/DPF**
1. Likelihood/Observation fusion	×	×	
2. State estimation fusion		×	×
3. Gaussian approximation for the global likelihood	×	×	
4. Gaussian approximation for the global posterior			×
5. Requires consensus convergence	×	×	×
6. Application specific	×	×	
7. Restrict the proposal to the transitional distribution	×		
8. Recovery from loss of information		×	×
9. Communication complexity	*low*	*high*	*medium*

The organization of the chapter is as follows. The proposed CSS/DPF is presented in Section 3.1 followed by the CSS/DUPF in Section 3.2. The UCD/DPF implementation is presented in Section 3.4. Section 3.3 illustrates the effectiveness of the proposed framework in tracking applications through Monte Carlo simulations. Finally Section 3.5 concludes the chapter.

3.1 The CSS/DPF Implementation

In distributed Kalman filters, it is well known [32,114] that the mean of the observations recorded across the sensor network provides sufficient statistics to reconstruct the optimal estimate. Extending this sufficient statistics approach to nonlinear systems, the section proposes a constraint sufficient statistics-based distributed implementation of the particle filter (CSS/DPF) for bearing-only [102–104] and joint bearing/range [115] tracking problems. Following [116], I show that if the global likelihood satisfies certain constraints then it can be expressed as a function $\mathcal{S}(\cdot)$ of the known local statistics. In the CSS/DPF, I impose another constraint and restrict $\mathcal{S}(\cdot)$ to the summation operation so that the global statistics can be computed efficiently using average consensus.

3.1.1 Sufficient Statistic-Based Framework

In this section, the sufficient statistic based framework for distributed implementation of the particle filter is developed in terms of the local observations $\mathbf{z}^{(l)}(k)$ and the global observation $\mathbf{z}(k) = \{\mathbf{z}^{(l)}(k)\}_{l=1}^{N}$ with N denoting the total number of nodes in the network. The global likelihood $P(\mathbf{z}(k)|\mathbf{x}(k))$ and predicted density $P(\mathbf{x}(k)|\mathbf{z}(1\ k-1))$ provide a complete characterization of the estimation problem as previously shown in Eq. (2.10). Let $\mathcal{S}(\mathbf{z}(k))$ be the sufficient statistic corresponding to the global likelihood function $P(\mathbf{z}(k)|\mathbf{x}(k))$. Based on the Fisher-Neyman factorization theorem [117], the global likelihood is factorized as

$$P\Big(\mathbf{z}(k)|\mathbf{x}(k)\Big) = \mathcal{T}_1\Big(\mathbf{z}(k)\Big) \times \mathcal{T}_2\Big(\mathcal{S}\big(\mathbf{z}(k)\big), \mathbf{x}(k)\Big), \tag{3.1}$$

where $\mathcal{T}_1(\cdot)$ and $\mathcal{T}_2(\cdot)$ are functions of enclosed variables. $\mathcal{T}_1(\mathbf{z}(k))$ is independent of the state $\mathbf{x}(k)$ and can be considered as the normalization constant. In other words, when node l, $(1 \leq l \leq N)$, knows the sufficient statistic $\mathcal{S}(\mathbf{z}(k))$ it can evaluate the global likelihood $P(\mathbf{z}(k)|\mathbf{x}(k))$ locally for any given value of the state vector $\mathbf{x}(k)$ or its vector particle representation $\mathbb{X}_i^{(l)}(k)$. Below, I define the local and global sufficient statistics.

Definition 1. *Any sufficient statistic that pertains to the overall observation* $\mathbf{z}(k)$ *used to describe the global likelihood* $P(\mathbf{z}(k)|\mathbf{x}(k))$ *is called the global sufficient statistic (GSS)* $G(k)$.

Definition 2. *Any sufficient statistic that pertains to the local observation ($\mathbf{z}^{(l)}(k)$, for $1 \le l \le N$) used to describe the global sufficient statistics is referred to as the local sufficient statistic (LSS) $\mathcal{Y}^{(l)}(k)$.*

The following lemma provides the conditions for the existence of LSS and GSS, and relates the GSS of the global likelihood function to the LSSs at node l, $(1 \le l \le N)$.

Lemma 1. *If the global likelihood $P(\mathbf{z}(k)|\mathbf{x}(k))$ at iteration k satisfies the factorization defined in Eq. (2.14) and the local likelihood $P(\mathbf{z}^{(l)}(k)|\mathbf{x}(k))$ possesses a sufficient statistic $\mathcal{Y}^{(l)}(k)$, ($1 \le l \le N$), then $\{\mathcal{Y}^{(1)}(k), \ldots, \mathcal{Y}^{(N)}(k)\}$ are jointly sufficient for estimating $\mathbf{x}(k)$ in terms of the global likelihood function.*

The proof of Lemma 1 is included in Appendix A.1. With some additional constraints on the nature of the factorization admitted by $P(\mathbf{z}(k)|\mathbf{x}(k))$, there exists a function $\mathcal{S}(\cdot)$ such that the GSS $G(k)$ equals $\mathcal{S}(\mathcal{Y}^{(1)}(k), \ldots, \mathcal{Y}^{(N)}(k))$ as summarized in the following lemma.

Lemma 2. *Assuming the local observation are independent given the state variable which results in the following factorization of the global likelihood function*

$$P(\mathbf{z}(k)|\mathbf{x}(k)) = \prod_{l=1}^{N} P(\mathbf{z}^{(l)}(k)|\mathbf{x}(k)),$$

and let the global likelihood $P(\mathbf{z}(k)|\mathbf{x}(k))$ (similarly the local likelihood $P(\mathbf{z}^{(l)}(k)|\mathbf{x}(k))$ at node l) be factorizable, i.e.,

$$P(\mathbf{z}(k)|\mathbf{x}(k)) = h_1(\mathbf{z}(k))h_2(\mathbf{z}(k), \mathbf{x}(k))h_3(\mathbf{x}(k)) \tag{3.2}$$

with the conditions:

(i) $h_1(\mathbf{z}(k)) > 0$, and;

(ii) for nodes $i \ne j$

$$h_2^{(i)}(\mathbf{z}(k), \mathbf{x}(k))h_2^{(j)}(\mathbf{z}(k), \mathbf{x}(k)) = h_2\left(\phi(\mathbf{z}^{(i)}(k), \mathbf{z}^{(j)}(k)), \mathbf{x}(k)\right) h_4\left(\mathbf{z}^{(i)}(k), \mathbf{z}^{(j)}(k)\right), \tag{3.3}$$

then there exist LSSs $\{\mathcal{Y}^{(1)}(k), \ldots, \mathcal{Y}^{(N)}(k)\}$ and a function $\mathcal{S}(\cdot)$ such that the GSS is given by

$$G(k) = \mathcal{S}(\mathcal{Y}^{(1)}(k), \ldots, \mathcal{Y}^{(N)}(k)). \tag{3.4}$$

Note that $h_1(\cdot)$, $h_2(\cdot)$, $h_3(\cdot)$, $h_4(\cdot)$, as well as their localized counterparts $h_1^{(*)}(\cdot)$, $h_2^{(i)}(\cdot)$, $h_3^{(i)}(\cdot)$, $h_4^{(i)}(\cdot)$, and $\phi(\cdot)$ denote functions of the enclosed variables.

The proof of Lemma 2 is provided in Appendix A.2. Lemmas 1 and 2 show that the GSS can be represented as a function of the LSSs under the constraints specified in Eqs. (3.2)-(3.3). Several standard distributions satisfy these constraints including the Gaussian distribution for the observation noise $\zeta^{(l)}(k)$ at node l (a standard model used in the bearing and range tracking problems [103]). To provide more insight into the nature of the LSSs and GSSs, I consider the following simplified case of a distributed network with identical sensor nodes and Gaussian noise, i.e., all sensor nodes follow the same observation model

$$Z^{(l)}(k) = g(\mathbf{x}(k)) + \zeta^{(l)}(k), \tag{3.5}$$

for $(1 \leq l \leq N)$, where $\zeta^{(l)}(k) \sim \mathcal{N}(0, \sigma^{(l)^2}(k))$. Expressing the global likelihood (Eq. (2.14)) as

$$P(\mathbf{z}(k)|\mathbf{x}(k)) \propto \exp\Big\{ -\sum_{l=1}^{N} \underbrace{\frac{Z^{(l)^2}(k)}{2\sigma^{(l)^2}(k)}}_{\mathcal{Y}_1^{(l)}(k)} - g^2(\mathbf{x}(k)) \sum_{l=1}^{N} \underbrace{\frac{1}{2\sigma^{(l)^2}(k)}}_{\mathcal{Y}_2^{(l)}(k)} + g(\mathbf{x}(k)) \sum_{l=1}^{N} \underbrace{\frac{Z^{(l)}(k)}{\sigma^{(l)^2}(k)}}_{\mathcal{Y}_3^{(l)}(k)} \Big\}. \tag{3.6}$$

It is noted that node l, $(1 \leq l \leq N)$, has three LSSs, i.e., $\mathcal{Y}_1^{(l)}(k) = Z^{(l)^2}(k)/2\sigma^{(l)^2}(k)$, $\mathcal{Y}_2^{(l)}(k) = 1/2\sigma^{(l)^2}(k)$, and $\mathcal{Y}_3^{(l)}(k) = Z^{(l)}(k)/\sigma^{(l)^2}(k)$. The three LSSs will result in three GSSs as follows

$$G_1(k) = \sum_{l=1}^{N} \frac{Z^{(l)^2}(k)}{2\sigma^{(l)^2}(k)} = \sum_{l=1}^{N} \mathcal{Y}_1^{(l)}(k) \tag{3.7}$$

$$G_2(k) = \sum_{l=1}^{N} \frac{1}{2\sigma^{(l)^2}(k)} = \sum_{l=1}^{N} \mathcal{Y}_2^{(l)}(k) \tag{3.8}$$

$$G_3(k) = \sum_{l=1}^{N} \frac{Z^{(l)}(k)}{\sigma^{(l)^2}(k)} = \sum_{l=1}^{N} \mathcal{Y}_3^{(l)}(k). \tag{3.9}$$

The GSS can be computed by running average consensus algorithms on the GSS across the network. Note that the result of the average consensus algorithm needs to be multiplied by the number of nodes N to be used in Eq. (3.6). The number of nodes in the proposed CSS/DPF are assumed known. If not, one additional consensus run with all nodes set to 0 except for the originating node that is set to 1 can be used to determine the number of active nodes in the network.

In the CSS/DPF, I impose another constraint and restrict $\mathcal{S}(\cdot)$ defined in Lemma 2 to a summation such that a GSS can be computed efficiently using an average consensus algorithm. In other words, I design the LSSs and GSSs in the CSS/DPF such that $\mathcal{S}(\cdot)$ is given by the following summation

$$G(k) = \mathcal{S}\big(\mathcal{Y}^{(1)}(k), \ldots, \mathcal{Y}^{(N)}(k)\big) = \sum_{l=1}^{N} \mathcal{Y}^{(l)}(k). \tag{3.10}$$

Below, the bearing-only tracking (BOT) in two and three dimensions is considered, which is then extended to joint range/bearing tracking [102–104].

3.1.2 CSS/DPF for Bearing and Range Tracking

In applications with locally dependent observation models, $g^{(l)}(\mathbf{x}(k))$ is not only a function of the state variables $\mathbf{x}(k)$ but may also depend on additional local variables, say $\lambda^{(l)}(k)$. The BOT problem belongs to this category where the local observation model at node l, $(1 \le l \le N)$, is a function of the state variables and the coordinates $\{X^{(l)}(k), Y^{(l)}(k)\}$ of node l. In such scenarios, the observation model needs to be factorizable as follows

$$g^{(l)}(\mathbf{x}(k)) = g_1^{(l)}(\lambda^{(l)}(k)) \times g_2(\mathbf{x}(k)), \tag{3.11}$$

for which the LSSs and GSSs are computable. Next, the CSS/DPF is developed for the 2D bearing only tracking problems.

3.1.2.1 2D Bearing-only Tracking

Recall that bearing-only tracking (BOT) estimates the kinematics of the target (position and velocity). In the 2D tracking scenario, the state representing the target is defined as $\mathbf{x}(k) = [X(k)\ \ \dot{X}(k)\ \ Y(k)\ \ \dot{Y}(k)]^T$, where T denotes transposition, $[X, Y]$ the position, and $[\dot{X}, \dot{Y}]$ the velocity of the target. Sensor node l records the bearing between the sensor-target line of sight with respect to the platform of the sensor nodes referenced (clockwise positive) to the y-axis (azimuth) as

$$\mathbb{Z}_\theta^{(l)}(\mathbf{x}(k)) = \tan^{-1}\left(\frac{X(k) - X^{(l)}(k)}{Y(k) - Y^{(l)}(k)}\right), \tag{3.12}$$

where $\lambda^{(l)}(k) = (X^{(l)}(k), Y^{(l)}(k))$ are the known coordinates of node l. The scalar observation $Z_\theta^{(l)}(k)$ made at node l is the true bearing $\mathbb{Z}_\theta^{(l)}(\mathbf{x}(k))$ plus additive noise as follows

$$Z_\theta^{(l)}(k) = \mathbb{Z}_\theta^{(l)}(\mathbf{x}(k)) + \zeta_\theta^{(l)}(k). \tag{3.13}$$

The participating nodes can be either static or mobile. For mobile nodes, a cooperative self localization algorithm, based on the global positioning system (GPS) or using some other anchor-based algorithm [121] is required to ascertain the locations of the observation nodes. The CSS/DPF uses the following result to factorize the global likelihood for the 2D-BOT problem.

Theorem 2. *In an agent network comprising N local nodes with local bearing observations $Z_\theta^{(l)}(k)$, $(1 \leq l \leq N)$, and under conditions specified in Lemmas 1 and 2, the global likelihood function for the 2D BOT can be expressed as follows*

$$P\big(\mathbf{z}_\theta(k)|\mathbf{x}(k)\big) = \frac{1}{C_\theta(k)} \exp\Bigg\{ -\frac{1}{2}\Big[G_{\theta,1}(k) + X^2(k)G_{\theta,2}(k)$$
$$+ Y^2(k)G_{\theta,3}(k) - 2X(k)Y(k)G_{\theta,4}(k) + 2X(k)G_{\theta,5}(k) - 2Y(k)G_{\theta,6}(k)\Big]\Bigg\} \tag{3.14}$$

where

$$
\begin{aligned}
G_{\theta,1}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\left[\mathcal{Z}_\theta^{(l)}(k)\right]^2}{R_\theta^{(l)}(k)}}_{\mathcal{Y}_{\theta,1}^{(l)}(k)}, &
G_{\theta,2}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\cos^2\left(Z_\theta^{(l)}(k)\right)}{R_\theta^{(l)}(k)}}_{\mathcal{Y}_{\theta,2}^{(l)}(k)}, \\
G_{\theta,3}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\sin^2\left(Z_\theta^{(l)}(k)\right)}{R_\theta^{(l)}(k)}}_{\mathcal{Y}_{\theta,3}^{(l)}(k)}, &
G_{\theta,4}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\cos\left(Z_\theta^{(l)}(k)\right)\sin\left(Z_\theta^{(l)}(k)\right)}{R_\theta^{(l)}(k)}}_{\mathcal{Y}_{\theta,4}^{(l)}(k)}, \\
G_{\theta,5}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\mathcal{Z}_\theta^{(l)}(k)\cos\left(Z_\theta^{(l)}(k)\right)}{R_\theta^{(l)}(k)}}_{\mathcal{Y}_{\theta,5}^{(l)}(k)}, \quad \textit{and} &
G_{\theta,6}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\mathcal{Z}_\theta^{(l)}(k)\sin\left(Z_\theta^{(l)}(k)\right)}{R_\theta^{(l)}(k)}}_{\mathcal{Y}_{\theta,6}^{(l)}(k)}.
\end{aligned}
\tag{3.15}
$$

Parameter $R_\theta^{(l)}(k)$ is the variance of observation noise at node l, $C_\theta(k) = (2\pi)^{N/2}\prod_{i=1}^{N}(R_\theta^{(l)}(k))^{1/2}$, and

$$
\mathcal{Z}_\theta^{(l)}(k) = Y^{(l)}(k)\sin(Z_\theta^{(l)}(k)) - X^{(l)}(k)\cos(Z_\theta^{(l)}(k)),
\tag{3.16}
$$

with $(X^{(l)}(k), Y^{(l)}(k))$ the coordinate of node l at time k.

The proof of Theorem 2 is included in Appendix A.3. Terms $G_*(k)$ are the GSSs expressed as functions of the LSSs $\mathcal{Y}_*^{(l)}(k)$, $(1 \le l \le N)$. Theorem 2 shows that a total number of six global sufficient statistics (GSS) and an additional term $C_\theta(k)$ are needed at each local particle filter to be able to evaluate the global likelihood locally. Because the LSSs are only functions of local quantities, the six GSSs can be computed using six parallel average consensus algorithms. If needed, term $C_\theta(k)$ can also be computed distributively using another average consensus algorithm. After the consensus step, the global likelihood can be evaluated locally from the consensus values of the GSS $G_1(k)$ to $G_6(k)$ and $C_\theta(k)$. Based on Theorem 2, the CSS/DPF is explained in terms of Algorithm 3.

Algorithm 3 CSS/DPF IMPLEMENTATION FOR 2D BOT PROBLEM.

Input: $\{\mathbb{X}_i^{(l)}(k-1), W_i^{(l)}(k-1)\}_{i=1}^{N_p^{(l)}}$ and $\mathbf{z}^{(l)}(k)$.

Output: $\{\mathbb{X}_i^{(l)}(k), W_i^{(l)}(k)\}_{i=1}^{N_p^{(l)}}$, $\bar{\mathbf{x}}^{(l)}(k)$ and $\boldsymbol{P}^{(l)}(k)$.

Local node l, $(1 \leq l \leq N)$, performs the following steps to update its particle set for iteration k.

1: **Compute LSSs**: Node l, $(1 \leq l \leq N)$, computes the LSSs $(\mathcal{Y}_1^{(l)}(k) - \mathcal{Y}_6^{(l)}(k))$ from its local observation $Z_\theta^{(l)}(k)$ based on Eq. (3.23).

2: **Compute GSSs (Consensus Step)**: A total of 6 parallel average consensus algorithms are performed to compute the GSSs $(G_1(k) - G_6(k))$ as defined in Eq. (3.23).

3: **Particle Generation Step**: For each particle $\mathbb{X}_i^{(l)}(k-1)$, for $(1 \leq i \leq N_p^{(l)})$, a new *predicted particle* $\mathbb{X}_i^{(l)}(k)$ is sampled form the transitional density $P(\mathbf{x}(k)|\mathbf{x}(k-1))|_{x(k-1)=\mathbb{X}_i^{(k)}(k-1)}$ (the proposal distribution).

4: **Weight Update**: The weights associated with the predicted particles $\mathbb{X}_i^{(l)}(k)$ (computed in Step 2) are calculated based on the global likelihood (Eq. (3.14)) using the values of the GSSs computed in Step 3 as follows

$$W_i^{(l)}(k) = \frac{P\big(\mathbf{z}(k)|\mathbb{X}_i^{(l)}(k)\big)}{\sum_{i=1}^{N_s} P\big(\mathbf{z}(k)|\mathbb{X}_i^{(l)}(k)\big)}$$

5: **Compute State Estimates**: An approximation of the global MMSE state estimate $\bar{\mathbf{x}}^{(l)}(k)$ at node l is computed from $\{\mathbb{X}_i^{(l)}, W_i^{(l)}(k)\}_{i=1}^{N_s}$ and its corresponding error covariance $\boldsymbol{P}^{(l)}(k)$ as follows

$$\bar{\mathbf{x}}^{(l)}(k) = \sum_{i=1}^{N_p^{(l)}} W_i^{(l)}(k)\mathbb{X}_i^{(l)}(k) \tag{3.17}$$

$$\boldsymbol{P}^{(l)}(k) = \frac{1}{N_p^{(l)}} \sum_{i=1}^{N_p^{(l)}} \big(\mathbb{X}_i^{(l)}(k) - \bar{\mathbf{x}}^{(l)}(k)\big)\big(\mathbb{X}_i^{(l)}(k) - \bar{\mathbf{x}}^{(l)}(k)\big)^T \tag{3.18}$$

6: **Resampling**: To avoid degeneracy, the updated particles $\mathbb{X}_i^{(l)}(k)$ are resampled using Algorithm 2.

3.1.2.2 3D Bearing-Only Tracking

In this section, I extend the CSS/DPF to the 3D BOT problem with state vector $\mathbf{x}(k) = [X(k), \dot{X}(k), Y(k), \dot{Y}(k), Z(k), \dot{Z}(k)]^T$. Compared to 2D BOT, the Z-coordinate $Z(k)$ and its velocity component $\dot{Z}(k)$ are included in the state vector. For 3D BOT, measurements often involve a pairwise combination between azimuth bearing, conical bearing, or elevation bearing [102]. Without loss of generality, I consider the pair of azimuth and elevation bearings with the azimuth bearing given by Eq. (3.12). The elevation bearing is defined as

$$\mathbb{Z}_{\phi}^{(l)}(\mathbf{x}(k)) = \tan^{-1}\left(\frac{Z(k) - Z^{(l)}(k)}{\mathbb{Z}_{\mathrm{R}}^{(l)}(\mathbf{x}(k))}\right), \tag{3.19}$$

with the overall observation model

$$\begin{bmatrix} Z_{\theta}^{(l)}(k) \\ Z_{\phi}^{(l)}(k) \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\frac{X(k)-X^{(l)}(k)}{Y(k)-Y^{(l)}(k)}\right) \\ \tan^{-1}\left(\frac{Z(k)-Z^{(l)}(k)}{\mathbb{Z}_{\mathrm{R}}^{(l)}(\mathbf{x}(k))}\right) \end{bmatrix} + \begin{bmatrix} \zeta_{\theta}^{(l)}(k) \\ \zeta_{\phi}^{(l)}(k) \end{bmatrix} \tag{3.20}$$

at node l. Term $\mathbb{Z}_{\mathrm{R}}^{(l)}(\mathbf{x}(k))$ is the true range between the sensor node and the target as follows

$$\mathbb{Z}_{\mathrm{R}}^{(l)}(\mathbf{x}(k)) = \sqrt{\left(X(k)-X^{(l)}(k)\right)^2 + \left(Y(k)-Y^{(l)}(k)\right)^2} \tag{3.21}$$

and $(X^{(l)}(k), Y^{(l)}(k), Z^{(l)}(k))$ is the 3D coordinate of the sensor node l.

Theorem 3. *In an agent network comprising of N sensor nodes with elevation bearing observations $Z_{\phi}^{(l)}(k)$ and under conditions specified in Lemmas 1 and 2, the global likelihood function for the 3D BOT problem can be expressed collectively in terms of Eqs.* (3.14) *and the following equation*

$$\begin{aligned} P\big(\mathbf{z}_{\phi}(k)|\mathbf{x}(k)\big) &= \frac{1}{C_{\phi}(k)} \exp\left\{\frac{-1}{2}\Big(G_{\phi,1}(k) - 2Z(k)G_{\phi,2}(k) + 2X(k)G_{\phi,3}(k)\right. \\ &+ 2Y(k)G_{\phi,4}(k) + Z^2(k)G_{\phi,5}(k) + X^2(k)G_{\phi,6}(k) + Y^2(k)G_{\phi,7}(k) \\ &- \left. 2X(k)Z(k)G_{\phi,8}(k) - 2Y(k)Z(k)G_{\phi,9}(k) + 2X(k)Y(k)G_{\phi,10}(k)\Big)\right\} \end{aligned} \tag{3.22}$$

where

$$\begin{aligned}
G_{\phi,1}(k) &= \sum_{l=1}^{N} \underbrace{\frac{(\mathcal{Z}_{\phi}^{(l)}(k))^2}{R_{\phi}^{(l)}(k)}}_{\mathcal{Y}_{\phi,1}^{(l)}(k)}, \\
G_{\phi,2}(k) &= \sum_{l=1}^{N} \underbrace{\frac{(\mathcal{Z}_{\phi}^{(l)}(k))^2 \cos^2(Z_{\phi}^{(l)}(k))}{(R_{\phi}^{(l)}(k)}}_{\mathcal{Y}_{\phi,2}^{(l)}(k)}, \\
G_{\phi,3}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\mathcal{Z}_{\phi}^{(l)}(k) \sin(Z_{\theta}^{(l)}(k)) \sin(Z_{\phi}^{(l)}(k))}{R_{\phi}^{(l)}(k)}}_{\mathcal{Y}_{\phi,3}^{(l)}(k)}, \\
G_{\phi,4}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\mathcal{Z}_{\phi}^{(l)}(k) \cos(Z_{\theta}^{(l)}(k)) \sin(Z_{\phi}^{(l)}(k))}{R_{\phi}^{(l)}(k)}}_{\mathcal{Y}_{\phi,4}^{(l)}(k)}, \\
G_{\phi,5}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\cos^2(Z_{\phi}^{(l)}(k))}{R_{\phi}^{(l)}(k)}}_{\mathcal{Y}_{\phi,5}^{(l)}(k)}, \\
G_{\phi,6}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\cos^2(Z_{\theta}^{(l)}(k)) \sin^2(Z_{\phi}^{(l)}(k)))}{R_{\phi}^{(l)}(k)}}_{\mathcal{Y}_{\phi,6}^{(l)}(k)}, \\
G_{\phi,7}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\sin^2(Z_{\theta}^{(l)}(k)) \sin^2(Z_{\phi}^{(l)}(k))}{R_{\phi}^{(l)}(k)}}_{\mathcal{Y}_{\phi,7}^{(l)}(k)}, \\
G_{\phi,8}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\sin(Z_{\theta}^{(l)}(k)) \sin(Z_{\phi}^{(l)}(k)) \cos(Z_{\phi}^{(l)}(k))}{R_{\phi}^{(l)}(k)}}_{\mathcal{Y}_{\phi,8}^{(l)}(k)} \\
G_{\phi,9}(k) &= \sum_{l=1}^{N} \underbrace{\frac{\cos(Z_{\theta}^{(l)}(k)) \sin(Z_{\phi}^{(l)}(k)) \cos(Z_{\phi}^{(l)}(k))}{R_{\phi}^{(l)}(k)}}_{\mathcal{Y}_{\phi,9}^{(l)}(k)}, \\
G_{\phi,10}(k) &= \sum_{l=1}^{N} \sum_{l=1}^{N} \underbrace{\frac{\cos(Z_{\theta}^{(l)}(k)) \sin(Z_{\theta}^{(l)}(k)) \cos(Z_{\theta}^{(l)}(k))}{R_{\phi}^{(l)}(k)}}_{\mathcal{Y}_{\phi,10}^{(l)}(k)},
\end{aligned} \tag{3.23}$$

with

$$C_{\phi}(k) = (2\pi)^{N/2}\prod_{i=1}^{N}(R_{\phi}^{(l)}(k))^{1/2}, \tag{3.24}$$

the elevation bearing noise variance [119] given by

$$R_{\phi}^{(l)}(k)=E\{X^2(k)+Y^2(k)+Z^2(k)\}(1-\exp^{-4\sigma_{\phi}^{(l)^2}})/4, \tag{3.25}$$

and parameter

$$\mathcal{Z}_{\phi}^{(l)}(k)=Z_{\phi}^{(l)}(k)\cos(Z_{\phi}^{(l)}(k))-X^{(l)}(k)\sin(Z_{\phi}^{(l)}(k))\sin(Z_{\theta}^{(l)}(k))-Y^{(l)}(k)\sin(Z_{\phi}^{(l)}(k))\cos(Z_{\theta}^{(l)}(k)). \tag{3.26}$$

The proof of Theorem 3 is included in Appendix A.4. The CSS/DPF algorithm for 3D BOT tracking is similar to the 2D BOT scenario except for Steps 1 and 2, where LSSs $\{\mathcal{Y}_{\phi,i}^{(l)}(k)\}$ and associated GSSs for elevation $\{G_{\phi,i}^{(l)}(k)\}$, $(1 \leq i \leq 10)$, are needed in addition to the LSSs $\{\mathcal{Y}_{\theta,i}^{(l)}(k)\}$ and GSSs $\{G_{\theta,i}^{(l)}(k)\}$, $(1 \leq i \leq 6)$, for azimuth. The number of consensus runs is 16 in this case.

3.1.2.3 2D Joint Bearing and Range Tracking

In 2D joint bearing and range tracking, the range measurements (as defined below) are available in addition to the bearing measurements (Eq. (3.12)) at all local nodes. The overall observation model is given by

$$\begin{bmatrix} Z_{\theta}^{(l)}(k) \\ Z_{\mathrm{R}}^{(l)}(k) \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\frac{X(k)-X^{(l)}(k)}{Y(k)-Y^{(l)}(k)}\right) \\ \sqrt{\left(X(k)-X^{(l)}(k)\right)^2+\left(Y(k)-Y^{(l)}(k)\right)^2} \end{bmatrix} + \begin{bmatrix} \zeta_{\theta}^{(l)}(k) \\ \zeta_{\mathrm{R}}^{(l)}(k) \end{bmatrix}, \tag{3.27}$$

where the range observation noise $\zeta_{\mathrm{R}}^{(l)}(\cdot)$ is assumed to be independent of bearing observation noise $\zeta_{\theta}^{(l)}(k)$. The global likelihood for the range observations is expressed in terms of the LSSs and GSSs in the following theorem.

Theorem 4. *In an agent network comprising of N local nodes with range and bearing observations $\{Z_R^{(l)}(k),\ Z_\theta^{(l)}(k)\}$, $(1 \le l \le N)$, and under conditions specified in Lemmas 1 and 2, the global likelihood function can be expressed as Eqs.* (3.14) *and* (3.28) *given by*

$$P\big(\mathbf{z}_R(k)|\mathbf{x}(k)\big) = \frac{1}{C_R(k)} \exp\Bigg\{ -\frac{1}{2}\Big[G_{R,1}(k) + X^2(k)G_{R,2}(k)$$
$$+Y^2(k)G_{R,3}(k) + 2X(k)Y(k)G_{R,4}(k) - 2X(k)G_{R,5}(k) - 2Y(k)G_{R,6}(k)\Big]\Bigg\} \tag{3.28}$$

where

$$G_{R,1}(k)=\sum_{l=1}^{N}\underbrace{\frac{\left[\mathcal{Z}_R^{(l)}(k)\right]^2}{R_R^{(l)}(k)}}_{\mathcal{Y}_{R,1}^{(l)}(k)}, \qquad G_{R,2}(k)=\sum_{l=1}^{N}\underbrace{\frac{\sin^2\left(Z_\theta^{(l)}(k)\right)}{R_R^{(l)}(k)}}_{\mathcal{Y}_{R,2}^{(l)}(k)},$$
$$G_{R,3}(k)=\sum_{l=1}^{N}\underbrace{\frac{\cos^2\left(Z_\theta^{(l)}(k)\right)}{R_R^{(l)}(k)}}_{\mathcal{Y}_{R,3}^{(l)}(k)}, \qquad G_{R,4}(k)=-\sum_{l=1}^{N}\underbrace{\frac{\cos\left(Z_\theta^{(l)}(k)\right)\sin\left(Z_\theta^{(l)}(k)\right)}{R_R^{(l)}(k)}}_{\mathcal{Y}_{R,4}^{(l)}(k)}, \tag{3.29}$$
$$\text{and} \quad G_{R,5}(k)=\sum_{l=1}^{N}\underbrace{\frac{\mathcal{Z}_R^{(l)}(k)\sin\left(Z_\theta^{(l)}(k)\right)}{R_R^{(l)}(k)}}_{\mathcal{Y}_{R,5}^{(l)}(k)}, \qquad G_{R,6}(k)=\sum_{l=1}^{N}\underbrace{\frac{\mathcal{Z}_R^{(l)}(k)\cos\left(Z_\theta^{(l)}(k)\right)}{R_R^{(l)}(k)}}_{\mathcal{Y}_{R,6}^{(l)}(k)},$$

where $C_R(k)$ is the normalization factor independent of the state variables, $R_R^{(l)}(k)$ is the variance of node l's range observation noise, and

$$\mathcal{Z}_R^{(l)}(k) = Z_R^{(l)}(k) + X^{(l)}(k)\sin(Z_\theta^{(l)}(k)) + Y^{(l)}(k)\cos(Z_\theta^{(l)}(k)). \tag{3.30}$$

The proof of Theorem 4 is included in Appendix A.5. Algorithm 3 can again be applied to estimate the states except for Steps 1 and 2, where LSSs $\{\mathcal{Y}_{R,i}^{(l)}(k)\}$ and associated GSSs for range $\{G_{R,i}^{(l)}(k)\}$, $(1 \le i \le 6)$, are needed in addition to the LSSs $\{\mathcal{Y}_{\theta,i}^{(l)}(k)\}$ and GSSs $\{G_{\theta,i}^{(l)}(k)\}$, $(1 \le i \le 6)$, for azimuth. The number of consensus runs is 12 in this case.

3.1.2.4 Adaptation of the CSS/DPF to Dynamic Networks

In this section, we investigate the application of sufficient statistics-based cooperative target localization approach (CSS/DPF) to dynamic networks, where nodes join and leave the cooperation at any time. In this context, two situations are observed which are describe next. Note that in the CSS/DPF, the LSSs are being communicated between the neighbouring nodes. Next we assume, without loss of generality, that tnode m joins/rejoins the network at iteration k of the CSS/DPF.

1. **A Brand New Node Joins the Cooperation:** The new node has no previous state estimates available and needs to go through an initialization stage. One of the neighbouring nodes transfers the global filtering density $P(\mathbf{x}(k-1)|\mathbf{z}(1:k-1))$ to the new node. Since this is an one time initialization, therefore, the initialization overhead is bearable and a good aproximation of $P(\mathbf{x}(k-1)|\mathbf{z}(1:k-1))$ (such as the Gaussian Mixture Model (GMM) [17] and Parzen representation [27]) of the noeighbouring node can be transfered to the new node, which now joins the network. At iteration k, the new node makes an observation $\mathbf{z}^{(m)}(k)$ and calculates the LSSs which are based on only its local observation. It now starts contributing to the consensus step of the CSS/DPF. Once the consensus step converges, all nodes including the new node has access to the GSSs. Given $P(\mathbf{x}(k-1)|\mathbf{z}(1:k-1))$ and the GSS, the new node can form its own global state estimates and is now a full member of the network.

2. **A previously cooperating node that had left the network rejoins the cooperation:** In this scenario, we assume that the node was making its own observations prior to rejoining and has its own local estimates as well as the local filtering density $P(\mathbf{x}(k-1)|\mathbf{z}^{(m)}(1:k-1))$. Node m has two options. It can either treat itself as a new node joining the network and follow the peocedure outlined for Case 1. Allternatively, it can combine/fuse its local filtering

density $P(\mathbf{x}(k\text{–}1)|\mathbf{z}^{(m)}(1:k\text{–}1))$ with the global filtering density $P(\mathbf{x}(k\text{–}1)|\mathbf{z}(1:k\text{–}1))$ obtained from one of the neighbouring nodes and rejoin the network as a full member cooperating in the consensus step for computing the GSSs. However, there is an issue with combining $P(\mathbf{x}(k-1)|\mathbf{z}^{(m)}(1:k-1))$ with $P(\mathbf{x}(k-1)|\mathbf{z}(1:k-1))$. Due to existing correlations, direct fusion is not feasable as it results in double counting of common information and degrades the overall performance. A conservative approach, e.g., covariance intersection can be applied.

3.1.2.5 Communication Complexity

The overall communication complexity of the CSS/DPF for the angle-only target localization problem at each node (i.e., the number of messages transferred at each iteration of the distributed particle filter) is of $\mathrm{O}((n_{GSS}+1)\Delta_G N_c(\boldsymbol{U}))$ where $N_c(\boldsymbol{U})$ is the total number of consensus iterations required for convergence. Recall that the consensus matrix $\boldsymbol{U}$ is a function of the connectivity of the network. It can be shown [120] that $N_c(\boldsymbol{U}) = -1/\max_{2\leq i\leq N}\log(|\lambda_i(\boldsymbol{U})|)$, where $\lambda_i(U)$ are the eigenvalues of the consensus matrix $\boldsymbol{U}$. The communication complexity of the CSS/DPF is, therefore, related to the properties of the communication network. For [23,24,59], the communication complexity is of $\mathrm{O}(n_x^2\Delta_G N_c(\boldsymbol{U}))$, which implies an improvement by a factor of $n_x^2/(n_{GSS}+1)$ in favor of the CSS/DPF. The computational complexity of the CSS/DPF is difficult to compute due to presence of the non-linear terms. Note, however, that the computational burden in the CSS/DPF is distributed evenly across the nodes, while the fusion center performs most of the computations in the centralized particle filter. In general, the number of computations at each node in the distributed implementation is significantly lower than these of the fusion centre in its centralized counterpart. This places an additional power energy constraint on the fusion center causing the system to fail if the power of the fusion center drains out.

In conclusion, the CSS/DPF is a distributed implementation of the SIR filter and belongs to the DPF via likelihood/observation fusion category (Section 2.5). The CSS/DPF implementation significantly reduces the number of required consensus runs for bearing-only and joint bearing-range tracking applications. The CSS/DPF requires the global likelihood function to satisfy conditions of Lemma 1 and 2. Though fairly straightforward and simple to implement, the CSS/DPF has the following drawbacks.

1. The CSS/DPF is designed specifically for bearing-only and joint bearing and range tracking applications. Extending the CSS/DPF to other applications is generally not straightforward.

2. Choosing the transitional distribution $P(\mathbf{x}(k)|\mathbf{x}(k-1))$ as the proposal distribution is not optimal.

3. In the CSS/DPF, some function of the local observations are transferred to neighbouring nodes. Communicating state posteriors is advantageous over communicating functions of local observations or local likelihoods because the later would result loss of information in case packets are lost. If instead, information on the state posteriors is communicated, lost information can be recovered since it is implicitly present in the future state posterior.

4. The CSS/DPF is limited to the Gaussian likelihoods.

5. As is the case for the existing consensus-based distributed particle filter implementations [18, 20, 23, 24], the CSS/DPF assumes that the consensus algorithm converges within the time interval available between two successive observations. Such an assumption in large networks is non-realistic and the consensus step loses synchronization with localized filters.

Next, I extend the proposed framework (CSS/DPF) to distributed implementation of the unscented particle filter, referred to as the CSS/DUPF which addresses drawbacks 2 and 3 of the CSS/DPF as follows:

1. Instead of choosing the transitional distribution $P(\mathbf{x}(k)|\mathbf{x}(k-1))$ as the proposal distribution, the CSS/DUPF uses an approximation of the optimal proposal distribution and therefore uses all available global observation including the most recent ones in deriving the proposed distribution.

2. Unlike the CSS/DPF where only local observations were transferred between the neighbouring nodes, in the CSS/DUPF local state estimates are also communicated between neighbouring nodes. Therefore, in the CSS/DUPF, lost information (e.g., due to link and/or node failure) can be recovered since it is implicitly present in the future state estimates.

3.2 The CSS/DUPF Implementation

The CSS/DUPF couples a distributed unscented Kalman filter (D/UKF) with the CSS/DPF such that the optimal proposal distribution function (Eq. (2.78)) is approximated with a Gaussian distribution whose statistics (mean and error covariance matrix) are computed using the D/UKF estimates. The CSS/UDPF is assumed to be in steady state and at iteration $k-1$, i.e., all nodes have computed the global state estimates ($\bar{\mathbf{x}}^{(l)}(k-1)$ and $\bar{\boldsymbol{P}}^{(l)}(k-1)$) at time instant $k-1$ (based on Step 5 of Algorithm 3). A new measurement $\mathbf{z}^{(l)}(k)$ is now available at the local nodes.

Step 1. Similar to CSS/DPF, node l, for $(1 \leq l \leq N)$, computes its LSSs and fuse them distributively to form the GSSs. Based on the computed GSSs, node l, can locally. evaluate the global likelihood $P(\mathbf{z}(k)|\mathbf{x}(k))$ for any given particles.

Step 2. Node l generates a set of $(2n_x+1)$ deterministic samples (referred to as the sigma points) $\mathcal{S} = \{\mathcal{W}_i^{(l)}, \chi_i^{(l)}(k)\}_{i=0}^{2n_x}$ based on the following selection procedure

$$\chi_i^{(l)}(k-1) = \bar{\mathbf{x}}^{(l)}(k-1) \pm \left\{\sqrt{(n_x+\kappa)\bar{\boldsymbol{P}}^{(l)}(k-1)}\right\}_i, \tag{3.31}$$

where term $\{\sqrt{(n_x+\kappa)\bar{\boldsymbol{P}}^{(l)}(k-1)}\}_i$ corresponds to the i^{th} column of the square root of matrix $(n_x+\kappa)\bar{\boldsymbol{P}}^{(l)}(k-1)$ and the initial condition is given by $\boldsymbol{\chi}_0^{(l)}(k) = \bar{\mathbf{x}}^{(l)}(k-1)$. The corresponding weights for the Sigma points $\{\mathcal{W}_i\}_{i=1}^{2n_x}$ are given by $\mathcal{W}_i^{(l)} = 1/(2(n_x+\kappa))$, where κ is a scaling parameter and the initial condition for the sigma points is $\mathcal{W}_0^{(l)} = \kappa/(n_x+\kappa)$.

Step 3. Node l, $(1 \le l \le N)$, computes an estimate of its local posterior as follows.

Step 3.1 The sigma points computed in Step 1 are propagated through the state model (Eq. (2.3)) to generate the predicted sigma points

$$\boldsymbol{\chi}_i^{(l)}(k|k-1) = \boldsymbol{f}(\boldsymbol{\chi}_i^{(l)}(k-1)), \quad \text{for } i = 0, \ldots, 2n_x. \tag{3.32}$$

Step 3.2 The predicted sigma points $\boldsymbol{\chi}_i^{(l)}(k|k-1)$ are then propagated through the observation model (Eq. (3.12) and/or Eq. (3.21)) to generate the predicted observation sigma points

$$\boldsymbol{\mathcal{Z}}_i^{(l)}(k|k-1) = \boldsymbol{g}(\boldsymbol{\chi}_i^{(l)}(k|k-1)), \quad \text{for } i = 0, \ldots, 2n_x. \tag{3.33}$$

Step 3.3 The predicted state estimate $\mathbf{x}_{\text{UKF}}^{(l)}(k|k-1)$, its error covariance matrix $\boldsymbol{P}_{\text{UKF}}^{(l)}(k|k-1)$, and the predicted observation estimate $\hat{\mathbf{z}}_{\text{UKF}}^{(l)}(k|k-1)$ are computed as follows

$$\hat{\mathbf{x}}_{\text{UKF}}^{(l)}(k|k-1) = \sum_{i=0}^{2n_x} \mathcal{W}_i^{(l)} \boldsymbol{\chi}_i^{(l)}(k|k-1), \tag{3.34}$$

$$\boldsymbol{P}_{\text{UKF}}^{(l)}(k|k-1) = \sum_{i=0}^{2n_x} \mathcal{W}_i^{(l)} \Big(\boldsymbol{\chi}_i^{(l)}(k|k-1) - \mathbf{x}_{\text{UKF}}^{(l)}(k|k-1)\Big)\Big(\boldsymbol{\chi}_i^{(l)}(k|k-1) - \mathbf{x}_{\text{UKF}}^{(l)}(k|k-1)\Big)^T \tag{3.35}$$

$$\hat{\mathbf{z}}_{\text{UKF}}^{(l)}(k|k-1) = \sum_{i=0}^{2n_x} \mathcal{W}_i^{(l)} \boldsymbol{\mathcal{Z}}_i^{(l)}(k|k-1). \tag{3.36}$$

Step 3.4 The autocovariance $\boldsymbol{P}_{zz}(k|k-1)$ of predicted observations, the cross-covariance $\boldsymbol{P}_{xz}(k|k-$

1) between predicted observation and predicted state estimates are computed as

$$\boldsymbol{P}_{zz}^{(l)}(k|k-1) = \sum_{i=0}^{2n_x} \mathcal{W}_i^{(l)} \Big(\boldsymbol{\mathcal{Z}}_i^{(l)}(k|k-1) - \mathbf{z}^{(l)}(k+1|k)\Big)\Big(\boldsymbol{\mathcal{Z}}_i^{(l)}(k|k-1) - \mathbf{z}^{(l)}(k|k-1)\Big)^T, \quad (3.37)$$

$$\boldsymbol{P}_{xz}^{(l)}(k|k-1) = \sum_{i=0}^{2n_x} \mathcal{W}_i^{(l)} \Big(\boldsymbol{\chi}_i^{(l)}(k|k-1) - \mathbf{x}_{\text{UKF}}^{(l)}(k|k-1)\Big)\Big(\boldsymbol{\mathcal{Z}}_i^{(l)}(k|k-1) - \mathbf{z}_{\text{UKF}}^{(l)}(k|k-1)\Big)^T \quad (3.38)$$

Step 3.5 The final step is to estimate the statistics of the proposal distribution as follows

$$\hat{\mathbf{x}}_{\text{UKF}}^{(l)}(k) = \hat{\mathbf{x}}_{\text{UKF}}^{(l)}(k|k-1) + \boldsymbol{\mathcal{K}}^{(l)}(k)\Big(\mathbf{z}^{(l)}(k) - \hat{\mathbf{z}}_{\text{UKF}}^{(l)}(k|k-1)\Big) \quad (3.39)$$

$$\boldsymbol{P}_{\text{UKF}}^{(l)}(k) = \boldsymbol{P}_{\text{UKF}}^{(l)}(k|k-1) - \boldsymbol{\mathcal{K}}^{(l)}(k)\boldsymbol{P}_{zz}^{(l)}(k|k-1)[\boldsymbol{\mathcal{K}}^{(l)}(k)]^T, \quad (3.40)$$

where the Kalman gain is given by

$$\boldsymbol{\mathcal{K}}^{(l)}(k) = \boldsymbol{P}_{xz}^{(l)}(k|k-1)\boldsymbol{P}_{zz}^{(l)}(k|k-1)^{-1}. \quad (3.41)$$

Step 4. The next step in the CSS/DUPF is to cooperatively compute statistics of the proposal distribution. Based on the Chong-Mori-Chang track-fusion theorem [127], the CSS/DUPF algorithm uses the following fusion to fuse local statistics $\{\hat{\mathbf{x}}_{\text{UKF}}^{(l)}(k), \boldsymbol{P}_{\text{UKF}}^{(l)}(k)\}_{l=1}^{N}$ into a common set of global statistics denoted by $\hat{\mathbf{x}}_{\text{UKF}}^{(l,\text{Fused})}(k)$ and $\boldsymbol{P}_{\text{UKF}}^{(l,\text{Fused})}(k)$

$$[\boldsymbol{P}_{\text{UKF}}^{(l,\text{Fused})}(k)]^{-1} = [\boldsymbol{P}_{\text{UKF}}^{(l)}(k|k-1)]^{-1} + \underbrace{\sum_{j=1}^{N}[\boldsymbol{P}_{\text{UKF}}^{(j)}(k)]^{-1} - [\boldsymbol{P}_{\text{UKF}}^{(j)}(k|k-1)]^{-1}}_{\boldsymbol{P}_c(\infty)} \quad (3.42)$$

$$\hat{\mathbf{x}}_{\text{UKF}}^{(l.\text{Fused})}(k) = [\boldsymbol{P}_{\text{UKF}}^{(l,\text{Fused})}(k)]^{-1}\Big[[\boldsymbol{P}_{\text{UKF}}^{(l)}(k|k-1)]^{-1}\mathbf{x}_{\text{UKF}}^{(l)}(k|k-1) + \underbrace{\sum_{j=1}^{N}[\boldsymbol{P}_{\text{UKF}}^{(j)}(k)]^{-1}\mathbf{x}_{\text{UKF}}^{(j)}(k) - [\boldsymbol{P}_{\text{UKF}}^{(j)}(k|k-1)]^{-1}\mathbf{x}_{\text{UKF}}^{(j)}(k|k-1)}_{\mathbf{x}_c(\infty)}\Big], \quad (3.43)$$

In Eqs. (3.42) and (3.43), $\{\mathbf{x}_c(\infty), \boldsymbol{P}_c(\infty)\}$ are obtained by iterating the following average consensus equations where $\epsilon \in (0, 1/\Delta_{\mathcal{G}})$ [32].

$$\mathbf{x}_c^{(l)}(t+1) = \mathbf{x}_c^{(l)}(t) + \epsilon \sum_{j\in\aleph^{(l)}} (\mathbf{x}_c^{(j)}(t) - \mathbf{x}_c^{(l)}(t)) \quad (3.44)$$

$$\boldsymbol{P}_c^{(l)}(t+1) = \boldsymbol{P}_c^{(l)}(t) + \epsilon \sum_{j\in\aleph^{(l)}} (\boldsymbol{P}_c^{(j)}(t) - \boldsymbol{P}_c^{(l)}(t)), \quad (3.45)$$

till converge to $\{\mathbf{x}_c(\infty), \boldsymbol{P}_c(\infty)\}$. The initial conditions are

$$\boldsymbol{P}_c^{(l)}(t=0) = [\boldsymbol{P}_{\text{UKF}}^{(j)}(k)]^{-1} - [\boldsymbol{P}_{\text{UKF}}^{(j)}(k|k-1)]^{-1} \tag{3.46}$$

$$\mathbf{x}_c^{(l)}(t=0) = [\boldsymbol{P}_{\text{UKF}}^{(j)}(k)]^{-1}\hat{\mathbf{x}}_{\text{UKF}}^{(j)}(k) - [\boldsymbol{P}_{\text{UKF}}^{(j)}(k|k-1)]^{-1}\hat{\mathbf{x}}_{\text{UKF}}^{(j)}(k|k-1). \tag{3.47}$$

In other words, Eq. (2.116) is used to reach consensus with $\mathbf{x}_c^{(l)}(t)$ used instead of $X_c^{(l)}(t)$ for the first consensus run and $\boldsymbol{P}_c^{(l)}(t)$ used instead of $X_c^{(l)}(t)$ for the second run.

Step 5. Node l, for $(1 \leq l \leq N)$, generates N_s random particles $\mathbb{X}_i^{(l)}(k)$ from the following proposal distribution

$$\mathbb{X}_i(k) \sim q\big(\mathbf{x}(k)|\mathbb{X}_i(0\!:k-1), \mathbf{z}(1\!:k)\big) \triangleq \mathcal{N}\big(\hat{\mathbf{x}}_{\text{UKF}}^{(l,\text{Fused})}(k), \boldsymbol{P}_{\text{UKF}}^{(1,\text{Fused})}(k)\big), \tag{3.48}$$

and computes their associated weights $W_i^{(l)}(k)$ based on the following weight update equation

$$W_i^{(l)}(k) = \frac{P\big(\mathbf{z}(k)|\mathbb{X}_i^{(l)}(k)\big)P\big(\mathbb{X}_i^{(l)}(k)|\mathbb{X}_i^{(l)}(k-1)\big)}{\mathcal{N}\big(\mathbb{X}_i^{(l)}(k); \hat{\mathbf{x}}_{\text{UKF}}^{(l,\text{Fused})}(k), \boldsymbol{P}_{\text{UKF}}^{(l,\text{Fused})}(k)\big)}, \tag{3.49}$$

where the global likelihood function $P(\mathbf{z}(k)|\mathbf{x}(k))$ is computed based on the GSSs. The implementation of the CSS/DUPF is outlined in Algorithm 4.

Similar to the CSS/DPF, the CSS/DUPF is applicable specifically to bearing-only and joint bearing/range tracking applications. Extending the CSS/DUPF to other applications is generally not straightforward. Besides, the CSS/DUPF restricts the global likelihood a Gaussian distribution. Next, I propose the UCD/DPF implementation of the particle filter which is applicable to more general problems and addresses does not require the global likelihood to be a Gaussian distribution.

3.3 Simulation Results for the CSS/DPF and CSS/DUPF

In this section, the performances of the proposed CSS/DPF, CSS/DUPF, and UCD/DPF are evaluated through Monte Carlo simulations. All simulations were performed using a commer-

Algorithm 4 CSS/DUPF IMPLEMENTATION

Input: $\{\mathbb{X}_i^{(l)}(k-1), W_i^{(l)}(k-1)\}_{i=1}^{N_p^{(l)}}$, $\bar{\mathbf{x}}^{(l)}(k-1)$, $\bar{\boldsymbol{P}}^{(l)}(k-1)$, and $\mathbf{z}^{(l)}(k)$.

Output: $\{\mathbb{X}_i^{(l)}(k), W_i^{(l)}(k)\}_{i=1}^{N_p^{(l)}}$, $\bar{\mathbf{x}}^{(l)}(k)$ and $\bar{\boldsymbol{P}}^{(l)}(k)$.

Local node l performs the following steps to update its particle set for iteration (k).

1: **Compute LSSs**: Same as Step 1 of Algorithm 1.

2: **Compute Statistics of the Proposal Distribution**:

3A: **Compute GSSs (Consensus Step)**: Same as Step 2 of Algorithm 1.

- **Local UKF Step**: A local state estimate $\hat{\mathbf{x}}(k)$ is computed via local UKF based on (i) The previous global statistics ($\bar{\mathbf{x}}^{(l)}(k-1)$ and $\bar{\boldsymbol{P}}^{(l)}(k-1)$), and; (ii) Local observation $\mathbf{z}^{(l)}(k)$.
- **Fusion of Local UKFs (Consensus Step)**: Local state estimates and their corresponding error covariance matrix are combined to compute the statistics of the proposal distribution ($\hat{\mathbf{x}}_{\text{UKF}}^{(l,\text{Fused})}(k)$ and $\boldsymbol{P}_{\text{UKF}}^{(l,\text{Fused})}(k)$) using the fusion rules given by Eqs. (3.42) and (3.43).

3B: **Particle Generation Step**: For each particle $\mathbb{X}_i^{(l)}(k-1)$, for $(1 \le i \le N_p^{(l)})$, a new *predicted particle* $\mathbb{X}_i^{(l)}(k)$ is sampled form the following proposal distribution

$$\mathbb{X}_i^{(l)}(k) \sim \mathcal{N}\big(\hat{\mathbf{x}}_{\text{UKF}}^{(l,\text{Fused})}(k), \boldsymbol{P}_{\text{UKF}}^{(l,\text{Fused})}(k)\big)$$

where its statistics are available from Step 3A.

4: **Weight Update**: The weights associated with the predicted particles $\mathbb{X}_i^{(l)}(k)$ (computed in Step 2) are calculated from the global likelihood Eq. (3.14) and the proposal distribution computed in Step 3 as follows

$$W_i^{(l)}(k) = \frac{P\big(\mathbf{z}(k)|\mathbb{X}_i^{(l)}(k)\big)P\big(\mathbb{X}_i^{(l)}(k)|\mathbb{X}_i^{(l)}(k-1)\big)}{\mathcal{N}\big(\mathbb{X}_i^{(l)}(k); \hat{\mathbf{x}}_{\text{UKF}}^{(l,\text{Fused})}(k), \boldsymbol{P}_{\text{UKF}}^{(l,\text{Fused})}(k)\big)},$$

and then normalized.

5: **Compute State Estimates**: Same as Step 5 of Algorithm 1.

6: **Resampling**: Same as Step 6 of Algorithm 1.

cial software package (MATLAB R2012a, The MathWorks, Inc., Natick, Massachusetts, United States). Simulations were performed on a computer with Intel Core i5 CPU 2.27 GHz with 4 GB of RAM.

First, a distributed 2D BOT application [102] is used to quantify the performance of the proposed CSS/DPF and CSS/DUPF implementations. As stated in Section 2.6.1, the objective is to design a practical filter capable of estimating the kinematics (position $[X, Y]$ and velocity $[\dot{X}, \dot{Y}]$) of the target from the bearing measurements and prior knowledge of the target's motion. The state vector is, therefore, given by $\mathbf{x}(k) = [X(k), Y(k), \dot{X}(k), \dot{Y}(k)]$. BOT is inherently a non-linear application with its non-linearity incorporated either in the state dynamics or in the measurement model depending on the choice of the coordinate system used to formulate the problem. The nonlinear state model is given by $\mathbf{x}(k{+}1) = \boldsymbol{f}(\mathbf{x}(k))\mathbf{x}(k) + \boldsymbol{\xi}(k{+}1)$ where the target's motion $\boldsymbol{f}(\mathbf{x}(k))$ is described using different models such as: (i) Constant velocity (CV) model; (ii) Clockwise coordinated turn (CCT) model; (iii) Anticlockwise coordinated turn (ACT) model; (iv) Constant acceleration (CA) model, or; (v) some combination of (i)-(iv). In this section, $\boldsymbol{f}(\mathbf{x}(k))$ is considered to be the non-linear CCT kinematic motion model given by

$$\boldsymbol{f}(\mathbf{x}(k)) = \begin{bmatrix} 1 & 0 & \frac{\sin(\Omega(k)\Delta T)}{\Omega(k)} & -\frac{1-\cos(\Omega(k)\Delta T)}{\Omega(k} \\ 0 & 1 & \frac{1-\cos(\Omega(k)\Delta T)}{\Omega(k)} & \frac{\sin(\Omega(k)\Delta T)}{\Omega(k)} \\ 0 & 0 & \cos(\Omega(k)\Delta T) & -\sin(\Omega(k)\Delta T) \\ 0 & 0 & \sin(\Omega(k)\Delta T) & \cos(\Omega(k)\Delta T) \end{bmatrix}, \tag{3.50}$$

with the mode-conditioned turning rate $\Omega(k)$ given by

$$\Omega(k) = \frac{A_m}{\sqrt{(\dot{X}(k))^2 + (\dot{Y}(k))^2}} \tag{3.51}$$

The typical manoeuvre acceleration parameter for the filters was set to $a_m = 1.08{\times}10^{-5}\text{m/s}^2$ [103].

In the following simulations, an AN/SN is considered comprising of $N = 20$ observation nodes where sensors are distributed randomly in a (15×15) m^2 square region, unless stated otherwise.

Within the area under surveillance, each sensor communicates only within a connectivity radius of $\sqrt{2\log(N)/N}$ meters as previously used by [24]. In addition, the network is assumed to be connected with each node linked to at least one other node in the network. The measurements $Z^{(l)}(k)$ available at node l are the target's bearings with respect its platform referenced (clockwise positive) to the y-axis, i.e.,

$$Z^{(l)}(k) \quad = \quad \operatorname{atan}\left(\frac{X(k) - X^{(l)}}{Y(k) - Y^{(l)}}\right) + \zeta^{(l)}(k), \tag{3.52}$$

where $\{X^{(l)}, Y^{(l)}\}$ are the coordinates of node l. Both state and observation noises are assumed to be normally distributed, i.e., $\boldsymbol{\xi}(k) \sim \mathcal{N}(0, \boldsymbol{Q})$ and $\boldsymbol{\zeta}(k) \sim \mathcal{N}(0, \boldsymbol{R})$. Further, the observation noise model is assumed to be state dependent such that the bearing noise variance $\sigma^2_{\zeta^{(l)}}(k)$ at node l depends on the distance $r^{(l)}(k)$ between the observer and target. Based on [166], the variance of the observation noise at node l is given by

$$\sigma^2_{\zeta^{(l)}}(k) = B_m r^{(l)^2}(k) + 0.1150 r^{(l)}(k) + 0.7405, \tag{3.53}$$

where different values for parameter B_m are used to test various signal to noise ratios (SNR). In other words, $\boldsymbol{R}(k) = \operatorname{diag}[\sigma^2_{\zeta^{(l)}}(k)]$. In each run, the target starts its track from coordinates $\{10, 10\}$, with the initial course set at $-110°$ with the standard deviation of the process noise $\sigma_{\boldsymbol{\xi}(k)} = 1.6 \times 10^{-2}$ meter. Matrix $\boldsymbol{Q}$ depends on $\sigma_{\boldsymbol{\xi}(k)}$ as defined in [103]. Eqs. (3.50)-(3.53) define the state-space model completely (Eqs. (2.3) and (2.4)). The performance metric used to evaluate different implementation is the root mean square position error (RMS) [103] given by

$$\text{RMS}(k) = \sqrt{\frac{1}{n_{\text{MC}}}\frac{1}{N}\sum_{j=1}^{n_{\text{MC}}}\sum_{l=1}^{N}\left(X_j(k) - \hat{X}_j^{(l)}(k)\right)^2 + \left(Y_j(k) - \hat{Y}_j^{(l)}(k)\right)^2}, \tag{3.54}$$

where n_{MC} is the number of Monte Carlo simulations. In the following simulations, 100 Monte Carlo runs are implemented. Both the centralized and distributed BOT tracking algorithms require an initialization step which is described next.

3.3.1 BOT Initialization:

To derive the initial values for the state vector

$$\hat{\mathbf{x}}^{(l)}(0) = [\hat{X}^{(l)}(0), \hat{Y}^{(l)}(0), \hat{\dot{X}}^{(l)}(0), \hat{\dot{Y}}^{(l)}(0)]^T$$

at node l, the initialization procedure proceeds as follows. Given the first bearing measurement $Z^{(l)}(1)$ at node l, the relative position components $\{X(0), Y(0)\}$ of the target state vector are computed based on the procedure described in [103], i.e.,

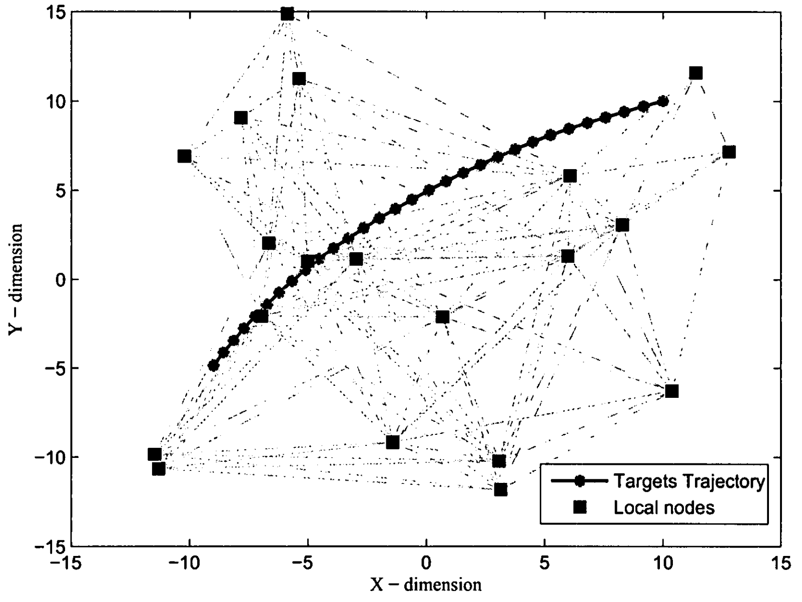
$$|X(0) - X^{(l)}| = \bar{r}^{(l)} \sin(Z^{(l)}(1)) \qquad \text{and} \qquad |Y(0) - Y^{(l)}| = \bar{r}^{(l)} \cos(Z^{(l)}(1)),$$

where $\{X^{(l)}$ and $Y^{(l)}\}$ are coordinates of node l assumed known. The range $r^{(l)}$ of the target from node l is initialized at random from other normal distributions, i.e., $r^{(l)} \sim \mathcal{N}(\bar{r}^{(l)}, \sigma_r^2)$. The velocity components are initialized using a similar procedure by selecting from a random distribution, i.e., $s \sim \mathcal{N}(\bar{s}, \sigma_s^2)$ and $c \sim \mathcal{N}(\bar{c}, \sigma_c^2)$, respectively. The velocity components of the target state vector is then initialized as $\dot{X}(0) = \bar{s}\sin(\bar{c})$ and $\dot{Y}(0) = \bar{s}\cos(\bar{c})$. The means $\bar{r}^{(l)}$, $\bar{s}$, and $\bar{c}$ along with their corresponding variances σ_r^2, σ_s^2, and σ_c^2 are assumed known. The initial error covariance matrix associated with $\mathbf{x}(0)$ is modeled as follows

$$\boldsymbol{P}(0) = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & 0 & 0 \\ \sigma_{yx}^2 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & \sigma_{\dot{x}}^2 & \sigma_{\dot{x}\dot{y}}^2 \\ 0 & 0 & \sigma_{\dot{y}\dot{x}}^2 & \sigma_{\dot{y}}^2 \end{bmatrix} \qquad (3.55)$$

where the constituent elements in $\boldsymbol{P}(0)$ are derived based on [103].

In the distributed implementations, the initialization step is performed at each node individually with the initial observation noise variance of $\sigma_\theta = 2.5^o$. Below four different scenarios are considered to evaluate the performance of the proposed distributed estimation framework.



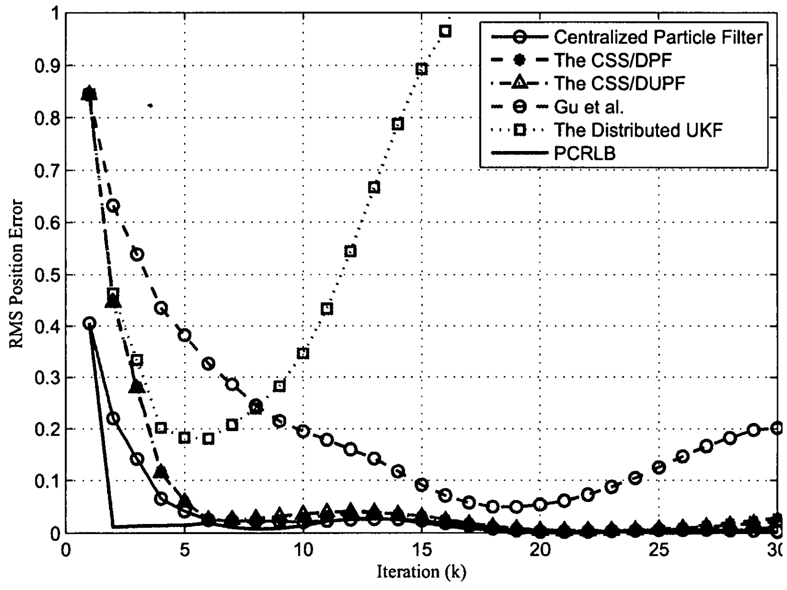
(a)

Figure 3.1: **Scenario 1**: Realization of the sensor placement along with the target's trajectory. The number of iterations required for achieving consensus in this network is $N_c(\boldsymbol{U}) = 5$.

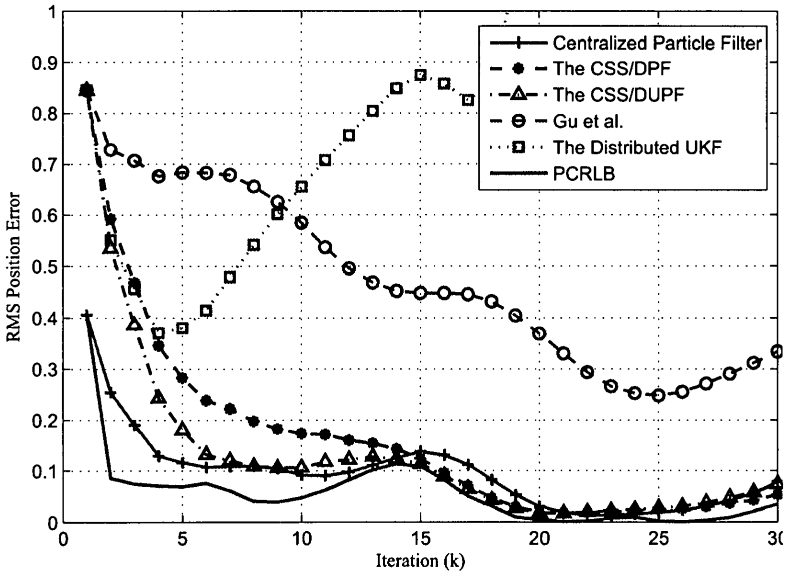
3.3.2 Scenario 1

To quantify the tracking performance of the proposed CSS/DPF and CSS/DUPF, five different estimation algorithms are considered: (i) Centralized scenario where one node has access to the observations of all other nodes. (ii) Distributed scenario using the CSS/DPF, (iii) Distributed scenario using the CSS/DUPF, (iv) Distributed unscented Kalman filter proposed in [7], referred to as distributed UKF, and; (v) Distributed particle filter proposed in [23], referred to as Gu et al. For comparison, we also plot the posterior Cramér Rao lower bound (PCRLB)-a lower bound on the performance of the optimal distributed estimators The PCRLB is computed based on a centralized recursive algorithm presented in [148]. The theory of the PCRLB is introduced in Chapter 5 where we present novel distributed algorithms to compute the PCRLB. The initialization parameters for the simulation run is obtained by following the filter initialization procedure described above with the standard deviations for the measurement and velocity models given by $\sigma_r = .7$, $\sigma_c = \pi/\sqrt{12}$, and $\sigma_s = .7$, and the mean values given by $\bar{c} = -110^\circ$ and $\bar{s} = 0.4$ meter. The mean value $\bar{r}^{(l)}$ of range is the noise corrupted true range between node l and the moving target. Resampling in the particle filtering was carried out if $N_{\text{eff}}(k) < N_s/3$. The number N_s of vector particles used at the fusion center in the centralized implementation is $10,000$, while the number of particles (N_{GCF} or N_{UPF}) used at each node in the distributed implementations is 1000. Fig. 3.1 shows one realization of the sensor placement along with the target trajectory.

Due to state-dependent noise variance, the signal to noise ratio (SNR) is time-varying and differs from one node to the other depending on the location of the target. Two different SNR cases (averaged across all nodes and time) are considered: (i) *High SNR,* where the SNRs at different nodes varies form 16dB to 29dB (Fig. 3.2(a)), (ii) *Low SNR,* where the SNRs ranges from 5dB to 17dB across the network (Fig. 3.2(b)). In Figs. 3.2(a) and (b) the RMS error computed based on Eq. (3.54) corresponding to the CSS/DPF and CSS/DUPF (schemes (ii) to (iii)) are compared versus that of the centralized particle filter (scheme (i)), schemes (iv) to (v), and the dPCRLB lower bound [51]. In Figs. 3.2(a) and (b) the consensus step is allowed to converge between two iteration of the localized filters. Each node initializes its local filter separately, therefore, the initial state estimates $\hat{\mathbf{x}}^{(l)}(0)$ are potentially different. In the centralized particle filter implementation, only one node (fusion centre) runs the particle filter based on the initial state estimate of that node. It is observed from Figs. 3.2(a) and (b) that the performance of the CSS/DPF and the CSS/DUPF are fairly close to each other and that of the centralized particle filter and approaches the PCRLB. Both CSS/DPF and CSS/DUPF outperform the distributed particle filter implementation proposed by Gu et al. (scheme (v)). The distributed UKF implementation (scheme (iv)) totally loses the track and eventually diverges.



(a)



(b)

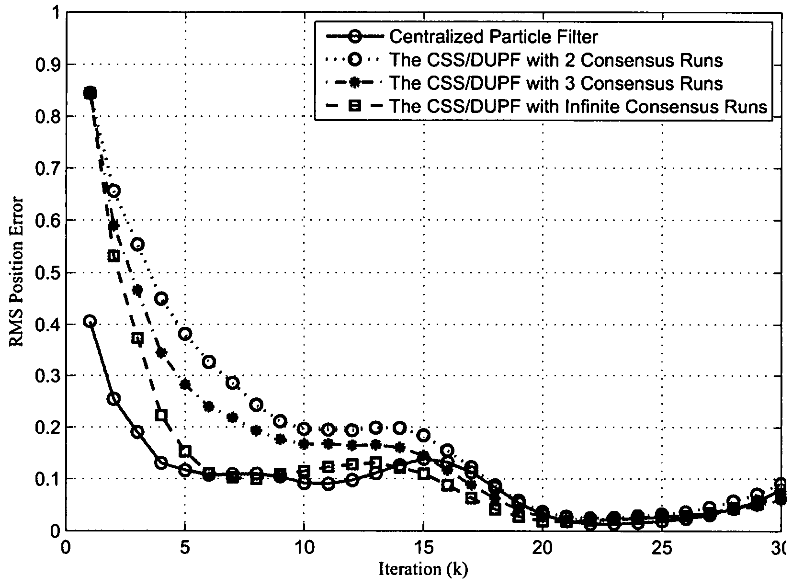
Figure 3.2: **Scenario 1**: Comparison between the centralized particle filter, the CSS/DPF, the CSS/DPF, distributed UKF [7], Gu *et al.* [23], and the PCRLB: (a) High SNR, and; (b) Low SNR.

3.3.3 Scenario 2

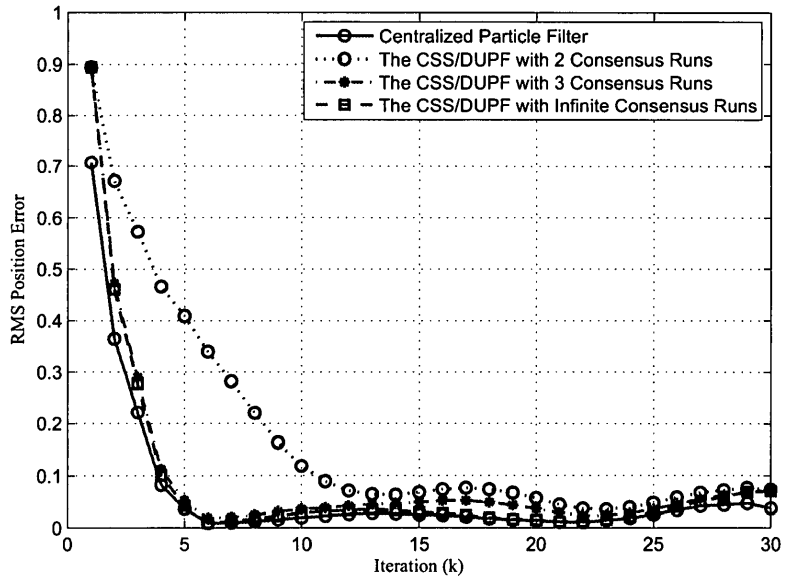
In the second scenario, the performance of the proposed CSS/DUPF using a limited number of consensus iterations is compared with that of the centralized particle filter. The purpose of this set of simulations is to determine the impact of a limited number of consensus iterations on the proposed CSS/DUPF. The consensus algorithms are stopped abruptly after a fixed number of iterations without allowing them to converge. The three remaining distributed implementations diverge if the consensus algorithm is not allowed to converge and are not plotted here since their RMS errors go out of scale. The results are shown in Fig. 3.3 where Fig. 3.3(a) shows the RMS error plots for the CSS/DUPF implemented in the network shown in Fig. 3.1 where the number of consensus iterations kept at 2 and 3. It is observed that the CSS/DUPF with only 2 consensus runs catches up with the centralized particle filter. Fig. 3.3(b) depicts the RMS plots for another network topology and target track where the number of iterations required for achieving the consensus in this network is twice that of the network shown in Fig. 3.1. The implemented CSS/DUPF runs a reduced number of consensus iterations. Results for 1, 2, and 3 consensus iterations are shown. The results confirms that the RMS error from the CSS/DUPF remains bounded and approaches that of the centralized particle filter.

3.3.4 Scenario 3

Fig. 3.4 shows the RMS error plots for joint bearing/range tracking problem. The bearing measurements are generated based on the description given in Scenario 1. The range measurements are corrupted by Gaussian noise with standard deviation 0.14m. The CSS/DUPF with one to three consensus iterations is compared with the centralized particle filter. It is observed that the CSS/DUPF with even one consensus iteration provides reasonable results. The performance of the CSS/DUPF with two and three consensus iterations are converging to the centralized plot.



(a)



(b)

Figure 3.3: **Scenario 2**: Comparison between the centralized particle filter and the CSS/DUPF with different number of consensus iteration: (a) Based on the network shown in Fig. 3.1, and; (b) Based on another network where the number of iterations required for achieving the consensus is twice.

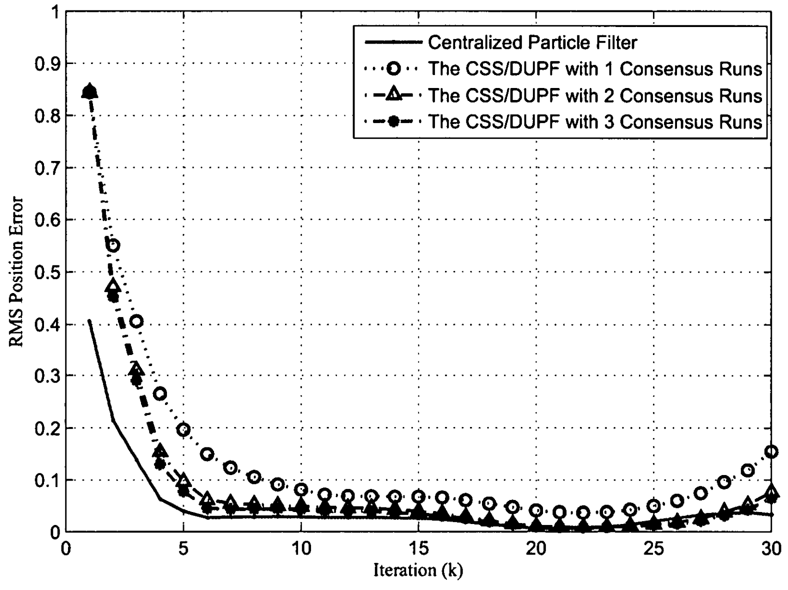
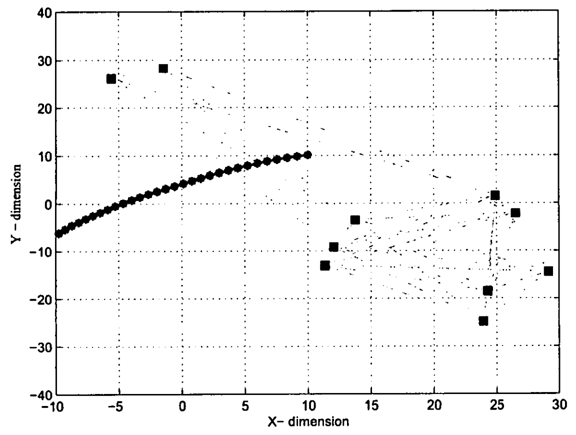


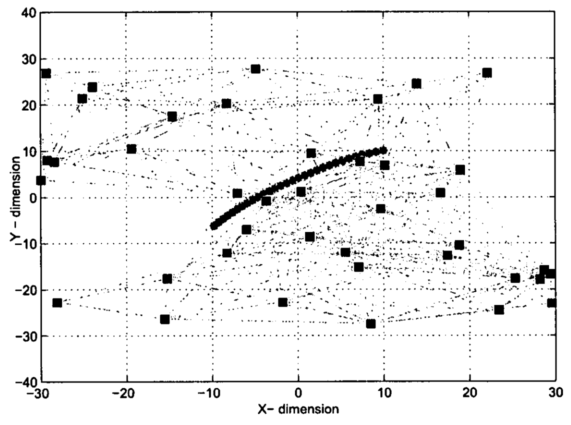
Figure 3.4: **Scenario 3**: Comparison of the centralized particle filter and the CSS/DUPF for joint bearing/range tracking problem.

3.3.5 Scenario 4

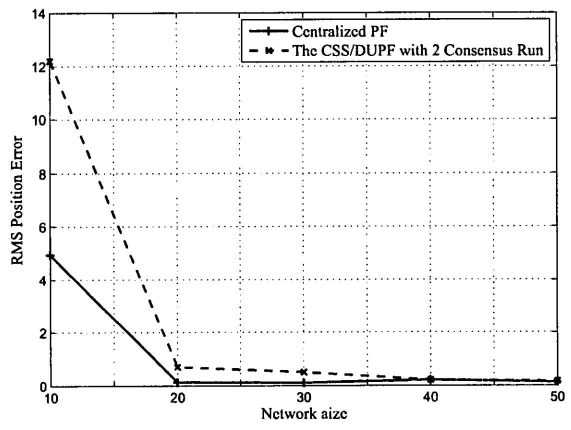
The purpose of this scenario is to evaluate the performance of the CSS/DUPF as a function of the number of active nodes in the network. In this scenario, an AN/SN is considered comprising of ($10 \leq N \leq 50$) observation nodes where sensors are distributed randomly in a (60×60) m^2 square region. Other parameters for this simulation are the same as in Scenario 1. Two examples of the sensor placements are shown in Fig. 3.5(a) and (b) where Fig. 3.5(a) shows the realization of the sensor placement along with the target's trajectory for $N = 10$. Most of the time the target is outside the surveillance region of the local nodes. Because of the state-dependent nature of the observation noises, large errors are expected in this scenario even for the centralized implementation. In other words, $N = 10$ observation nodes are not enough to track the target in this scenario. Fig. 3.5(b) shows the realization of the sensor placement along



(a)



(b)



(c)

Figure 3.5: **Scenario 4**: (a) Realization of the sensor placement along with the target's trajectory for $N = 10$. (b) Realization of the sensor placement along with the target's trajectory for $N = 40$. (c) RMS tracking performance at iteration $k = 20$ for varying network sizes and for the centralized filter, the CSS/DUPF with two consensus runs and the CSS/DUPF with three consensus runs.

with the target's trajectory for $N = 40$. It is observed that in contrary to Fig. 3.5(a), the sensor nodes collectively have a better coverage in this case. Fig. 3.5(c) shows the RMS error plots for different numbers of nodes ($10 \leq N \leq 50$) at iteration $k = 20$ for the centralized particle filter and the CSS/DUPF with only two iteration for each consensus run. It is observed that when the number of nodes is $N = 20$ and higher, the performance of the CSS/DUPF with a limited number of consensus iterations catches up with its centralized counterpart. Compared to the previous scenarios (Scenario 1 and 2), the surveillance region considered here is relatively larger which makes the tracking more challenging and increases the corresponding error.

3.4 The UCD/DPF Implementation

The unscented, consensus-based, distributed implementation of the particle filter (UCD/DPF) couples the unscented Kalman filter (UKF) [44] with the particle filter such that the UKF estimates the Gaussian approximation of the proposal distribution which is then used to generate local particles. The UCD/DPF involves the following four steps:

1. Individual sensor nodes run localized, unscented particle filters to approximate their local posterior distributions.

2. A pre-specified set of local statistics of the state variables are computed at each node from the local posterior distributions.

3. At each node, a consensus algorithm fuses local statistics computed in Step 2 into global statistics.

4. Once the global statistics are available, an unscented Kalman filter (UKF) propagates the global statistics into the proposal distributions to be used during the next iteration of the UCD/DPF.

In terms of contributions, the UCD/DPF makes two important improvements to the existing DPF framework

1. Unlike existing distributed implementations [24, 27] of the particle filter, the UCD/DPF uses all available global observations including the most recent ones in deriving the proposal distribution based on the distributed UKF. In other words, the UCD/DPF computes the local proposal density based on both the global statistics as well as the local observations.

2. Computation of the global estimates from local estimates during the consensus step is based on an optimal fusion rule which compensates for the problem of common information between the local state estimates.

Improvement 2 replaces the commonly used local averaging approach and, along with improvement 1, enhances the performance of the UCD/DPF. Further, the UCD/DPF paves the way for incorporating future developments in consensus-based distributed Kalman filters to the distributed particle filtering framework. Below, the main steps followed at node l, for $(1 \leq l \leq N)$, of the UCD/DPF are outlined. The filter is assumed to be in steady state and at iteration $k-1$, when all nodes are assumed to have reached a consensus with values $\hat{\mathbf{x}}^{(l,\text{Fused})}(k-1)$ and $\boldsymbol{P}^{(l,\text{Fused})}(k-1)$. A new measurement $\mathbf{z}^{(l)}(k)$ is now available at each local node.

Step 1. This step is similar to Step 2 of the CSS/DUPF with one difference, i.e., node l, for $(1 \leq l \leq N)$, generates the Sigma points $\{\boldsymbol{\chi}_i^{(l)}(k-1)\}_{i=0}^{2n_x}$ based on Eq. (3.31) using $\hat{\mathbf{x}}^{(l,\text{Fused})}(k-1)$ and $\boldsymbol{P}^{(l,\text{Fused})}(k-1)$ instead of $\bar{\mathbf{x}}^{(l)}(k-1)$ and $\bar{\boldsymbol{P}}^{(l)}(k-1)$.

Step 2. This step is similar to Step 3 of the CSS/DUPF where node l, for $(1 \leq l \leq N)$, computes the statistics of its local proposal distribution $(\hat{\mathbf{x}}_{\text{UKF}}^{(l)}(k), \boldsymbol{P}_{\text{UKF}}^{(l)}(k))$ using Eqs. (3.39)-(3.40).

Step 3. For $(1 \le l \le N)$, node l generates N_s random particles $\mathbb{X}_i^{(l)}(k)$ from its proposal distribution defined as follows

$$\mathbb{X}_i^{(l)}(k) \sim \mathcal{N}\Big(\hat{\mathbf{x}}_{\text{UKF}}^{(l)}(k), \boldsymbol{P}_{\text{UKF}}^{(l)}(k)\Big). \tag{3.56}$$

Step 4. Node l, for $(1 \le l \le N)$, computes the corespondent weight $W_i^{(l)}(k)$ of its particles as follows

$$W_i^{(l)}(k) \propto W_i^{(l)}(k-1)\frac{P\big(\mathbf{z}^{(l)}(k)|\mathbb{X}_i^{(l)}(k)\big)P\big(\mathbb{X}_i^{(l)}(k)|\mathbb{X}_i^{(l)}(k-1)\big)}{\mathcal{N}\big(\mathbb{X}_i^{(l)}(k);\hat{\mathbf{x}}_{\text{UKF}}^{(l)}(k),\boldsymbol{P}_{\text{UKF}}^{(l)}(k)\big)}, \tag{3.57}$$

After this step, node l has a set of particles and their associated weights that approximate the local filtering distribution $P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k), \mathbf{z}(1: k-1))$.

Step 5. Based on Eqs. (3.17)-(3.18), node l computes the MMSE estimate $\bar{\mathbf{x}}^{(l)}(k)$ and its corresponding error covariance $\bar{\boldsymbol{P}}^{(l)}(k)$ (local statistics) of the state variables.

Step 6. The final step of the UCD/DPF algorithm is the consensus step used to compute a consistent set of values for the global statistics $\hat{\mathbf{x}}^{(l,\text{Fused})}(k)$ and $\boldsymbol{P}^{(l,\text{Fused})}(k)$ at time k. The UCD/DPF uses the following fusion rules (instead of Eqs. (3.42)-(3.43) used in the CSS/DUPF)

$$\begin{aligned}
\big[\boldsymbol{P}^{(l,\text{Fused})}(k)\big]^{-1} &= \big[\boldsymbol{P}_{\text{UKF}}^{(l)}(k|k-1)\big]^{-1} + \underbrace{\sum_{j=1}^{N}\big[\bar{\boldsymbol{P}}^{(j)}(k)\big]^{-1} - \big[\boldsymbol{P}_{\text{UKF}}^{(j)}(k|k-1)\big]^{-1}}_{\boldsymbol{P}_c(\infty)} && (3.58)\\
\hat{\mathbf{x}}^{(l,\text{Fused})}(k) &= \big[\boldsymbol{P}^{(l,\text{Fused})}(k)\big]^{-1}\Big[\big[\boldsymbol{P}_{\text{UKF}}^{(l)}(k|k-1)\big]^{-1}\mathbf{x}_{\text{UKF}}^{(l)}(k|k-1)\\
&+ \underbrace{\sum_{j=1}^{N}\big[\bar{\boldsymbol{P}}^{(j)}(k)\big]^{-1}\bar{\mathbf{x}}^{(j)}(k) - \big[\boldsymbol{P}_{\text{UKF}}^{(j)}(k|k-1)\big]^{-1}\mathbf{x}_{\text{UKF}}^{(j)}(k|k-1)}_{\mathbf{x}_c(\infty)}\Big], && (3.59)
\end{aligned}$$

where $\{\mathbf{x}_c(\infty)$ and $\boldsymbol{P}_c(\infty)\}$ are obtained using Eqs. (3.44)-(3.47).

In conclusion, the UCD/DPF implementation of the particle filter belongs to the DPF via state estimation fusion category (Section 2.5) and addresses the first four drawbacks of the CSS/DPF listed right before Section 3.4. Though the UCD/DPF is more generally applicable than the CSS/DPF, it has the following drawbacks:

1. The UCD/DPF approximates the global posterior density with a Gaussian distribution and computes its statistics via consensus algorithms based on a set of optimal nonlinear fusion rules.

2. Similar to the CSS/DPF, the UCD/DPF assumes that the consensus is reached between two successive observations. Such an assumption is only reasonable in applications where communication is relatively inexpensive as compared to sensing, e.g., in rendezvous control or coordination of mobile sensors.

In Chapter 4, I develop the CF/DPF framework which does not restrict the global posterior density to a Gaussian distribution and removes the time constraint on the consensus convergence.

Finally, I note that the three proposed DPF implementations in this chapter suffer from one common drawback, i.e., they require the consensus to be reached between two successive observations. The performance of these methods degrades if consensus is not reached within two consecutive iterations of the local particle filters. Chapter 3 extends the distributed estimation framework to unreliable networks with intermittent connectivity. Intermittent network connectivity results in information loss, significant delays in the convergence of the consensus algorithm, and loss in synchronization between the localized filters. In the next chapter, I study a generic framework for distributed estimation in intermittently connected networks from the consensus-convergence perspective where the fundamental question is: *How can loss of synchronization between the localized filters and the fusion step can be adequately resolved to compensate for delays in the convergence of the consensus algorithms?*

3.4.1 Simulation Results for the UCD/DPF

As stated in Section 3.4, the UCD/DPF can be considered as a generalized version of the CSS/DUPF for applications other than BOT and joint bearing/range tracking where the CSS/DPF and CSS/DUPF are not applicable. Although the UCD/DPF is more general than the CSS/DUPF, but due to the absence of a sufficient statistic based step for computing the global likelihood in the UCD/DPF, we expect the performance of the CSS/DUPF to be superior in BOT and joint bearing/range tracking scenarios. Therefore, the UCD/DPF is evaluated separately for a tracking scenario where the CSS/DUPF is not applicable.

In this section, the range-only tracking application is considered to quantify the performance of the proposed UCD/UPF. Similar to the previous simulations, a single CCT model (Eq. (3.50)) with known statistics of the process noise $\boldsymbol{\xi}(k)$ is considered. An AN/SN with $N = 20$ nodes with random geometric graph model is considered where sensors are distributed randomly in a (15×15) m^2 square region. The observations are now range-only measurements given by

$$Z^{(l)}(k) = \sqrt{\big(X(k) - X^{(l)}(k)\big)^2 + \big(Y(k) - Y^{(l)}(k)\big)^2} + \zeta^{(l)}(k), \tag{3.60}$$

where $\{X^{(l)}(k), Y^{(l)}(k)\}$ are the coordinates of node l. Two scenario are considered in this section to evaluate performance of the UCD/DPF. The first scenario considers a constant value for the variance of the observation noise across the network while in the second scenario the variance of the observation noise at node l is state dependent as follows

$$\sigma^2_{\zeta^{(l)}}(k) = 0.001 r^{(l)^2}(k) + 0.01 r^{(l)}(k). \tag{3.61}$$

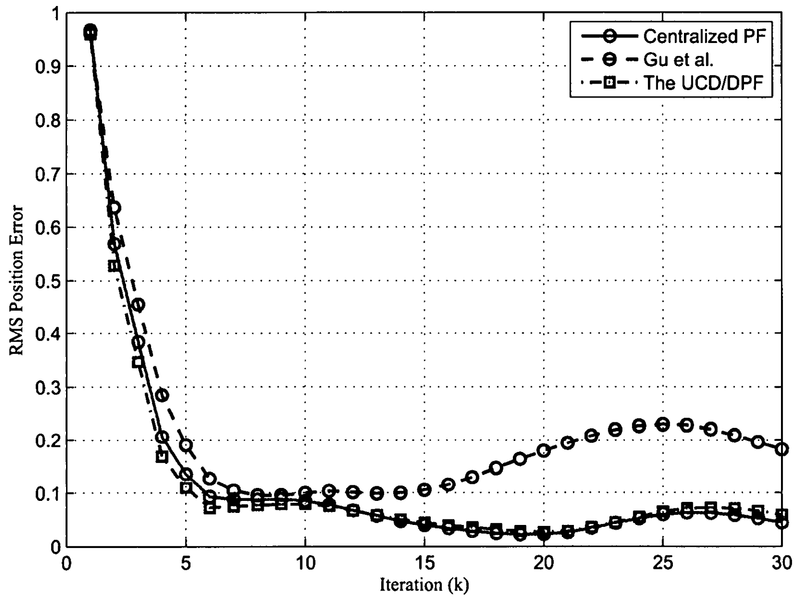
The target starts its track from coordinates $\{10, 10\}$ meters. The initial course is set at $-110°$ with the standard deviation of the process noise $\sigma_{\boldsymbol{\xi}(k)} = 1.6 \times 10^{-2}$. The initialization is performed at each node by selecting an initial location $\hat{\mathbf{x}}^{(l)}(0)$, for $1 \leq l \leq N$, from the following initial

Gaussian distribution $\mathcal{N}(\mathbf{x}(0)+0.5, \sigma^2_{\xi(k)})$. Other parameters are the same as the ones used in Scenario 1 above.

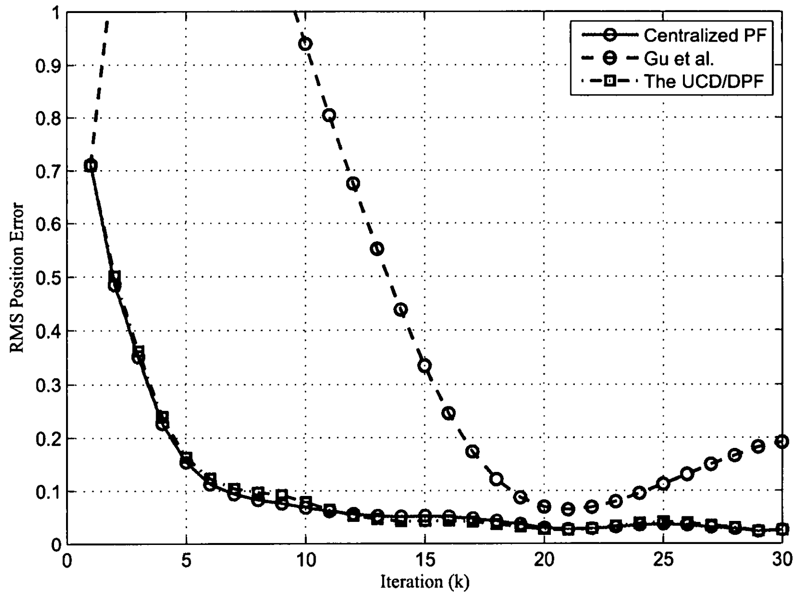
To quantify the tracking performance of the UCD/DPF three schemes are considered: (i) Centralized scenario where each node has access to the local observations of all other nodes. The performance of the centralized UPF is considered as the base performance; (ii) The proposed UCD/DPF implementation, and; (iii) Distributed particle filter proposed in [23], referred to as Gu et al. Fig. 3.6 shows the RMS error plots corresponding to schemes (i)-(iii) for a range-only tracking application. Fig. 3.6(a) shows the result for the constant high SNR scenario. Fig. 3.6(b) shows the RMS error plots for the variable high SNR scenario. It is observed that the performance of the proposed UCD/DPF remains close to its centralized counterpart in both scenarios. However, while the centralized and UCD/DPF implementations show low RMS errors, the other distributed implementation shows a significant increase in error.

3.5 Summary

In this chapter, I proposed three consensus-based, distributed implementations of the particle filters. First, a constraint sufficient statistic based distributed implementation of the particle filter (CSS/DPF) is proposed for bearing-only and joint bearing/range tracking applications where I exploit the property that the global sufficient statistics (GSS) attributed to the global likelihood function can be expressed as a summation of the local sufficient statistics (LSS) under certain constrains. I further derived explicit expressions for LSSs and their corresponding GSSs for 2D and 3D bearing-only tracking and 2D joint bearing and range tracking. The CSS/DPF implementation is a two stage algorithm based on first computing the GSS from the means of the LSS via consensus algorithms, and then updating the local particle filters using the modified GSS. The communication overhead of the CSS/DPF is reduced significantly in comparison with the



(a)



(b)

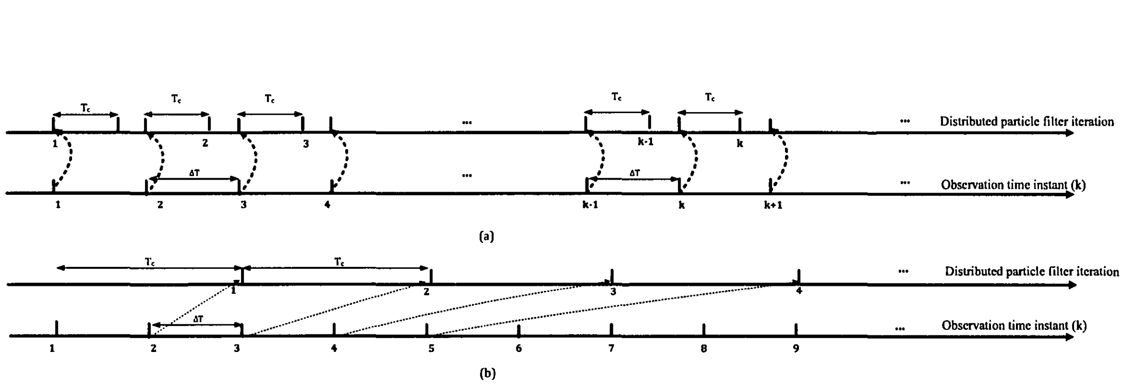
Figure 3.6: Comparison between the centralized particle filter, the UCD/DPF, and Gu *et al.* [23]: (a) Constant SNR, and; (b) High SNR but varying from a node to another.

other state-of-the-art distributed implementation of the particle filter. Second, the CSS/DUPF is proposed which improves the CSS/DPF by introducing a proposal distribution other than the transitional density which incorporates the global observations and therefore a is closer approximation of the optimal proposal distribution. Finally, consensus-based distributed implementation of the unscented particle filter (CD/UPF) is introduced which extends consensus-based distributed Kalman filtering framework to nonlinear systems. The CSS/DPF has the lowest computational complexity in comparison with other distributed implementations of the particle filter. Numerical simulations illustrate the superiority of the CSS/DUPF over other sufficient statistics based distributed particle filters. The performance of the CSS/DUPF catches up with that of the centralized particle filter even with a limited number of iterations per consensus run.

4 Distributed Particle Filter with Intermittent/Irregular Consensus Convergence

In Chapter 3, I proposed three full-order consensus-based distributed implementations of the particle filter: the CSS/DPF (Section 3.1), the CSS/DUPF (Section 3.2), and the UCD/DPF (Section 3.4). All of these proposed approaches have one common limitation, i.e., the requirement for each node to wait until consensus is reached before running the next iteration of the localized particle filters. To incorporate observations without delay, the consensus algorithm should converge between two consecutive observations. Such an assumption is reasonable in applications where communication as compared to sensing is relatively fast to allow for consensus convergence, e.g., in rendezvous control or coordination of mobile sensors. Fig. 4.1 considers an alternative scenario where the consensus convergence takes twice as long as the duration between two successive observations (ΔT). In such cases, the consensus algorithm continues to lag behind localized filters incorporating the local observations such that the global estimate for current particle filter implementation is delayed. Referred to as intermittent network connectivity [123, 124], this issue has been investigated broadly in the context of linear systems based on Kalman filter [123, 124] and have not yet been explored for non-linear systems.

In this chapter, I propose a multi-rate consensus/fusion based framework for distributed im-



T_c
ΔT
Distributed particle filter iteration
Observation time instant (k)
1
2
3
4
k-1
k
k+1
5
6
7
8
9
(a)
(b)

Figure 4.1: (a) Situations where CSS/DPF and UCD/DPF are applicable, i.e., consensus converges within the duration ΔT of two consecutive observations. (b) A scenario where the consensus convergence T_c is greater than ΔT. The lag between the global estimates and the local estimates grows exponentially.

plementation of the particle filter (CF/DPF)[6] for nonlinear systems. The CF/DPF offers two distinct advantages over its counterparts. First, the CF/DPF framework is suitable for nonlinear systems with intermittent network connectivity and consensus can not be reached between two consecutive observations. Second, the CF/DPF is not limited to the Gaussian approximation for the global posterior density. Below, I summarize the key contributions of the chapter.

1. Fusion filter: In addition to the localized particle filters, referred to as the local filters, the CF/DPF introduces separate consensus-based filters, referred to as the fusion filters, to derive the global posterior distribution by consistently fusing local filtering densities in a distributed fashion. The localized implementation of the particle filter and the fusion filter used to achieve consensus are run in parallel, possibly at different rates. Achieving consensus between two successive iterations of the local filters is, therefore, no longer a requirement. The CF/DPF compensates for the common past information between local estimates based on an optimal non-linear Bayesian fusion rule [127]. The fusion concept used in the CF/DPF is similar to [27] and [42], where separate

[6]The conventional particle filter has been chosen in developing the CF/DPF as a proof of concept. The proposed framework can be generalized to other variants of the particle filter such as the marginalized particle filter [125], the approximate condition mean particle filter [126] and the unscented particle filter [44] with some modifications.

channel filters (one for each communication link) are deployed to consistently fuse local estimates. In the CF/DPF, the number of fusion filters are limited to one per processing node, a considerable saving over [27] and [42].

Fig. 4.2 compares the proposed CF/DPF framework with the channel filter framework and centralized estimation. In the centralized estimation (Fig. 4.2(a)), all the nodes forward their raw observations (either directly or via help of other nodes) to the FC where the state vector is estimated. In the channel filter framework (Fig. 4.2(b)), one channel filter is associated with each communication link to fuse the local estimates of two neighbouring nodes and finally compute the global estimate. These filters are in addition to the localized filters run at the nodes. Note that, the channel filter approach can only be implemented for a tree-connect network topology [27] as shown in Fig. 4.2(b) and can not be extended to any arbitrary network, for example the one shown in Fig. 4.2(a). In the CF/DPF (Fig. 4.2(c)) each node only implements one additional fusion filter per node irrespective of the neighbouring connections thus reducing the number of fusion filters compared to [27] and [42]. Further, the CF/DPF is applicable to any network configuration.

2. Modified Fusion filters: In the CF/DPF, the fusion filters can run at a rate different form that of the local filters. I further investigate this multi-rate nature of the proposed framework, recognize three different scenarios, and describe how the CF/DPF handles each of them. For the worse-case scenario with the fusion filters lagging the local filters exponentially, I derive a modified-fusion filter algorithm that limits the lag to an affordable delay.

Table 4.1 provides a comparison of the CF/DPF with the approaches discussed in Chapter 3. The CF/DPF belongs to the state estimation fusion category and offers two advantages over its counterparts. The CF/DPF does not impose any restriction on the form of the global likelihood or global posterior distribution and it is resilient to the intermittence in the connectivity of the network.

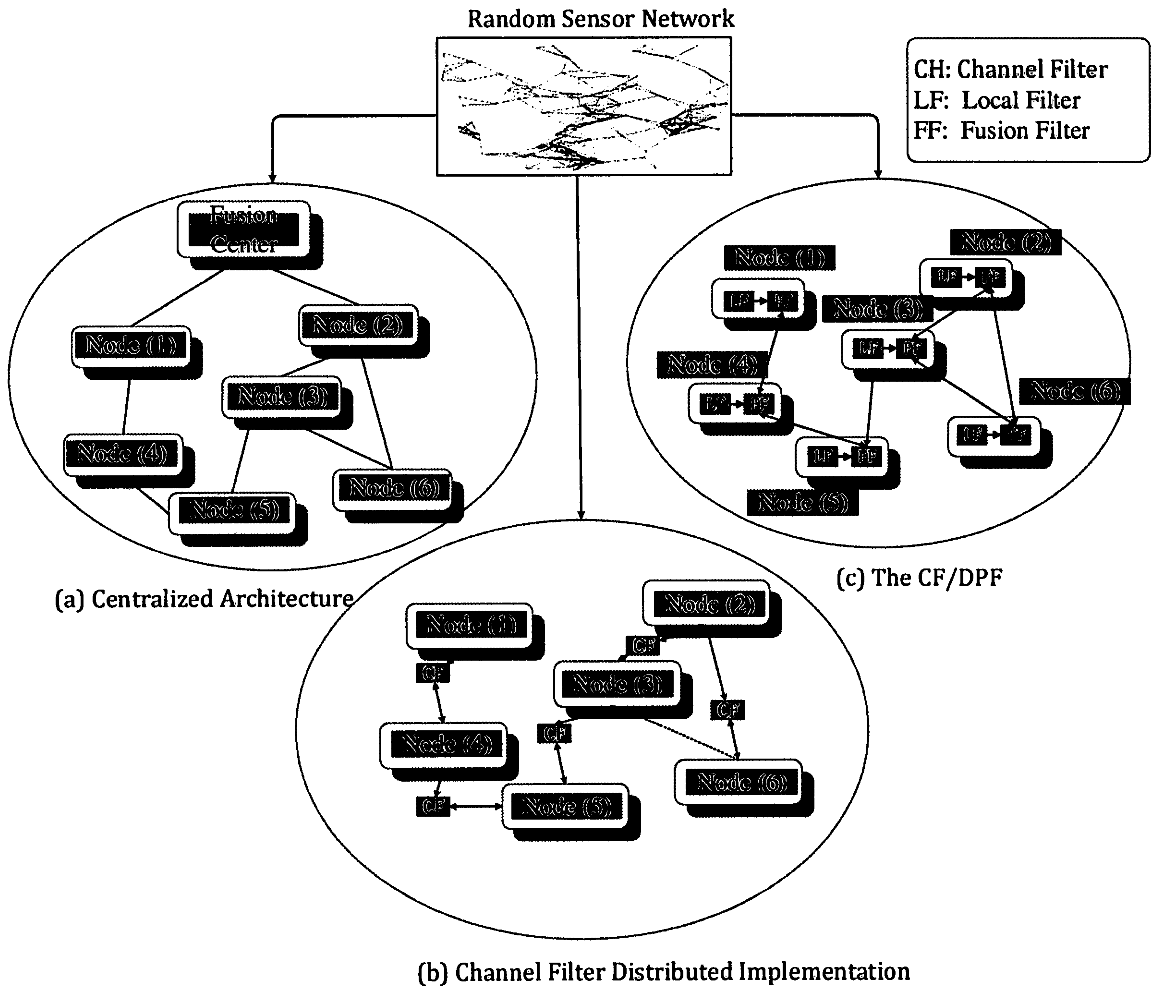


Figure 4.2: (a) Centralized implementation where all nodes communicate their local estimates to the fusion center. (b) Distributed implementation using channel filters where a separate filter is required for each communication link. (c) The proposed CF/DPF implementation where sensor nodes connect through their fusion filters (one fusion filter per node). In terms of the number of extra filters, the CF/DPF falls between the centralized and channel filters.

Table 4.1: Comparison of different full-order DPF implementations.

Characteristics	**CSS/DPF**	**UCD/DPF**	**CSS/DUPF**	**CF/DPF**
1. Likelihood/Observation fusion	×	×		
2. State estimation fusion		×	×	×
3. Gaussian approximation for the global likelihood	×		×	
4. Gaussian approximation for the global posterior		×		
5. Requires consensus convergence	×	×	×	
6. Application specific	×	×		
7. Restrict the proposal to the transitional distribution	×			
8. No restriction on the form of likelihood/posterior				×
9. Resilience to intermittent connectivity				×
10. Recovery from loss of information		×	×	×
11. Communication complexity	*low*	*midium*	*high*	*high*

The chapter is organized as follows. The proposed CF/DPF algorithm and the fusion filter are described in Section 4.1. The modified fusion filter is presented in Section 4.2. Section 4.3 illustrates the effectiveness of the proposed framework in tracking applications through Monte Carlo simulations. Finally, Section 4.4 concludes the chapter.

4.1 The CF/DPF Implementation

As shown in Fig. 4.2(c), the CF/DPF implementation runs two localized particle filters at each sensor node. The first filter, referred to as the local filter, comes from the distributed implementation of the particle filter described in Section 4.1.1 and is based only on the local observations $\mathbf{z}^{(l)}(1\!:\!k)$. The CF/DPF introduces a second particle filter at each node, referred to as the fusion filter, which estimates the global posterior distribution $P(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k))$ from the local filtering distributions $P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k))$ and local prediction distributions $P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k\text{–}1))$ as described in Section 4.1.2.

4.1.1 Distributed Configuration and Local Filters

Recall that the distributed estimation framework as presented in Section 2.5 (Eqs. (2.126)-(2.127)) is given by

$$\begin{aligned}\mathbf{x}(k) &= \boldsymbol{f}(\mathbf{x}(k-1)) + \boldsymbol{\xi}(k) &(4.1)\\ \mathbf{z}^{(l)}(k) &= \boldsymbol{g}^{(l)}(\mathbf{x}(k)) + \boldsymbol{\zeta}^{(l)}(k), &(4.2)\end{aligned}$$

for sensor nodes $(1 \leq l \leq N)$. In the CF/DPF, the entire state vector $\mathbf{x}(k)$ is estimated by running localized particle filters at each node. These filters, referred to as the local filters, come from the distributed implementation of the particle filter and are based only on local observations $\mathbf{z}^{(l)}(1 : k)$. In addition to updating the particles and their associated weights, the local filter at node l provides estimates of the local prediction distribution $P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1 : k-1))$ from the particles as explained below.

Computation and Sampling of the Prediction Distribution: From the Chapman-Kolmogorov equation (Eq. (2.13)), a sample based approximation of the prediction density $P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1 :$

$k-1)$) is expressed as

$$P\left(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k-1)\right) = \sum_{i=1}^{N_s} W_i^{(l,\mathrm{LF})}(k-1) P\left(\mathbf{x}(k)|\mathbb{X}_i^{(l,\mathrm{LF})}(k-1)\right), \tag{4.3}$$

which is a continuous mixture. To generate random particles from such a mixture density, a new sample $\mathbb{X}_i^{(l,\mathrm{LF})}(k|k-1)$ is generated from its corresponding mixture $P(\mathbf{x}(k)|\mathbb{X}_i^{(l,\mathrm{LF})}(k-1))$ in Eq. (4.3). Its weight $W_i^{(l,\mathrm{LF})}(k-1)$ is the same as the corresponding weight for $\mathbb{X}_i^{(l,\mathrm{LF})}(k-1)$. The prediction density is given by

$$P\left(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k-1)\right) = \sum_{i=1}^{N_s} W_i^{(l,\mathrm{LF})}(k-1)\delta\left(\mathbf{x}(k) - \mathbb{X}_i^{(l,\mathrm{LF})}(k|k-1)\right).$$

Once the random samples are generated, the mean square error estimates (MSE) of the parameters can be computed.

4.1.2 Fusion Filter

The CF/DPF introduces a second particle filter at each node, referred to as the fusion filter, which computes an estimate of the global posterior distribution $P(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k))$. Being a particle filter itself, implementation of the fusion filter requires the proposal distribution and the weight update equation. Theorem 5 expresses the global posterior distribution in terms of the local filtering densities, which is used for updating the weights of the fusion filter. The selection of the proposal distribution will be explained later in Section 4.1.5. Each node, for $(1 \le l \le N)$, propagates forward in time two sets of particles: $\{\mathbb{X}_i^{(l,\mathrm{LF})}(k), W_i^{(l,\mathrm{LF})}(k)\}_{i=1}^{N_s}$ associated with the local filters and $\{\mathbb{X}_i^{(l,\mathrm{FF})}(k), W_i^{(l,\mathrm{FF})}(k)\}_{i=1}^{N_{\mathrm{FF}}}$ associated with the fusion filter.

Theorem 5. *Assuming that the observations conditioned on the state variables and made at node l are independent of those made at node j, $(j \neq l)$, the global posterior distribution for an*

N-sensor network is

$$P\Big(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k)\Big) \propto \frac{\prod_{l=1}^{N} P\Big(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k)\Big)}{\prod_{l=1}^{N} P\Big(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k-1)\Big)} \times P\Big(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k-1)\Big), \tag{4.4}$$

where the last term may be factorized as follows

$$P\Big(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k-1)\Big) = P\Big(\mathbf{x}(k)|\mathbf{x}(k-1)\Big)P\Big(\mathbf{x}(0\!:\!k-1)|\mathbf{z}(1\!:\!k-1)\Big). \tag{4.5}$$

The proof of Theorem 5 is included in Appendix B.1. Note that the optimal distributed protocol defined in Eq. (4.4) consists of three terms: (i) Product of the local filtering distribution $\prod_{i=1}^{N} P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k))$ which depends on local observations; (ii) Product of local prediction densities $\prod_{i=1}^{N} P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k-1))$, which is again only based on the local observations and represent the common information between neighboring nodes, and; (iii) Global prediction density $P\left(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k-1)\right)$ based on Eq. (4.5). The fusion rule, therefore, requires consensus algorithms to be run for terms (i) and (ii). The proposed CF/DPF computes the two terms separately (as described later) by running two consensus algorithms at each iteration of the fusion filter. An alternative is to compute the ratio of two terms at each node and run one consensus algorithm for computing the ratio term. In the CF/DPF, I propose to estimate the numerator and denominator of Eq. (4.4) separately because maintaining the local filtering and prediction distributions is advantageous in networks with intermittent connectivity as it allows the CF/DPF to recover from loss of information due to delays in convergence.

4.1.3 Weight Update Equation

Assume that the local filters have reached steady state at iteration k, i.e., the local filter's computation is completed up to and including time iteration k where a particle filter based estimate of the local filtering distribution is available. The weight update equation for the fusion filter is given by

$$W_i^{(l,\mathrm{FF})}(k) = \frac{P\left(\mathbb{X}_i^{(l,\mathrm{FF})}(k)|\mathbf{z}(1:k)\right)}{q\left(\mathbb{X}_i^{(l,\mathrm{FF})}(k)|\mathbf{z}(1:k)\right)}. \tag{4.6}$$

Given particles $\mathbb{X}_i^{(l,\mathrm{FF})}(k-1)$, the values of the particles $\mathbb{X}_i^{(l,\mathrm{FF})}(k)$ at time instant k are updated by generating random particles from the proposal distribution $q(\mathbf{x}(0:k)|\mathbf{z}(1:k))$. As stated previously in Section 2.2.2, the proposal distribution is chosen such that it satisfies the following factorization

$$q\big(\mathbf{x}(0:k)|\mathbf{z}(1:k)\big) = q\big(\mathbf{x}(0:k-1)|\mathbf{z}(1:k-1)\big)q\big(\mathbf{x}(k)|\mathbf{x}(1:k-1),\mathbf{z}(1:k)\big), \tag{4.7}$$

then one can obtain particles $\mathbb{X}_i^{(l,\mathrm{FF})}(0:k) \sim q\big(\mathbf{x}(0:k)|\mathbf{z}(1:k)\big)$ by augmenting each of the existing samples $\mathbb{X}_i^{(l,\mathrm{FF})}(0:k-1) \sim q\big(\mathbf{x}(0:k-1)|\mathbf{z}(1:k-1)\big)$ with the new particles generated as follows

$$\text{Prediction Step:} \qquad \mathbb{X}_i^{(l,\mathrm{FF})}(k) \sim q\big(\mathbf{x}(k)|\mathbf{x}(0:k-1),\mathbf{z}(1:k)\big). \tag{4.8}$$

A filtered estimate of the state variables $P(\mathbf{x}(k)|\mathbf{z}(1:k))$ at each iteration is of interest, therefore, following [43] I approximate $q(\mathbf{x}(k)|\mathbf{x}(1:k-1),\mathbf{z}(1:k)) = q(\mathbf{x}(k)|\mathbf{x}(k-1),\mathbf{z}(k))$. The proposal density is then dependent only on $\mathbf{x}(k)$ and $\mathbf{z}(k)$. In such a scenario, one can discard the history of the particles $\mathbb{X}_i^{(l,\mathrm{FF})}(0:k-2)$ at previous iterations [43]. Substituting Eq. (4.5) in Eq. (4.4) and using the result together with Eq. (4.7) in Eq. (4.6), the weight update equation is given by

$$W_i^{(l,\mathrm{FF})}(k) \propto W_i^{(l,\mathrm{FF})}(k-1)\frac{\prod_{l=1}^{N} P\left(\mathbb{X}_i^{(l,\mathrm{FF})}(k)|\mathbf{z}^{(l)}(1:k)\right)}{\prod_{l=1}^{N} P\left(\mathbb{X}_i^{(l,\mathrm{FF})}(k)|\mathbf{z}^{(l)}(1:k-1)\right)}\;\frac{P\left(\mathbb{X}_i^{(l,\mathrm{FF})}(k)|\mathbb{X}_i^{(l,\mathrm{FF})}(k-1)\right)}{q\left(\mathbb{X}_i^{(l,\mathrm{FF})}(k)|\mathbb{X}_i^{(l,\mathrm{FF})}(k-1),\mathbf{z}(k)\right)}, \tag{4.9}$$

where

$$W_i^{(l,\mathrm{FF})}(k-1) = \frac{P\left(\mathbb{X}_i^{(l,\mathrm{FF})}(k-1)|\mathbf{z}(1:k-1)\right)}{q\left(\mathbb{X}_i^{(l,\mathrm{FF})}(k-1)|\mathbf{z}(1:k-1)\right)}. \tag{4.10}$$

Given the weights $W_i^{(l,\mathrm{FF})}(k-1)$ from the previous iteration, Eq. (4.9) requires all nodes to participate in the computation of the following two terms

$$\prod_{l=1}^{N} P\left(\mathbb{X}_i^{(l,\mathrm{FF})}(k)|\mathbf{z}^{(l)}(1:k)\right) \qquad \text{and} \qquad \prod_{l=1}^{N} P\left(\mathbb{X}_i^{(l,\mathrm{FF})}(k)|\mathbf{z}^{(l)}(1:k-1)\right). \tag{4.11}$$

The numerator of the second fraction in Eq. (4.9) requires the transitional distribution $P(\mathbf{x}(k)|\mathbf{x}(k-1))$, which is known from the state model. Its denominator requires the proposal distribution $q(\mathbf{x}(k)|\mathbf{x}(k-1), \mathbf{z}(k))$. Below, I show how two terms (Eq. (4.11)) and the proposal distribution are determined.

4.1.4 Distributed Computation of Product Densities

The two terms in (4.11) are not determined by transferring the whole particle vectors and their associated weights between the neighboring nodes due to an impractically large number of information transfers. A second issue lies due to representing the localized posteriors as a Dirac mixture in the particle filter. Two separate Dirac mixtures may not have the same support and their multiplication could possibly be zero. In order to tackle these problems, a transformation is required on the Dirac function particle representations by converting them to continuous distributions prior to communication and fusion. Gaussian distributions [4, 5, 7, 23, 24, 59], grid-based techniques [47], Gaussian Mixture Model (GMM) [17] and Parzen representations [27] are different parametric continuous distributions used in the context of the distributed particle filter implementations. The channel filter framework [27] fuses only two local distributions, therefore, the local probability density functions can be modeled [27] with such complex distributions. Incorporating these distributions in the CF/DPF framework is, however, not a trivial task because the CF/DPF computes the product of N local distributions. The use of a complex distribution like GMM is, therefore, computationally prohibitive.

In order to tackle this problem, I approximate the product terms in Eq. (4.9) with Gaussian distribution which results in local filtering and prediction densities to be normally distributed as

$$P\left(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k)\right) \propto \mathcal{N}\left(\mu^{(l)}(k), \boldsymbol{P}^{(l)}(k)\right) \text{ and } P\left(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k-1)\right) \propto \mathcal{N}\left(\boldsymbol{\nu}^{(l)}(k), \boldsymbol{R}^{(l)}(k)\right), \tag{4.12}$$

where $\boldsymbol{\mu}^{(l)}(k)$ and $\boldsymbol{P}^{(l)}(k)$ are, respectively, the mean and covariance of local particles at node l during the filtering step of iteration k. Similarly, $\boldsymbol{\nu}^{(l)}(k)$ and $\boldsymbol{R}^{(l)}(k)$ are, respectively, the mean and covariance of local particles at node l during the prediction step. It should be noted that I only approximate the product density for updating the weights with a Gaussian distribution and the global posterior distribution is not restricted to be Gaussian. The local statistics at node l are computed as

$$\begin{aligned} \boldsymbol{\mu}^{(l)}(k) &= \sum_{i=1}^{N_s} W_i^{(l,\mathrm{LF})}(k)\mathbb{X}_i^{(l,\mathrm{LF})}(k) \\ \text{and} \qquad \boldsymbol{P}^{(l)}(k) &= \sum_{i=1}^{N_s} W_i^{(l,\mathrm{LF})}(k)\left(\mathbb{X}_i^{(l,\mathrm{LF})}(k)-\boldsymbol{\mu}^{(l)}(k)\right)\left(\mathbb{X}_i^{(l,\mathrm{LF})}(k)-\boldsymbol{\mu}^{(l)}(k)\right)^T. \end{aligned} \tag{4.13}$$

Reference [129] shows that the product of N multivariate normal distributions is also normal, i.e.,

$$\prod_{l=1}^{N} P\left(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k)\right) \triangleq \prod_{l=1}^{N}\mathcal{N}\left(\boldsymbol{\mu}^{(l)}(k),\boldsymbol{P}^{(l)}(k)\right) = \frac{1}{C}\times\mathcal{N}(\boldsymbol{\mu}(k),\boldsymbol{P}(k)), \tag{4.14}$$

where C is a normalization term (Reference [129] includes the proof). Parameters $\boldsymbol{\mu}(k)$ and $\boldsymbol{P}(k)$ are given by

$$\boldsymbol{P}(k) = (\sum_{l=1}^{N}\underbrace{\left(\boldsymbol{P}^{(l)}(k)\right)^{-1}}_{\mathbf{X}_{c1}^{(l)}(0)})^{-1} \quad \text{and} \quad \boldsymbol{\mu}(k) = \boldsymbol{P}(k)\times\sum_{l=1}^{N}\underbrace{\left(\boldsymbol{P}^{(l)}(k)\right)^{-1}\boldsymbol{\mu}^{(l)}(k).}_{\mathbf{x}_{c2}^{(l)}(0)} \tag{4.15}$$

Similarly, the product of local prediction densities (Term (4.11)) is modeled with a Gaussian density

$\mathcal{N}(\mathbf{x}(k);\boldsymbol{v}(k),\boldsymbol{R}(k))$, where the parameters $\boldsymbol{v}(k)$ and $\boldsymbol{R}(k)$ are computed as follows

$$\boldsymbol{R}(k) = (\sum_{l=1}^{N}\underbrace{\left(\boldsymbol{R}^{(l)}(k)\right)^{-1}}_{\mathbf{X}_{c3}^{(l)}(0)})^{-1} \quad \text{and} \quad \boldsymbol{v}(k) = \boldsymbol{R}(k)\times\sum_{l=1}^{N}\underbrace{\left(\boldsymbol{R}^{(l)}(k)\right)^{-1}\boldsymbol{v}^{(l)}(k).}_{\mathbf{x}_{c4}^{(l)}(0)} \tag{4.16}$$

The parameters of the product distributions only involves average quantities and can be provided using average consensus algorithms as follows:

(i) Node l, $(1 \le l \le N)$, initializes its consensus states to $\mathbf{X}_{c1}^{(l)}(0) = (\boldsymbol{P}^{(l)}(k))^{-1}$, $\mathbf{x}_{c2}^{(l)}(0) = (\boldsymbol{P}^{(l)}(k))^{-1}\boldsymbol{\mu}^{(l)}(k)$, $\mathbf{X}_{c3}^{(l)}(0) = (\boldsymbol{R}^{(l)}(k))^{-1}$, and $\mathbf{x}_{c4}^{(l)}(0) = (\boldsymbol{R}^{(l)}(k))^{-1}\boldsymbol{v}^{(l)}(k)$, then Eq. (2.116) is used to reach consensus with $\mathbf{X}_{c1}^{(l)}(t)$ used instead of $X_c^{(l)}(t)$ in Eq. (2.116) for the first consensus run. Similarly, $\mathbf{x}_{c2}^{(l)}(t)$ is used instead of $X_c^{(l)}(t)$ for the second run and so on.

(ii) Once consensus is reached, parameters $\boldsymbol{\mu}^{(l)}(k)$ and $\boldsymbol{P}^{(l)}(k)$ are computed as follows

$$\boldsymbol{P}(k) = 1/N \times \lim_{t\to\infty}\left\{\left(\mathbf{X}_{c1}^{(l)}(t)\right)^{-1}\right\} \quad \text{and} \quad \boldsymbol{\mu}(k) = \lim_{t\to\infty}\left\{\left(\mathbf{X}_{c1}^{(l)}(t)\right)^{-1} \times \mathbf{x}_{c2}^{(l)}(t)\right\} \tag{4.17}$$

$$\boldsymbol{R}(k) = 1/N \times \lim_{t\to\infty}\left\{\left(\mathbf{X}_{c3}^{(l)}(t)\right)^{-1}\right\} \quad \text{and} \quad \boldsymbol{v}(k) = \lim_{t\to\infty}\left\{\left(\mathbf{X}_{c3}^{(l)}(t)\right)^{-1} \times \mathbf{x}_{c4}^{(l)}(t)\right\}. \tag{4.18}$$

Based on aforementioned approximation, the weight update equation of the fusion filter (Eq. (4.9)) is given by

$$W_i^{(l,\mathrm{FF})}(k) \propto W_i^{(l,\mathrm{FF})}(k-1)\frac{\mathcal{N}(\mathbb{X}_i^{(l,\mathrm{FF})}(k);\boldsymbol{\mu}(k),\boldsymbol{P}(k))P(\mathbb{X}_i^{(l,\mathrm{FF})}(k)|\mathbb{X}_i^{(l,\mathrm{FF})}(k-1))}{\mathcal{N}(\mathbb{X}_i^{(l,\mathrm{FF})}(k);\boldsymbol{v}(k),\boldsymbol{R}(k))q(\mathbb{X}_i^{(l,\mathrm{FF})}(k)|\mathbb{X}_i^{(l,\mathrm{FF})}(k-1),\mathbf{z}(k))}. \tag{4.19}$$

Eq. (4.19) requires the proposal distribution $q(\mathbf{x}(k)|\mathbf{x}(k-1),\mathbf{z}(k))$ which is introduced next.

4.1.5 Proposal Distribution

In this section, I describe three different proposal distributions which can be used in the CF/DPF.

4.1.5.1 SIR Fusion Filter

The most common strategy is to sample from the probabilistic model of the state evolution, i.e., to use transitional density $P(\mathbf{x}(k)|\mathbf{x}(k+1))$ as proposal distribution. The simplified weight update equation for the SIR fusion filter is obtained from Eq. (4.19) as follows

$$W_i^{(l,\mathrm{FF})}(k) \propto W_i^{(l,\mathrm{FF})}(k-1)\frac{\mathcal{N}(\mathbb{X}_i^{(l,\mathrm{FF})}(k);\boldsymbol{\mu}(k),\boldsymbol{P}(k))}{\mathcal{N}(\mathbb{X}_i^{(l,\mathrm{FF})}(k);\boldsymbol{v}(k),\boldsymbol{R}(k))}. \tag{4.20}$$

This SIR fusion filter fails if a new measurement appears in the tail of the transitional distribution or when the likelihood is too peaked in comparison with the transitional density.

4.1.5.2 Product Density as Proposal Distribution

We are free to choose any proposal distribution that appropriately considers the effect of new observations and is close to the global posterior distribution. The product of local filtering densities is a reasonable approximation of the global posterior density as such a good candidate for the proposal distribution, i.e.,

$$q(\mathbf{x}(k)|\mathbf{x}(k-1), \mathbf{z}(1:k)) \triangleq \prod_{l=1}^{N} P\left(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k)\right), \tag{4.21}$$

which implies that the fusion filter particles $\{\mathbb{X}_i^{(l,\mathrm{FF})}(k)\}_{i=1}^{N_s}$ are generated from $\mathcal{N}(\boldsymbol{\mu}(k), \boldsymbol{P}(k))$. In such a scenario, the weight update equation (Eq. (4.19)) simplifies to

$$W_i^{(l,\mathrm{FF})}(k) \propto W_i^{(l,\mathrm{FF})}(k-1)\frac{P(\mathbb{X}_i^{(l,\mathrm{FF})}(k)|\mathbb{X}_i^{(l,\mathrm{FF})}(k-1))}{\mathcal{N}(\mathbb{X}_i^{(l,\mathrm{FF})}(k); \boldsymbol{v}(k), \boldsymbol{R}(k))}. \tag{4.22}$$

Next I justify that the product term is a good and reasonable choice for a proposal distribution that incorporates all the new observations available across the network. Assume at iteration k, node l, for $(1 \leq l \leq N)$ computes an unbiased local estimate $\bar{\mathbf{x}}^{(l)}(k)$ of the state variables $\mathbf{x}(k)$ from its particle-based representation of the filtering distribution with the corresponding error and error covariance denoted by $\Delta_x^{(l)}(k) = \mathbf{x}(k) - \bar{\mathbf{x}}^{(l)}(k)$ and $\bar{\boldsymbol{P}}^{(l)}(k)$. When the estimation error $\Delta_x^{(i)}(k)$ and $\Delta_x^{(j)}(k)$, for $(1 \leq i, j \leq N)$ and $i \neq j$ are uncorrelated, the optimal fusion of N unbiased local estimates $\bar{\mathbf{x}}^{(l)}(k)$ in linear minimum variance scene is shown [76] to be given by

$$\bar{\boldsymbol{P}}(k) = (\sum_{l=1}^{N}\left(\bar{\boldsymbol{P}}^{(l)}(k)\right)^{-1})^{-1} \quad \text{and} \quad \bar{\mathbf{x}}(k) = (\sum_{l=1}^{N}\left(\bar{\boldsymbol{P}}^{(l)}(k)\right)^{-1})^{-1} \times \sum_{l=1}^{N}\left(\bar{\boldsymbol{P}}^{(l)}(k)\right)^{-1}\bar{\mathbf{x}}^{(l)}(k). \tag{4.23}$$

where $\bar{\mathbf{x}}(k)$ is the overall estimate obtained from $P(\mathbf{x}(k)|\mathbf{z}(1:k))$ with error covariance $\bar{\boldsymbol{P}}(k)$. Eq. (4.23) is the same as Eq. (4.15), which describes the statistics of the product of N normally distributed densities. The optimal proposal distribution is also a filtering density [43], therefore, the proposal distribution defined in Eq. (4.21) is a good choice that simplifies the update equation of the fusion filter. Further, Eq. (4.21) is a reasonable approximation of the optimal proposal distribution. From the framework of unscented Kalman filter and unscented particle filter, it is well known [44] that approximating distributions will be advantageous over approximating non-linear functions. The drawback with this proposal density is the impractical assumption that the local estimates are uncorrelated. I improve the performance of the fusion filter using a better approximation of the optimal proposal distribution, which is described next.

4.1.5.3 Gaussian Approximation of The Optimal Proposal Distribution

I consider the optimal solution to the fusion protocol (Eq. (4.4)) when local filtering densities are normally distributed. In such a case, $P(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k-1))$ is also normally distributed [127] with mean $\mathbf{x}^{(l,\text{global})}(k)$ and covariance $\boldsymbol{P}^{(l,\text{global})}(k)$

$$\bar{\boldsymbol{P}}^{(l,\text{global})^{-1}}(k) = \left(\boldsymbol{R}^{(l)}(k)\right)^{-1} + \underbrace{\sum_{j=1}^{N} \boldsymbol{P}^{(j)^{-1}}(k)}_{\mathbf{X}_{c1}^{(l)}(\infty)} - \underbrace{\sum_{j=1}^{N} \boldsymbol{R}^{(j)^{-1}}(k)}_{\mathbf{X}_{c3}^{(l)}(\infty)} \tag{4.24}$$

$$\begin{aligned}\bar{\mathbf{x}}^{(l,\text{global})}(k) &= \bar{\boldsymbol{P}}^{(l,\text{global})^{-1}}(k)\Big[\left(\boldsymbol{R}^{(l)}(k)\right)^{-1}\boldsymbol{v}^{(l)}(k) \\ &+ \underbrace{\sum_{j=1}^{N} \boldsymbol{P}^{(j)^{-1}}(k)\boldsymbol{\mu}^{(j)}(k)}_{\mathbf{x}_{c2}^{(l)}(\infty)} - \underbrace{\sum_{j=1}^{N}\left(\boldsymbol{R}^{(j)}(k)\right)^{-1}\boldsymbol{v}^{(j)}(k)}_{\mathbf{x}_{c4}^{(l)}(\infty)}\Big].\end{aligned} \tag{4.25}$$

The four terms $\mathbf{X}_{c1}^{(l)}(\infty)$, $\mathbf{x}_{c2}^{(l)}(\infty)$, $\mathbf{X}_{c3}^{(l)}(\infty)$, and $\mathbf{x}_{c4}^{(l)}(\infty)$ are already computed and available at local nodes as part of computing the product terms. Fusion rules in Eqs. (4.24) and (4.25) are obtained based on the track fusion without feedback [127]. In such a scenario, particles $\mathbb{X}_i^{(l,\text{FF})}(k)$ are drawn from $\mathcal{N}(\mathbf{x}^{(l,\text{global})}(k),\ \boldsymbol{P}^{(l,\text{global})}(k))$ and the weight update equation (Eq. (4.22)) is given by

$$W_i^{(l,\text{FF})}(k) \propto W_i^{(l,\text{FF})}(k-1)\frac{\mathcal{N}(\mathbb{X}_i^{(l,\text{FF})}(k);\boldsymbol{\mu}(k),\boldsymbol{P}(k))P(\mathbb{X}_i^{(l,\text{FF})}(k)|\mathbb{X}_i^{(l,\text{FF})}(k-1))}{\mathcal{N}(\mathbb{X}_i^{(l,\text{FF})}(k);\boldsymbol{v}(k),\boldsymbol{R}(k))\mathcal{N}(\mathbb{X}_i^{(l,\text{FF})}(k);\mathbf{x}_{(k)}^{(l,\text{global})},\boldsymbol{P}_{(k)}^{(l,\text{global})})}. \tag{4.26}$$

Algorithm 5 FUSION FILTER($\{\mathbb{X}_i^{(l,\text{FF})}(k-1), W_i^{(l,\text{FF})}(k-1)\}_{i=1}^{N_{\text{FF}}}$)

```
Input: ${\mathbb{X}_i^{(l,FF)}(k-1), W_i^{(l,FF)}(k-1)}_{i=1}^{N_{l,FF}}$ - Fusion filter's particles and associated weights.
Output: ${\mathbb{X}_i^{(l,FF)}(k), W_i^{(l,FF)}(k)}_{i=1}^{N_s}$ Fusion filter's updated particles and associated weights.
1: for $l = 1 : N$, do
       $(\boldsymbol{\mu}^{(l)}(k), \boldsymbol{P}^{(l)}(k), \boldsymbol{v}^{(l)}(k), \boldsymbol{R}^{(l)}(k)) = \text{LocalFilter}\Big(\{\mathbb{X}_i^{(l)}(k-1), W_i^{(l)}(k-1)\}_{i=1}^{N_s}, \mathbf{z}^{(l)}(k)\Big)$
2: end for
3: DoFusion$(\{\boldsymbol{\mu}^{(l)}(k), \boldsymbol{P}^{(l)}(k)\}_{l=1}^{N})$ computes $\{\boldsymbol{\mu}^{(l,FF)}(k), \boldsymbol{P}^{(l,FF)}(k)\}$ for numerator of Eq. (4.4).
4: DoFusion$(\{\boldsymbol{v}^{(l)}(k), \boldsymbol{R}^{(l)}(k)\}_{l=1}^{N})$ computes $\{\boldsymbol{v}^{(l,FF)}(k), \boldsymbol{R}^{(l,FF)}(k)\}$ for denominator of (4.4).
5: for $i = 1 : N$, do
       • Generate particles $\left\{\mathbb{X}_i^{(l,FF)}(k)\right\}_{i=1}^{N_{l,FF}}$ by sampling proposal distribution defined in Section 4.1.5.
       • Compute weights $W^{(l,FF)}(k)$ using Eq. (4.22).
6: end for
7: Resampling: $\Big(\{\mathbb{X}_i^{(l,FF)}(k), W_i^{(l,FF)}(k)\}_{i=1}^{N_{FF}}\Big) = \text{Resample}\Big(\{\mathbb{X}_i^{(l,FF)}(k), W_i^{(l,FF)}(k)\}_{i=1}^{N_{FF}}\Big)$.
```

The various steps of the fusion filter are outlined in Algorithm 5. The filtering step of the CD/DPF is based on running the localized filters at each node followed by the fusion filter, which computes the global posterior density by running consensus algorithm across the network. At the completion of the consensus step, all nodes have the same global posterior available.

4.1.6 Computational complexity

In this section, I provide a rough comparison of the computational complexity of the CF/DPF versus that of the centralized implementation. Because of the non-linear dynamics of the particle filter, it is somewhat difficult to derive a generalized expression for its computational complexity.

There are steps that can not be easily evaluated in the complexity computation of the particle filter such as the cost of evaluating a non-linear function (as is the case for the state and observation models) [131]. In order to provide a rough comparison, we consider below a simplified linear state model with Gaussian excitation and uncorrelated Gaussian observations. Following the approach proposed in [131], the computational complexity of two implementations of the particle filter is expressed in terms of flops, where a flop is defined as addition, subtraction, multiplication or division of two floating point numbers. The computational complexity of the centralized particle filter for N–node network with N_s particles is of $\mathrm{O}\left((n_x^2 + N)N_s\right)$. The CF/DPF runs the local filter at each observation node which is similar in complexity to the centralized particle filter except that the observation (target's bearing at each node) is a scalar. Setting $N = 1$, the computational complexity of the local filter is of $\mathrm{O}\left(n_x^2 N_{\mathrm{LF}}\right)$ per node, where N_{LF} is the number of particles used by the local filter. There are two additional components in the CF/DPF: (i) The fusion filter which has a complexity of $\mathrm{O}(n_x^2 N_{\mathrm{FF}})$ per node where N_{FF} is the number of particles used by the fusion filter, and; (ii) The CF/DPF introduces an additional consensus step which has a computational complexity of $\mathrm{O}(n_x^2 \Delta_{\mathcal{G}} N_c(\boldsymbol{U}))$. The associated convergence time $N_c(\boldsymbol{U}) = 1/\log(1/r_{\mathrm{asym}}(\boldsymbol{U}))$ provides the asymptotic number of consensus iterations required for the error to decrease by the factor of $1/e$ and is expressed in terms of the asymptotic convergence rate $r_{\mathrm{asym}}(\boldsymbol{U})$. Based on [31], $N_c(\boldsymbol{U}) = -1/\max_{2\leq i\leq N}\log(|\lambda_i(\boldsymbol{U})|)$, where $\lambda_i(\boldsymbol{U})$ is the eigenvalue of the consensus matrix $\boldsymbol{U}$. The overall computational complexity of the CF/DPF is, therefore, given by $\max\left\{\mathrm{O}(N n_x^2(N_{\mathrm{LF}} + N_{\mathrm{FF}})), \mathrm{O}(n_x^2 \Delta_{\mathcal{G}} N_c(\boldsymbol{U}))\right\}$ compared to the computational complexity $\mathrm{O}\left((n_x^2 + N)N_s\right)$ of the centralized implementation.

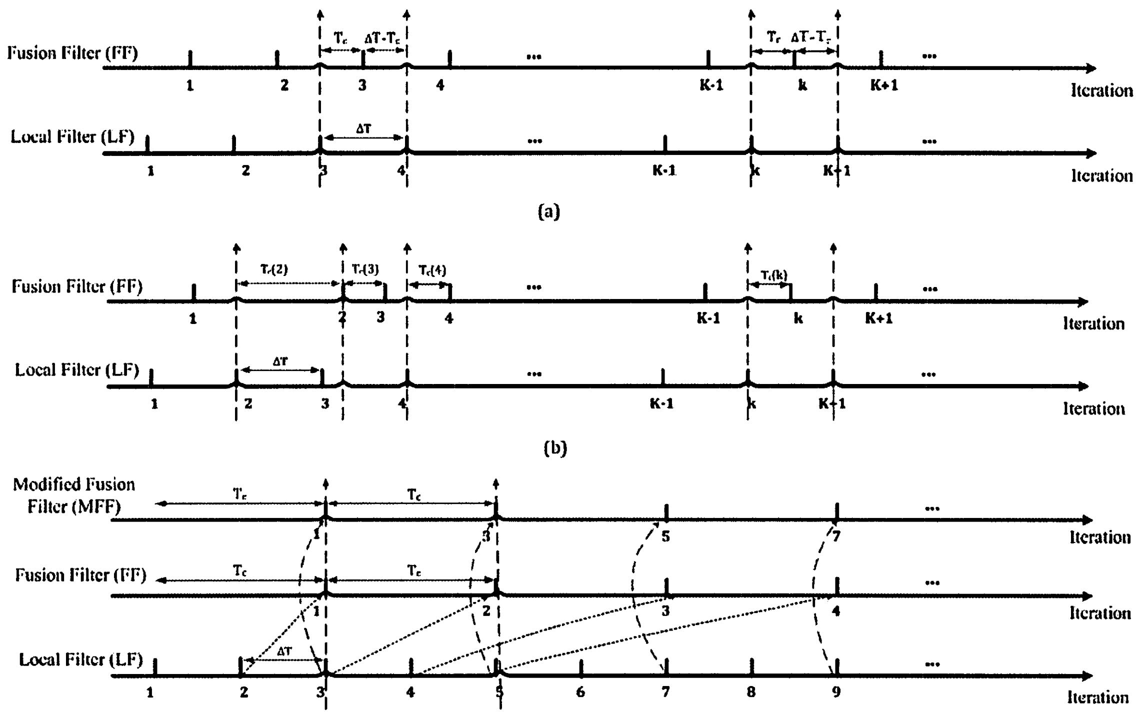


Figure 4.3: Multi-rate implementation of the local and fusion filters. (a) The ideal scenario where the fusion filter's consensus step converges before the new iteration of the local filter. (b) The convergence rate of the fusion filter varies according to the network connectivity. (c) The lag between the fusion filter and the local filter grows exponentially.

4.2 Modified Fusion Filter

In the CF/DPF, the local filters and the fusion filters can run out of synchronization due to intermittent network connectivity. The local filters are confined to their sensor node and unaffected by loss of connectivity. The fusion filters, on the other hand, run consensus algorithms. The convergence of these consensus algorithms is delayed in cases where connectivity is temporarily lost or the communication bandwidth is reduced. In this section I develop ways of dealing with such intermittent connectivity issues. First, let me introduce the notation. I assume that the observations arrive at constant time intervals of ΔT. Each iteration of the local filters is performed within this interval, which I will refer to as the local filter's estimation interval. The duration (the fusion filters's estimation interval) of the update cycle of the fusion filter is denoted by T_c. Fig. 4.3 illustrates three scenarios dealing with different fusion filter's estimation intervals. Fig. 4.3(a) is the ideal scenario where $T_c \leq \Delta T$ and the fusion filter's consensus step converges before the new iteration of the local filter. In such a scenario, the local and fusion filters stay synchronized. Fig. 4.3(b) shows the second scenario when the convergence rate of the fusion filter varies according to the network connectivity. Under regular connectivity $T_c < \Delta T$ and with limited connectivity losses, the fusion filters manages to catch up with the localized filters in due time. Fig. 4.3(c) considers a more problematic scenario when $T_c > \Delta T$. Even with ideal connectivity, the fusion filter will continue to lag the localized filters with no hope of its catching up. The bottom two timing diagrams in Fig. 4.3(c) refer to this scenario with $T_c = 2\Delta T$. As illustrated, the lag between the fusion filter and the localized filters grows exponentially with time in this scenario. An improvement to the fusion filter is suggested in the top timing diagram of Fig. 4.3(c), where the fusion filter uses the most recent local filtering density of the localized filters. This allows the fusion filter to catch up with the localized filter even for cases $T_c > \Delta T$. Such a modified fusion implementation requires an updated fusion rule for the global posterior density, which is considered next.

At iteration $k+m$, I assume that node l, for $(1 \leq l \leq N)$, has a particle-based approximation of the local filtering distributions $P(\mathbf{x}(k+m)|\mathbf{z}^{(l)}(1:k+m))$, while its fusion filter has a particle-based approximation of the global posterior distribution $P(\mathbf{x}(0:k)|\mathbf{z}(1:k))$ for iteration k. In other words, the fusion filters are lagging the localized filters by m iterations. In the conventional fusion filter the statistics of $P(\mathbf{x}(k+1)|\mathbf{z}^{(l)}(1:k+1))$, for $(1 \leq l \leq N)$ are used in the next consensus step of the fusion filter which then computes the global posterior $P(\mathbf{x}(0:k+1)|\mathbf{z}(1:k+1))$

based on Theorem 5. The modified fusion filter uses the most recent local filtering distributions $P(\mathbf{x}(k+m)|\mathbf{z}^{(l)}(1\!:k+m))$ according to Theorem 6.

Theorem 6. *Conditioned on the state variables, assume that the observations made at node l are independent of the observations made at node j, $(j \neq l)$. The global posterior distribution for a N-sensor network at iteration $k+m$ is then given by*

$$P(\mathbf{x}(0\!:\!k+m)|\mathbf{z}(1\!:\!k+m)) \propto \prod_{l=1}^{N} \frac{\prod_{k'=k+1}^{k+m} P\left(\mathbf{x}(k')|\mathbf{z}^{(l)}(1\!:\!k')\right)}{\prod_{k'=k+1}^{k+m} P\left(\mathbf{x}(k')|\mathbf{z}^{(l)}(1\!:\!k'-1)\right)} \prod_{k'=k+1}^{k+m} P\left(\mathbf{x}(k')|\mathbf{x}(k'-1)\right) \times P\left(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k)\right). \quad (4.27)$$

The proof of Theorem 6 is included in Appendix B.2. In the consensus step of the modified fusion filter, two average consensus algorithms are used to compute $\prod_{l=1}^{N}\prod_{k'=k+1}^{k+m} P(\mathbf{x}(k')|\mathbf{z}^{(l)}(1\!:k'))$ and $\prod_{l=1}^{N}\prod_{k'=k+1}^{k+m} P(\mathbf{x}(k')|\mathbf{z}^{(l)}(1\!:k'-1))$, i.e.,

$$\prod_{l=1}^{N}\prod_{k'=k+1}^{k+m} P\left(\mathbf{x}(k')|\mathbf{z}^{(l)}(1\!:\!k')\right) \propto \prod_{l=1}^{N} \mathcal{N}(\boldsymbol{\mu}^{(l)}(k+1\!:\!k+m), \boldsymbol{P}^{(l)}(k+1\!:\!k+m)) \quad (4.28)$$

$$\text{and} \prod_{l=1}^{N}\prod_{k'=k+1}^{k+m} P\left(\mathbf{x}(k')|\mathbf{z}^{(l)}(1\!:\!k'-1)\right) \propto \prod_{l=1}^{N} \mathcal{N}(\boldsymbol{v}^{(l)}(k+1\!:\!k+m), \boldsymbol{R}^{(l)}(k+1\!:\!k+m)), \quad (4.29)$$

instead of computing $\prod_{l=1}^{N} P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k))$ and $\prod_{l=1}^{N} P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k-1))$ as was the case for the conventional fusion filter. The modified fusion filter starts with a set of particles $\mathbb{X}_i^{(\mathrm{MFF},l)}(k)$, $W_i^{(\mathrm{MFF},l)}(k)$ approximating $P(\mathbf{x}(0:k)|\mathbf{z}(1:k))$ and generates updated particles $\mathbb{X}_i^{(\mathrm{MFF},l)}(k+m)$, $W_i^{(\mathrm{MFF},l)}(k+m)$ for $P(\mathbf{x}(0\!:\!k+m)|\mathbf{z}(1\!:\!k+m))$ using the following weight update equation

$$W_i^{(l,\mathrm{MFF})}(k+m) \propto W_i^{(l,\mathrm{MFF})}(k) \times \frac{\prod_{k'=k+1}^{k+m} P\left(\mathbb{X}_i^{(l,\mathrm{MFF})}(k')|\mathbb{X}_i^{(l,\mathrm{MFF})}(k'-1)\right)}{\mathcal{N}(\mathbb{X}_i^{(l,\mathrm{MFF})}(k+m); \boldsymbol{v}(k+1\!:\!k+m), \boldsymbol{R}(k+1\!:\!k+m))}, \quad (4.30)$$

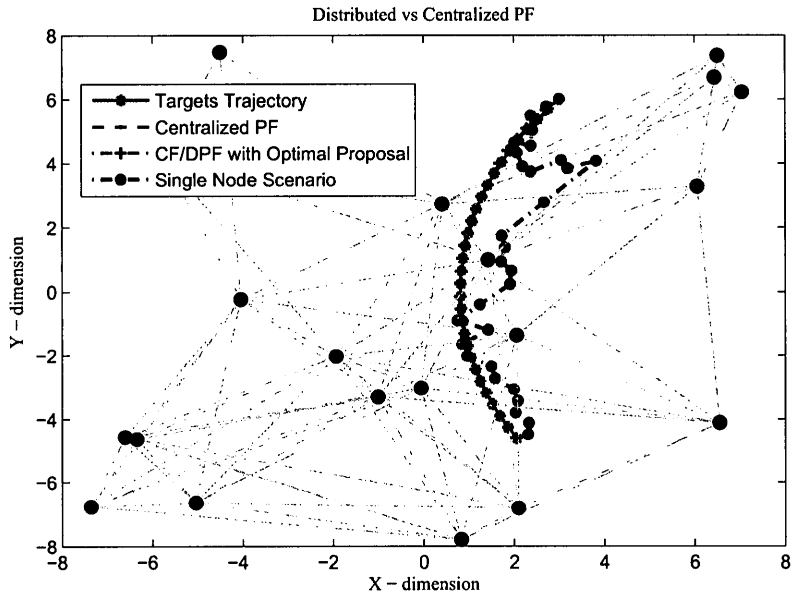
which is obtained directly from Eq. (4.27). Note that the normal approximation in Eqs. (4.28)–(4.30) are similar to the ones used in the conventional fusion filter. Furthermore, I note that the modification requires prediction of the particles from iteration k all the way to $k+m$ in order to evaluate the second term on the right hand side of Eq. (4.30). Algorithm 6 outlines this step and summarizes the modified fusion filter.

Algorithm 6 Modified Fusion Filter

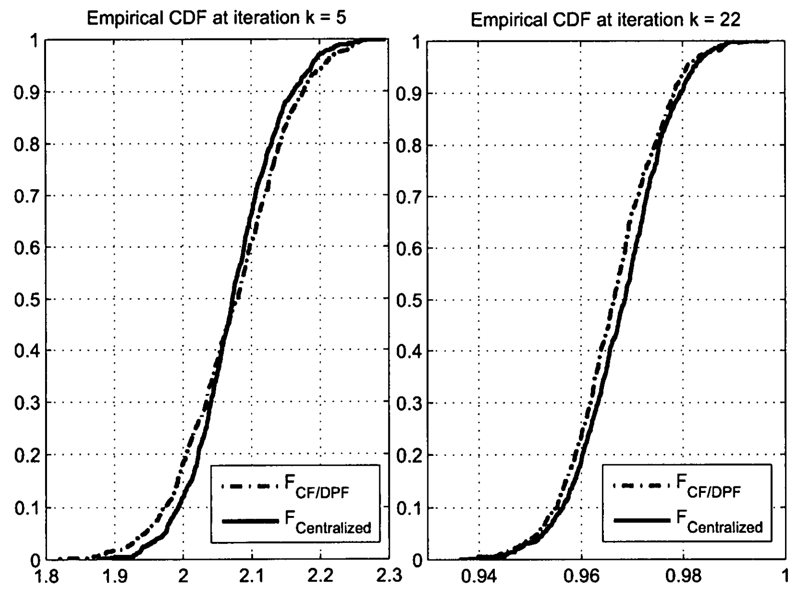
```
Input: $\{\mathbb{X}_i^{(l,\mathrm{MFF})}(k), W_i^{(l,\mathrm{MFF})}(k)\}_{i=1}^{N_{l,\mathrm{MFF}}}$ - Fusion filter's particles and associated weights.
Output: $\{\mathbb{X}_i^{(l,\mathrm{MFF})}(k+m), W_i^{(l,\mathrm{MFF})}(k+m)\}_{i=1}^{N_s}$ updated particles and associated weights.
1: for $k' = k+1 : k+m$, do
       $\mathcal{N}(\boldsymbol{\mu}^{(l)}(k'), \boldsymbol{P}^{(l)}(k')) = \text{SaveGaussian}\Big(\{\mathbb{X}_i^{(l)}(k'), W_i^{(l)}(k')\}_{i=1}^{N_s}\Big)$
       $\mathcal{N}(\boldsymbol{v}^{(l)}(k'), \boldsymbol{R}^{(l)}(k')) = \text{SaveGaussian}\Big(\{\mathbb{X}_i^{(l)}(k'+1|k'), W_i^{(l)}(k')\}_{i=1}^{N_s}\Big)$
2: end for
3: $\mathcal{N}(\boldsymbol{\mu}^{(l)}(k+1:k+m), \boldsymbol{P}^{(l)}(k+1:k+m)) = \text{SaveGaussian}\Big(\prod_{k'=k+1}^{k+m} \mathcal{N}(\boldsymbol{\mu}^{(l)}(k'), \boldsymbol{P}^{(l)}(k'))\Big)$.
4: $\mathcal{N}(\boldsymbol{v}^{(l)}(k+1:k+m), \boldsymbol{R}^{(l)}(k+1:k+m)) = \text{SaveGaussian}\Big(\prod_{k'=k+1}^{k+m} \mathcal{N}(\boldsymbol{v}^{(l)}(k'), \boldsymbol{R}^{(l)}(k'))\Big)$.
5: $\{\boldsymbol{\mu}^{(l,\mathrm{MFF})}(k+1:k+m), \boldsymbol{P}^{(l,\mathrm{MFF})}(k+1:k+m)\} = \text{DoFusion}(\{\boldsymbol{\mu}^{(l)}(k+1:k+m), \boldsymbol{P}^{(l)}(k+1:k+m)\}_{l=1}^{N})$.
6: $\{\boldsymbol{v}^{(l,\mathrm{MFF})}(k+1:k+m), \boldsymbol{R}^{(l,\mathrm{MFF})}(k+1:k+m)\} = \text{DoFusion}(\{\boldsymbol{v}^{(l)}(k+1:k+m), \boldsymbol{R}^{(l)}(k+1:k+m)\}_{l=1}^{N})$.
7: for $i = 1 : N_{\mathrm{FF}}$, do
8:     for $k' = k+1 : k+m-1$, do
           $\mathbb{X}_i^{(l,\mathrm{MFF})}(k') \sim P\big(\mathbf{x}(k')|\mathbb{X}_i^{(l,\mathrm{MFF})}(k'-1)\big)$.
9:     end for
       $\mathbb{X}_i^{(l,\mathrm{MFF})}(k+m) \sim \mathcal{N}\big(\boldsymbol{\mu}^{(l,\mathrm{MFF})}(k+1:k+m), \boldsymbol{P}^{(l,\mathrm{MFF})}(k+1:k+m)\big)$.
       Compute weights $W_i^{(l,\mathrm{MFF})}(k+m)$ using Eq. (4.30).
10: end for
```

4.3 Simulation Results

In this section, different scenarios with non-linear target kinematics and non-Gaussian observation model are considered to investigate the properties of the proposed CF/DPF implementation. As stated previously, the CF/DPF and the UCD/DPF are not application specific and are applicable to any nonlinear dynamical system. Appendix B.3 provides a rough comparison of the computa-



(a)



(b)

Figure 4.4: Scenario 1: (a) Target's tracks obtained from the centralized, CF/DPF and stand-alone algorithms (the consensus is allowed to converge). (b) CDFs for the X-coordinate of the target from the centralized and CF/DPF approaches for $k = 5, 22$.

tional complexity of the UCD/DPF and CF/DPF versus that of the centralized implementation.

A sensor network of $N = 20$ nodes with random geometric graph model in a square region of dimension (16×16) m^2 is considered. Each sensor communicates only with its neighboring nodes within a connectivity radius of $\sqrt{2\log(N)/N}$ units. In addition, the network is assumed to be connected with each node linked to at least one other node in the network. Measurements are the target's bearings with respect to the platform of each node referenced (clockwise positive) to the y-axis as defined in Eq. (3.52). The observations are assumed to be corrupted by the non-Gaussian *target glint noise* [165] modeled as a mixture model of two zero-mean Gaussians [165], one with a high probability of occurrence and small variance and the other with relatively a small probability of occurrence and high variance. The likelihood model at node l, for $(1 \leq l \leq N)$, is described as

$$P(\mathbf{z}^{(l)}|\mathbf{x}(k)) = (1-\epsilon) \times \mathcal{N}(\mathbf{x}; 0, \sigma^2_{\varsigma^{(l)}}(k)) + \epsilon \times \mathcal{N}(\mathbf{x}; 0, 10^4\sigma^2_{\varsigma^{(l)}}(k)), \tag{4.31}$$

where $\epsilon = 0.09$ in the simulations. Furthermore, the observation noise is assumed to be state dependent such that the bearing noise variance $\sigma^2_{\varsigma^{(l)}}(k)$ at node l depends on the distance $r^{(l)}(k)$ between the observer and target. Based on [166], the variance of the observation noise at node l is, therefore, given by

$$\sigma^2_{\varsigma^{(l)}}(k) = 0.08{r^{(l)}}^2(k) + 0.1150r^{(l)}(k) + 0.7405. \tag{4.32}$$

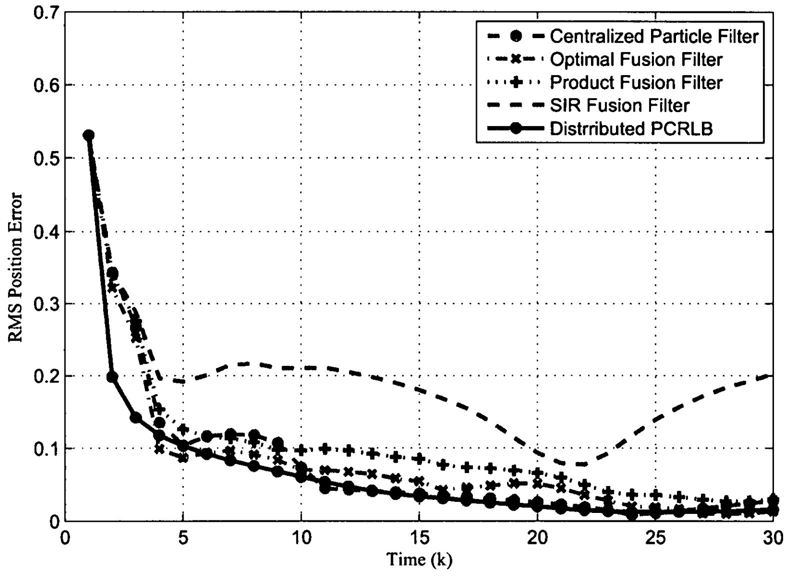
Due to state-dependent noise variance, the signal to noise ratio (SNR) is time-varying and differs (within a range of -10dB to 20dB) from one sensor node to the other depending on the location of the target. Averaged across all nodes and time, the mean SNR is 5.5dB. In the simulations, I chose to incorporate observations made at all nodes in the estimation, however, sensor selection based on the proposed distributed PCRLB can be used, instead, which will be considered later in Section 6. Both centralized and distributed filters are initialized based on the procedure described in Section 3.3.1.

The target starts from coordinates $(3, 6)$ units The position of target the target $([X, Y])$ in first three iterations are $(2.6904, 5.6209)$, $(2.3932, 5.2321)$, and $(2.1098, 4.8318)$. The initial course is set at $-140°$ with the standard deviation of the process noise $\sigma_v = 1.6 \times 10^{-3}$ unit. The number N_s of vector particles for centralized implementation is $N_s = 10,000$. The number N_{LF} and N_{FF} of vector particles used in each local filter and fusion filter is 500. The number of particles for the CF/DPF are selected to keep its computational complexity the same as that of the centralized implementation. To quantify the tracking performance of the proposed methods three scenarios are considered. In Scenario 1 and 2, the nonlinear CCT state model (Eq. (3.50)) presented in Section 3.3 is used. Scenario 3 considers distributed unicycle mobile robot localization problem as introduced in Section 2.6.4 where the state model is given by Eqs. (4.33)-(4.34).

4.3.1 Scenario 1

Scenario 1 accomplishes two goals. First, the performance of the proposed CF/DPF is compared versus the centralized implementation. The fusion filters used in the CF/DPF are allowed to converge between two consecutive iterations of the localized particle filters (i.e., following the timing subplot (a) of Fig. 4.3). Second, the impact of the three proposal distributions listed in Section 4.1.5 on the CF/DPF are compared. The performance of the CF/DPF is computed for each of these proposal distributions using Monte Carlo simulations.

Fig. 4.4(a) plots one realization of the target track and the estimated tracks obtained from: (i) The CF/DPF; (ii) the centralized implementation, and; (iii) a single node estimation (stand alone case). In the CF/DPF, the Gaussian approximation of the optimal proposal distribution is used as the proposal distribution (Case 3 in Section 4.1.5). The two estimates from the CF/DPF and the centralized implementation are fairly close to the true trajectory of the target so much so as that they overlap. The stand alone scenario based on running a particle filter at a single node



(a)

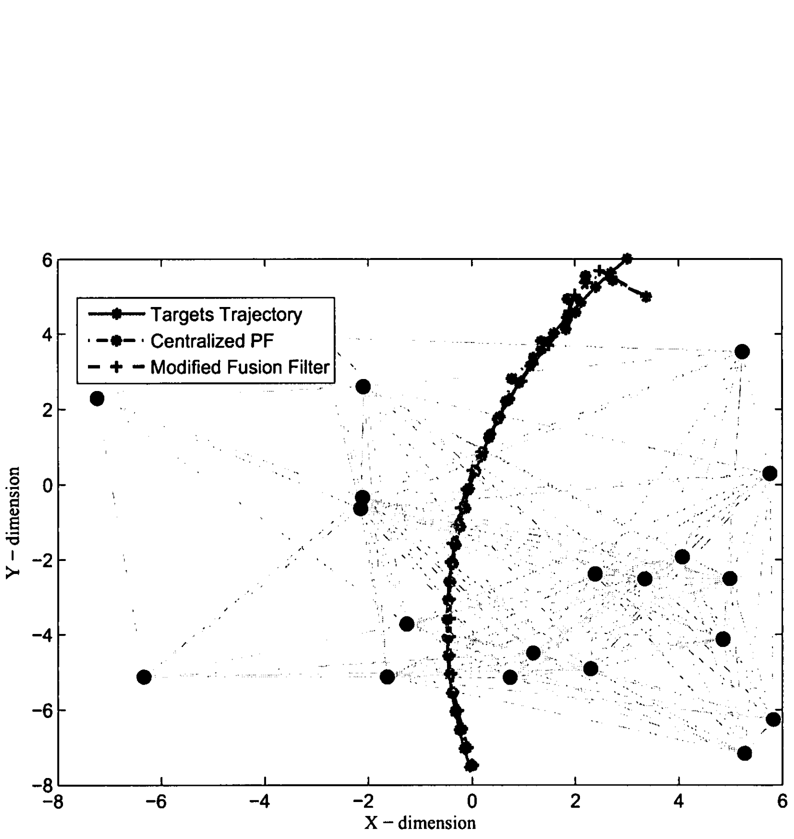
Figure 4.5: Scenario 1: Comparison of the RMS errors resulting from the centralized versus distributed implementations.

(shown as the red circle in Fig. 4.4(a)) fails to track the target. Fig. 4.4(b) plots the cumulative distribution function (CDF) for the X-coordinate of the target estimated using the centralized and CF/DPF implementations for iterations $k = 5$ and 22. We note that the two CDFs are close to each other. Fig. 4.4 illustrates the near-optimal nature of the CF/DPF.

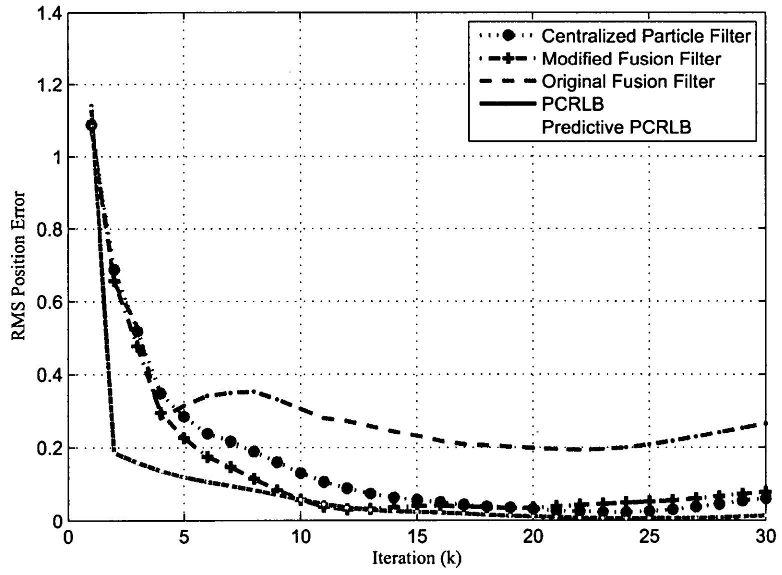
Fig. 4.5 compares the RMS error curves for the target's position. Based on a Monte-Carlo simulation of 100 runs, Fig. 4.5 plots the RMS error curves for the estimated target's position via three CF/DPF implementations obtained using different proposals distributions. It is observed that the SIR fusion filter performs the worst in this highly non-linear environment with non-Gaussian observation noise, while the outputs of the centralized and the other two distributed implementations are fairly close to each other and approach the PCRLB. Since the product fusion filter requires less computations, the simulations in Scenario 2 are based on the CF/DPF implementation using the product fusion filter.

4.3.2 Scenario 2

The second scenario models the timing subplot (c) of Fig. 4.3. The convergence of the fusion filter takes up to two iterations of the localized filters. The original fusion filter (Algorithm 5) is unable to converge within two consecutive iterations of the localized particle filters. Therefore, the lag between fusion filters and the localized filters in the CF/DPF continues to increase exponentially. The modified fusion filter described in Algorithm 6 is implemented to limit the lag to two localized filter iterations. The target's track are shown in Fig. 4.6(a) for the centralized implementation, original and modified fusion filter. Fig. 4.6(b) shows the RMS error curves for the target's position including the RMS error resulting from Algorithm 5 and the extended PCRLB (Appendix E). Since consensus is not reached, therefore, the fusion estimate from Algorithm 5 is different from one node to another. Result from one randomly selected node is included. The node performs



(a)



(b)

Figure 4.6: Scenario 2: (a) Actual target's track alongside the estimated tracks obtained from the centralized and modified fusion filter. Here, the consensus algorithm converges after every two iterations of the local particle filters. (b) Comparison of the RMS errors resulting from the centralized, original fusion filter and modified fusion filter.

poorly due to consensus not reached. The performance of the modified fusion filter remains close to its centralized counterpart, therefore, it seems capable of handling intermittent consensus steps. In Fig. 4.6(b), the extended PCRLB overlaps with its centralized counterpart.

4.3.3 Scenario 3

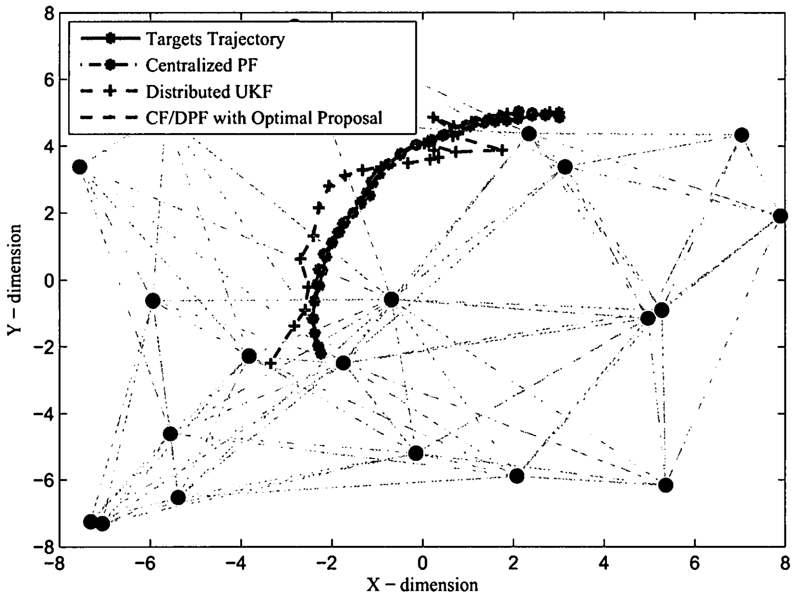
In the third scenario, a distributed mobile robot localization problem [6, 7] is considered based on angle-only measurements. This is a good benchmark since the underlying dynamics is non-linear with non-additive forcing terms resulting in a non-Gaussian transitional state model. This scenario is introduced to check if the CF/DPF can handle non-Gaussian state models, therefore, the consensus is assumed to converge between two consecutive observations. As stated previously in Section 2.6.4, the state vector of the unicycle robot is defined by $\mathbf{x} = [X, Y, \theta]^T$, where (X, Y) is the 2D coordinate of the robot and θ is its orientation. The velocity and angular velocity are denoted by $\tilde{V}(k)$ and $\tilde{W}(k)$, respectively. The following discrete-time non-linear unicycle model [6] represents the state dynamics of the robot

$$X(k+1) = X(k)+\frac{\tilde{V}(k)}{\tilde{W}(k)}(\sin(\theta(k)+\tilde{W}(k)\Delta T)-\sin(\theta(k))) \tag{4.33}$$

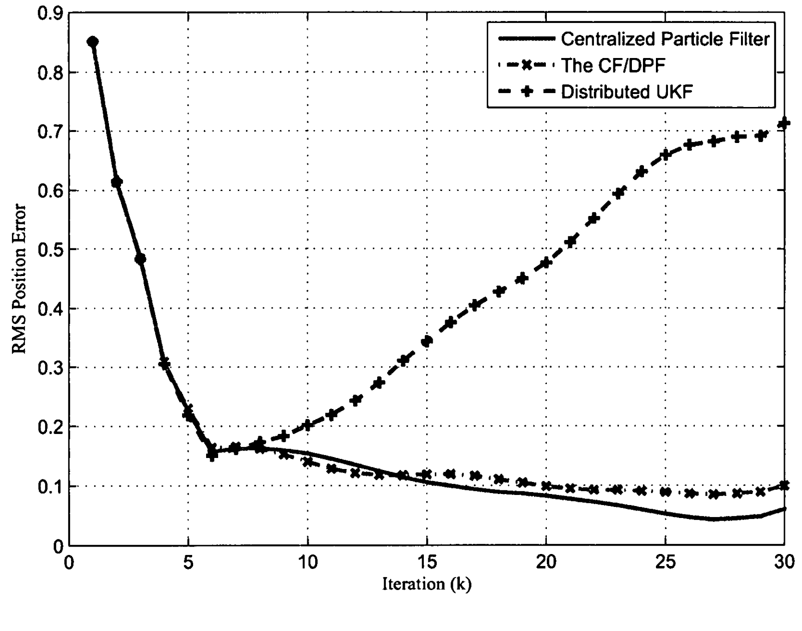
$$Y(k+1) = Y(k)+\frac{\tilde{V}(k)}{\tilde{W}(k)}(\cos(\theta(k)+\tilde{W}(k)\Delta T)-\cos(\theta(k))) \tag{4.34}$$

$$\text{and} \qquad \theta(k+1) = \theta(k) + \tilde{W}(k)\Delta T + \xi_\theta \Delta T, \tag{4.35}$$

where ΔT is the sampling time and ξ_θ is the orientation noise term. The design parameters are: $\Delta T = 1$, a mean velocity of 30 cm/s with a standard deviation of 5 cm/s, and a mean angular velocity of 0.08 rad/s with a standard deviation of 0.01 rad. The observation model is similar to the one described for Scenario 1 with non-Gaussian and state-dependent observation noise. The robot starts at coordinates $(3, 5)$. Fig. 4.7(a) shows one realization of the sensor placement along with the robot's trajectories estimated from the proposed CF/DPF, centralized particle



(a)



(b)

Figure 4.7: (a) Robot trajectories estimated from the CF/DPF, centralized, and distributed UKF implementations. (b) RMS error plots for the three implementations.

filter and distributed unscented Kalman filter (UKF) [7] implementations. We observe that both centralized particle filter and CF/DPF closely follow the robot trajectory, while the distributed UKF deviates after some initial iterations. Fig. 4.7(b) plots the RMS error plots obtained from Monte-Carlo simulation of 100 runs, which corroborate our earlier observation that the CF/DPF and the centralized particle filter provide better estimates that are close to each other, while the UKF produces a different result with the highest RMS error.

4.4 Summary

In this chapter, I propose a multi-rate consensus/fusion based framework, referred to as the CF/DPF, for distributed implementation of the particle filter. In the proposed framework, two particle filters run at each sensor node. The first filter, referred to as the local filter, recursively runs the particle filter based only on the local observations. I introduce a second particle filter at each node, referred to as the fusion filter, which consistently assimilate local estimates into a global estimate by extracting new information. The proposed CF/DPF implementation allows the fusion filter to run at a rate different from that of the local filters. Achieving consensus between two successive iterations of the localized particle filter is no longer a requirement. The fusion filter and its consensus-step are now separated from the local filters, which enables the consensus step to converge without any time limitations. Numerical simulations verify the near-optimal performance of the CF/DPF. The CF/DPF estimates follow the centralized particle filter closely approaching the PCRLB at the SNRs that we tested.

5 Posterior Cramér-Rao Lower Bound for Distributed Architectures (dPCRLB)

The Cramér-Rao lower bound (CRLB) is widely used for assessing the performance of an estimation algorithm. In the simplest form, the CRLB provides a lower limit on the error variance of an unbiased estimator of a deterministic parameter. An unbiased estimator that achieves the CRLB is considered to be efficient. In dealing with stochastic dynamical models, the state variables are often random necessitating the need for a Bayesian estimator with the bound on the error variance taken with respect to a posterior density function. In cases where statistics related to a random variable are being estimated, a lower bound [132] that is analogous to the CRLB is referred to as the posterior Cramér-Rao lower bound (PCRLB) (at times also referred to as the Bayesian CRLB or the Van Trees version of the CRLB). A common form of the PCRLB is the conventional (non-conditional) PCRLB determined primarily from the state model, observation model, and prior knowledge of the initial state of the system. Most PCRLB formulations does not allow for a recursive implementation and suffer from computational complexity as the dimension of the state vector grows in time.

The chapter derives recursive distributed algorithms for online computation of the optimal PCRLB for distributed sensor/agent networks (AN/SN). The motivation for this work comes from sensor selection decisions [133–141] especially in geographically dispersed networks deploying an unrestrictedly large number of sensor nodes. Limitations in power, frequency, and bandwidth restrict the maximum number of active sensors that can simultaneously participate in the decentralized estimation process. The problem of sensor selection is to determine the optimal way of dynamically selecting a subset of sensors over time that provides the best estimation performance. Among other criteria proposed for sensor resource management, the PCRLB [15, 142–147] provides a predictive measure of the achievable optimal performance. More importantly, this PCRLB is independent of the estimation mechanism. In the past, sensor management algorithms based on the PCRLB have only been presented for the centralized networks with a fusion centre. No such work has been pursued for distributed estimation networks primarily because of the difficulty in computing the PCRLB distributively. The chapter addresses this gap and as a first step derives optimal recursive PCRLB expressions, referred to as the distributed PCRLB (dPCRLB), for sensor networks configured using distributed architectures. *I reiterate that the centralized computation of the PCRLB cannot be realized [15] for dynamic resource allocation in decentralized networks due to the absence of the fusion centre and the only alternative is real-time, recursive computation of the dPCRLB in a distributed fashion especially for sensor selection.*

The seminal work of Tichavsky *et al.* [148] provides a recursive formula to update the Fisher information matrix (FIM), i.e., the inverse of the PCRLB, iteratively for a general multidimensional, discrete time, nonlinear, estimation problem in the centralized architecture while keeping the dimensions of the FIM constant. Based on [148], there has been a surge of interest in extending the PCRLB to more practical scenarios, e.g., to include measurement origin uncertainty [149, 150], to consider issues related to the quantization of sensor data, to compute approximated online PCRLB [151], and to derive online conditional PCRLB [152]. Subsequently, the PCRLB theory has been extended to several applications, e.g., for adaptive resource management [146], dynamic sensor selection [145], bearing-only tracking [154] and multiple target tracking [155]. As stated earlier, previous derivations of the PCRLB are limited to the centralized [146, 149, 152] and hierarchical estimation architectures [145] and only recently a suboptimal PCRLB expression [15] has been derived for the distributed architectures. In this chapter, optimal dPCRLB algorithms are derived for full-order distributed approaches, where the entire state vector is estimated locally at each observation node without resorting to a fusion centre. In the full-order dPCRLB computation, average consensus algorithms are used to distributively compute the summation terms involving local statistics such as the local FIMs. In the discussions that follow, a connected network with at least one path traversing the complete network is assumed. Also, observability over the entire network is assumed though local observability is not required. Some of the results presented in this chapter have been appeared previously in [53–55, 64]

To summarize, the chapter makes the following important contributions.

1. Exact expressions for computing the *non-conditional* (conventional) dPCRLB for full-order distributed architectures are derived. A Riccati-type recursion that sequentially determines the optimal distributed FIM from localized FIMs of the distributed estimators is derived, which is used to compute the full order dPCRLB (FO/dPCRLB).

2. As an alternative to the *non-conditional* (conventional) dPCRLB (Item 1), the *conditional* dPCRLB is proposed for full-order distributed estimation in AN/SN systems. The conventional PCRLB considers observations and state variables as random, therefore, the expectations are taken with respect to the joint probability distribution of the states and observations. As mentioned previously, the conventional PCRLB is determined primarily from the state model, observation model, and prior knowledge of the initial state of the system leading to an *offline* bound with actual observations averaged out over time. An alternative is to express the PCRLB as a function of the past history of observations, which inherently contains information of the current realization of the system state. The resulting

PCRLB is referred to as the conditional PCRLB [152], which is an *online* bound leading to a more accurate representation of the systems's performance and a better criteria for sensor-selection. Current conditional PCRLB expressions [152] are limited to centralized architectures utilizing a FC, which make them inappropriate for distributed topologies.

The rest of the chapter is organized as follows. Section 5.1 revisits old notation as well as introduces new ones and reviews the centralized PCRLB. Section 5.2 derives an expression for computing the non-conditional dPCRLB for a full-order distributed architecture. Section 5.3 extends the result to the conditional dPCRLB for a full-order distributed architecture. Section 5.4 illustrates the effectiveness of the proposed bounds through Monte Carlo simulations. Finally, Section 5.5 concludes the chapter.

5.1 PCRLB for Centralized Architecture

As previously stated in Chapter 2, $k > 1$, $\hat{\mathbf{x}}(k)$ is defined to be the estimate (i.e., the expected value) of the state vector $\mathbf{x}(k)$ at time step k based on observations taken up to k, and $\hat{\boldsymbol{P}}(k)$ is defined to be the mean squared error (covariance) associated with estimate $\hat{\mathbf{x}}(k)$, i.e.,

$$\hat{\mathbf{x}}(k) \triangleq \mathbb{E}\{\mathbf{x}(k)|\mathbf{z}(1:k)\} \tag{5.1}$$

$$\text{and} \quad \hat{\boldsymbol{P}}(k) \triangleq \mathbb{E}\{\big(\mathbf{x}(k)-\hat{\mathbf{x}}(k)\big)\big(\mathbf{x}(k)-\hat{\mathbf{x}}(k)\big)^T\}, \tag{5.2}$$

where $\mathbb{E}\{\cdot\}$ is the expectation operator and $\mathbf{z}(1:k)$ are the accumulative observations upto k. Similarly, the predicted value of the state vector and its associated error covariance are

$$\hat{\mathbf{x}}(k+1|k) \triangleq \mathbb{E}\{\mathbf{x}(k+1)|\mathbf{z}(1:k)\} \tag{5.3}$$

$$\text{and} \quad \hat{\boldsymbol{P}}(k+1|k) \triangleq \mathbb{E}\{\big(\mathbf{x}(k+1)-\hat{\mathbf{x}}(k+1|k)\big)\big(\mathbf{x}(k+1)-\hat{\mathbf{x}}(k+1|k)\big)^T\}. \tag{5.4}$$

In a distributed estimation setup, each node establishes its local estimates based on its own observations. Such a local estimate at node l, $(1 \leq l \leq N)$, is defined as

$$\hat{\mathbf{x}}^{(l)}(k) \triangleq \mathbb{E}\{\mathbf{x}^{(l)}(k)|\mathbf{z}^{(l)}(1:k)\} \tag{5.5}$$

$$\text{and} \quad \hat{\boldsymbol{P}}^{(l)}(k) \triangleq \mathbb{E}\{(\mathbf{x}^{(l)}(k)-\hat{\mathbf{x}}^{(l)}(k))(\mathbf{x}^{(l)}(k)-\hat{\mathbf{x}}^{(l)}(k))^T\}. \tag{5.6}$$

Likewise, the locally predicted state estimate at node l is

$$\hat{\mathbf{x}}^{(l)}(k+1|k) \triangleq \mathbb{E}\{\mathbf{x}^{(l)}(k+1)|\mathbf{z}^{(l)}(1:k)\} \tag{5.7}$$

$$\text{with} \quad \hat{\boldsymbol{P}}^{(l)}(k+1|k) \triangleq \mathbb{E}\{(\mathbf{x}^{(l)}(k+1)-\hat{\mathbf{x}}^{(l)}(k+1|k))(\mathbf{x}^{(l)}(k+1)-\hat{\mathbf{x}}^{(l)}(k+1|k))^T\} \tag{5.8}$$

5.1.1 PCRLB for Centralized Architecture

The PCRLB inequality [148] states that the mean square error (MSE) associated with the estimate $\hat{\mathbf{x}}(0\colon k)$ of the state vector $\mathbf{x}(0\colon k)$ is lower bounded by

$$\mathbb{E}\{(\mathbf{x}(0\colon k)-\hat{\mathbf{x}}(0\colon k))(\mathbf{x}(0\colon k)-\hat{\mathbf{x}}(0\colon k))^T\} \geq [\boldsymbol{J}(\mathbf{x}(0\colon k))]^{-1}.$$

Matrix $\boldsymbol{J}(\mathbf{x}(0\colon k))$ is referred to as the Fisher information matrix (FIM) [148], i.e., the inverse of the PCRLB, derived from the joint probability density $P(\mathbf{x}(0\colon k),\mathbf{z}(1\colon k))$. Let ∇ and Δ denote, respectively, the operators for the first and second order partial derivatives as follows

$$\begin{aligned} \nabla_{\mathbf{x}(k)} &= [\frac{\partial}{\partial X_1(k)},\ldots,\frac{\partial}{\partial X_{n_x}(k)}]^T \\ \text{and} \quad \Delta_{\mathbf{x}(k-1)}^{\mathbf{x}(k)} &= \nabla_{\mathbf{x}(k-1)}\nabla_{\mathbf{x}(k)}^T. \end{aligned}$$

A common form [148] of the FIM is defined as

$$\boldsymbol{J}(\mathbf{x}(0\colon k)) = \mathbb{E}\{-\Delta_{\mathbf{x}(0:k)}^{\mathbf{x}(0:k)}\log P(\mathbf{x}(0\colon k),\mathbf{z}(1\colon k))\}, \tag{5.9}$$

where the expectation is with respect to the joint distribution of the states and observations. An alternative expression for the FIM is derived by expressing

$$P(\mathbf{x}(0:k),\mathbf{z}(1:k)) = P(\mathbf{x}(0:k)|\mathbf{z}(1:k))P(\mathbf{z}(1:k)). \tag{5.10}$$

Since $P(\mathbf{z}(1\colon k))$ is assumed independent of the state, Eq. (5.10) leads to the following definition for the FIM.

Definition 3. *The Fisher information matrix for the state vector* $\mathbf{x}(0\colon k)$ *from time* 0 *to* k *is given by*

$$\begin{aligned} \boldsymbol{J}\big(\mathbf{x}(0:k)\big) &= \mathbb{E}\big\{-\Delta^{\mathbf{x}(0:k)}_{\mathbf{x}(0:k)} \log P(\mathbf{x}(0:k)|\mathbf{z}(1:k))\big\} \\ &= -\int \Delta^{\mathbf{x}(0:k)}_{\mathbf{x}(0:k)} \log P(\mathbf{x}(0\colon k)|\mathbf{z}(1\colon k))P(\mathbf{x}(0\colon k),\mathbf{z}(1\colon k))d\mathbf{x}, \end{aligned} \tag{5.11}$$

where the expectation is taken with respect to $P(\mathbf{x}(0\colon k),\mathbf{z}(1\colon k))$ *and the integration is multidimensional depending on the state dimensions.*

The global FIM $\boldsymbol{J}\big(\mathbf{x}(0:k)\big)$ is factorized as follows [148]

$$\boldsymbol{J}(\mathbf{x}(0:k)) \triangleq \begin{bmatrix} \mathbb{A}(k) & \mathbb{B}(k) \\ \mathbb{B}(k)^T & \mathbb{C}(k) \end{bmatrix} = \left[\begin{array}{c:c} \mathbb{E}\Big\{-\Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(0:k-1)} \log P_c(k)\Big\} & \mathbb{E}\Big\{-\Delta^{\mathbf{x}(k)}_{\mathbf{x}(0:k-1)} \log P_c(k)\Big\} \\ \hdashline \mathbb{E}\Big\{-\Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(k)} \log P_c(k)\Big\} & \mathbb{E}\Big\{-\Delta^{\mathbf{x}(k)}_{\mathbf{x}(k)} \log P_c(k)\Big\} \end{array}\right] \tag{5.12}$$

where $P_c(k) = P(\mathbf{x}(0\colon k)|\mathbf{z}(1\colon k))$. The FIM $\boldsymbol{J}(\mathbf{x}(k))$ associated with the estimate $\hat{\mathbf{x}}(k)$ is obtained by taking the inverse of $(n_x \times n_x)$ right-lower square block of $[\boldsymbol{J}(\mathbf{x}(0:k))]^{-1}$ using the following Lemma [152].

Lemma 3. *Matrix inversion Lemma:*

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{\Omega}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{\Phi}^{-1} \\ -\boldsymbol{\Phi}^{-1}\boldsymbol{B}^T\boldsymbol{A}^{-1} & \boldsymbol{\Phi}^{-1} \end{bmatrix}, \tag{5.13}$$

where subblocks $\{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}\}$ *have conformable dimensions,* $\boldsymbol{\Omega} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{B}^T$, *and* $\boldsymbol{\Phi} = \boldsymbol{C} - \boldsymbol{B}^T\boldsymbol{A}^{-1}\boldsymbol{B}$.

Based on Lemma 3, the FIM $\boldsymbol{J}(\mathbf{x}(k))$ is given by

$$\boldsymbol{J}(\mathbf{x}(k)) = \mathbb{C}(k) - \mathbb{B}(k)^T\mathbb{A}(k)^{-1}\mathbb{B}(k), \tag{5.14}$$

Proposition 1 (derived in [148]) presents the centralized sequential formulation of the FIM $\boldsymbol{J}(\mathbf{x}(k))$ that requires a central fusion centre but without the need of computing the inverse of $\boldsymbol{J}(\mathbf{x}(0\colon k))$ or inverse of other large matrices, e.g., $\mathbb{A}(k)$.

Proposition 1. *The centralized FIM* $\{\boldsymbol{J}(\mathbf{x}(k))\}$ *associated with the filtered estimate* $\hat{\mathbf{x}}(k)$ *recurses as*

$$\boldsymbol{J}\big(\mathbf{x}(k+1)\big) = \boldsymbol{D}^{22}(k) - \boldsymbol{D}^{21}(k)\big(\boldsymbol{J}\big(\mathbf{x}(k)\big) + \boldsymbol{D}^{11}(k)\big)^{-1}\boldsymbol{D}^{12}(k), \tag{5.15}$$

where

$$\boldsymbol{D}^{11}(k) = \mathbb{E}\big\{ - \Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}, \tag{5.16}$$

$$\boldsymbol{D}^{12}(k) = \big[\boldsymbol{D}^{21}(k)\big]^T = \mathbb{E}\big\{ - \Delta_{\mathbf{x}(k)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}, \tag{5.17}$$

$$\boldsymbol{D}^{22}(k) = \mathbb{E}\big\{ - \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\} + \underbrace{\mathbb{E}\big\{ - \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{z}(k+1)|\mathbf{x}(k+1)\big)\big\}}_{\boldsymbol{J}(\mathbf{z}(k+1))}. \tag{5.18}$$

The initial condition is $\boldsymbol{J}\big(\mathbf{x}(0)\big) = \mathbb{E}\{-\Delta_{\mathbf{x}(0)}^{\mathbf{x}(0)} \log P(\mathbf{x}(0))\}$.

In the following discussion, I derive a bound similar to $\boldsymbol{J}(\mathbf{x}(k+1))$ except for the state prediction estimate $\hat{\mathbf{x}}(k+1|k)$ as defined below.

Definition 4. *Term* $\boldsymbol{J}(\mathbf{x}(0\colon k+1|k))$ *denotes the FIM corresponding to the predicted estimate of* $\mathbf{x}(0 : k+1)$ *derived from the prediction density* $P(\mathbf{x}(0\colon k+1)|\mathbf{z}(1\colon k))$.

As for $\boldsymbol{J}(\mathbf{x}(k))$, the FIM $\boldsymbol{J}(\mathbf{x}(k+1|k))$ associated with the predicted estimate $\hat{\mathbf{x}}(k+1|k)$ can be computed by taking the inverse of the $(n_x \times n_x)$ right-lower block of $[\boldsymbol{J}(\mathbf{x}(0\colon k+1|k))]^{-1}$. This procedure is computationally intense. Instead, Proposition 2 derives an alternative expression for computing $\boldsymbol{J}\big(\mathbf{x}(k+1|k)\big)$ from $\boldsymbol{J}(\mathbf{x}(k))$.

Proposition 2. *The centralized FIM* $\{\boldsymbol{J}(\mathbf{x}(k{+}1|k))\}$ *for the predicted estimate* $\hat{\mathbf{x}}(k{+}1|k)$ *recurses as*

$$\boldsymbol{J}\big(\mathbf{x}(k{+}1|k)\big) = \boldsymbol{B}^{22}(k) - \boldsymbol{D}^{21}(k)\big(\boldsymbol{J}\big(\mathbf{x}(k)\big) + \boldsymbol{D}^{11}(k)\big)^{-1}\boldsymbol{D}^{12}(k), \tag{5.19}$$

where $\boldsymbol{J}\big(\mathbf{x}(k)\big)$ *is derived from Proposition 1. Terms* $\boldsymbol{D}^{11}(k)$*,* $\boldsymbol{D}^{12}(k)$*, and* $\boldsymbol{D}^{21}(k)$ *are given by Eqs. (5.16)-(5.17) and the additional term*

$$\boldsymbol{B}^{22}(k) = \mathbb{E}\big\{ - \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}. \tag{5.20}$$

The proof of Proposition 2 is included in Appendix C.1. In centralized estimation, where all raw observations are forwarded to the central processing unit (fusion centre) for processing, Proposition 1 provides a recursive procedure for updating $\boldsymbol{J}(\mathbf{x}(k))$ without the need for computing $\boldsymbol{J}(\mathbf{x}(0\colon k))$. The predicted FIM $\boldsymbol{J}(\mathbf{x}(k+1|k))$, when needed, can be obtained from $\boldsymbol{J}(\mathbf{x}(k))$ using Proposition 2. A second configuration that uses a centralized fusion centre is the hierarchical architecture where each node communicates its local estimates or other statistics based on its local observations to the fusion centre. The latter forms the global estimate and updates the global posterior density $P(\mathbf{x}(0\colon k)|\mathbf{z}(1\colon k))$. Reference [145] shows that the PCRLB equations for the centralized architecture are also valid for the hierarchical architecture. Therefore, Propositions 1 and 2 can be used for both centralized and hierarchical architectures.

The focus of this chapter is on distributed estimation, where a fusion centre is not implemented and all processing is performed locally at the nodes constituting the network. The primary motivation for this work is development of distributed PCRLB based resource management techniques to dynamically select a subset of candidate sensor nodes participating in distributed state estimation. Due to the absence of the fusion centre, such sensor selection approaches necessitate the PCRLB to be computed online in a distributed fashion as is discussed next.

5.2 dPCRLB for Full-order Distributed Estimation

The problem I want to solve is to compute the theoretical lower bound, i.e., PCRLB, on the error in the global state estimate. Below, I explain the proposed dPCRLB computation algorithm for the full-order state estimation. In Appendix E, I show that the equations used to compute the global FIM as a function of the local FIMs are similar in nature to those for reduced-order state estimation with some modifications.

5.2.1 Full-order dPCRLB (FO/dPCRLB)

This section derives the recursive expression for computing the full-order dPCRLB, i.e., expresses the global information sub-matrix, denoted by $\boldsymbol{J}_{\mathrm{FO}}\big(\mathbf{x}(k+1)\big)$, as a function of its value $\boldsymbol{J}_{\mathrm{FO}}\big(\mathbf{x}(k)\big)$ for the previous iteration, local FIMs $\boldsymbol{J}^{(l)}_{\mathrm{FO}}\big(\mathbf{x}(k+1)\big)$, and local prediction FIMs $\boldsymbol{J}^{(l)}_{\mathrm{FO}}\big(\mathbf{x}(k+1|k)\big)$, $1 \le l \le N$.

Definition 5. *Term $\boldsymbol{J}^{(l)}_{FO}\big(\mathbf{x}(0\!:k)\big)$, for $1 \le l \le N$, denotes the local FIM corresponding to the local estimate $\hat{\mathbf{x}}^{(l)}(0\!:k)$ of $\mathbf{x}(0\!:k)$ derived from the local posterior density $P(\mathbf{x}(0\!:k)|\mathbf{z}^{(l)}(1\!:k))$ for a full order local estimator defined as*

$$\boldsymbol{J}^{(l)}_{FO}\big(\mathbf{x}(0\!:k)\big) = \mathbb{E}_{P(\mathbf{x}(0:k),\mathbf{z}^{(l)}(1:k))}\big\{-\Delta^{\mathbf{x}(0:k)}_{\mathbf{x}(0:k)}\log P(\mathbf{x}(0\!:k)|\mathbf{z}^{(l)}(1\!:k))\big\}. \tag{5.21}$$

Definition 6. *Term $\boldsymbol{J}^{(l)}_{FO}\big(\mathbf{x}(0:k\!+\!1|k)\big)$ denotes the local FIM corresponding to the local prediction estimate $\hat{\mathbf{x}}^{(l)}(0\!:k\!+\!1|k)$ of $\mathbf{x}(0:k\!+\!1)$ derived from the local prediction density $P(\mathbf{x}(0:k\!+\!1)|\mathbf{z}^{(l)}(1:k))$ for a full-order local estimator defined as*

$$\boldsymbol{J}^{(l)}_{FO}\big(\mathbf{x}(0:k+1|k)\big) = \mathbb{E}_{P(\mathbf{x}(0:k+1),\mathbf{z}^{(l)}(1:k))}\big\{-\Delta^{\mathbf{x}(0:k+1)}_{\mathbf{x}(0:k+1)}\log P(\mathbf{x}(0\!:k\!+1)|\mathbf{z}^{(l)}(1\!:k))\big\}. \tag{5.22}$$

Note that the inverse of the local filtering FIM, i.e., $[\boldsymbol{J}^{(l)}_{\mathrm{FO}}(\mathbf{x}(k))]^{-1}$, is equal to the $(n_x \times n_x)$ right-lower block of $[\boldsymbol{J}^{(l)}_{\mathrm{FO}}(\mathbf{x}(0\!:k))]^{-1}$ as explained previously for $\boldsymbol{J}(\mathbf{x}(k))$ based on Lemma 1.

The expressions for recursively computing $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$ are similar in nature to Eqs. (5.15)-(5.18) except that the likelihood function $P(\mathbf{z}(k+1)|\mathbf{x}(k+1))$ originally used in $\boldsymbol{J}(\mathbf{z}(k+1)) = \mathbb{E}\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P(\mathbf{z}(k+1)|\mathbf{x}(k+1))\}$ (defined in Eq. (5.18)) is replaced by its corresponding local likelihood $P(\mathbf{z}^{(l)}(k+1)|\mathbf{x}(k+1))$, i.e., $\boldsymbol{J}(\mathbf{z}^{(l)}(k+1)) = \mathbb{E}\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P(\mathbf{z}^{(l)}(k+1)|\mathbf{x}(k+1))\}$. Similarly, computation of $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1|k))$ is also based on Proposition 7 except $\boldsymbol{J}(\mathbf{x}(k))$ gets replaced by $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$.

In deriving the optimal recursive expressions for computing the dPCRLB, another form of the local FIM (denoted by $\tilde{\boldsymbol{J}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$) associated with the local state estimate is encountered as defined below, which is derived from the local filtering distribution $P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k))$, i.e.,

$$\tilde{\boldsymbol{J}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k)) = \mathbb{E}_{P(\mathbf{x}(k),\mathbf{z}^{(l)}(1:k))}\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \log P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k))\}. \tag{5.23}$$

Similarly, the prediction FIM $\tilde{\boldsymbol{J}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1|k))$ associated with the local prediction estimate is given by

$$\tilde{\boldsymbol{J}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1|k)) = \mathbb{E}_{P(\mathbf{x}(k+1),\mathbf{z}^{(l)}(1:k))}\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \log P(\mathbf{x}(k+1)|\mathbf{z}^{(l)}(1:k))\}. \tag{5.24}$$

Difference between $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$ and $\tilde{\boldsymbol{J}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$: The localized FIM $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$ is obtained by inverting the $(n_x \times n_x)$ right lower square block of $[\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(0:k))]^{-1}$ using Eqs. (5.12)-(5.14) directly or its recursive implementation using Eq. (5.15). On the other hand, its counterpart $\tilde{\boldsymbol{J}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$ is derived directly from Eq. (5.23) by taking the expectation and Laplacian of the local conditional posterior. A way of obtaining term $\tilde{\boldsymbol{J}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$ is by re-initializing (renewing) the system prior probability density function (PDF) at time k with the posterior PDF, i.e., $P_0(\mathbf{x}(k)) = P(\mathbf{x}(k)|\mathbf{z}^{(l)}(k))$. While $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$ can be computed recursively, determining $\tilde{\boldsymbol{J}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$ is not generally straightforward [146]. For linear systems with Gaussian excitation, it has been shown [152] that the two FIMs are the same. For nonlinear systems, the two FIMs are generally different. A comparison of $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$ and $\tilde{\boldsymbol{J}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$ is difficult due to complex integral terms. Further explanation on the differences between $J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k))$ and $\tilde{J}^{(l)}_{\mathrm{FO}}(\mathbf{x}(k))$ is presented in [146, 152]. A similar difference exists between the localized predictive FIMs $J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1|k))$ derived from $J^{(l)}_{\mathrm{FO}}(\mathbf{x}(0\!:k))$ using Eq. (5.19) and $\tilde{J}^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1|k))$ obtained from Eq. (5.24).

Scenario 1 (Estimation based only on local measurements): Theorem 7 presented below provides the optimal recursive formula for computing the distributed FIM corresponding to the global estimation from the local FIMs $J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k))$ and local prediction FIMs $J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1|k))$ for Scenario 1 (Section 2.1.2.1).

Theorem 7. *The sequence $\{J_{FO}(\mathbf{x}(k))\}$ of information sub-matrices for the global estimates follows the recursion*

$$J_{FO}(\mathbf{x}(k+1)) = C^{22}_{FO}(k) - C^{21}_{FO}(k)\big(J_{FO}(\mathbf{x}(k)) + C^{11}_{FO}(k)\big)^{-1}C^{12}_{FO}(k) \tag{5.25}$$

where terms $C^{11}_{FO}(k)$, $C^{21}_{FO}(k)$, $C^{12}_{FO}(k)$ and $C^{22}_{FO}(k)$ are given by

$$\begin{aligned} C^{11}_{FO}(k) &= \mathbb{E}\big\{-\Delta^{\mathbf{x}(k)}_{\mathbf{x}(k)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}, && (5.26)\\ C^{12}_{FO}(k) &= \big[C^{21}_{FO}(k)\big]^T = \mathbb{E}\big\{-\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}, && (5.27)\\ \textit{and } C^{22}_{FO}(k) &= \sum_{l=1}^{N}\tilde{J}^{(l)}_{FO}(\mathbf{x}(k+1)) - \sum_{l=1}^{N}\tilde{J}^{(l)}_{FO}(\mathbf{x}(k+1|k)) + \mathbb{E}\big\{-\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}. && (5.28)\end{aligned}$$

In order to approximately compute the dPCRLB and specifically to compute $C^{22}_{\mathrm{FO}}(k)$, I propose to replace $\tilde{J}^{(l)}_{\mathrm{FO}}(\mathbf{x}(k))$ with $J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k))$ (and similarly $\tilde{J}^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1|k))$ with $J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1|k))$) in Eq. (5.28)), i.e.,

$$C^{22}_{\mathrm{FO}}(k) \approx \sum_{l=1}^{N} J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1)) - \sum_{l=1}^{N} J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1|k)) + \mathbb{E}\big\{-\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}. \tag{5.29}$$

Note that Eq. (5.29) is an approximation given that $J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k))$ may be different from $\tilde{J}^{(l)}_{\mathrm{FO}}(\mathbf{x}(k))$ for nonlinear systems. In our simulations for a nonlinear/Gaussian system, I illustrate through Monte

Carlo simulations that Eq. (5.29) provides reasonably accurate results. The proof of Theorem 7 for Scenario 1 is included in Appendix C.2.

Scenario 2 (Estimation based on local measurements and previous global estimate): I extend Theorem 7 to compute the global FIM as a function of the local FIMs for Scenario 2 (Section 2.1.2.1) where the local estimator at node l, for $1 \leq l \leq N$, is still restricted to local observations but additionally uses the previous estimated global state.

Corollary 1. *Theorem 7 provides the optimal expression for Scenario 2 except for* (5.28) *involving* $C^{22}_{FO}(k)$, *which changes to*

$$\begin{aligned} C^{22}_{FO}(k) &= \sum_{l=1}^{N} \mathbb{E}\{-\Delta^{\mathbf{x}(k)}_{\mathbf{x}(k)} \log P(\mathbf{x}(k+1)|\mathbf{z}(1:k), \mathbf{z}^{(l)}(k+1))\} \\ &- \sum_{l=1}^{N} \tilde{J}^{(l)}_{FO}(\mathbf{x}(k+1|k)) + \mathbb{E}\{\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)} \log P(\mathbf{x}(k+1)|\mathbf{x}(k))\} \end{aligned} \tag{5.30}$$

where the first term on the right hand side (RHS) of Eq. (5.30) *associated with the local state estimate is derived from the local filtering distribution* $P(\mathbf{x}(k)|\mathbf{z}(1:k), \mathbf{z}^{(l)}(k+1))$.

The proof of Corollary 1 is included in Appendix C.3. Eq. (5.30) can be further approximated as

$$C^{22}_{\mathrm{FO}}(k) \approx \sum_{l=1}^{N} J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1)) - N J_{\mathrm{FO}}(\mathbf{x}(k+1|k)) + \mathbb{E}\{-\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)} \log P(\mathbf{x}(k+1)|\mathbf{x}(k))\}, \tag{5.31}$$

where I use the local FIM $J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1))$ instead of the first term on the RHS of (5.30).

The following observations are made from Theorem 7.

– For updating $J_{\mathrm{FO}}(\mathbf{x}(k+1))$, reference [15] derives the following approximate expression

$$\begin{aligned} \hat{J}_{\mathrm{FO}}(\mathbf{x}(k+1)) &= \sum_{l=1}^{N} (J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1)) - J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1|k))) + \mathbb{E}\{-\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)} \log P(\mathbf{x}(k+1)|\mathbf{x}(k))\} \\ &- C^{21}_{\mathrm{FO}}(k)(J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k)) + C^{11}_{\mathrm{FO}}(k))^{-1} C^{12}_{\mathrm{FO}}(k). \end{aligned} \tag{5.32}$$

There is one notable difference between Eq. (5.32) and Theorem 7. The third term on the RHS of (5.32) is based on the previous **local** FIM $J^{(l)}_{\mathrm{FO}}(\mathbf{x}(k))$ at node l thus making it node-dependent. The corresponding term in (Eq. (5.25)) is based on the overall FIM from the previous iteration. When the PCRLB is computed in a distributed manner, Eq. (5.32) differs from one node to another. Theorem 7 is, therefore, an exact result.

– Theorem 7 is optimal but computationally more intense that the approximated Eq. (5.32), which is the price paid for increased accuracy.

– In additive Gaussian state-space models, the forcing term $\boldsymbol{\xi}(k)$ and observation noise $\boldsymbol{\zeta}^{(l)}(k)$ in Eqs. (2.16)-(2.17) are assumed to be uncorrelated and normally distributed with zero mean and covariance matrices $\boldsymbol{Q}(k)$ and $\boldsymbol{R}^{(l)}(k)$, respectively. Eqs. (5.26)-(5.29) are then reduced to

$$\begin{aligned} \boldsymbol{C}^{11}_{\mathrm{FO}}(k) &= \mathbb{E}\{[\nabla_{\mathbf{x}(k)}\boldsymbol{f}^T(\mathbf{x}(k))]\boldsymbol{Q}^{-1}(k)[\nabla_{\mathbf{x}(k)}\boldsymbol{f}^T(\mathbf{x}(k))]^T\} && (5.33)\\ \boldsymbol{C}^{12}_{\mathrm{FO}}(k) &= (\boldsymbol{C}^{21}_{\mathrm{FO}}(k))^T = -\mathbb{E}\{[\nabla_{\mathbf{x}(k)}\boldsymbol{f}^T(\mathbf{x}(k))]\boldsymbol{Q}^{-1}(k)\} && (5.34)\\ \text{and}\quad \boldsymbol{C}^{22}_{\mathrm{FO}}(k) &= \sum_{l=1}^{N}\Big(\underbrace{\boldsymbol{J}^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1)) - \boldsymbol{J}^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1|k))}_{\boldsymbol{X}^{(l)}_{c1}(0)}\Big) + \boldsymbol{Q}^{-1}(k). && (5.35) \end{aligned}$$

– Theorem 7 provides a recursive framework for computing the FO/dPCRLB. Knowing the state transition model $P(\mathbf{x}(k+1)|\mathbf{x}(k))$, Terms $\boldsymbol{C}^{11}_{\mathrm{FO}}(k+1)$ and $\boldsymbol{C}^{12}_{\mathrm{FO}}(k+1)$ can be computed locally at each node. In Section 5.2.2, I describe how $\boldsymbol{C}^{22}_{\mathrm{FO}}(k)$ is computed distributively.

– Theorem 7 computes the FO/dPCRLB with communication occurring at every observation time step. Below, I present an extension of Theorem 7 to cases where the global FO/dPCRLB is computed after every $m > 1$ iterations. This typically happens in networks with intermittent communications. The local FIM includes no communication and can be computed as soon as the local observation is made. The global FIM needs to fuse local FIMs, which in this case will be possible only when communication is restored. Assume that the global FIM $\boldsymbol{J}_{\mathrm{FO}}(\mathbf{x}(j))$ is available for iteration $j=(k+1-m)$ and the next fusion occurs at iteration $k+1$. For such a scenario, Theorem 7 is extended as

$$\boldsymbol{J}_{\text{FO}}\big(\mathbf{x}(k+1)\big) = \boldsymbol{C}_{\text{FO}}^{22}(k) - \boldsymbol{C}_{\text{FO}}^{21}(k)\big(\boldsymbol{J}_{\text{FO}}\big(\mathbf{x}(k|j)\big) + \boldsymbol{C}_{\text{FO}}^{11}(k)\big)^{-1}\boldsymbol{C}_{\text{FO}}^{12}(k), \tag{5.36}$$

where $C_{\text{FO}}^{11}(k)$ and $C_{\text{FO}}^{12}(k) = [C_{\text{FO}}^{21}(k)]^T$ are given by Eqs. (5.26) and (5.27), and

$$\boldsymbol{C}_{\text{FO}}^{22}(k) = \sum_{l=1}^{N}\Big[\boldsymbol{J}_{\text{FO}}^{(l)}(\mathbf{x}(k+1)) - \boldsymbol{J}_{\text{FO}}^{(l)}(\mathbf{x}(k+1|j))\Big] + \mathbb{E}\big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\} \tag{5.37}$$

Term $\boldsymbol{J}_{\text{FO}}(\mathbf{x}(k|j))$ is the global m-step-ahead predictive FIM. Similarly, $\boldsymbol{J}_{\text{FO}}^{(l)}(\mathbf{x}(k+1|j))$ is the local predictive FIM. For more details on predictive FIMs, please refer to [156].

– A lack of invertibility of the local FIM $\boldsymbol{J}_{\text{FO}}^{(l)}(\mathbf{x}(k))$ indicates that the states are locally unobservable. This happens if the condition number $\kappa(\boldsymbol{J}_{\text{FO}}^{(l)}(\mathbf{x}(k)))$, i.e., the common logarithm of the ratio of its largest eigenvalue $\lambda_{\text{max}}^{(l)}$ to its smallest eigenvalue $\lambda_{\text{min}}^{(l)}$, given by $\kappa(\boldsymbol{J}_{\text{FO}}^{(l)}(\mathbf{x}(k))) = \log_{10}(\lambda_{\text{max}}^{(l)}/\lambda_{\text{min}}^{(l)})$, is a large number. When the local FIM at node l is singular, the local node can not track the target on the basis of only its local observations. Therefore, it can not update its local FIM. In cases when the local FIM at node l is not invertible, the dPCRLB algorithm drops node l from the consensus step. Consensus is achieved using the remaining nodes. The local FIM $\boldsymbol{J}_{\text{FO}}^{(l)}(\mathbf{x}(k))$ at node l is then updated using the global FIM obtained from the consensus step.

– Finally, I investigate the communication overhead for the FO/dPCRLB. When average consensus is used to distributively compute the summation terms in Eq. (5.25), the communication overhead is of $\text{O}(n_x^2|\aleph^{(l)}|N_c)$ at each node, where n_x is number of states, $|\aleph^{(l)}|$ the number of nodes in the neighbourhood of node l, and N_c is the number of consensus iterations. The communication overhead for the approximate expression (Eq. (5.32)) is the same.

5.2.2 Distributed Computation of the Full-order dPCRLB

Assume submatrices $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$, $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k{+}1|k))$, and $\boldsymbol{J}_{\mathrm{FO}}(\mathbf{x}(k))$ are available from iteration k of the dPCRLB update (or via initialization). Below, I explain iteration $(k+1)$ for updating the dPCRLB.

Step 1: Node l, for $1 \le l \le N$, computes terms $\boldsymbol{C}_{\mathrm{FO}}^{11}(k)$, $\boldsymbol{C}_{\mathrm{FO}}^{21}(k)$, and $\boldsymbol{C}_{\mathrm{FO}}^{12}(k)$ using Eqs. (5.26)-(5.27). Since these terms are based on the global state mode (Eq. (2.16)), they can be computed locally at each node without requiring any communication with the neighbouring nodes.

Step 2: Compute term $\boldsymbol{C}_{\mathrm{FO}}^{22}(k)$ using (5.29). This involves the local FIMs $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1))$ and $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1|k))$ representing the bound on the local estimator at node l. Term $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1))$, for example, is computed by extending Proposition 1 to the distributed estimation model as

$$\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1)) = \left[\boldsymbol{D}_{\mathrm{FO}}^{22}(k)\right]^{(l)} - \left[\boldsymbol{D}_{\mathrm{FO}}^{21}(k)\right]^{(l)} \left(\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k)) + \left[\boldsymbol{D}_{\mathrm{FO}}^{11}(k)\right]^{(l)}\right)^{-1} \left[\boldsymbol{D}_{\mathrm{FO}}^{12}(k)\right]^{(l)}, \tag{5.38}$$

where

$$\left[\boldsymbol{D}_{\mathrm{FO}}^{11}(k)\right]^{(l)} = \mathbb{E}\left\{ -\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\right\} \tag{5.39}$$

$$\left[\boldsymbol{D}_{\mathrm{FO}}^{12}(k)\right]^{(l)} = \left(\left[\boldsymbol{D}_{\mathrm{FO}}^{21}(k)\right]^{(l)}\right)^{T} = \mathbb{E}\left\{ -\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\right\} \tag{5.40}$$

and

$$\left[\boldsymbol{D}_{\mathrm{FO}}^{22}(k)\right]^{(l)} = \mathbb{E}\left\{ -\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\right\} + \mathbb{E}\left\{ -\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{z}^{(l)}(k{+}1)|\mathbf{x}(k{+}1)\big)\right\}. \tag{5.41}$$

Scenario 2 replaces Eq. (5.38) with

$$\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1)) = \left[\boldsymbol{D}_{\mathrm{FO}}^{22}(k)\right]^{(l)} - \left[\boldsymbol{D}_{\mathrm{FO}}^{21}(k)\right]^{(l)} \left(\boldsymbol{J}_{\mathrm{FO}}(\mathbf{x}(k)) + \left[\boldsymbol{D}_{\mathrm{FO}}^{11}(k)\right]^{(l)}\right)^{-1} \left[\boldsymbol{D}_{\mathrm{FO}}^{12}(k)\right]^{(l)}, \tag{5.42}$$

with the local FIM at iteration k on the RHS of Eq. (5.38) replaced by the global FIM at iteration k. Note that Eqs. (5.38)-(5.41) only require information available locally at each node.

The expression for computing $J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k{+}1|k))$ is based on Proposition 2 expanded as follows

$$\begin{aligned} J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1|k)) &= [B_{\mathrm{FO}}^{22}(k)]^{(l)} - [D_{\mathrm{FO}}^{21}(k)]^{(l)}\big(J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k)) + [D_{\mathrm{FO}}^{11}(k)]^{(l)}\big)^{-1}[D_{\mathrm{FO}}^{12}(k)]^{(l)} && (5.43)\\ \text{where } [B_{\mathrm{FO}}^{22}(k)]^{(l)} &= \mathbb{E}\big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}. && (5.44)\end{aligned}$$

Across the network, $J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k{+}1))$ and $J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k{+}1|k))$ will have different values. Having computed $J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k{+}1))$ and $J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k{+}1|k))$, term $\mathbf{X}_{c1}^{(l)}(0)$ in Eq. (5.35) (summation terms $\Sigma J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k{+}1))$ and $\Sigma J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k{+}1|k))$) can be computed using an average consensus algorithm [32] in a distributed fashion as explained next[7]. Node l, for $1 \le l \le N$, initializes its consensus state as

$$\mathbf{X}_{c1}^{(l)}(0) = J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1)) - J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1|k)) \tag{5.45}$$

and continues to iterate

$$\mathbf{X}_{c1}^{(l)}(t+1) = \mathbf{X}_{c1}^{(l)}(t) + \epsilon \sum_{j \in \aleph^{(l)}} \Big(\mathbf{X}_{c1}^{(j)}(t) - \mathbf{X}_{c1}^{(l)}(t)\Big) \tag{5.46}$$

till convergence to

$$\mathbf{X}_{c1}(\infty) = \frac{1}{N}\sum_{l=1}^{N} \Big(J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1)) - J_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1|k))\Big) \tag{5.47}$$

is achieved. In Eq. (5.46), ϵ is a small value satisfying $\epsilon \in (0, \frac{1}{\Delta_G}]$ and $\Delta_{\mathcal{G}_f} = \max_l D^{(l)}$ is the maximum degree for fusion graph $\mathcal{G}_f$ and $D^{(l)}$ is the number of neighboring nodes for fusion node l. Once the consensus converges, each fusion node substitutes the result of Eq. (5.47) in Eq. (5.29) to compute $C_{\mathrm{FO}}^{22}(k)$. Note that the consensus approach in Eq. (5.46) is a distributed algorithm where each node communicates only with its neighboring nodes. The final expectation term in (5.29) depends only on the state model and can be derived locally.

Step 3: Theorem 7 is now used to compute the dPCRLB, which is the same at all nodes.

[7]The derivation of a summation term using average consensus algorithm requires information on the total number N of active nodes. Since the prime motivation for computing the dPCRLB is sensor selection, therefore, the number of active nodes should be known beforehand. I note that when N is unknown, an additional average consensus step with the value of one node set to 1 and others to 0 can instead be used to determine the number of nodes in the network. Average consensus will converge to $1/N$ and its reciprocal will provide the value of N.

Note that only Step 2 requires cooperation among the neighbouring nodes achieved using a consensus algorithm across the network, while Steps 1 and 3 can be computed locally at each processing node. Finally, I note that when the dPCRLB is computed using average consensus algorithms with (i) the network being connected; (ii) fast connectivity allowing for consensus to be achieved between two consecutive observations, the proposed dPCRLB coincides with its centralized value. This is in fact exploited by the dPCRLB algorithm. I note that, assumptions (i) and (ii) are commonly used in the consensus-based literature related to distributed implementation of the particle filter and the Kalman filter [1,32]. Such assumptions are reasonable in applications where compared to sensing communication is relatively inexpensive, e.g., in rendezvous control or coordination of mobile sensors.

5.2.3 Particle Filter Realization for full-order dPCRLB

In nonlinear dynamical systems, direct computation of $\{C^{11}_{\text{FO}}(k), C^{12}_{\text{FO}}(k), C^{21}_{\text{FO}}(k), C^{22}_{\text{FO}}(k)\}$ as well as localized terms $\{[D^{11}_{\text{FO}}(k)]^{(l)}, [D^{12}_{\text{FO}}(k)]^{(l)}, [D^{21}_{\text{FO}}(k)]^{(l)}, [D^{22}_{\text{FO}}(k)]^{(l)}\}$ is difficult due to the involvement of nonlinear terms within the expectation operator [157]. Sequential Monte Carlo methods (such as the particle filter [158, 159]) are usually used to compute these terms. For completeness, the following section explains how the expectation terms in the FO/dPCRLB are computed using particle filters specifically in terms of the CF/DPF proposed in Chapter 4. In the CF/DPF, an additional higher order particle filter (referred to as the global/fusion/consensus filter) is introduced that assimilates the local statistics from these local filters into global statistics[8]. In the sequel, $\{\mathbb{X}_i^{(l,\text{FF})}(k), W_i^{(l,\text{FF})}\}$ refers to the global particle set computed at node l, for

[8]Other distributed implementations of the particle filter [7,19,20,24] do not maintain separate local and global particle sets. Only one set of particles is maintained. Information or statistics from local particle sets is then fused in a distributed way to update the particle set to better represent the global posterior. The proposed distributed computation of dPCRLB is also applicable in such cases as long as the global particle sets are available at each node.

$1 \leq l \leq N_s$, using the higher order global filter. In a general case, the global particle set and associated weights can be used to implement Steps 1-3 of the full order dPCRLB computational algorithm described in Section 5.2.2. For the sake of completeness, I summarize Eqs. (5.26)-(5.28) in terms of the global particle set $\{\mathbb{X}_i^{(l,\mathrm{FF})}(k), W_i^{(l,\mathrm{FF})}\}$ of the distributed particle filter followed by their equivalent representation for the case where the forcing terms are additive Gaussian. Representing (5.26) in terms of the global particle filter set, I get

$$\widehat{C}_{\mathrm{FO}}^{11}(k) \approx -\sum_{i=1}^{N_p} W_i^{(l,\mathrm{FF})}(k)\Big(\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \log P(\mathbf{x}(k{+}1)|\mathbf{x}(k))]\Big)\Big|_{\mathbf{x}(k)=\mathbb{X}_i^{(l,\mathrm{FF})}(k)}. \tag{5.48}$$

For the additive Gaussian forcing terms, Eq. (5.48) simplifies to

$$\widehat{C}_{\mathrm{FO}}^{11}(k) \approx \sum_{i=1}^{N_p} W_i^{(l,\mathrm{FF})}(k)\Big([\nabla_{\mathbf{x}(k)}\boldsymbol{f}^T(\mathbf{x}(k))]\boldsymbol{Q}^{-1}(k)[\nabla_{\mathbf{x}(k)}\boldsymbol{f}^T(\mathbf{x}(k))]^T\Big)\Big|_{\mathbf{x}(k)=\mathbb{X}_i^{(l,\mathrm{FF})}(k)}. \tag{5.49}$$

Similarly, Eq. (5.27) in terms of the global particle set is

$$\widehat{C}_{\mathrm{FO}}^{12}(k) = \Big[\widehat{C}_{\mathrm{FO}}^{21}(k)\Big]^T \approx -\sum_{i=1}^{N_p} W_i^{(l,\mathrm{FF})}(k)\Big(\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k+1)} \log P(\mathbf{x}(k{+}1)|\mathbf{x}(k))\Big)\Big|_{\mathbf{x}(k)=\mathbb{X}_i^{(l,\mathrm{FF})}(k)}, \tag{5.50}$$

which for the additive Gaussian forcing terms simplifies to

$$\widehat{C}_{\mathrm{FO}}^{12}(k) = \Big[\widehat{C}_{\mathrm{FO}}^{21}(k)\Big]^T \approx -\sum_{i=1}^{N_p} W_i^{(l,\mathrm{FF})}(k)\Big([\nabla_{\mathbf{x}(k)}\boldsymbol{f}^T(\mathbf{x}(k))]\boldsymbol{Q}^{-1}(k)\Big)\Big|_{\mathbf{x}(k)=\mathbb{X}_i^{(l,\mathrm{FF})}(k)}. \tag{5.51}$$

Term $\boldsymbol{C}_{\mathrm{FO}}^{22}(k)$ in Eq. (5.29) requires participation of all the local fusion nodes to compute the submatrices $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1))$ and $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1|k))$ of the local FIM. Submatrix $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1))$ is computed based on Eq. (5.38) with terms $[\boldsymbol{D}_{\mathrm{FO}}{}^{11}(k)]^{(l)}$, $[\boldsymbol{D}_{\mathrm{FO}}{}^{12}(k)]^{(l)}$, and $[\boldsymbol{D}_{\mathrm{FO}}{}^{22}(k)]^{(l)}$ having particle filter representations similar to the ones expressed for Eqs. (5.26)-(5.27). Below, I write these terms for the Gaussian case

$$[\widehat{\boldsymbol{D}}_{\mathrm{FO}}^{11}(k)]^{(l)} \approx \sum_{i=1}^{N_p} W_i^{(l,\mathrm{LF})}(k)\Big([\nabla_{\mathbf{x}(k)}\boldsymbol{f}^T(\mathbf{x}(k))]\boldsymbol{Q}^{-1}(k)[\nabla_{\mathbf{x}(k)}\boldsymbol{f}^T(\mathbf{x}(k))]^T\Big)\Big|_{\mathbf{x}(k)=\mathbb{X}_i^{(l,\mathrm{LF})}(k)} \tag{5.52}$$

$$[\widehat{\boldsymbol{D}}_{\mathrm{FO}}^{12}(k)]^{(l)} = \Big[[\widehat{\boldsymbol{D}}_{\mathrm{FO}}^{21}(k)]^{(l)}\Big]^T \approx \sum_{i=1}^{N_p} W_i^{(l,\mathrm{LF})}(k)\Big([\nabla_{\mathbf{x}(k)}\boldsymbol{f}^T(\mathbf{x}(k))]\boldsymbol{Q}^{-1}(k)\Big)\Big|_{\mathbf{x}(k)=\mathbb{X}_i^{(l,\mathrm{LF})}(k)} \tag{5.53}$$

and

$$\left[\widehat{\boldsymbol{D}}^{22}_{\mathrm{FO}}(k)\right]^{(l)} \approx \boldsymbol{Q}^{-1}(k) + \sum_{i=1}^{N_p} W_i^{(l,\mathrm{LF})}(k)\Big(\left[\nabla_{\mathbf{x}(k+1)}\boldsymbol{g}^{(l)T}(k+1)\right]\boldsymbol{R}^{-1}(k+1)\left[\nabla_{\mathbf{x}(k+1)}\boldsymbol{g}^{(l)T}(k+1)\right]^T\Big)\Big|_{\substack{\mathbf{x}(k)= \\ \mathbb{X}_i^{(l,\mathrm{LF})}(k+1|k)}} \tag{5.54}$$

where particles $\mathbb{X}_i^{(l,\mathrm{LF})}(k+1|k)$ are computed by propagating particles $\mathbb{X}_i^{(l,\mathrm{LF})}(k)$ through the transitional density $P(\mathbf{x}(k+1)|\mathbf{x}(k))$ obtained from the state equation (Eq. (2.16)). Note that the required terms in Eqs. (5.52)-(5.54) are computed based on the available particles for iteration k. Eqs. (5.43)-(5.44) are then used to compute $\boldsymbol{J}^{(l)}_{\mathrm{FO}}(\mathbf{x}(k+1|k))$.

The aforementioned procedure using particles and weights associated to the distributed particle filter can readily be extended to non-Gaussian forcing terms.

5.3 Conditional Full-order dPCRLB

In the previous section (Section 5.2), I derived expressions for computing the *non-conditional* dPCRLB distributively for full-order state estimation. In this section, I extend my non-conditional dPCRLB framework to conditional dPCRLB for full-order estimation. Compared to the non-conditional PCRLB, the conditional PCRLB is a function of the past history of observations made and, therefore, a more accurate representation of the estimator's performance and, consequently, a better criteria for sensor selection. Previous algorithms to compute the conditional PCRLB are limited to centralized architectures, which involve a fusion centre, thus making them unsuitable for decentralized topologies. The section addresses this gap. Extending the *non-conditional* dPCRLB to *conditional* dPCRLB is challenging due to the following issues:

1. The underlying expectations in the conditional dPCRLB are with respect to the conditional posterior, hence, the Chong-Mori-Chang theorem can not be used directly. A new factorization expression for the conditional posterior is required.

2. The recursive expressions for the conditional Fisher information matrix (FIM), i.e., inverse of the PCRLB, utilize an auxiliary FIM corresponding to the previous iteration (instead of its own previous value), therefore, distributed expressions for computing the auxiliary FIM are now needed.

3. In addition, recursive expressions for computing the predictive conditional PCRLB from the auxiliary FIM are required.

I start by introducing the centralized conditional PCRLB in the next sub-section.

5.3.1 Centralized Conditional PCRLB

Before introducing the conditional PCRLB, I define first the auxiliary FIM which is constructed by performing the expectation in Eq. (5.9) with respect to the posterior distribution $P(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k))$ leading to the following definition

$$\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(0\!:\!k)) \triangleq \mathbb{E}_{P(\mathbf{x}(0:k)|\mathbf{z}(1:k))}\big\{-\Delta_{\mathbf{x}(0:k)}^{\mathbf{x}(0:k)} \log P(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k))\big\}. \tag{5.55}$$

Reference [152] has derived recursive expressions for computing $\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(k))$ (the inverse of $(n_x \times n_x)$ right-lower square block of the inverse of $\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(0:k))$). Similar to $\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(0:k))$ (and $\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(k))$) the predictive auxiliary FIM $\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(0\!:\!k|k\!-\!1))$ is defined as

$$\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(0\!:\!k|k\!-\!1)) \triangleq \mathbb{E}_{P(\mathbf{x}(0:k)|\mathbf{z}(1:k-1))}\big\{-\Delta_{\mathbf{x}(0:k)}^{\mathbf{x}(0:k)} \log P(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k\!-\!1))\big\}, \tag{5.56}$$

My scheme extends [152] to distributed topologies.

The conditional PCRLB provides a bound on the performance of estimating $\mathbf{x}(0\!:\!k)$ given that the past observations $\mathbf{z}(1\!:\!k\!-\!1)$ are known [152]. The *conditional* MSE in the estimate $\hat{\mathbf{x}}(0:k)$ of the state vector $\mathbf{x}(0\!:\!k)$ is lower bounded by

$$\boldsymbol{I}(\mathbf{x}(0\!:\!k)) \triangleq \mathbb{E}_{P(\mathbf{x}(0:k),\mathbf{z}(k)|\mathbf{z}(1:k-1))}\big\{-\Delta_{\mathbf{x}(0:k)}^{\mathbf{x}(0:k)} \log P(\mathbf{x}(0:k),\mathbf{z}(k)|\mathbf{z}(1:k-1))\big\}, \tag{5.57}$$

where $P_c(k) \triangleq P(\mathbf{x}(0\!:\!k), \mathbf{z}(k)|\mathbf{z}(1:k-1))$. The conditional FIM $\boldsymbol{L}(k)$ is defined the inverse of the $(n_x \times n_x)$ right-lower block of $[\boldsymbol{I}(\mathbf{x}(0\!:\!k))]^{-1}$. A centralized recursive expression for updating $\boldsymbol{L}(\mathbf{x}(k))$ is derived in Reference [152]. For deriving the conditional dPCRLB, I need recursive expressions for computing the *predictive* conditional PCRLB defined as

$$\boldsymbol{I}(0:k+1|k) \triangleq \mathbb{E}_{P(\mathbf{x}(0:k+1)|\mathbf{z}(1:k))}\big\{-\Delta_{\mathbf{x}(0:k+1)}^{\mathbf{x}(0:k+1)} \log P(\mathbf{x}(0\!:\!k{+}1)|\mathbf{z}(1\!:\!k))\big\}. \tag{5.58}$$

Term $\boldsymbol{L}(\mathbf{x}(k{+}1|k))$ is defined as the inverse of the $(n_x \times n_x)$ right-lower block of $[\boldsymbol{I}(\mathbf{x}(0\!:\!k{+}1|k))]^{-1}$. Using the factorization

$$P(\mathbf{x}(0:k+1)|\mathbf{z}(1:k)) = P(\mathbf{x}(k+1)|\mathbf{x}(k))P(\mathbf{x}(1:k)|\mathbf{z}(1:k)),$$

The following Lemma 4 recursively computes $\boldsymbol{L}(\mathbf{x}(k{+}1|k))$ from $\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(k))$.

Lemma 4. *The predicted conditional FIM $\{\boldsymbol{L}(\mathbf{x}(k{+}1|k))\}$ recurses as follows*

$$\boldsymbol{L}(\mathbf{x}(k+1|k)) \;=\; \boldsymbol{B}_p^{22}(k) - \boldsymbol{B}^{21}(k)\big(\boldsymbol{J}_{AUX}(\mathbf{x}(k)) + \boldsymbol{B}^{11}(k)\big)^{-1}\boldsymbol{B}^{12}(k), \tag{5.59}$$

where

$$\begin{aligned} \boldsymbol{B}^{11}(k) &= \mathbb{E}\big\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}, && (5.60)\\ \boldsymbol{B}^{12}(k) &= [\boldsymbol{B}^{21}(k)]^T = \mathbb{E}\big\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}, && (5.61)\\ \textit{and} \quad \boldsymbol{B}_p^{22}(k) &= \mathbb{E}_{P_p(k+1)}\big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}. && (5.62) \end{aligned}$$

Next, I compute the conditional dPCRLB distributively.

5.3.2 Distributed Conditional dPCRLB

The section computes the global conditional FIM from the local conditional FIMs, which are defined below.

Definition 7. *The local conditional FIM* $\boldsymbol{I}_{FO}^{(l)}(0\!:\!k\!+\!1)$ *corresponding to the local estimate* $\hat{\mathbf{x}}^{(l)}(0\!:\!k\!+\!1)$*, for* $(1 \le l \le N)$*, is defined as follows*

$$\boldsymbol{I}_{FO}^{(l)}(\mathbf{x}(0:k\!+\!1)) \triangleq \mathbb{E}_{P(\mathbf{x}(0:k+1),\mathbf{z}^{(l)}(k+1)|\mathbf{z}^{(l)}(1:k))}\big\{-\Delta_{\mathbf{x}(0:k+1)}^{\mathbf{x}(0:k+1)} \log P(\mathbf{x}(0\!:\!k\!+\!1), \mathbf{z}^{(l)}(k\!+\!1)|\mathbf{z}^{(l)}(1\!:\!k))\big\}, \tag{5.63}$$

The local bound $\boldsymbol{L}_{\mathrm{FO}}^{(l)}(k\!+\!1)$ on $\mathbf{x}^{(l)}(k\!+\!1|k\!+\!1)$, is given by the inverse of the $(n_x \times n_x)$ right-lower block of $[\boldsymbol{I}_{\mathrm{FO}}^{(l)}(\mathbf{x}(0\!:\!k\!+\!1))]^{-1}$.

Definition 8. *The local predictive conditional FIM* $\boldsymbol{I}_{FO}^{(l)}(\mathbf{x}(0\!:\!k\!+\!1|k))$ *is defined as follows*

$$\boldsymbol{I}_{FO}^{(l)}(\mathbf{x}(0:k+1|k)) \triangleq \mathbb{E}_{P(\mathbf{x}(0:k+1)|\mathbf{z}^{(l)}(1:k))}\big\{-\Delta_{\mathbf{x}(0:k+1)}^{\mathbf{x}(0:k+1)} \log P(\mathbf{x}(0\!:\!k+1)|\mathbf{z}^{(l)}(1\!:\!k))\big\}, \tag{5.64}$$

The local bound $\boldsymbol{L}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k+1|k))$ on $\mathbf{x}^{(l)}(k+1|k)$ is given by the inverse of the $(n_x \times n_x)$ right-lower block of $[\boldsymbol{I}_{\mathrm{FO}}^{(l)}(\mathbf{x}(0:k\!+\!1|k))]^{-1}$. Note that centralized bound [152] can be used to compute both $\boldsymbol{L}_{\mathrm{FO}}^{(l)}(k\!+\!1)$ and $\boldsymbol{L}_{\mathrm{FO}}^{(l)}(k\!+\!1|k)$ with relevant local distributions replacing the global ones. The local auxiliary FIMs $[\boldsymbol{J}_{\mathrm{FO,AUX}}(\mathbf{x}(k))]^{(l)}$ and $[\boldsymbol{J}_{\mathrm{FO,AUX}}(\mathbf{x}(k|k\!-\!1))]^{(l)}$ are derived from $[\boldsymbol{J}_{\mathrm{FO,AUX}}(\mathbf{x}(0\!:\!k))]^{(l)}$ and $[\boldsymbol{J}_{\mathrm{FO,AUX}}(\mathbf{x}(0\!:\!k|k\!-\!1))]^{(l)}$, which have definitions similar to Eqs. (5.55) and (5.56) except that the local distributions are used. Another format for the local FIMs is

$$\begin{aligned}
\widetilde{\boldsymbol{L}}_{\mathrm{FO}}^{(l)}(k+1) &= \mathbb{E}\big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1), \mathbf{z}^{(l)}(k+1)|\mathbf{z}^{(l)}(1:k)\big)\big\} && (5.65)\\
\widetilde{\boldsymbol{L}}_{\mathrm{FO}}^{(l)}(k+1|k) &= \mathbb{E}\big\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \log P\big(\mathbf{x}(k+1)|\mathbf{z}^{(l)}(1:k)\big)\big\}. && (5.66)
\end{aligned}$$

where the expectations are with respect to $P(\mathbf{x}(k+1), \mathbf{z}^{(l)}(k+1)|\mathbf{z}^{(l)}(1:k))$. It can be shown that in Gaussian linear systems, $\widetilde{\boldsymbol{L}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k\!+\!1))$ and $\boldsymbol{L}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k\!+\!1))$ (similarly $\widetilde{\boldsymbol{L}}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k\!+\!1|k))$ and $\boldsymbol{L}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k\!+\!1|k))$) are equivalent.

Now that I have defined the local conditional FIMs, Theorem 8 provides the optimal recursive formula for computing the overall conditional FIM as a function of these local terms.

Theorem 8. *The sequence $\{\boldsymbol{L}_{FO}(\mathbf{x}(k{+}1))(\mathbf{x}(k))\}$ of the global information sub-matrices follows the recursion*

$$\boldsymbol{L}_{FO}(\mathbf{x}(k{+}1)) = \boldsymbol{C}^{22}_{FO}(k) - \boldsymbol{C}^{21}_{FO}(k)\Big(\boldsymbol{J}_{FO,AUX}(\mathbf{x}(k)) + \boldsymbol{C}^{11}_{FO}(k)\Big)^{-1}\boldsymbol{C}^{12}_{FO}(k) \tag{5.67}$$

$$\boldsymbol{C}^{11}_{FO}(k) = \mathbb{E}_{P_c(k+1)}\{-\Delta^{\mathbf{x}(k)}_{\mathbf{x}(k)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\}, \tag{5.68}$$

$$\boldsymbol{C}^{12}_{FO}(k) = \mathbb{E}_{P_c(k+1)}\{-\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\}, \tag{5.69}$$

and

$$\boldsymbol{C}^{22}_{FO}(k) = \sum_{l=1}^{N}\widetilde{\boldsymbol{L}}^{(l)}_{FO}(\mathbf{x}(k{+}1)) - \sum_{l=1}^{N}\widetilde{\boldsymbol{L}}^{(l)}_{FO}(\mathbf{x}(k{+}1|k)) + \mathbb{E}_{P_c(k+1)}\{-\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\}. \tag{5.70}$$

Theorem 8 proposed for computing the conditional dPCRLB is similar in structure to the recursive expression for computing the conventional dPCRLB derived in Section 5.2. with two differences: (i) The local conditional FIMs ($\boldsymbol{L}^{(l)}_{\mathrm{FO}}(\mathbf{x}(k{+}1))$ and $\boldsymbol{L}^{(l)}_{\mathrm{FO}}(\mathbf{x}(k{+}1|k))$) are used instead of their non-conditional counterparts, and; (ii) The global FIM for previous time $\boldsymbol{J}_{\mathrm{FO}}(\mathbf{x}(k))$ is replaced by the global auxiliary FIM $\boldsymbol{J}_{\mathrm{FO,AUX}}(\mathbf{x}(k))$.

In order to compute the conditional dPCRLB, term $\widetilde{\boldsymbol{L}}^{(l)}_{\mathrm{FO}}(k{+}1)$ is replaced with $\boldsymbol{L}^{(l)}_{\mathrm{FO}}(k{+}1)$ and similarly $\widetilde{\boldsymbol{L}}^{(l)}_{\mathrm{FO}}(k{+}1|k)$ is replaced with $\boldsymbol{L}^{(l)}_{\mathrm{FO}}(k{+}1|k)$. Later, I derive distributed recursive expression for computing $\boldsymbol{J}_{\mathrm{FO,AUX}}(\mathbf{x}(k))$. Theorem 8 is proved by extending Chong-Mori-Chang track-fusion theorem [127] to conditional posterior as follows.

Lemma 5. *Assuming that the observations conditioned on the state variables are independent, the global posterior for a N-sensor network is factorized as follows*

$$P(\mathbf{x}(0{:}k{+}1),\mathbf{z}(k{+}1)|\mathbf{z}(1{:}k)) \propto \frac{\prod_{l=1}^{N}P\big(\mathbf{x}(k{+}1),\mathbf{z}^{(l)}(k{+}1)|\mathbf{z}^{(l)}(1{:}k)\big)\prod_{l=1}^{N}P\big(\mathbf{x}(k)|\mathbf{z}^{(l)}(1{:}k)\big)}{\prod_{l=1}^{N}P\big(\mathbf{x}(k{+}1)|\mathbf{z}^{(l)}(1{:}k)\big)\prod_{l=1}^{N}P\big(\mathbf{x}(k)|\mathbf{z}^{(l)}(1{:}k{-}1)\big)}$$
$$\times\, P\big(\mathbf{x}(k{+}1)|\mathbf{x}(k)\big)P\big(\mathbf{x}(k)|\mathbf{x}(k{-}1)\big)P\big(\mathbf{x}(0{:}k{-}1)|\mathbf{z}(1{:}k{-}1)\big). \tag{5.71}$$

The proof of Lemma 9 is provided in Appendix C.4 followed by proof of Theorem 8 in Appendix C.5.

In general, there is no recursive method to calculate $\boldsymbol{J}_{\text{FO,AUX}}(\mathbf{x}(k))$. An approximated centralized recursive expression is proposed in [152]. Next, Proposition 3 presents a decentralized recursive expression for computing $\boldsymbol{J}_{\text{FO,AUX}}(\mathbf{x}(k))$ using the approximation stated in [152].

Proposition 3. *The global sequence* $\{\boldsymbol{J}_{FO,AUX}(\mathbf{x}(k))\}$ *of information sub-matrices follows the approximated recursion, i.e.,*

$$\boldsymbol{J}_{FO,AUX}(\mathbf{x}(k)) \approx \boldsymbol{M}_{FO}^{22}(k-1) - \boldsymbol{M}_{FO}^{21}(k-1)\Big(\boldsymbol{J}_{FO,AUX}(k-1)+\boldsymbol{M}_{FO}^{11}(k-1)\Big)^{-1}\boldsymbol{M}_{FO}^{12}(k-1) \quad (5.72)$$

where

$$\boldsymbol{M}_{FO}^{11}(k-1) = \mathbb{E}_{P_a(k)}\{-\Delta_{\mathbf{x}(k-1)}^{\mathbf{x}(k-1)}\log P(\mathbf{x}(k)|\mathbf{x}(k-1))\}, \quad (5.73)$$

$$\boldsymbol{M}_{FO}^{12}(k-1) = \mathbb{E}_{P_a(k)}\{-\Delta_{\mathbf{x}(k-1)}^{\mathbf{x}(k)}\log P(\mathbf{x}(k)|\mathbf{x}(k-1))\}, \quad (5.74)$$

$$\begin{aligned}\boldsymbol{M}_{FO}^{22}(k-1) &= \sum_{l=1}^{N}[\boldsymbol{J}_{AUX}(\mathbf{x}(k))]^{(l)} - \sum_{l=1}^{N}[\boldsymbol{J}_{AUX}(\mathbf{x}(k|k-1))]^{(l)} \\ &+ \mathbb{E}_{P_a(k)}\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)}\log P(\mathbf{x}(k)|\mathbf{x}(k-1))\}. \end{aligned}\quad (5.75)$$

The proof of Proposition 3 is similar to that for the non-conditional dPCRLB.

5.3.3 Practical Application of the Conditional dPCRLB

Recent advances in sensor technology allow deployment of a large number of sensor nodes. Limitations in power, frequency, and bandwidth restrict the maximum number of active sensors with only an active subset participating in the estimation process at each iteration. For such activation decisions, the PCRLB has been utilized as an effective criteria [15, 67, 146, 152], since it can be computed predictively and is independent of the estimation mechanism. The conditional dPCRLB expressions proposed in the thesis are derived in this context. Two different types of nodes are considered [15]: (i) *Sensor nodes*: with limited power used only to record measurements, and; (ii) *Processing nodes*: responsible for sensor-selection within their neighbourhoods and for performing decentralized estimation. Below, I consider two different dPCRLB-based sensor-selection scenarios. Case 1 [67] is near-optimal but requires high communication overhead. By computing the dPCRLB within local neighbourhoods, Case 2 [15] does not require consensus and has a reduced overhead.

Case 1: The conditional dPCRLB (Theorem 1) is computed over the entire network and used for sensor-selection. The global submatrix $\boldsymbol{J}_{\text{FO,AUX}}(k)$ and local submatrices $[\boldsymbol{J}_{\text{AUX}}(\mathbf{x}(k))]^{(l)}$ are assumed available from iteration k at node l. Iteration $(k{+}1)$ for computing conditional dPCRLB is as follows.

Step 1: Compute terms $\boldsymbol{C}^{11}_{\text{FO}}(k)$, $\boldsymbol{C}^{21}_{\text{FO}}(k)$, and $\boldsymbol{C}^{12}_{\text{FO}}(k)$ using (5.68) and (5.69), and terms $\boldsymbol{M}^{11}_{\text{FO}}(k)$, $\boldsymbol{M}^{21}_{\text{FO}}(k)$, and $\boldsymbol{M}^{12}_{\text{FO}}(k)$ using (5.73) and (5.74). Although these terms are global, they are based on the state model and computed locally.

Step 2: Compute the local FIMs $\boldsymbol{J}^{(l)}_{\text{FO}}(k{+}1)$ and $\boldsymbol{J}^{(l)}_{\text{FO}}(k{+}1|k)$ and local auxiliary FIMs $[\boldsymbol{J}_{\text{FO,AUX}}(k{+}1)]^{(l)}$ and $[\boldsymbol{J}_{\text{FO,AUX}}(k{+}1|k)]^{(l)}$ as explained in Section 5.3.2.

Step 3: Compute $\boldsymbol{C}^{22}_{\text{FO}}(k)$ using (5.70). Term $\sum_{l=1}^{N}\{\boldsymbol{J}^{(l)}_{\text{FO}}(\mathbf{x}(k+1)) - \boldsymbol{J}^{(l)}_{\text{FO}}(\mathbf{x}(k+1|k))\}$ is computed distributively across the network using consensus. Similarly, $\boldsymbol{M}^{22}_{\text{FO}}(k)$ in (5.75) includes a summation term that is also requires consensus.

Step 4: Theorem 8 computes the conditional dPCRLB. Likewise, for next iteration, Proposition 3 computes $\boldsymbol{J}_{\text{FO,AUX}}(k+1)$.

Under Case 1, Step 3 involves communication overhead. If average consensus is used to distributively compute the summation terms, the communication overhead at each processing node is of $\text{O}(n_x^2|\aleph^{(l)}|N_c)$, where n_x is number of states, $|\aleph^{(l)}|$ the number of nodes in the neighbourhood of node l, and N_c is the number of consensus iterations. In decentralized sensor-selection, this overhead is restricted to the processing nodes.

Case 2: fuses local conditional PCRLBs within local neighbourhoods [15] for sensor-selection. Consensus is not needed that reduces overhead. Steps 1 and 2 are the same as in Case 1.

Step 3: Processing node l computes $\sum_{\aleph^{(l)}}\{\boldsymbol{J}_{\text{FO}}^{(l)}(\mathbf{x}(k+1))-\boldsymbol{J}_{\text{FO}}^{(l)}(\mathbf{x}(k+1|k))\}$ over local neighbourhoods $\aleph^{(l)}$.

Step 4: Theorem 8 is used at processing node l to compute the conditional dPCRLB within local neighbourhoods $\aleph^{(l)}$. Proposition 3 computes $\boldsymbol{J}_{\text{FO,AUX}}(k+1)$ but within local neighbourhoods.

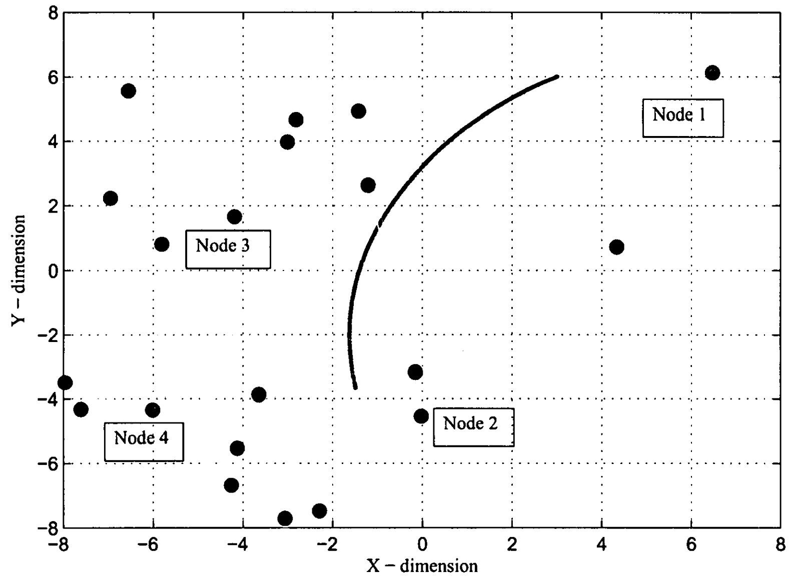
In Case 2, the communication overhead at each processing node is of $\mathrm{O}(n_x^2|\aleph^{(l)}|)$, an improvement of a factor of N_c over Case 1. Instead of computing the dPCRLB over the entire network that leads to a high overhead, Case 2 only fuses dPCRLB within local neighbourhoods of the processing nodes.

5.4 Simulation Results

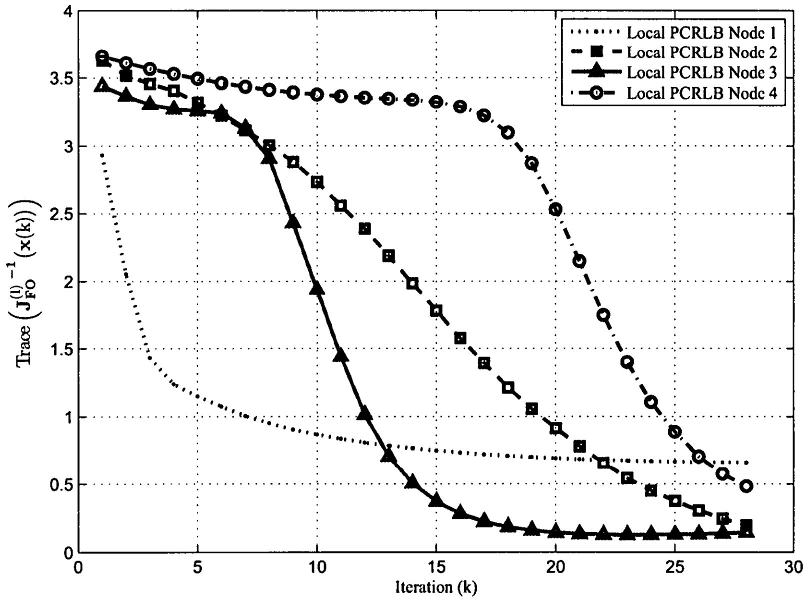
In this section, Monte Carlo simulations are performed to determine the accuracy of the proposed dPCRLB expressions for full-order (Theorem 7, Section 5.2.2 and Theorem 8, Section 5.3.2) systems by comparing them with the results obtained using the centralized PCRLB (Proposition 1) as well as from the approximated bound proposed in [15].

5.4.1 Non-Conditional dPCRLB Computational Algorithms

A distributed bearing-only target tracking (BOT) application [102] as explained in Section 3.3 is used to demonstrate the accuracy of the proposed full-order dPCRLB. The dPCRLB comparison includes results from three : (i) The centralized PCRLB (Proposition 1); (ii) Proposed FO/dPCRLB approach (Sections 5.2.2); (iii) Approximated expression for dPCRLB given in [15] (Eq. (5.32)). Two different scenarios are considered, which described next.



(a)



(b)

Figure 5.1: (a) Target's track alongside the location of the local observation nodes. (b) Trace of the local PCRLBs computed at Nodes 1-4 based on Eq. (5.38)-(5.41). All nodes shown in Fig. 5.1(a) are used in the dPCRLB algorithm.

5.4.1.1 Scenario 1

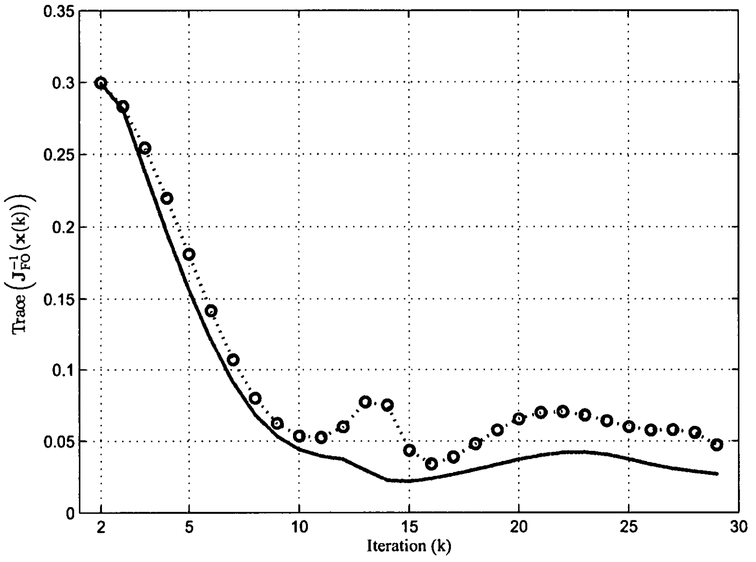
The first simulation [103] is based on a fixed target trajectory (i.e., the same track is used in each Monte Carlo run) and the true values of the state variables is used to compute different bounds. The proposed algorithm for full-order systems outlined in Section 5.2.2 is then used to compute the dPCRLB. In Step 3, Theorem 7 (Eqs. (5.33)-(5.35)) is used. This is a test case included to evaluate the correctness of the proposed dPCRLB and to see how close the proposed dPCRLB can potentially be to the centralized PCRLB. In reality, the exact state values are not known. Scenario 2 covers a more realistic case. Fig. 5.1(a) shows one realization of the sensor placement along with the target's track. Fig. 5.1(b) depicts the trace of the local PCRLBs computed using Eqs. (5.38)-(5.41) at four randomly selected nodes highlighted as Nodes 1-4 in Fig. 5.1(a). The local performance of nodes varies due to state dependent nature of the problem. The dPCRLB is then computed from all the local PCRLBs based on Theorem 7. Fig. 5.2 compares the proposed full-order dPCRLB, the centralized PCRLB based on a fusion centre (included here as the ground truth), and the suboptimal dPCRLB based on [15] over 200 Monte Carlo runs with the same sensor network configuration. Due to the state-dependent observation noise variance, we note that the SNR is time-varying and differs from one sensor node to the other depending on the location of the target. Two different SNR cases (averaged across all nodes and time) are considered: (i) *High SNR,* where the SNRs at different nodes varies form 17dB to 24dB with a mean value of 20dB, (ii) *Low SNR,* where the SNRs ranges from 0dB to 11dB across the network with a mean value of 6dB. Fig. 5.2(a) plots the PCLRBs for the high SNR case, while Fig. 5.2(b) plots the bounds for the low SNR scenario. As illustrated in Figs. 5.2(a) and 5.2(b), the centralized and distributed PCRLBs virtually overlap. The proposed bound predicts the estimator's performance more accurately than the approximated approach [15]. Finally, we note that for low SNR scenarios, the approximated full-order dPCRLB (Eq. (5.32)) degrades significantly from the true bound due to the localized nature of the previous FIM (third term on the RHS of Eq. (5.32)). As illustrated in the first bullet after Lemma 2, the approximated expression uses $\boldsymbol{J}_{\mathrm{FO}}^{(l)}(\mathbf{x}(k))$ instead of the global FIM $\boldsymbol{J}_{\mathrm{FO}}(\mathbf{x}(k))$ which results in additional inaccuracies and as well as variations in the dPCRLB from one node to another.

5.4.1.2 Scenario 2

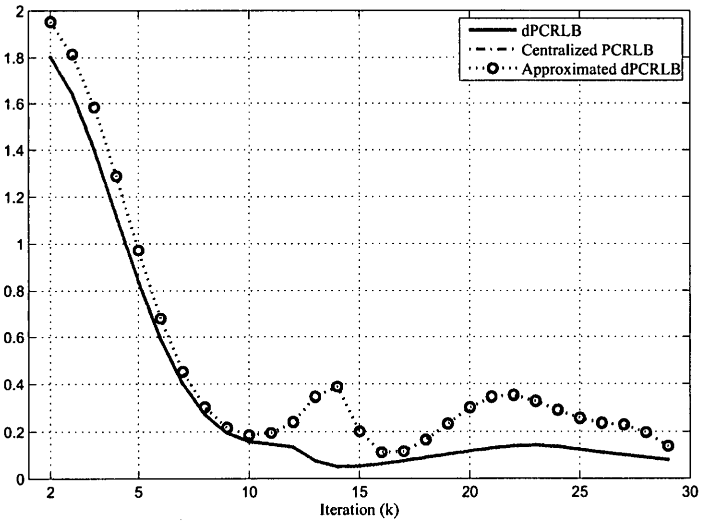
Scenario 2 uses the BOT model specified in Scenario 1 with the following differences: (i) The target track is not fixed (i.e., unlike Scenario 1 with fixed track, the track varies from one iteration to another in the Monte Carlo simulation); (ii) The dPCRLB is based on the estimated state values obtained from the particle filter [158] (as opposed to the true state values utilized in Scenario 1) in both centralized and distributed computation of the PCRLBs; (iii) In each Monte Carlo run (Monte Carlo simulation of 200 runs is performed), a different sensor network configuration is considered, with $N = 20$ nodes randomly scattered in a square region of dimension (16×16) m^2. Because of these differences, the baseline (centralized PCRLB) and the comparison results are different between Figs. 5.2 and 5.3.

The full-order dPCRLB algorithm explained in Section 5.2 is used to compute the dPCRLB with Step 3 (incorporating Theorem 7) based on Eqs. (5.51)-(5.54), which includes expectations. We use consensus/fusion based distributed implementation of the particle filter (CF/DPF) [50] to compute the expectation terms over possible realizations of the state and observation sequences. For the BOT problem, the computation of the Jacobian terms $\nabla_{\mathbf{x}(k)} \boldsymbol{f}^T(k)$ and $\nabla_{\mathbf{x}(k+1)} \boldsymbol{g}^{(l)^T}(k+1)$ used in Eqs. (5.51)-(5.54) and the initialization step are further described in [103]. Matrix $\left[\boldsymbol{D}_{\mathrm{FO}}^{22}(k)\right]^{(l)}$ in Eq. (5.54) is derived based on the particle based approximation given in [141].

We note that both the centralized PCRLB and dPCRLB use state estimates from the particle filters. The centralized PCRLB uses state estimates computed by the centralized particle



(a)



(b)

Figure 5.2: Scenario 1 in Full-order System: Comparison between the centralized, proposed and approximated [15] dPCRLBs at: (a) High SNR (average 20dB), and; (b) Low SNR (average 6dB). The exact full-order dPCRLB from Theorem 1 computed using Eq. (36) is shown in red solid line, the centralized PCRLB from Proposition 1 in green dotted line, and the approximated dPCRLB from Eq. (39) in blue dotted line with circles.

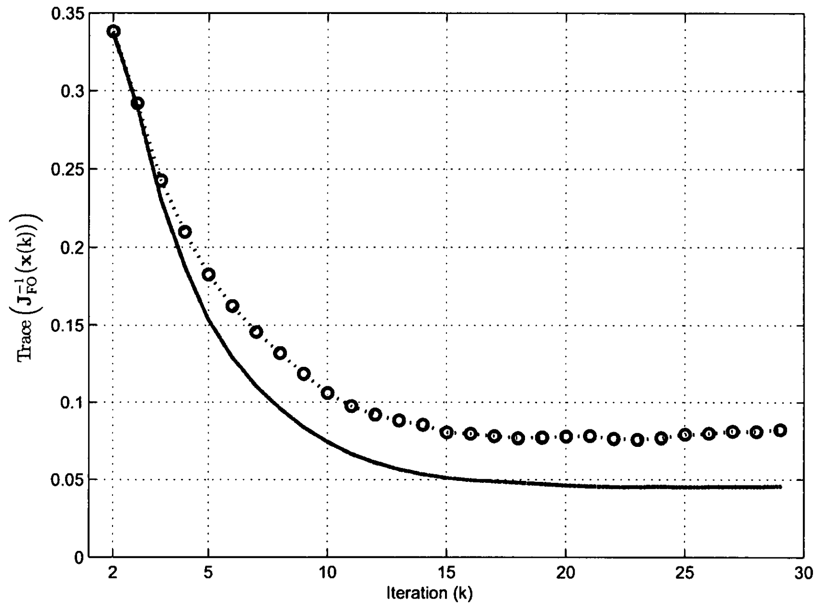
filter, while the distributed PCRLB uses estimates from the distributed particle filter such as the CF/DPF [25]. Consequently, any drop in the accuracy of the state estimates (due to for example a reduction in the SNR) affects both bounds. As long as the distributed particle filter is an optimal implementation of the centralized particle filter, the centralized PCRLB and dPCRLB should result in similar bounds.

Fig. 5.3(a) is for the high SNR case, while Fig. 5.3(b) plots the bounds for the low SNR scenario. In both cases, the centralized PCRLB and proposed dPCRLB are close (almost overlapping), while the approximated dPCRLB [15] fluctuates from the true value. In Figs. 5.3(a) and 5.3(b), the PCRLBs are higher than Figs. 5.2(a) and 5.2(b) because estimated values for states are used instead of the actual values and the target track varies between different runs of the Monte Carlo simulation.

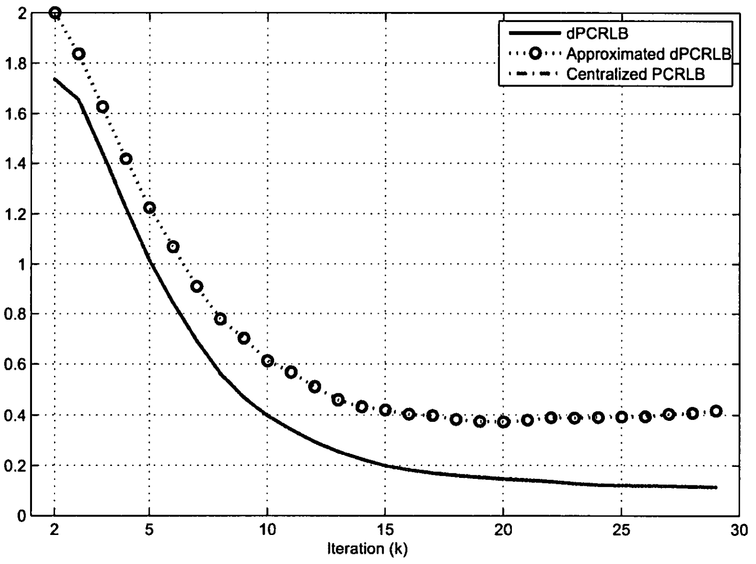
5.4.2 Conditional dPCRLB Computational Algorithms

In this section, the proposed recursive algorithm (Eqs. (5.67)-(5.70)) for computing the online conditional dPCRLB is evaluated as an alternative to the offline non-conditional dPCRLB . Previous conditional PCRLB algorithms are limited to centralized architectures using a fusion centre which makes them inappropriate for decentralized sensor management. The proposed conditional dPCRLB is an accurate representation of its centralized counterpart. Since it is a function of the past observations made, the conditional PCRLB is a more reliable criteria for decentralized sensor-selection applications.

Another distributed BOT application [67] based on a sensor network of $N = 30$ nodes compares the proposed conditional dPCRLB with the centralized conditional PCRLB. A nonlinear clockwise coordinate turn (CCT) motion model (Eq. (3.50)) is considered for the target. Node l's observation is the target's bearings as outlined in Section 5.4 where both process and observation

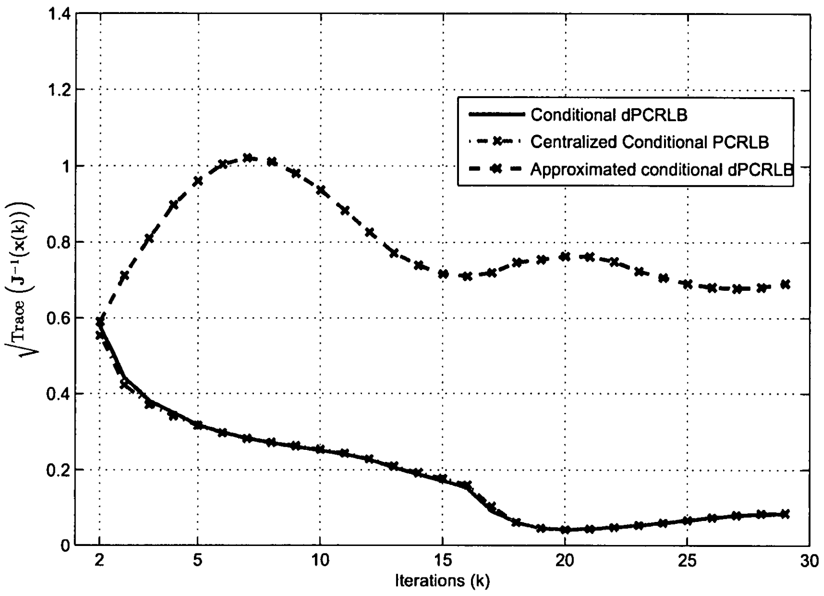


(a)

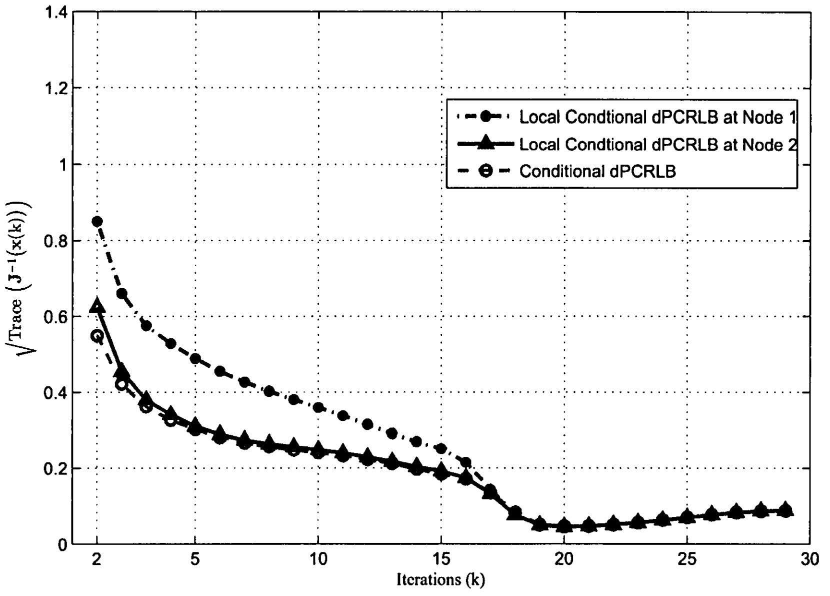


(b)

Figure 5.3: Scenario 2 in Full-order System: Same as Fig. 5.2 except particles set $\{\mathbb{X}_i(k), W_i(k)\}$ is used to compute expectations in Eqs. (5.38)-(5.41).



(a)



(b)

Figure 5.4: (a) Case 1: Comparison of the proposed conditional dPCRLB, centralized conditional PCRLB and an approximate conditional dPCRLB (similar to [15]). (b) Same as (a) except for Case 2 (local fusion without consensus): PCRLB comparison between two randomly selected nodes and from Case 1.

noises are normally distributed with the observation noise model assumed to be state dependent such that the bearing noise variance at node l depends on the distance between the observer and target. Simulations consider the two Cases described in Section 5.3.3. For Case 1, Fig. 5.4(a) compares the proposed conditional dPCRLB (obtained from Theorem 8), the centralized conditional PCRLB (using the centralized bound [152]), and the approximated conditional dPCRLB based only on the first two terms on the RHS of Eq. (5.70) (similar to [15]). It is observed that the proposed conditional dPCRLB and the centralized bound overlap across various iterations. The approximated PCRLB fluctuates widely over time. Having justified that the proposed dPCRLB is an accurate representation of its centralized counterpart, Fig. 5.4(b) plots the conditional dPCRLB results for Case 2 (local fusion with no consensus). Results from two randomly selected nodes are plotted in Fig. 5.4(b). Due to localized fusion in Case 2, some variation in the conditional dPCRLBs is observed at the two nodes but the proposed bound is still superior to the approximated bound as plotted in Fig. 5.4(a). Fig. 5.4(b) also suggests that some nodes are self-confined where global fusion is not needed. On the basis of local information, some nodes are, however, unable to reach the true bound and extra communication may be needed if higher accuracy is desired. Still, Case 2 is sufficient for sensor selection decisions in the current form.

5.5 Summary

The chapter derives the dPCRLB for distributed full-order and reduced order estimation architecture in distributed AN/SN systems without the need of a central processing unit. The centralized PCRLB can not be computed for these networks. The chapter proposes the distributed PCRLB (dPCRLB) algorithms for full-order (FO/dPCRLB) expressed in terms of Theorem 1. Theorem 1 is applicable when the estimates of the entire state vector is available locally at each node. In reduced-order estimation, a different subset of the state vector is estimated at the local nodes. The dPCRLB for reduced-order estimation is derived in Appendix E. Motivated by resource management decisions in sensor networks, optimal and near-optimal expressions for recursively computing the FO/dPCRLB are derived. The proposed dPCRLBs and their practical implementations are compared for a variety of full-order systems using Monte Carlo simulations. Our results indicate that the proposed dPCRLB algorithms provide an exact bound and overlap the PCRLB plot derived from the centralized architecture.

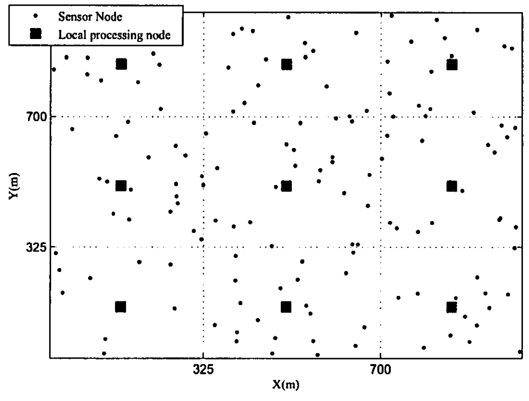
6 Sensor Selection in Distributed Networks

Recent developments in sensor hardware and advances in communication have paved the way for deploying an unrestrictively large number of sensor nodes for long periods of time. Limitations in power, frequency, and bandwidth, however, restrict the maximum number of sensors that can be simultaneously active. Algorithms dealing with the activation of the sensor nodes (or alternatively, the scheduling of the sensing activities) are referred to as sensor selection algorithms, since they select which nodes participate in the sensing task. Adaptive sensor selection refers to the dynamical activation of the sensor nodes within a sensing task. In other words, the active sensors may change from one iteration of the algorithm to another, adaptive sensor selection is, therefore, introduced as an essential task in geographically distributed networks. Adaptive sensor selection [133–137] is a stochastic problem that involves optimization of a pre-defined cost function, e.g., the volume of the uncertainty ellipsoid [138], the estimated states' mean square error (MSE) [139] or information driven methods [140]. Adaptive sensor selection arises in several applications, e.g., cellular networks [161], distributed tracking in wireless Ad hoc sensor networks [162], robotic localization and underwater networks [7].

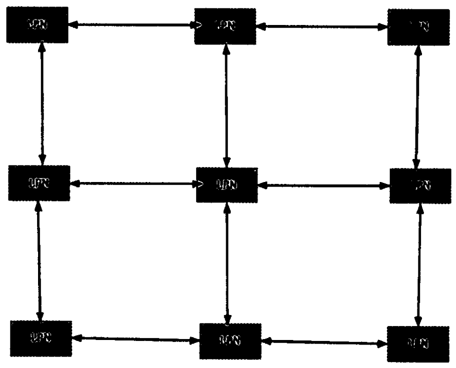
My previous work [49–52, 59, 61–63] and likewise, a large majority of the existing state-of-art distributed, non-linear estimation algorithms for agent/sensor networks (AN/SN) [18–21, 23, 24, 27, 29] incorporate observations locally in a distributed fashion from *all* observation nodes. The chapter focuses on a more challenging distributed estimation problem that optimizes an additional constraint of limiting the number of active nodes and selecting a subset of observation nodes (sensors) at each iteration. For such an adaptive sensor selection problem, the PCRLB [141, 147,160] has been proposed as an effective criteria because it provides a near-optimal bound of the achievable tracker's performance and can be calculated predictively. Further, it is independent and not constrained by the estimation methodology employed. I propose a distributed diffusive PCRLB-based sensor selection procedure for distributed AN/SN systems where the performance of each local estimator is characterized by its local FIM. Local FIMs can be used as a criteria for local sensor selection decisions. Such decisions are limited to local observations and the global sensor information is not incorporated. A fusion rule is, therefore, needed to combine local FIMs into the global FIM for taking globally optimal sensor subset selection decisions. The non-conditional dPCRLB computational algorithm, presented in Chapter 5, is proposed as the objective function for distributed adaptive sensor selection. A combination of minimum and average consensus algorithms are then used to select a subset of observation nodes.

The chapter extends the non-conditional dPCRLB framework to conditional dPCRLB for full-order adaptive sensor selection problems. As stated previously, the conventional (non-conditional) PCRLB considers observations and state variables as random, consequently, it is determined primarily from the state model, observation model, and prior knowledge of the initial state of the system leading to an *offline* bound with actual observations averaged out over time. The conditional PCRLB, on the other hand, is a function of the past history of observations which, therefore, leads to a more accurate representation of the system's performance and a better criteria for adaptive sensor-selection.

Finally, the chapter addresses another critical restriction in large, geographically distributed AN/SN systems imposed by limitations in power budget, system bandwidth, and communication capabilities, i.e., only quantized observations are exchanged between the sensors and process-



(a)



(b)

Figure 6.1: (a) A sample distributed scenario [15] consisting of 9 local processing nodes and 150 observation nodes (sensors). (b) Fusion-to-fusion communication constraints.

ing nodes. Within its observation neighbourhood, a local processing node, therefore, activates a small subset of sensors to receive the quantized version of their observations. The chapter derives distributed computational techniques for determining the conditional dPCRLB for quantized, distributed AN/SN systems, referred to as CQ/dPCRLB. Analytical expressions for the CQ/dPCRLB are derived, which are particularly useful for particle filter-based estimators.

The rest of chapter is organized as follows. Section 6.1 provides necessary background on the sensor selection model. Section 6.3 presents the non-conditional dPCRLB based sensor selector. Section 6.4 extends the sensor selection framework based on conditional dPCRLB. Section 6.5 extends the dPCRLB algorithm to quantized local observations. Section 6.6.1 illustrates the effectiveness of the proposed framework in tracking applications through Monte Carlo simulations. Finally, Section 6.7 concludes the chapter.

6.1 System Description

Unlike Chapter 2 to 5, where no distinction is made between the observation and processing nodes, the chapter considers a distributed AN/SN topology (as shown in Fig. 6.1(a)) with two different types of nodes [15]: (i) *Observation nodes (sensors)*: with limited power used to record measurements, and; (ii) *Local processing nodes*: responsible for selecting sensors within their neighbourhoods, process the data locally, and cooperate distributively with other connected processing nodes to reach a consensual tracking estimate for the target. Such a configuration has the added advantage of *not* requiring global knowledge of the network topology at the local processing nodes and is suitable for any Ad hoc AN/SN. Within its neighbourhood, each local processing node activates a small subset of sensors. These sensors forward their observations to the associated local processing node. After processing the local observations, local processing nodes communicate some statistics related to the localized state estimates within themselves, typically using a gossip type algorithm [1], to form the global state estimate. Fig. 6.1(b) illustrates the fusion-fusion neighbourhood. To prevent data incest [137] (i.e., to avoid observation redundancy and correlation between locally estimated tracks), I impose a commonly used assumption [15] that a sensor node once selected for information processing by a local processing node does not forward its observations to a second processing node during the same iteration.

6.1.1 Distributed Sensor Selection Model

An AN/SN is considered comprising of N_f local processing nodes (e.g., in Fig. 6.1(a) $N_f = 9$). The distributed sensor selection entails a scenario where each local processing node can communicate only with sensors and other local processing nodes within its surveillance region (immediate neighbourhood). Local processing node l, $(1 \leq l \leq N_f)$, is associated with a set of $N_{ss}^{(l)}$ sensors within its local neighbourhood. The total number of observation nodes in the network is, therefore, given by

$$N_{ss} = \sum_{l=1}^{N_f} N_{ss}^{(l)}. \tag{6.1}$$

For example in Fig. 6.1(a) $N_{ss} = 150$. Due to physical limitations, only a subset $\aleph_{\text{obs}}^{(l)}(k)$ of $N_{ss}^{(l)}$ sensors connected to processing node l, for $(1 \leq l \leq N_f)$, is active at iteration k. Further, only a maximum number $N_{\text{max}}^{(l)}(k)$ of sensors can be activated by node l, i.e, $|\aleph_{\text{obs}}^{(l)}(k)| \leq N_{\text{max}}^{(l)}(k)$ where $|.|$ denotes cardinality operator. The total number $N_{\text{max}}(k)$ of observation nodes simultaneously active in the network is also restricted, i.e.,

$$\sum_{l=1}^{N_f} N_{s,\text{max}}^{(l)}(k) \leq N_{\text{max}}(k) \leq N_{ss}. \tag{6.2}$$

The observation subset $\aleph_{\text{obs}}^{(l)}(k)$ at local processing node l can only be changed after N_{change} iterations.

Each sensor in the network observes a set of n_x state variables $\mathbf{x} = [X_1, X_2, \ldots, X_{n_x}]^T$. The observation model (Eq. (2.3)) corresponding to sensor m in the fusion neighbourhood of the processing node l is given by

$$Z^{(l,m)}(k) = g^{(l,m)}(\mathbf{x}(k)) + \zeta^{(l,m)}(k), \tag{6.3}$$

where $g^{(l,m)}(\cdot)$ and $\zeta^{(l,m)}(\cdot)$ are, respectively, the local observation model and uncertainty at sensor node m connected to processing node l. For sensor selection problem, term $\mathbf{z}^{(\aleph_{\text{obs}}^{(l)})}(k)$ denotes the local observation vector $\mathbf{z}^{(l)}(k)$ in Eq. (2.2). The collection of all observations associated with the processing node l at time instant k is given by

$$\mathbf{z}^{(\aleph_{\text{obs}}^{(l)})}(k) = \{Z^{(l,m)}(k) : m \in \aleph_{\text{obs}}^{(l)}(k)\}, \quad \text{for } (1 \leq l \leq N_f). \tag{6.4}$$

The full-order state-space model (Eqs. (2.127) and (2.126)) for the distributed sensor selection problem is modified as follows

$$\begin{aligned}\mathbf{x}(k) &= \boldsymbol{f}(\mathbf{x}(k-1)) + \boldsymbol{\xi}(k) & (6.5)\\ \mathbf{z}^{(\aleph_{\text{obs}}^{(l)})}(k) &= \boldsymbol{g}^{(\aleph_{\text{obs}}^{(l)})}(\mathbf{x}(k)) + \boldsymbol{\zeta}^{(\aleph_{\text{obs}}^{(l)})}(k), & (6.6)\end{aligned}$$

for local processing nodes ($1 \leq l \leq N_f$). The entire state $\mathbf{x}(k)$ is estimated by running localized filter at each local processing node, while observations are restricted to $\mathbf{z}^{(\aleph_{\text{obs}}^{(l)})}(k)$ obtained from the observation nodes in the fusion neighbourhood $\aleph_{\text{obs}}^{(l)}(k)$ selected by fusion node l. Since $\aleph_{\text{obs}}^{(l)}(k)$ varies with time, the dimensions of the observation vector is not fixed. I also define a fusion-to-fusion neighbourhood $\aleph_{\text{fuse}}^{(l)}$ that includes the set of processing nodes connected to the local processing node l. Fig. 6.1(b) shows an example of the fusion-to-fusion neighbourhood $\aleph_{\text{fuse}}^{(l)}$.

6.2 Sensor Selection Objective Function

Sensor selection is a stochastic problem that involves optimization of a pre-defined objective function, e.g., the estimated states' mean square error (MSE) [139] or an entropy-based information measure such as the expected maximum likelihood [135, 140]. Recently the PCRLB [136, 137, 141, 145–147, 152, 160] has been proposed as an effective cost function for centralized sensor selection because it provides a near-optimal bound of the achievable tracker's performance and can be calculated predictively [15]. Further, it is independent and not constrained by the estimation methodology employed.

In this chapter the dPCRLB from Section 5.2 is used as the objective function for sensor selection. The subscript FO is omitted from dPCRLB expressions to keep the notion simple. As stated previously, the MSE of the estimate $\hat{\mathbf{x}}(0\!:\!k)$ of the state variables $\mathbf{x}(0\!:\!k)$ is lower bounded by the PCRLB as follows

$$\mathbb{E}\{(\hat{\mathbf{x}}(0:k) - \mathbf{x}(0:k))(\hat{\mathbf{x}}(0:k) - \mathbf{x}(0:k))^T\} \geq [\boldsymbol{J}(\mathbf{x}(0:k))]^{-1} \qquad (6.7)$$

where $\mathbb{E}$ denotes expectation. Matrix $\boldsymbol{J}(\mathbf{x}(0:k))$ is derived from the joint probability density function $P(\mathbf{x}(0:k), \mathbf{z}(1:k))$ and is referred to as the Fisher information matrix (FIM). Different forms of the FIM $\boldsymbol{J}(\mathbf{x}(0:k))$ are introduced in Section 5.2.1. Term $\boldsymbol{J}^{(l)}(\mathbf{x}(0:k))$ denotes the local FIM at processing node l, for $(1 \leq l \leq N_f)$ corresponding to the local estimate $\widehat{\mathbf{x}}^{(l)}(0:k)$ based on the local posterior density $P(\mathbf{x}(0:k)|\mathbf{z}^{\aleph^{(l)}_{\text{obs}}}(1:k))$. Similarly, $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(0:k))$ denotes the local single-observation FIM at processing node l based on observation made at the observation node m_l only.

The global FIM at the processing nodes is computed in a distributed configuration using the dPCRLB expressions stated under Theorem 7. Below, iteration $(k+1)$ for updating the dPCRLB is explained in terms of Steps 1-3. Submatrices $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k))$, $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1|k))$, and $\boldsymbol{J}(\mathbf{x}(k))$ are available from iteration k of the dPCRLB update. Besides, each processing node runs the CF/DPF distributed implementation of the particle filter introduced in Chapter 4 and has available two particle sets: The first set results from the local filter and is denoted by $\{\mathbb{X}_i^{(l,\text{LF})}, W_i^{(l,\text{LF})}\}$. The second set results from the fusion filter and is denoted by $\{\mathbb{X}_i^{(l,\text{FF})}, W_i^{(l,\text{FF})}\}$. At the end of a CF/DPF iteration, the fusion filter has achieved consensus such that its particles (though different at the processing nodes) represent the same global posterior distribution. Since iteration k of the CF/DPF is also complete, therefore, the local particles $\{\mathbb{X}_i^{(l,\text{LF})}(k), W_i^{(l,\text{LF})}(k)\}$ and fusion particles $\{\mathbb{X}_i^{(l,\text{FF})}(k), W_i^{(l,\text{FF})}(k)\}$ are also available.

Step 1: Based on particles $\{\mathbb{X}_i^{(l,\text{FF})}(k), W_i^{(l,\text{FF})}\}$ of the fusion filter, processing node l computes terms $\boldsymbol{C}^{11}(k)$, $\boldsymbol{C}^{21}(k)$, and $\boldsymbol{C}^{12}(k)$ using Eqs. (5.26)-(5.27). Since the fusion particles represent the same global posterior distribution, the resulting values are similar at all processing nodes.

Step 2: Processing node l computes term $\boldsymbol{C}^{22}(k)$ using (5.28). This involves the local FIMs $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1))$ and $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1|k))$, which are computed based on the framework presented in Section 5.2. Since these are local entities, these are based on the particles $\{\mathbb{X}_i^{(l,\mathrm{LF})}(k), W_i^{(l,\mathrm{LF})}(k)\}$ of the local filters at the fusion nodes. Consequently, $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1))$ and $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1|k))$ will have different values at the processing nodes. Based on the sensor selection model introduced in Section 6.1.1, term $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1))$, for example, is computed as follows

$$\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1)) = \left[\boldsymbol{D}^{22}(k)\right]^{(l,m_l)} - \left[\boldsymbol{D}^{21}(k)\right]^{(l)}\left(\boldsymbol{J}^{(l)}(\mathbf{x}(k))+\left[\boldsymbol{D}^{11}(k)\right]^{(l)}\right)^{-1}\left[\boldsymbol{D}^{12}(k)\right]^{(l)}, \tag{6.8}$$

where

$$\left[\boldsymbol{D}^{11}(k)\right]^{(l)} = \mathbb{E}\big\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\} \tag{6.9}$$

$$\left[\boldsymbol{D}^{12}(k)\right]^{(l)} = \left(\left[\boldsymbol{D}^{21}(k)\right]^{(l)}\right)^T = \mathbb{E}\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k+1)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\} \tag{6.10}$$

$$\begin{aligned}\left[\boldsymbol{D}^{22}(k)\right]^{(l,m_l)} &= \mathbb{E}\big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\} \\ &+ \mathbb{E}\big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log P\big(\mathbf{z}^{(l,m_l)}(k+1)|\mathbf{x}(k+1)\big)\big\}.\end{aligned} \tag{6.11}$$

Step 3: Theorem 7 is now used to compute the dPCRLB, which is the same at all processing nodes.

As a special case and without loss of generality, I develop the distributed particle filter tracker and the dPCRLB-based sensor selector for 2D bearing-only tracking (BOT) applications. As stated previously, the objective in BOT is to estimate the kinematics (position $[X, Y]$ and velocity $[\dot{X}, \dot{Y}]$) of the target from the bearing angle measurements (referenced clockwise positive to the y-axis), i.e.,

$$Z^{(l,m_l)}(k) = \operatorname{atan}\left(\frac{X(k) - X^{(l,m_l)}}{Y(k) - Y^{(l,m_l)}}\right) + \zeta^{(l,m_l)}(k), \tag{6.12}$$

where $(X^{(l,m_l)}, Y^{(l,m_l)})$ are the coordinates of sensor node l.

Gaussian forcing terms: A common BOT model [103] assumes that the forcing term $\boldsymbol{\xi}(k)$ and observation noise $\boldsymbol{\zeta}^{(l,m_l)}(k)$ in Eqs. (6.5) and (6.3) to be uncorrelated and normally distributed with zero mean and covariance matrices $\boldsymbol{Q}(k)$ and $\boldsymbol{R}^{(l,m_l)}(k)$, respectively. In such cases, Eqs. (5.33)-(5.35) are used instead of Eq. (5.26)-(5.28). For the sensor selection model used in this chapter, Eqs. (5.33) and (5.34) remain the same and Eq. (5.35) changes as follows

$$\boldsymbol{C}^{22}(k) = \sum_{l=1}^{N_f} \sum_{m_l \in \aleph_{\text{obs}}^{(l)}} \Big(\underbrace{\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1)) - \boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1|k))}_{\mathbf{x}_{c1}^{(l,m_l)}(0)} \Big) + \boldsymbol{Q}^{-1}(k) \tag{6.13}$$

Eqs. (5.33) and (5.34) can be expressed in terms of the fusion filter's particles as follows

$$\widehat{\boldsymbol{C}}^{11}(k) \approx \sum_{i=1}^{N_p} W_i^{(l,\text{FF})}(k) \times \Big([\nabla_{\mathbf{x}(k)} \boldsymbol{f}^T(k)] \boldsymbol{Q}^{-1}(k) [\nabla_{\mathbf{x}(k)} \boldsymbol{f}^T(k)] \Big) \Big|_{\mathbf{x}(k)=\mathbb{X}_i^{(l,\text{FF})}(k)}, \tag{6.14}$$

$$\widehat{\boldsymbol{C}}^{12}(k) = \Big[\widehat{\boldsymbol{C}}^{21}(k)\Big]^T \approx \sum_{i=1}^{N_p} W_i^{(l,\text{FF})}(k) \times \Big([\nabla_{\mathbf{x}(k)} \boldsymbol{f}^T(k)] \boldsymbol{Q}^{-1}(k) \Big) \Big|_{\mathbf{x}(k)=\mathbb{X}_i^{(l,\text{FF})}(k)}, \tag{6.15}$$

Term $\boldsymbol{C}^{22}(k)$ requires participation of all local processing nodes to compute$\sum_l \sum_m \mathbf{x}_{c1}^{(l,m_l)}(0)$, which depends on the submatrices $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1))$ and $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1|k))$ of the local FIM. Submatrix $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1))$ is computed using Eq. (6.8) with terms $\big[\boldsymbol{D}^{11}(k)\big]^{(l)}$, $\big[\boldsymbol{D}^{12}(k)\big]^{(l)}$, and $\big[\boldsymbol{D}^{22}(k)\big]^{(l,m_l)}$ approximated as

$$\big[\widehat{\boldsymbol{D}}^{11}(k)\big]^{(l)} \approx \sum_{i=1}^{N_s} W_i^{(l,\text{LF})}(k) \times \Big([\nabla_{\mathbf{x}(k)} \boldsymbol{f}^T(k)] \boldsymbol{Q}^{-1}(k) [\nabla_{\mathbf{x}(k)} \boldsymbol{f}^T(k)] \Big) \Big|_{\mathbf{x}(k)=\mathbb{X}_i^{(l,\text{LF})}(k)}, \tag{6.16}$$

$$\big[\widehat{\boldsymbol{D}}^{12}(k)\big]^{(l)} = \Big[\big[\widehat{\boldsymbol{D}}^{21}(k)\big]^{(l)}\Big]^T \approx \sum_{i=1}^{N_s} W_i^{(l,\text{LF})}(k) \times \Big([\nabla_{\mathbf{x}(k)} \boldsymbol{f}^T(k)] \boldsymbol{Q}^{-1}(k) \Big) \Big|_{\mathbf{x}(k)=\mathbb{X}_i^{(l,\text{LF})}(k)} \tag{6.17}$$

$$\begin{aligned}\big[\widehat{\boldsymbol{D}}^{22}(k)\big]^{(l,m_l)} \approx\ & \boldsymbol{Q}^{-1}(k) + \frac{1}{R^{(l,m_l)}(k)} \sum_{i=1}^{N_s} W_i^{(l,\text{LF})}(k) \\ & \times \begin{bmatrix} H_{(1,1)}^{(l,m_l)}(k) & H_{(1,2)}^{(l,m_l)}(k) & 0 & 0 \\ H_{(2,1)}^{(l,m_l)}(k) & H_{(2,2)}^{(l,m_l)}(k) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \Bigg|_{\substack{\mathbf{x}(k+1)= \\ \mathbb{X}_i^{(l,\text{LF})}(k+1|k)}}\end{aligned} \tag{6.18}$$

with

$$H^{(l,m_l)}_{(1,1)}(k) = \frac{(Y(k+1) - Y^{(l,m_l)})^2}{[(X(k+1) - X^{(l,m_l)})^2 + (Y(k+1) - Y^{(l,m_l)})^2]^2} \tag{6.19}$$

$$H^{(l,m_l)}_{(1,2)}(k) = H^{(l,m_l)}_{(2,1)}(k) = \frac{-(X(k+1) - X^{(l,m_l)})(Y(k+1) - Y^{(l,m_l)})}{[(X(k+1) - X^{(l,m_l)})^2 + (Y(k+1) - Y^{(l,m_l)})^2]^2} \tag{6.20}$$

$$H^{(l,m_l)}_{(2,2)}(k) = \frac{(X(k+1) - X^{(l,m_l)})^2}{[(X(k+1) - X^{(l,m_l)})^2 + (Y(k+1) - Y^{(l,m_l)})^2]^2}. \tag{6.21}$$

Approximations (6.16)-(6.18) use local filter particles instead of particles from the fusion filters from the CF/DPF. Prediction particles $\mathbb{X}_i^{(l,\mathrm{LF})}(k+1|k)$ in (6.18) are computed by propagating $\mathbb{X}_i^{(l,\mathrm{LF})}(k)$ through the transitional density $P(\mathbf{x}(k+1)|\mathbf{x}(k))$ obtained from the state equation (Eq. (6.5)). Note that all required terms in Eqs. (6.16)-(6.21) are computed based on the available particles for iteration k. Having computed $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1))$ and $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1|k))$, term $\sum_l \sum_m \mathbf{x}_{c1}^{(l,m_l)}(0)$ in Eq. (6.13) is obtained using an average consensus algorithm in a distributed fashion as discussed in Section 5.2.

This completes the review of the computation and fusion of local FIMs $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1))$. Finally, note that the approach for computing $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1|k))$ is similar, please refer to Section 5.2 for more details. Next the dPCRLB-based sensor selection algorithm is presented.

6.3 dPCRLB based Sensor Selection

In this section, I present the dPCRLB based distributed sensor selection algorithm for full-order distributed estimation problems. The dPCRLB from Section 6.2 is used as the objective function for sensor selection. The dPCRLB based sensor selection is illustrated in Fig. 6.2 where iteration k has just been completed. At each node, the CF/DPF has its local particle set $\{\mathbb{X}_i^{(l,\mathrm{LF})}(k), W_i^{(l,\mathrm{LF})}(k)\}_{i=1}^{N_s}$ and fusion particle set $\{\mathbb{X}_i^{(l,\mathrm{FF})}(k), W_i^{(l,\mathrm{FF})}(k)\}_{i=1}^{N_{\mathrm{FF}}}$ available based on the active network configuration determined by the sensor selection algorithm. From the previous iteration at time index k of the sensor selection, the following quantities are available: local

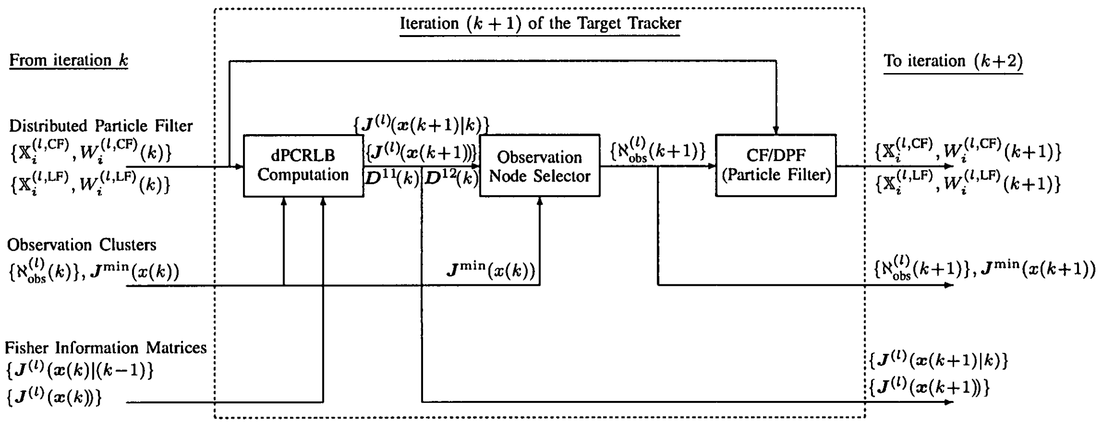

Iteration $(k+1)$ of the Target Tracker
From iteration k
To iteration $(k+2)$
Distributed Particle Filter
$\{\mathbb{X}_i^{(l,\text{CF})}, W_i^{(l,\text{CF})}(k)\}$
$\{\mathbb{X}_i^{(l,\text{LF})}, W_i^{(l,\text{LF})}(k)\}$
dPCRLB Computation
$\{J^{(l)}(x(k+1)|k)\}$
$\{J^{(l)}(x(k+1))\}$
$D^{11}(k)$
$D^{12}(k)$
Observation Node Selector
$\{\aleph_{\text{obs}}^{(l)}(k+1)\}$
CF/DPF (Particle Filter)
$\{\mathbb{X}_i^{(l,\text{CF})}, W_i^{(l,\text{CF})}(k+1)\}$
$\{\mathbb{X}_i^{(l,\text{LF})}, W_i^{(l,\text{LF})}(k+1)\}$
Observation Clusters
$\{\aleph_{\text{obs}}^{(l)}(k)\}, J^{\min}(x(k))$
$J^{\min}(x(k))$
$\{\aleph_{\text{obs}}^{(l)}(k+1)\}, J^{\min}(x(k+1))$
Fisher Information Matrices
$\{J^{(l)}(x(k)|(k-1)\}$
$\{J^{(l)}(x(k))\}$
$\{J^{(l)}(x(k+1)|k)\}$
$\{J^{(l)}(x(k+1))\}$

Figure 6.2: Iteration $(k+1)$ of the proposed dPCRLB based distributed target tracker with the observation node selection feature.

PCRLBs $J^{(l,m_l)}(\mathbf{x}(k))$, for $(1 \le l \le N)$ and $(m_l \in \aleph_{\text{obs}}^{(l)})$, the global dPCRLB $J(\mathbf{x}(k))$ optimized for $\aleph_{\text{obs}}^{(l)}$ at k. Iteration $k+1$ uses the overall dPCRLB to compute the local and global PCRLBs as explained in Section 6.2.

After computing the local and overall FIMs for the dPCRLB (the "dPCRLB computation" block in Fig. 6.2), the next stage constitutes the observation node selector for the processing nodes. As shown in Fig. 6.2, the selector requires the following inputs:

PCRLB Parameters (from dPCRLB computation block):

$$J(\mathbf{x}(k+1)), J^{(l)}(\mathbf{x}(k+1)), J^{(l)}(\mathbf{x}(k+1|k)), D^{11}(k), D^{12}(k)$$

Selector Parameter (from the previous selector iteration:)

$$J^{\min}(\mathbf{x}(k)), \text{i.e., the overall dPCRLB optimized for } \aleph_{\text{obs}}^{(l)} \text{ at } k.$$

I illustrate the proposed sensor selection approach in terms of the BOT problem. The overall cost function $\mathcal{C}(k+1)$ used by the BOT selectors is based on the dPCRLBs related to the (x, y)

coordinates of the target, i.e.,

$$\mathcal{C}(k+1) = [\boldsymbol{J}(\mathbf{x}(k+1))]^{-1}_{xx} + [\boldsymbol{J}(\mathbf{x}(k+1))]^{-1}_{yy}. \tag{6.22}$$

where $[\boldsymbol{J}(\mathbf{x}(k+1))]^{-1}_{xx}$ is the dPCLRB corresponding to the x-coordinate at iteration $k+1$. Similarly, $[\boldsymbol{J}(\mathbf{x}(k+1))]^{-1}_{yy}$ is the dPCLRB corresponding to the y-coordinate at iteration $k+1$. In general, sensor selection is an NP-hard combinatorial optimization problem [163]. Finding the optimal solution in real time is difficult especially when the number of possible combinations is impractically large, hence, a near-optimal procedure is generally used. The observation node selection is carried out in several iterations $t \geq 1$. To select the best observation node at each local processing node, the following local cost function (expressed in terms of processing-node-observation-node (l, m_l) combination) is used

$$\mathcal{C}^{(l,m_l)}(t) = [\boldsymbol{J}^{(\aleph^{(l)}_{\text{obs}})}(t)]^{-1}_{xx} + [\boldsymbol{J}^{(\aleph^{(l)}_{\text{obs}})}(t)]^{-1}_{yy}. \tag{6.23}$$

where $[\boldsymbol{J}^{(\aleph^{(l)}_{\text{obs}})}(t)]^{-1}_{xx}$ and $[\boldsymbol{J}^{(\aleph^{(l)}_{\text{obs}})}(t)]^{-1}_{yy}$ are the dPCLRB corresponding to the x and y-coordinates in

$$\boldsymbol{J}^{(\aleph^{(l)}_{\text{obs}})}(t) = [\boldsymbol{C}^{22}(t)]^{(l,m_l)} - \boldsymbol{C}^{21}(k)\Big(\boldsymbol{J}^{\text{min}}(x(k)) + \boldsymbol{C}^{11}(k)\Big)^{-1}\boldsymbol{C}^{12}(k), \tag{6.24}$$

with

$$[\boldsymbol{C}^{22}(t)]^{(l,m_l)} = \boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1)) + \boldsymbol{J}^{\text{min}}_{k+1|k+1}(t) - \boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1|k)) - \boldsymbol{J}^{\text{min}}_{k+1|k}(t) + \boldsymbol{Q}^{-1}(k). \tag{6.25}$$

Note that Eqs. (6.24) and (6.25) are representations of Eqs. (5.25) and (5.28) for a single processing-node-observation-node (l, m_l) combination. Notation $\boldsymbol{J}^{(\aleph^{(l)}_{\text{obs}})}(t)$ correspond to the FIM for estimates obtained from the iterating neighbourhood $\aleph^{(l)}_{\text{obs}}(t)$ as it is being optimized. Once optimized, $\aleph^{(l)}_{\text{obs}}(k+1) = \aleph^{(l)}_{\text{obs}}(t)$. Parameters $\boldsymbol{C}^{21}(k) = [\boldsymbol{C}^{12}(k)]^T$ and $\boldsymbol{C}^{11}(k)$ are available from the dPCRLB computation block and are fixed for various iterations of the senor selector. Parameter $\boldsymbol{J}^{(\text{min})}(\mathbf{x}(k))$ corresponds to the dPCRLB from the previously optimized neighbourhood in the last k iteration. Parameter $[C^{22}(t)]^{(l,m_l)}$ is local for the (l, m_l) processing-node-observation-node combination and is obtained from Eq. (6.25). Parameter $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1))$ and $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1|k))$ are the dPCRLBs corresponding to the filtering and prediction estimates obtained at processing node l from a single observation at observation node m_l. Finally, $\boldsymbol{J}^{(\min)}_{k+1|k}(t)$ and $\boldsymbol{J}^{(\min)}_{k+1|k+1}(t)$ are the FIMs corresponding to the filtered and predicted estimates obtained from the iterating neighbourhood $\aleph^{(l)}_{\text{obs}}(t)$. Having defined the cost function, I describe the iterative consensus-based sensor selection approach expressed in terms of the following two steps.

6.3.1 Initial Sensor Selection Step

The initial step of the distributed sensor selection has the following sub-steps.

1.1. At local processing node l, for $(1 \leq l \leq N_f)$, the local FIMs $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1))$ and the cost function $\mathcal{C}^{(l,m_l)}(1)$ corresponding to the processing-node-observation-node (l, m_l) combination are computed based on Eqs. (6.23)-(6.25).

1.2. From all (l, m_l) combinations, node l selects one observation node for which $\mathcal{C}^{(l,m_l)}(1)$ is minimum. In other words, a single observation node is selected by each local processing node that provides the optimal performance at that node when at the most one observation is used.

1.3. At this stage, a complete enumeration encompassing all processing nodes $(1 \leq l \leq N_f)$ is performed. One processing-node-observation-node combination $(q = l, m_q = m_l)$ is selected with the minimum cost function associated to it across the network. A minimum consensus algorithm accomplishes Step 1.3.

1.4. Matrices

$$\boldsymbol{J}^{(q,m_q)}(\mathbf{x}(k+1)) \triangleq \boldsymbol{J}^{(\min)}_{k+1|k}(1) \qquad \text{and} \qquad \boldsymbol{J}^{(q,m_q)}(\mathbf{x}(k+1|k)) \triangleq \boldsymbol{J}^{(\min)}_{k+1|k+1}(1)$$

corresponding to the FIMs for the combination (q, m_q) are communicated across the network. The neighbourhood structure is given by $\aleph(1) = \{\aleph_{\text{obs}}^{(l)}(1)\}^{N_f}$. After the initial selection, all $\aleph_{\text{obs}}^{(l)}(1) = \{\}$ (i.e., empty sets) except for $l = q$ where $\aleph_{\text{obs}}^{(q)} = \{m_q\}$. Note that I have added time index $t = 1$ to each neighbourhood to indicate the iteration number for the selection stage. The FIMs $\boldsymbol{J}^{(l,m_l)}(\mathbf{x}(k+1))$ computed in Step 1.1 are limited to the observation nodes within the communication range of node l.

6.3.2 Subsequent Sensor Selection Step

Each local processing node l, $(1 \leq l \leq N_f)$, selects an observation node in its immediate neighbourhood and for it computes the cost function taking into account the previously selected neighbourhood $(\aleph_{\text{obs}}^{(l)}(t))$ and the associated FIMs $\boldsymbol{J}_{k+1|k}^{(\min)}(t)$ and $\boldsymbol{J}_{k+1|k+1}^{(\min)}(t)$. The subsequent selection is based on the following sub-steps.

2.1. Local processing node l computes $[C^{22}(t)]^{(l,m_l)}$, for $(m_l \notin \aleph_{\text{obs}}^{(l)}(t))$, using Eq. (6.25). The predicted dPCRLB $\boldsymbol{J}^{(\aleph_{\text{obs}}^{(l)})}(t)$ is based on Eq. (6.24).

2.2. Given $\boldsymbol{J}^{(\aleph_{\text{obs}}^{(l)})}(t)$, Eq. (6.23) is used to compute the local cost function $\mathcal{C}^{(l,m_l)}(t)$.

2.3. Select the local processing node $\mathcal{L}$ and observation node $m_{\mathcal{L}}$ combination corresponding to the minimum overall cost function using a minimum consensus algorithm.

2.4. Append the neighbourhood structure to include the new combination $N_{\text{obs}}^{(l)}(t+1) = \{N_{\text{obs}}^{(l)}(t)\}$ appended with the new combination. The overall FIM corresponding to the appended neighbourhood combination is denoted by $\boldsymbol{J}^{(\mathcal{L},m_{\mathcal{L}})}(\mathbf{x}(k+1))$.

2.5. Matrix $\boldsymbol{J}^{(\min)}(\mathbf{x}(k+1))$ now equals to $\boldsymbol{J}^{(\mathcal{L},m_{\mathcal{L}})}(\mathbf{x}(k+1))$, which now corresponds to the overall FIM corresponding to the selected sensors. The new value of matrix $\boldsymbol{J}^{(\min)}(\mathbf{x}(k+1))$ is communicated across the network.

The selection is terminated, if $N_{\max}$ has been reached, otherwise Step 6.3.2 is continued.

In this section a dPCRLB-based sensor selection algorithm is proposed. The next section extends the non-conditional dPCRLB framework to conditional dPCRLB for full-order adaptive sensor selection problems. The non-conditional dPCRLB [148] considers observations and state variables as random, consequently, it is determined primarily from the state model, observation model, and prior knowledge of the initial state of the system. The conditional dPCRLB, on the other hand, is a function of the past history of observations and, therefore, leads to a more accurate representation of the systems's performance and a better criteria for adaptive sensor-selection.

6.4 Conditional dPCRLB based Sensor Selection

As stated previously, the conditional PCRLB provides a bound on the performance of estimating $\mathbf{x}(0\!:\!k)$ given that the past observations $\mathbf{z}(1\!:\!k-1)$ are known [152]. Contrary to its conventional counterpart, the conditional PCRLB does not assume the observations to be random. Instead the actual observations are used. The cost function $\mathcal{C}(k+1)$ used by the sensor selectors is now based on the conditional dPCRLBs related to the (x, y) coordinates of the target, i.e.,

$$\mathcal{C}(k+1) = [\boldsymbol{L}(\mathbf{x}(k+1))]^{-1}_{xx} + [\boldsymbol{L}(\mathbf{x}(k+1))]^{-1}_{yy}. \tag{6.26}$$

where $[\boldsymbol{L}(\mathbf{x}(k+1))]^{-1}_{xx}$ is the conditional dPCLRB corresponding to the x-coordinate at iteration $k+1$. Similarly, $[\boldsymbol{L}(\mathbf{x}(k+1))]^{-1}_{yy}$ is the conditional dPCLRB corresponding to the y-coordinate at iteration $k+1$. Similar to the previous section, the observation node selection is carried out in several iterations. During initialization at each iteration, the best observation node for each processing node is picked. One observation node among N_f selected sensors forms the initial neighbourhood. The process is repeated till the desired number of observation nodes is included in the neighbourhood set. To select the best observation node at each processing node, the following local cost function (expressed in terms of processing-node-observation-node (l, m_l) combination)

$$\mathcal{C}^{(l,m_l)}(t) = [\boldsymbol{L}^{(\aleph_{\text{obs}}^{(l)})}(t)]_{xx}^{-1} + [\boldsymbol{L}^{(\aleph_{\text{obs}}^{(l)})}(t)]_{yy}^{-1}. \tag{6.27}$$

where $[\boldsymbol{L}^{(\aleph_{\text{obs}}^{(l)})}(t)]_{xx}^{-1}$ and $[\boldsymbol{L}^{(\aleph_{\text{obs}}^{(l)})}(\mathbf{x}(k+1))]_{yy}^{-1}$ are the conditional dPCLRB corresponding to the x and y-coordinates in

$$\boldsymbol{L}^{(\aleph_{\text{obs}}^{(l)})}(t) = [\boldsymbol{C}^{22}(t)]^{(l,m_l)} - \boldsymbol{C}^{21}(k)\Big(\boldsymbol{J}_{\text{AUX}}^{(\text{min})}(\mathbf{x}(k)) + \boldsymbol{C}^{11}(k)\Big)^{-1}\boldsymbol{C}^{12}(k), \tag{6.28}$$

with

$$[\boldsymbol{C}^{22}(t)]^{(l,m_l)} = \boldsymbol{L}^{(l,m_l)}(\mathbf{x}(k+1)) + \boldsymbol{L}_{k+1|k+1}^{\text{min}}(t) - \boldsymbol{L}^{(l,m_l)}(\mathbf{x}(k+1|k)) - \boldsymbol{L}_{k+1|k}^{(\text{min})}(t) + \boldsymbol{Q}^{-1}(k). \tag{6.29}$$

Notation $\boldsymbol{L}^{(\aleph_{\text{obs}}^{(l)})}(t)$ correspond to the FIM for estimates obtained from the iterating neighbourhood $\aleph_{\text{obs}}^{(l)}(t)$ as it is being optimized. Once optimized, $\aleph_{\text{obs}}^{(l)}(k+1) = \aleph_{\text{obs}}^{(l)}(t)$. Parameters $\boldsymbol{C}^{21}(k) = [\boldsymbol{C}^{12}(k)]^T$ and $\boldsymbol{C}^{11}(k)$ are available from the conditional dPCRLB computation block and are fixed for various iterations of the senor selector. Parameter $\boldsymbol{J}_{\text{AUX}}^{(\text{min})}(\mathbf{x}(k))$ corresponds to the auxiliary PCRLB from the previously optimized neighbourhood in the last k iteration. Parameter $[\boldsymbol{C}^{22}(t)]^{(l,m_l)}$ is local for the (l, m_l) processing-node-observation-node combination and is obtained from Eq. (6.29). Parameter $\boldsymbol{L}^{(l,m_l)}(\mathbf{x}(k+1))$ and $\boldsymbol{L}^{(l,m_l)}(\mathbf{x}(k+1|k))$ are the conditional dPCRLBs corresponding to the filtering and prediction estimates obtained at processing node l from a single observation at observation node m_l. Finally, $\boldsymbol{L}_{k+1|k}^{(\text{min})}(t)$ and $\boldsymbol{L}_{k+1|k+1}^{(\text{min})}(t)$ are the conditional FIMs corresponding to the filtered and predicted estimates obtained from the iterating neighbourhood $\aleph_{\text{obs}}^{(l)}(t)$. Having defined the cost function, the iterative consensus-based distributed sensor selection approach is described next in terms of the following two steps.

1. *Initial Selection:* has the following sub-steps: (a) At processing node l, for $(1 \leq l \leq N_f)$ the conditional FIMs $\boldsymbol{L}^{(l,m_l)}(\mathbf{x}(k+1))$ and the cost function $\mathcal{C}^{(l,m_l)}(1)$ corresponding to

the processing-node-observation-node (l, m_l) combination are computed based on (6.27)-(6.29). (b) From all (l, m_l) combinations, the processing node l selects one observation node for which $\mathcal{C}^{(l,m_l)}(1)$ is minimum. In other words, a single observation node is selected by each processing node that provides the optimal performance at that node when at the most one observation is used. (c) At this stage, a complete enumeration encompassing all processing nodes ($1 \leq l \leq N_f$) is performed. We select one processing-node-observation-node combination ($q = l, m_q = m_l$) with the minimum cost function associated to it across the network. A minimum consensus algorithm accomplishes Step 1.c. (d) Matrices

$$\boldsymbol{L}^{(q,m_q)}(\mathbf{x}(k+1)) \triangleq \boldsymbol{L}^{(\min)}_{k+1|k}(1)$$

and

$$\boldsymbol{L}^{(q,m_q)}(\mathbf{x}(k+1|k)) \triangleq \boldsymbol{L}^{(\min)}_{k+1|k+1}(1)$$

corresponding to the conditional FIMs for the combination (q, m_q) are communicated across the network. The neighbourhood structure is given by $\aleph(1) = \{\aleph^{(l)}_{\text{obs}}(1)\}^{N_f}$. After the initial selection, all $\aleph^{(l)}_{\text{obs}}(1) = \{\}$ (i.e., empty sets) except for $l = q$ where $\aleph^{(q)}_{\text{obs}} = \{m_q\}$. Note that we have added time index $t = 1$ to each neighbourhood to indicate the iteration number for the fusion selection stage. The FIMs $\boldsymbol{L}^{(l,m_l)}(\mathbf{x}(k+1))$ computed in Step 1.a are limited to the sensors within the neighbourhood of processing node l.

2. *Subsequent Selection:* is based on the following substeps: Each processing node l, ($1 \leq l \leq N_f$), selects an observation node in its immediate neighbourhood and for it computes the cost function taking into account the previously selected neighbourhood ($\aleph^{(l)}_{\text{obs}}(t)$) and the associated FIMs $\boldsymbol{L}^{(\min)}_{k+1|k}(t)$ and $\boldsymbol{L}^{(\min)}_{k+1|k+1}(t)$. (a) Processing node l computes $[\boldsymbol{C}^{22}(t)]^{(l,m_l)}$, for ($m_l \notin \aleph^{(l)}_{\text{obs}}(t)$), using (6.28) and (6.29). (b) Given $\boldsymbol{L}^{(\aleph^{(l)}_{\text{obs}})}(t)$, Eq. (6.27) is used to compute the local cost function $\mathcal{C}^{(l,m_l)}(t)$. (c) Select the processing node $\mathcal{L}$ and observation

node $m_{\mathcal{L}}$ combination corresponding to the minimum overall cost function using a minimum consensus algorithm. (d) Append the neighbourhood structure to include the new combination $N_{\text{obs}}^{(l)}(t{+}1) = \{N_{\text{obs}}^{(l)}(t)\}$, appended with the new combination. The overall FIM corresponding to the appended neighbourhood combination is denoted by $\boldsymbol{L}^{(\mathcal{L},m_{\mathcal{L}})}(\mathbf{x}(k{+}1))$. (e) Matrix $\boldsymbol{L}^{(\min)}(\mathbf{x}(k{+}1))$ now equals to $\boldsymbol{L}^{(\mathcal{L},m_{\mathcal{L}})}(\mathbf{x}(k{+}1))$, which now corresponds to the overall conditional FIM corresponding to the selected sensors. The new value of matrix $\boldsymbol{L}^{(\min)}(\mathbf{x}(k+1))$ is communicated across the network.

3. *Termination:* Check if $N_{\max}$ has been reached. Else, go to Step 2.

Although the conditional PCRLB is an effective sensor resource management criteria for large, geographically distributed sensor networks, the proposed algorithm for distributed computation of the conditional PCRLB (dPCRLB) is based on *raw observations* leading to significant communication overhead to the estimation mechanism. The next section derives distributed computational techniques for determining the conditional dPCRLB for quantized, distributed AN/SN systems, referred to as the CQ/dPCRLB. Analytical expressions for the CQ/dPCRLB are derived, which are particularly useful for particle filter-based estimators.

6.5 Conditional PCRLB for Quantized Distributed Particle Filters

The section extends the conditional dPCRLB framework to quantized observations with emphasis on particle filter estimators. Additional contributions of the section include: (a) Both computational and communication complexity of conditional dPCRLB (Section 5.3) are reduced in the proposed conditional dPCRLB with quantized observations (CQ/dPCRLB). (b) In Section 5.3 and Section 6.4 the conditional FIM, i.e., the inverse of the conditional dPCRLB, is expressed as a function of the auxiliary FIM which is updated distributively at each iteration. The CQ/dPCRLB updates the conditional dPCRLB directly without the need of computing the auxiliary FIM leading to significant communication savings. Next, I formulate the distributed estimation framework with quantized observations

6.5.1 Distributed Estimation with Quantized Observations

Similar to the model presented in Section 6.1, processing node l, $(1 \leq l \leq N_f)$, is connected to a set of sensor nodes with only a subset active at each iteration. The active sensors connected to node l constitute its local observation neighbourhood $\aleph_{\text{obs}}^{(l)}$. The total number of active sensors in the network is $N_{ss} = \sum_{l=1}^{N_f} |\aleph_{\text{obs}}^{(l)}|$, where $|\cdot|$ denotes the cardinality operator. Sensor m in the observation neighbourhood of node l, i.e., $m \in \aleph_{\text{obs}}^{(l)}$, makes observation $Z^{(l,m)}(k)$. Instead of transferring the raw observation, sensor m communicates its quantized version $Y^{(l,m)}(k)$ to the processing node l based on the following model

$$Y^{(l,m)}(k) = \boldsymbol{Q}^{(l,m_l)}\big(\underbrace{g^{(l,m)}(\mathrm{x}(k)) + \zeta^{(l,m)}(k)}_{Z^{(l,m)}(k)}\big), \tag{6.30}$$

where $\boldsymbol{Q}^{(l,m)}(\cdot)$ is the local quantization operator at node l, and $\boldsymbol{g}^{(l,m)}(\cdot)$ and $\zeta^{(l,m)}(\cdot)$ are, respectively, the local observation model and uncertainty at sensor m connected to processing node l. For simplicity and without loss of generality, the quantization operators $\boldsymbol{Q}^{(l,m)}(\cdot)$ are considered to be the same across the network (i.e., $\boldsymbol{Q}^{(l,m)}(\cdot) = \boldsymbol{Q}(\cdot)$). Collectively, the overall quantized observation vector at node l is denoted by

$$\boldsymbol{y}^{(l)}(k) = \{Y^{(l,m)}(k) : m \in \aleph_{\text{obs}}^{(l)}\}, \quad \text{for } (1 \leq l \leq N_f). \tag{6.31}$$

Depending on how many sensors are activated by the processing node l, the dimension of the observation vector $\boldsymbol{y}^{(l)}(k)$ is different at each processing node. As for the quantized observations $\boldsymbol{y}^{(l)}(k)$, vector $\mathbf{z}^{(l)}(k)$ is the collection of all raw observations associated with the processing node

l, i.e.,

$$\mathbf{z}^{(l)}(k) = \{Z^{(l,m)}(k) : m \in \aleph^{(l)}_{\text{obs}}\}, \quad \text{for } (1 \leq l \leq N_f). \tag{6.32}$$

In other words, $\boldsymbol{y}^{(l)}(k)$ is the quantized version of $\mathbf{z}^{(l)}(k)$. An N_L-bit quantization scheme is considered, where node m's quantized observation $Y^{(l,m)}(k)$ can take any discrete value between 0 and $2^{N_L} - 1$. The set of quantization threshold is denoted by $\boldsymbol{q} = [q_0, q_1, \ldots, q_{2^{N_L}-1}]$ where for brevity $q_0 = -\infty$ and $q_{2^{N_L}} = \infty$. The likelihood that $Y^{(l,m)}(k)$ is at level q_i is denoted by $\boldsymbol{h}_i^{(l,m)}(k) \triangleq P(Y^{(l,m)}(k) = q_i | \mathbf{x}(k))$ with

$$\begin{aligned} \boldsymbol{h}_i^{(l,m)}(k) &= P(q_i \leq Z^{(l,m)}(k) \leq q_{i+1} | \mathbf{x}(k)) \\ &= P\Big(\big[q_i - \boldsymbol{g}^{(l,m)}(\mathbf{x}(k))\big] \leq \boldsymbol{\zeta}^{(l,m)}(k) \leq \big[q_{i+1} - \boldsymbol{g}^{(l,m)}(\mathbf{x}(k))\big] \Big) \end{aligned} \tag{6.33}$$

Section 5.1 reviews the local conditional dPCRLB for raw observations as presented in Section 5.3 with one proposed modification.

6.5.2 Modified Conditional *d*PCRLB for Raw Observations

Based on the conditional PCRLB inequality, the mean square error (MSE) associated with the local estimate $\hat{\mathbf{x}}^{(l)}(0\!:\!k+1)$ of the state vector at node l is lower bounded as follows

$$\mathbb{E}_{P_c^{(l)}(k+1)} \{ \boldsymbol{e}^{(l)}(0\!:\!k+1) (\boldsymbol{e}^{(l)}(0\!:\!k+1))^T \} \geq [\boldsymbol{I}^{(l)}(\mathbf{x}(0\!:\!k+1))]^{-1},$$

where $P_c^{(l)}(k+1) \triangleq P(\mathbf{x}(0\!:\!k), \mathbf{z}^{(l)}(k+1) | \mathbf{z}^{(l)}(1\!:\!k))$, $\mathbb{E}\{\cdot\}$ denotes expectation, and $\boldsymbol{e}^{(l)}(0\!:\!k+1) \triangleq \mathbf{x}(0\!:\!k+1) - \hat{\mathbf{x}}^{(l)}(0\!:\!k+1)$ is the estimation error. The local <u>accumulated</u> conditional FIM $\boldsymbol{I}^{(l)}(\mathbf{x}(0\!:\!k+1))$ corresponds to the state trajectory $\hat{\mathbf{x}}^{(l)}(0\!:\!k+1)$ from iteration 0 to $k+1$ and is given by

$$\boldsymbol{I}^{(l)}(\mathbf{x}(0\!:\!k+1)) \triangleq \mathbb{E}_{P_c^{(l)}(k+1)} \{ -\Delta^{\mathbf{x}(0:k+1)}_{\mathbf{x}(0:k+1)} \log P_c^{(l)}(k+1) \}. \tag{6.34}$$

Another local FIM is the local <u>instantaneous</u> conditional FIM $\boldsymbol{L}^{(l)}(\mathbf{x}(k+1))$ associated with $\hat{\mathbf{x}}^{(l)}(k+1)$, which is obtained by taking the inverse of $(n_x \times n_x)$ right-lower block of $[\boldsymbol{I}^{(l)}(\mathbf{x}(0\colon k+1))]^{-1}$. Please refer to

Below, I further highlight the relationship between the local accumulated conditional FIM $\boldsymbol{I}^{(l)}(\mathbf{x}(0\colon k+1))$ and local instantaneous conditional FIM $\boldsymbol{L}^{(l)}(\mathbf{x}(k+1))$. The local instantaneous conditional FIM $\boldsymbol{L}^{(l)}(\mathbf{x}(k+1))$ is computed using either of the following three approaches: (i) Directly by inverting large matrix $\boldsymbol{I}^{(l)}(\mathbf{x}(0\colon k+1))$; (ii) Recursively as a function of the previous local instantaneous auxiliary FIM $\boldsymbol{J}^{(l)}_{\text{AUX}}(\mathbf{x}(k))$ (Section 5.3), and; (iii) Recursively as a function of the previous local instantaneous conditional FIM $\boldsymbol{L}^{(l)}(\mathbf{x}(k))$ presented below in Result 1. In approach (i), first the local accumulated conditional FIM $\boldsymbol{I}^{(l)}(\mathbf{x}(0\colon k+1))$ is factorized as follows

$$\boldsymbol{I}^{(l)}(0\colon k+1) = \begin{bmatrix} [A^{11}(k+1)]^{(l)} & [A^{12}(k+1)]^{(l)} \\ [A^{21}(k+1)]^{(l)} & [A^{22}(k+1)]^{(l)} \end{bmatrix} = \mathbb{E}\left\{ - \left[\begin{array}{c:c} \Delta^{\mathbf{x}(0:k)}_{\mathbf{x}(0:k)} & \Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(0:k)} \\ \hdashline \Delta^{\mathbf{x}(0:k)}_{\mathbf{x}(k+1)} & \Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)} \end{array}\right] \log P^{(l)}_{c}(k+1)) \right\}. \tag{6.35}$$

Then, the local instantaneous conditional FIM $\boldsymbol{L}^{(l)}(\mathbf{x}(k+1))$ associated with the estimate $\hat{\mathbf{x}}(k+1)$ is obtained by taking the inverse of the $(n_x \times n_x)$ right-lower square block of $[\boldsymbol{I}^{(l)}(\mathbf{x}(0\colon k+1))]^{-1}$ by applying the matrix inversion Lemma 3. Based on Lemma 3, the local instantaneous conditional FIM is given by

$$\boldsymbol{L}^{(l)}(k+1) = [\boldsymbol{A}^{22}(k+1)]^{(l)} - [\boldsymbol{A}^{21}(k+1)]^{(l)}[\boldsymbol{A}^{22}(k+1)]^{(l)^{-1}}[\boldsymbol{A}^{12}(k+1)]^{(l)}. \tag{6.36}$$

which requires inversion of large matrix $[A^{11}(k+1)]^{(l)}$. Next, I describe approach (iii) in more details. Node l updates its local conditional FIM $\boldsymbol{L}^{(l)}(\mathbf{x}(k+1))$ as follows.

Result 1. *The instantaneous local FIM* $\boldsymbol{L}^{(l)}(\mathbf{x}(k+1))$ *associated with estimate* $\hat{\mathbf{x}}^{(l)}(k+1)$ *at node*

l is computed as follows

$$\begin{aligned}
\boldsymbol{L}^{(l)}(\mathbf{x}(k+1)) &\approx \big[\boldsymbol{B}^{22}(k)\big]^{(l)} - \big[\boldsymbol{B}^{21}(k)\big]^{(l)}\Big(\boldsymbol{L}^{(l)}(\mathbf{x}(k))+\big[\boldsymbol{B}^{11}(k)\big]^{(l)}\Big)^{-1}\big[\boldsymbol{B}^{12}(k)\big]^{(l)}, && (6.37)\\
\big[\boldsymbol{B}^{11}(k)\big]^{(l)} &= \mathbb{E}\big\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}, && (6.38)\\
\big[\boldsymbol{B}^{12}(k)\big]^{(l)} &= \mathbb{E}\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k+1)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\} && (6.39)
\end{aligned}$$

and

$$\big[\boldsymbol{B}^{22}(k)\big]^{(l)} = \mathbb{E}\big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\} + \mathbb{E}\big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log P\big(\mathbf{z}^{(l)}(k+1)|\mathbf{x}(k+1)\big)\big\}. \quad (6.40)$$

The derivation of Result 1 is included in Appendix D.2. In Chapter 5, $\boldsymbol{L}^{(l)}(\mathbf{x}(k+1))$ is computed recursively from the local instantaneous auxiliary FIM $[\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(k))]^{(l)}$ which is the inverse of $(n_x \times n_x)$ right-lower square block of the accumulated auxiliary FIM $[\boldsymbol{J}_{\mathrm{AUX}}^{(l)}(\mathbf{x}(0\colon k))]^{-1}$. The latter is defined as

$$\boldsymbol{J}_{\mathrm{AUX}}^{(l)}(\mathbf{x}(0\colon k)) \triangleq \mathbb{E}_{P_a^{(l)}(k+1)}\big\{-\Delta_{\mathbf{x}(0:k)}^{\mathbf{x}(0:k)}\log P_a^{(l)}(k)\big\} \quad (6.41)$$

with $P_a^{(l)}(k) \triangleq P(\mathbf{x}(0\colon k)|\mathbf{z}^{(l)}(1\colon k))$. The algorithm proposed in Chapter 5, therefore, requires distributed fusion of both the local FIMs and the local auxiliary FIMs, while Result 1 eliminates the need for fusing the local instantaneous auxiliary FIMs and, therefore, cuts the communication overhead by half.

Distributed computation of the conditional PCRLB requires a recursive expression for the predictive local conditional FIM $\boldsymbol{L}^{(l)}(\mathbf{x}(k+1|k))$ which is similar to (6.37) except $[\boldsymbol{B}^{22}(k)]^{(l)}$ is substituted with $[\boldsymbol{B}_p^{22}(k)]^{(l)}$ as

$$\big[\boldsymbol{B}_p^{22}(k)\big]^{(l)}=\mathbb{E}\big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}. \quad (6.42)$$

Having computed the local FIMs $\boldsymbol{L}^{(l)}(\mathbf{x}(k+1))$ and the local prediction FIMs $\boldsymbol{L}^{(l)}(\mathbf{x}(k+1|k))$ at iteration $k+1$, the next step in the conditional dPCRLB is to fuse these local FIMs to compute the global instantaneous conditional FIM $\boldsymbol{L}^{(\mathrm{G})}(\mathbf{x}(k+1))$. In Chapter 5, I derived a fusion rule for assimilating local conditional FIMs into the global conditional FIM when raw observations are available at each local node. Section 6.5.3 extends the derivations to quantized observations and eliminates the need for fusion of local instantaneous auxiliary FIMs.

6.5.3 CQ/*d*PCRLB with Quantized Observations

In Result 1, raw observations $Z^{(l,m)}(k)$ are replaced with their quantized version $Y^{(l,m)}(k)$, which results in the quantized filtering conditional FIM $\boldsymbol{L}_{\mathrm{Q}}^{(l)}(\mathbf{x}(k+1))$. Since terms $[\boldsymbol{B}^{11}(k)]^{(l)}$, $[\boldsymbol{B}^{12}(k)]^{(l)}$, $[\boldsymbol{B}^{21}(k)]^{(l)}$ are based on the state model, they remain the same. Term $[\boldsymbol{B}^{22}(k)]^{(l)}$ in Eq. (6.40) is now computed using the quantized observation as follows

$$[\boldsymbol{B}_{\mathrm{Q}}^{22}(k)]^{(l)} = \mathbb{E}\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log P(\mathbf{x}(k+1)|\mathbf{x}(k))\} + \underbrace{\mathbb{E}\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log P(\boldsymbol{y}^{(l)}(k+1)|\mathbf{x}(k+1))\}}_{\boldsymbol{J}(\boldsymbol{y}^{(l)}(k+1))}. \quad (6.43)$$

To compute $\boldsymbol{J}(\boldsymbol{y}^{(l)}(k+1))$, the likelihood $P(\boldsymbol{y}^{(l)}(k+1)|\mathbf{x}(k+1))$ along with the second derivative of its logarithmic function is needed. Because of quantized observations, $P(\boldsymbol{y}^{(l)}(k+1)|\mathbf{x}(k+1))$ transforms into a probability mass function that is discrete with second derivative replaced by a double summation as described below. Given the state variables, local observations are assumed independent such that

$$\boldsymbol{J}(\boldsymbol{y}^{(l)}(k+1)) = \sum_{m\in\aleph_{\mathrm{obs}}^{(l)}(k)} \boldsymbol{J}(Y^{(l,m)}(k+1)), \quad (6.44)$$

$$\text{where} \quad \boldsymbol{J}(Y^{(l,m)}(k+1)) = \sum_{i=1}^{N_L} -\mathbb{E}\left\{\delta\left(Y^{(l,m)}(k+1)-i\right)\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)}\log\left(\boldsymbol{h}_i^{(l,m)}(k)\right)\right\} \quad (6.45)$$

and $\delta(\cdot)$ is the delta function. We note that $\mathbb{E}\{\delta(Y^{(l,m)}(k+1)-i)\} = \boldsymbol{h}_i^{(l,m)}(k)$, where $\boldsymbol{h}_i^{(l,m)}(k)$ was defined immediately after Eq. (6.33) previously and has the second derivative

$$\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \log(\boldsymbol{h}_i^{(l,m)}(k)) = \begin{bmatrix} \frac{\partial^2 \log(\boldsymbol{h}_i^{(l,m)}(k))}{(\partial(X_1(k)))^2} & \cdots & \frac{\partial^2 \log(\boldsymbol{h}_i^{(l,m)}(k))}{\partial(X_1(k))\partial(X_{n_x}(k))} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \log(\boldsymbol{h}_i^{(l,m)}(k))}{\partial(X_{n_x}(k))\partial(X_1(k))} & \cdots & \frac{\partial^2 \log(\boldsymbol{h}_i^{(l,m)}(k))}{(\partial(X_{n_x}(k)))^2} \end{bmatrix}. \tag{6.46}$$

Under mild regularity conditions, the expected value of (6.46) is equal to the variance of its first moment, i.e.,

$$\mathbb{E}\Big\{\frac{\partial^2 \log\big(\boldsymbol{h}_i^{(l,m)}(k)\big)}{\partial(X_j(k))\partial(X_u(k))}\Big\} = -\mathbb{E}\Big\{\frac{\frac{\partial \boldsymbol{h}_i^{(l,m)}(k)}{\partial(X_j(k))}\frac{\partial \boldsymbol{h}_i^{(l,m)}(k)}{\partial(X_u(k))}}{\big(\boldsymbol{h}_i^{(l,m)}(k)\big)^2}\Big\}. \tag{6.47}$$

Eqs. (6.44)-(6.47) are used to compute $[\boldsymbol{B}_{\mathrm{Q}}^{22}(k)]^{(l)}$. Finally, the local quantized filtering FIM is given by

$$\boldsymbol{L}_{\mathrm{Q}}^{(l)}(\mathbf{x}(k+1)) \approx [\boldsymbol{B}_{\mathrm{Q}}^{22}(k)]^{(l)} - [\boldsymbol{B}^{21}(k)]^{(l)}\Big(\boldsymbol{L}_{\mathrm{Q}}^{(l)}(\mathbf{x}(k)) + [\boldsymbol{B}^{11}(k)]^{(l)}\Big)^{-1}[\boldsymbol{B}^{12}(k)]^{(l)}. \tag{6.48}$$

Eq. (6.48) is derived by applying the following factorization

$$P(\mathbf{x}(0:k+1), \boldsymbol{y}^{(l)}(1:k+1)) = P\big(\mathbf{x}(0:k), \boldsymbol{y}^{(l)}(1:k)\big)P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)P(\boldsymbol{y}^{(l)}(k+1)|\mathbf{x}(k+1)), \tag{6.49}$$

to the quantized version of Eq. (6.34) and then taking the inverse of the $(n_x \times n_x)$ right lower block of $[\boldsymbol{I}_{\mathrm{Q}}^{(l)}(\mathbf{x}(0:k+1))]^{-1}$. The similarity between Eqs. (6.37) and (6.48) is intuitively pleasing. The local predictive FIM $\boldsymbol{L}_{\mathrm{Q}}^{(l)}(\mathbf{x}(k+1|k))$ is derived in the similar manner as (6.48) with $[\boldsymbol{B}^{22}(k)]^{(l)}$ replaced by (6.42)

Fusing Local FIMs (CQ/dPCRLB): Result 2 provides a fusion rule for assimilating the local FIMs with quantized observations to compute the global quantized FIM.

Result 2. *The sequence $\{\boldsymbol{L}_Q^{(G)}(\mathbf{x}(k+1))\}$ corresponding to the global information submatrix (CQ/dPCRLB) with quantized local observations follows the following recursion*

$$\begin{aligned}
\boldsymbol{L}_Q^{(G)}(\mathbf{x}(k{+}1)) &\approx \boldsymbol{C}_Q^{22}(k) - \boldsymbol{C}_Q^{21}(k)\big(\boldsymbol{L}_Q^{(G)}(\mathbf{x}(k)) + \boldsymbol{C}_Q^{11}(k)\big)^{-1}\boldsymbol{C}_Q^{12}(k) && (6.50)\\
\textit{where } \boldsymbol{C}_Q^{11}(k) &= \mathbb{E}\big\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}, && (6.51)\\
\boldsymbol{C}_Q^{12}(k) &= \mathbb{E}\big\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}, && (6.52)
\end{aligned}$$

and

$$\boldsymbol{C}_Q^{22}(k) \approx \sum_{l=1}^{N_f} \boldsymbol{L}_Q^{(l)}(\mathbf{x}(k+1)) - \sum_{l=1}^{N_f} \boldsymbol{L}_Q^{(l)}(k+1|k) + \mathbb{E}\big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}. \quad (6.53)$$

The proof of Result 2 is included in Appendix D.3.

Gaussian Observation Noise: The analytical expressions are derived for the case when local observations $Z^{(l,m)}(k)$ are zero-mean Gaussian with variance $R^{(l,m)}(k)$, i.e., $Z^{(l,m)}(k) \sim \mathcal{N}(0, R^{(l,m)}(k))$. The likelihood that $Y^{(l,m)}(k)$ is at level q_i is

$$\begin{aligned}
\boldsymbol{h}_i^{(l,m)}(k) &= \frac{1}{\sqrt{2\pi R^{(l,m)}(k)}} \int_{q_i - \boldsymbol{g}^{(l,m)}(\mathbf{x}(k))}^{q_{i+1} - \boldsymbol{g}^{(l,m)}(\mathbf{x}(k))} \exp\Big\{\frac{-t}{2R^{(l,m)}(k)}\Big\} dt\\
&= \Phi\left(\frac{q_i - \boldsymbol{g}^{(l,m)}(\mathbf{x}(k))}{\sqrt{R^{(l,m)}(k)}}\right) - \Phi\left(\frac{q_{i+1} - \boldsymbol{g}^{(l,m)}(\mathbf{x}(k))}{\sqrt{R^{(l,m)}(k)}}\right), \quad (6.54)
\end{aligned}$$

where $\Phi(\cdot)$ is the standard cumulative Gaussian distribution. Based on (6.54), each derivative term in Eq. (6.47) is represented as

$$\frac{\partial \boldsymbol{h}_i^{(l,m)}(k)}{\partial(X_u(k))} = -\frac{\frac{\partial \boldsymbol{g}^{(l,m)}(\mathbf{x}(k))}{\partial \mathbf{x}(k)}}{\sqrt{2\pi R^{(l,m)}(k)}} \left(\exp\Big(\frac{-(q_{i+1} - \boldsymbol{g}^{(l,m)}(\mathbf{x}(k)))^2}{2R^{(l,m)}(k)}\Big) - \exp\Big(\frac{-(q_i - \boldsymbol{g}^{(l,m)}(\mathbf{x}(k)))^2}{2R^{(l,m)}(k)}\Big)\right). \quad (6.55)$$

6.5.4 Computation of The Conditional dPCRLB

The analytical computation of the expectations in Result 2 is not practical and, therefore, particle filter-based approaches are proposed. If the state estimator is based on distributed particle filters [51], then the same particle set can be used in the CQ/dPCRLB algorithm. An active sensor communicates its quantized observation to the associated processing node. The processing nodes themselves communicate the local conditional FIMs and statistics of local posteriors (i.e., local state estimates and their corresponding covariance matrices) to the neighbouring processing nodes which are then fused in a distributed fashion to compute the global state estimate and the global conditional FIM. I explain the CQ/dPCRLB algorithm in the context of the CF/DPF implementation (Chapter 4) being used as the state estimator. Recall that the CF/DPF implements two particle filters at each node: (i) Local filter which approximates the local posterior at node l with a set of weighted particles $\{\mathbb{X}_i^{(l,\mathrm{LF})}(k), W_i^{(l,\mathrm{LF})}\}$, and; (ii) Fusion filter which combines the local posteriors to estimate the global posterior with a second set of particles $\{\mathbb{X}_i^{(l,\mathrm{FF})}(k), W_i^{(l,\mathrm{FF})}\}$. All information regarding the observations collected up to time k at node l, are presented in the local particles $\mathbb{X}_i^{(l,\mathrm{LF})}(k)$, while the information available across the network is provided by the global particles $\mathbb{X}_i^{(l,\mathrm{FF})}(k)$. The CQ/dPCRLB comprises of the following steps:

I. Local FIMs:

1. Eqs. (6.38)-(6.39) are computed at node l based on Monte-Carlo integration using local particles $\mathbb{X}_i^{(l,\mathrm{LF})}(k)$.

2. For computing Eq. (6.43), first, node l computes the predictive particles $\mathbb{X}_i^{(l,\mathrm{LF})}(k+1|k)$ by propagating $\mathbb{X}_i^{(l,\mathrm{LF})}(k)$ through $P(\mathbf{x}(k+1)|\mathbf{x}(k))$, and then computes Eq. (6.43) using $\mathbb{X}_i^{(l,\mathrm{LF})}(k)$ and $\mathbb{X}_i^{(l,\mathrm{LF})}(k+1|k)$.

3. The local FIMs are then computed using Eq. (6.48).

II. Global FIM:

4. The expectations in (6.51)-(6.53) are computed using the global particles $\mathbb{X}_i^{(l,\mathrm{FF})}(k)$ to derive the FIMs $C_{\mathrm{Q}}^{**}(k)$. Eq. (6.53) includes summation of local FIMs across the network typically

computed using the average consensus algorithms [55] in a distributed fashion.

5. Result 2 is used to compute the global FIM based on the local FIMs computed in Step 4.

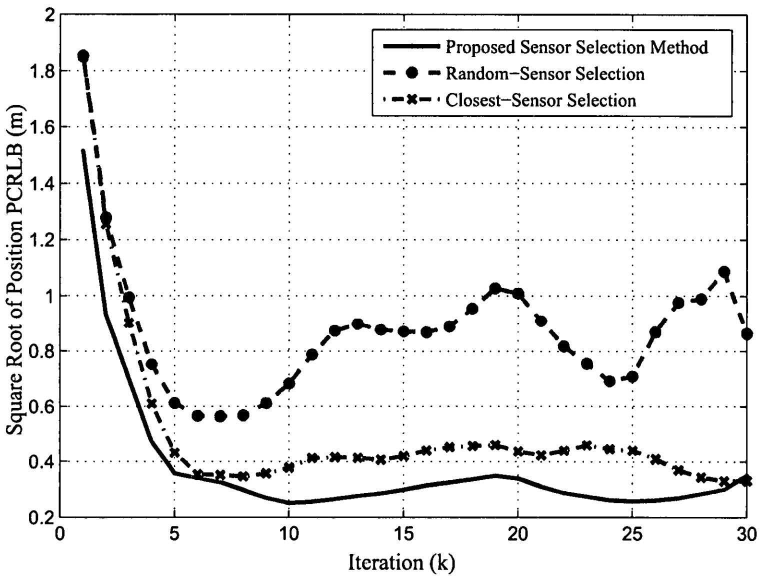
6.5.5 Communication Savings with CQ/dPCRLB

First, the transfer of quantized observation (instead of raw data) between sensors and associated processing nodes leads to significant communication savings. Second, the communication overhead for computing the global auxiliary FIM from the local auxiliary FIMs across the network is eliminated in the proposed CQ/dPCRLB algorithm. With average consensus [51], the second savings is of $O(n_x|\aleph^{(l)}_{\text{fuse}}|N_c)$ (i.e., the communication complexity reduces by half), where n_x is number of states, $|\aleph^{(l)}_{\text{fuse}}|$ the number of processing nodes in the neighbourhood of processing node l, and N_c the number of consensus iterations. The CQ/dPCRLB can be further extended to communicate quantized versions of the local state statistics (quantized local tracks [164]) and local FIMs between neighbouring processing nodes during the fusion filter stage which will be considered in future work.

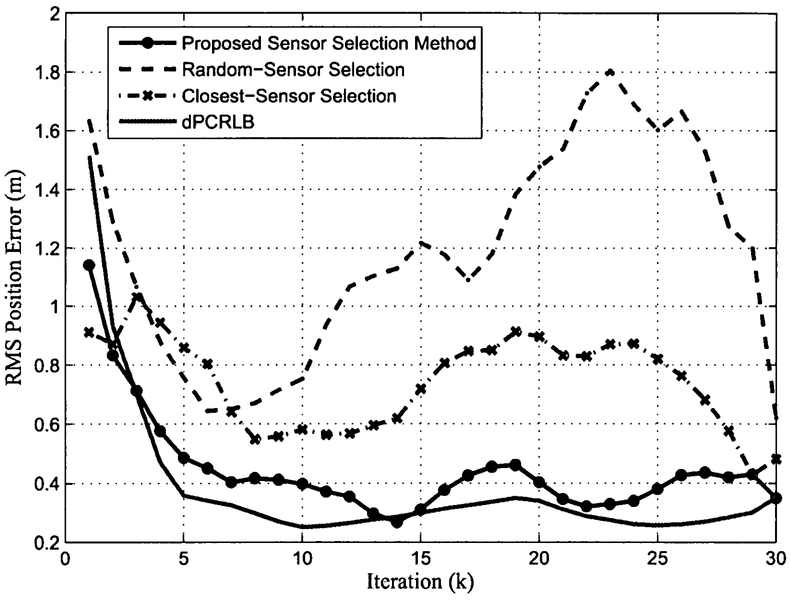
6.6 Simulation Results

In this section the proposed distributed sensor selection algorithms are implemented using the non-conditional dPCRLB in Section 6.6.1 and the conditional dPCRLB in Section 6.6.2, and are compared in performance with some of the existing sensor selection algorithms. In Section 6.6.3, the conditional dPCRLB for quantized distributed estimation proposed in Section 6.5 is likewise evaluated using the Monte Carlo simulations.

A large-scale distributed BOT application [103] based on Fig. 6.1 is simulated to test the proposed consensus-based dynamic sensor selection approaches. An AN/SN consisting of $N_{ss} = 225$ sensor nodes and $N_f = 9$ local processing nodes scattered in a square region of dimension



(a)



(b)

Figure 6.3: (a) The dPCRLB, and; (b) RMS error for target's position averaged over all processing nodes for the three approaches.

(1500×1500) m^2 is considered. For simplicity, the observation nodes are assumed distributed uniformly with the processing node at the centre of its rectangular (500×500)m neighbourhood. Each processing node communicates only with selected observation nodes within its rectangular (500×500)m neighbourhood and other processing nodes within a connectivity radius of 550 m. Each processing node linked to at least one other processing node in the network. The CCT kinematic motion model (Eq. 3.50) defines the state model. Measurements are the target's bearings with respect to the platform of each node referenced (clockwise positive) to the y-axis as follows

$$Z^{(l,m_l)}(k) = \text{atan}\left[\frac{X(k) - X^{(l,m_l)}}{Y(k) - Y^{(l,m_l)}}\right] + \zeta^{(l,m_l)}(k), \tag{6.56}$$

where $\{X^{(l,m_l)}, Y^{(l,m_l)}\}$ represents the coordinates of sensor (l, m_l), i.e., sensor m_l connected to processing node l, for $(1 \leq l \leq N_f)$. Both state and observation noises are normally distributed with the observation noise $(\zeta^{(l,m_l)}(k))$ assumed to be state dependent such that the variance of the observation noise at sensor node (l, m_l) given by

$$\sigma^2_{\zeta^{(l,m_l)}}(k) = 0.01150 r^{(l,m_l)}(k) + 0.7405, \tag{6.57}$$

depends on the distance $r^{(l,m_l)}(k)$ between sensor node (l, m_l) and target. Consequently, the SNR is time-varying and differs from one sensor node to the other depending on the location of the target.

6.6.1 Non-Conditional dPCRLB-based Sensor Selection

In this section, the sensor selection algorithm based on the non-conditional dPCRLB proposed in Section 6.3 is evaluated through Monte Carlo simulations. The maximum number of active observation nodes at each iteration is $N_{\max} = 32$ with the additional constraint that each processing node can at the most select four sensors. Since the distributed dynamical system is non-linear, the distributed particle filter implementation (CF/DPF (Chapter 4)) is used to track the targets, compute the local FIMs $\boldsymbol{J}^{(l)}(\mathbf{x}(k+1))$ and $\boldsymbol{J}^{(l)}(\mathbf{x}(k+1|k))$, for $(1 \le l \le N_f)$, and evaluate the global FIM $\boldsymbol{J}(\mathbf{x}(k+1))$. The number of vector particles used at each processing nodes is $N_s = 1000$. The dPCRLB sensor selection approach is compared with other distributed sensor selection approaches [15] as follows.

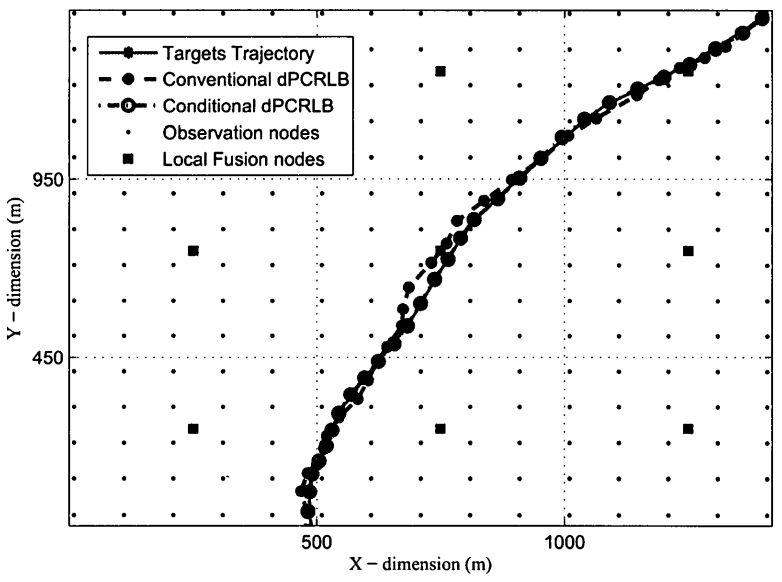
1. *Random-sensor approach:* Observation nodes are selected randomly by each processing node from within its neighbourhood.

2. *Closest-sensor approach:* If a target is present in the neighbourhood of a processing node, observation nodes closest to the estimated location of the target are selected. Else, sensors are selected randomly from the processing node's neighbourhood.

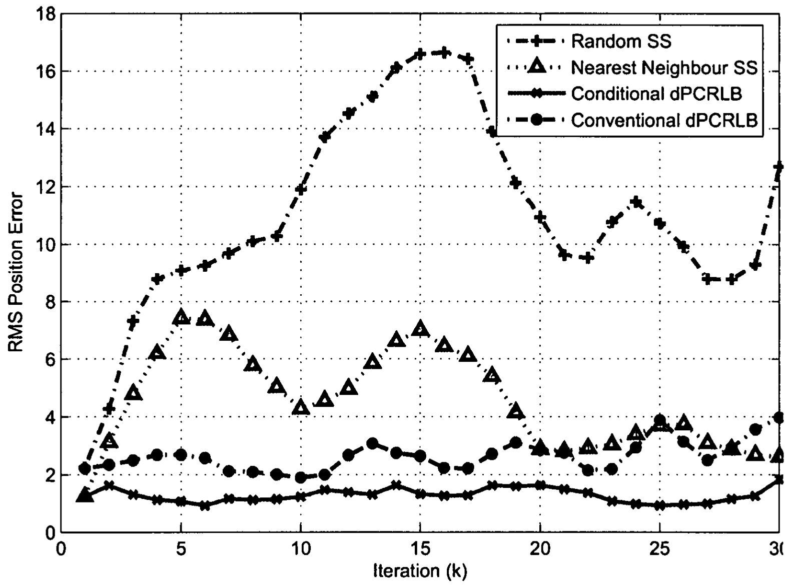
In the experiments, a single target starts its maneuver from coordinates $\{100, 1400\}$. The initial course is set at $-140°$ with the standard deviation of the process noise $\sigma_v = 1.6$. Fig. 6.3(a) shows the position PCRLB for the three sensor selection approaches based on the selected sensors. The RMS error for the three approaches with the CF/DPF as the estimation algorithm are plotted in Fig. 6.3(b). In Fig. 6.3(a), the dPCRLB based sensor selection approach provides the minimum lower error bound as well as the minimum RMSE as shown in Fig. 6.3(b). Next, the conditional dPCRLB based sensor selection algorithm is evaluated. Fig. 6.3 reinforces our earlier result of the superiority of the dPCRLB based sensor selection approach.

6.6.2 Conditional dPCRLB based Sensor Selection

In this section, the sensor selection algorithm based on the conditional dPCRLB proposed in Section 6.4 is evaluated through Monte Carlo simulations. In other words, the sensor selection procedure is the same as in Section 6.6.1 except for using the conditional PCRLB as the selection criteria versus non-conditional PCRLB used in Section 6.6.1. As in the previous section, a large-



(a)



(b)

Figure 6.4: (a) Target's position alongside with the sensor nodes and observation nodes positions. (b) RMSE for target's position averaged over all fusion nodes.

scale distributed BOT application is simulated based on an AN/SN consisting of $N_{ss} = 225$ sensor nodes and $N_f = 9$ fusion nodes scattered in a square region of dimension (1500×1500) m^2. A single target scenario is considered with the target starts its maneuver from coordinates $(1400, 1400)$. The initial course is set at $-140°$ with the standard deviation of the process noise $\sigma_v = 1.6$. The maximum number N_{max} of active observation nodes at each iteration is different from the earlier setup and set to 18 with the constraint that each processing node (shown as '■') can at the most select four sensors. The measurement equation is given by Eq. (6.56) and the target movies according to a CCT motion model given by Eq. (3.50) with maneuver acceleration parameter A_m set to $1.08 \times 10^{-5}\text{km/s}^2$. Fig. 6.4(a) shows the target tracks together with location of observation nodes and local processing nodes. The variance of the observation noise at sensor node (l, m_l) is given by Eq. (6.57) which considers a state dependent noise model such that the bearing noise variance at sensor node (l, m_l) depends on the distance $r^{(l,m_l)}(k)$ between sensor node (l, m_l) and target. Consequently, the SNR is time-varying and differs from one sensor node to the other depending on the location of the target. As stated previously the CF/DPF [50] is used to track the targets and compute the local FIMs. The conditional dPCRLB sensor selection approach is compared with other distributed approaches [15,67] as follows:

1. *Non-conditional dPCRLB-based sensor selection:* where the conventional dPCRLB is the selection criteria.

2. *Random-sensor approach:* Observation nodes are selected randomly by each processing node from within its neighbourhood.

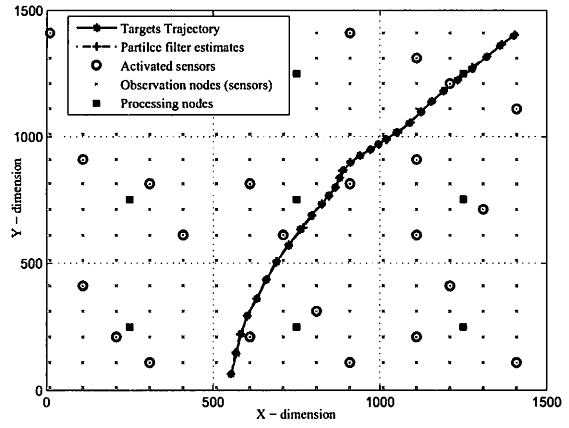
3. *Closest-sensor approach:* where the observation nodes closest to the estimated location of the target are selected.

Fig. 6.4(b) shows the position RMSE for the four sensor selection approaches. The conditional dPCRLB based sensor selection approach outperforms the other methods and provides the minimum RMSE as shown in Fig. 6.4(b). Next, the conditional dPCRLB for quantized distributed estimation proposed in Section 6.5 is considered.

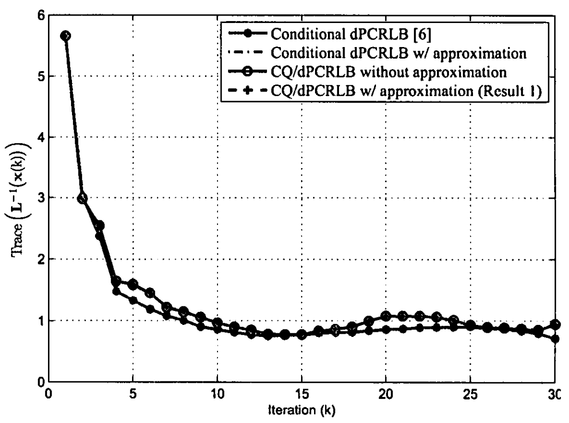
6.6.3 Conditional dPCRLB for Quantized Distributed AN/SN Systems

In this section, the performance of the CQ/dPCRLB algorithm proposed in Section 6.4 is evaluated through Monte Carlo simulations. Similar to the previous section, a large-scale distributed bearing-only tracker with nonlinear CCT model [51] given by Eq. (3.50) is considered. The observations are bearing measurements given by Eq. 6.56. Both process and observation noises are normally distributed with the observation noise ($\zeta^{(l,m_l)}(k)$) model assumed to be state dependent such that the bearing noise variance at sensor (l, m_l) depends on the distance between the observer and target. A agent network (Fig. 6.5(a)) consisting of 225 static sensors and $N_f = 9$ processing nodes scattered in a square region of dimension $(1500 \times 1500)\text{m}^2$ is implemented. Our goal is to evaluate the performance of the proposed CQ/dPCRLB, therefore, the activated sensors are selected at random and limited to three sensors per processing node.

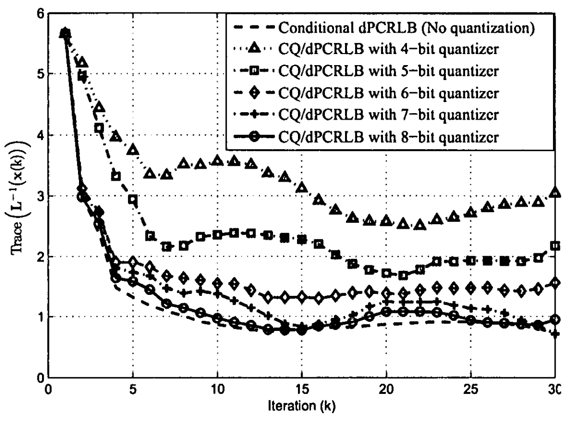
The objective of the Monte Carlo simulations in this section is three folds. The first objective is to validate the effectiveness of the conditional FIM approximation (i.e., to replace the global auxiliary FIM with the global conditional FIM) in Result 2. Fig. 6.5(b) plots the conditional dPCRLB and CQ/dPCRLB with and without the proposed global conditional FIM approximation. In each case, results for both raw (bottom two plots) and quantized (top two plots) observations are included. Within each set of plots in Fig. 6.5(b), the bounds virtually overlap verifying the effectiveness of the global conditional FIM approximation. The second objective is to compare the CQ/dPCRLB with quantized observations for accuracy against the conditional dPCRLB com-



(a)



(b)



(c)

Figure 6.5: (*a*) A sample decentralized bearing only tracking setup. (*b*) Comparison of the conditional dPCRLBs [55] using raw observations with the CQ/dPCRLBs using 8-bit quantized observations. (c) Effect of quantization on the CQ/dPCRLB for different (4, 5, 6, 7, and 8 bit) quantization levels.

puted from raw observations [55]. Comparing bounds across the two sets of plots in Fig. 6.5(b), it is observed that the respective plots do not overlap but are fairly close to each other. Despite using quantized observations, the CQ/dPCRLB is a reasonable approximation of the dPCRLB. Illustrated in Fig. 6.5(c), the third objective is to quantify the potential CQ/dPCRLB performance loss as a function of the number of quantization levels. The CQ/dPCRLB approaches the dPCRLB as the number of quantization levels are increased. The relative performance gain with an increased number of quantization levels decreases beyond an 8-bit quantizer in our setup. The CQ/dPCRLB from an 8-bit quantizer is a good approximation.

6.7 Summary

The PCRLB has recently been proposed [15] as an effective selection criteria for distributed sensor resource management in large, geographically distributed sensor networks. Existing PCRLB-based selection techniques are, however, primarily limited to centralized and hierarchical architectures, and when extended to distributed topologies use approximate expressions [15] for computing the PCRLB. The chapter addresses this gap and proposes the distributed PCRLB (dPCRLB) as the sensor selection criteria for distributed AN/SN systems without any need for central fusion. In the chapter, dynamic sensor selection for reactive non-linear tracking applications in distributed AN/SN systems is considered. I proposed a consensus-based sensor selection approach based on the dPCRLB for a network with two types of nodes: *observation nodes* with limited power, no processing ability, which make observations, and; *local processing nodes* without any power constraints for processing and communication. Each processing node computes its local track based only on the observations limited to the selected observation nodes in its neighbourhood. The processing nodes cooperate distributively with each other to compute the global state estimate. The cost function for the consensus-based distributed iterative local node selection approach is based on the dPCRLB. A distributed adaptive sensor-selection algorithm is then developed using the conditional dPCRLB. The conditional PCRLB is a function of the past history of observations made and, therefore, a more accurate representation of the estimator's performance and, consequently, a better criteria for distributed adaptive sensor selection. Finally, existing distributed algorithms for computing the PCRLB are typically based on raw observations resulting in a significant communication overhead. The chapter further derived the PCRLB for distributed estimators in an AN/SN system with quantized observations. Our numerical simulations verify the efficiency of the proposed distributed dPCRLB based sensor selection approaches. Through Monte Carlo simulations, we showed that the sensor selection algorithm based on the conditional dPCRLB is superior to the implementation using the conventional (non-conditional) dPCRLB. Finally, the proposed CQ/dPCRLB with quantized observations is compared for accuracy with its centralized counterpart through Monte-Carlo simulations.

7 Contributions and Future Research Directions

The chapter concludes the thesis with a list of important contributions made in the dissertation and some proposed directions for future work.

7.1 Summary of Contributions

A list of the main contributions of the thesis is as follows.

1. **Consensus-Based Distributed Implementation of the Particle Filter** [49, 50, 59–61]: I proposed three consensus-based, distributed implementations of the particle filter. First, a constrained sufficient statistic based distributed implementation of the particle filter (CSS/DPF) is proposed for bearing-only tracking (BOT) and joint bearing/range tracking problems encountered in a number of applications including radar target tracking and robot localization. Existing distributed consensus-based particle filter implementations proposed in the literature [20,22] require a large number of parallel consensus runs at each iteration of the particle filter which adds considerable consensus overhead to the distributed estimator. The CSS/DPF is proposed with the goal of developing a distributed particle filter that has reduced consensus overhead and affordable complexity. In the CSS/DPF, the number of parallel consensus runs is reduced to 6 for 2-D BOT, 16 for 3-D BOT, and 12 for joint bearing/range tracking. The proposed CSS/DPF still depends on the convergence of each

of the consensus runs which itself requires a large number of consensus iterations. To further reduce the consensus overhead, the CSS/DPF is extended to distributed implementation of the unscented particle filter, referred to as the CSS/DUPF which require limited number of consensus iterations.

Although computationally efficient, the CSS/DPF and CSS/DUPF are dependent on the dynamics of the system and are applicable to applications where the global sufficient statistics (GSS) can be expressed as a linear combination (summation) of the local sufficient statistics (LSS). The unscented, consensus-based, distributed implementation of the particle filter (UCD/DPF) is proposed which is generalizable to systems with any dynamics. The UCD/DPF couples the unscented Kalman filter (UKF) with the particle filter such that the UKF estimates the Gaussian approximation of the proposal distribution, which is used to generate new particles for the next iteration of the particle filter. In terms of contributions, the UCD/DPF makes two important improvements to the existing distributed particle filter framework: (i) Unlike existing distributed implementations [24,27] of the particle filter, the UCD/DPF uses all available global observations including the most recent ones in deriving the proposal distribution based on the distributed UKF, and; (ii) Computation of the global estimates from local estimates during the consensus step is based on an optimal fusion rule.

2. **The CF/DPF Framework** [51, 52, 62, 63]: A major problem in distributed estimation networks is unreliable communication (especially in large and multi-hop networks), which results in communication delays and information loss. Referred to as the intermittent network connectivity, this issue has been investigated broadly in the context of the Kalman filter. Such methods are, however, limited to linear systems and have not yet been extended to non-linear systems. The thesis addresses this gap. A multi-rate consensus/fusion based framework for distributed implementation of the particle filter, referred to as the CF/DPF,

is proposed. The CF/DPF framework is based on running localized particle filters to estimate the overall state vector at each observation node. Separate fusion filters are designed to consistently assimilate the local filtering distributions into the global posterior by compensating for the common past information between neighbouring nodes. The CF/DPF offers two distinct advantages over its counterparts. First, the CF/DPF framework is suitable for scenarios where network connectivity is intermittent and consensus can not be reached between two consecutive observations. Second, the CF/DPF is not limited to the Gaussian approximation for the global posterior density.

3. **Distributed Computation of the PCRLB** [53–55, 64]: In order to evaluate the performance of the proposed *distributed*, *non-linear* framework, the posterior Cramér-Rao lower bounds (PCRLB) are presented. The current PCRLB approaches assume a centralized or hierarchical architecture. The exact expression for distributed computation of the PCRLB is not yet available and only an approximate expression [15] has recently been derived. The thesis derives the exact expression, referred to as the dPCRLB, for computing the PCRLB for any AN/SN configured in a distributed fashion.

4. **Conditional dPCRLB**: Motivated by the distributed adaptive resource management problems, the thesis derives recursive expressions for the online computation of the *conditional* dPCRLB [55]. Compared to the *non-conditional* PCRLB, the conditional PCRLB is a function of the past history of observations made and, therefore, a more accurate representation of the estimator's performance and, consequently, a better criteria for sensor selection. Previous algorithms to compute the conditional PCRLB are limited to centralized architectures, which involve a fusion centre, thus making them unsuitable for distributed topologies. The distributed algorithms for computing the conditional and non-conditional dPCRLBs are exact with resulting bounds same as those for the centralized PCRLB.

5. **Distributed Sensor Selection** [56,67]: Finally, the thesis considers the problem of sensor resource management for distributed, nonlinear estimation applications with the objective of dynamically activating a time-variant subset of observation nodes to optimize the network's performance [67]. The PCRLB is a predictive benchmark of the tracker's achievable performance and has recently been proposed as a criteria for sensor selection. Existing PCRLB-based sensor selection techniques are, however, primarily limited to centralized and hierarchical architectures, and when extended to distributed topologies use approximate expressions for computing the PCRLB. I proposed a near-optimal dPCRLB-based sensor selection procedure for distributed sensor networks.

The algorithms listed under Items 1-5 are tested and compared with their state-of-art counterparts using Monte Carlo simulations for different tracking applications. In most cases the proposed algorithms outperform the existing state-of-art approaches.

7.2 Future Research Directions

Below, I highlight some directions for future research work.

1. In the thesis, I considered a single state model to represent the system's dynamics which is a common practice in distributed implementations of the particle filter [16–19, 23, 24, 27]. Extending the proposed distributed particle filter implementations to multiple state models [167] as is the case for source tracking applications where the source can manoeuvre differently is one direction for future research.

2. In the thesis, the SIR and unscented particle filters have been chosen as proof of concepts to develop distributed particle filter implementations. The proposed frameworks can be extended/generalized to other variants of the particle filter such as the marginalized particle

filter [125] and the approximate condition mean particle filter [126], with some modifications which is another direction for future work.

3. **Consensus and Innovation based Distributed Particle Filter Implementation**: The consensus-based distributed implementations of the particle filter require the consensus step to converge between two consecutive observations. In large sensor networks, convergence often requires a large number of consensus iterations which adds considerable consensus overhead to the distributed estimator. The impractically large number of consensus iterations in distributed consensus-based particle filters motivates future work to either come up with more efficient consensus algorithms or with distributed particle filter implementations that can cope with situations where consensus is limited to few (one to three) iterations between two consecutive observations.

   The thesis proposed the CSS/DPF which requires a reduced number of consensus runs per iteration, but still requires the consensus step to converge. To further reduce the consensus overhead, the CSS/DUPF is then proposed which can be considered as a consensus and innovation [168] distributed non-linear estimator. In other words, it can be shown that the CSS/DUPF is the non-linear (particle filter based) counterpart of the linear consensus and innovation filters [168] where its mean squared error (MSE) remains bounded when the number of consensus iterations between two consecutive observations is less than the number of iterations required for the consensus convergence. An interesting future research direction is to extend the CSS/DUPF to scenarios with communication constraints where the consensus is limited to one iteration between two consecutive observations (i.e., the communication time scale and sensing time scale are the same as shown in Fig. 7.1).

4. **Incorporating non-Parametric Statistical Models in the CF/DPF**: The localized

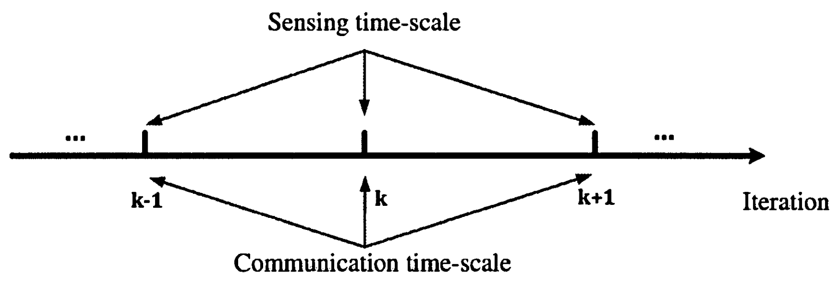


Figure 7.1: Time-scales of sensing (dynamic estimation) and communication (consensus iterations). Consensus + innovation Kalman filtering where the consensus time (communication time) and the sensing/filtering time are the same.

posteriors in CF/DPF are represented as a Dirac mixture in the particle filter. Two separate Dirac mixtures may not have the same support and their multiplication could possibly be zero. In order to tackle this problem, a transformation is required on the Dirac function particle representations by converting them to continuous distributions prior to communication and fusion. The CF/DPF uses Gaussian approximation of the local filtering and prediction densities. Alternative parametric distributions which can be used in the CF/DPF are: grid-based techniques [47], Gaussian Mixture Model (GMM) [17] and Parzen representations [27]. Another interesting alternative solution is to use non-parametric statistical models instead of the above parametric models. For example, recently the support vector machines (SVM) have shown to perform well for density estimation problems where the PDF of the IID sample set can be learned and the entire sample set can be represented by a few support vectors and the associated kernel functions [170]. Another direction for future research is to incorporate SVMs in the CF/DPF implementation which should improve the estimation performance of the CF/DPF.

5. **Distributed Estimation with Measurement Origin Uncertainty:** Extending the pro-

posed computational algorithms to account for the measurement origin uncertainty [149,160] is another direction of research that can be pursued to generalize the distributed particle filter implementations as well as computing the associated dPCRLBs. For example, Reference [149] has introduced a general framework for determining the PCRLBs that allows a marriage of non-linear measurements and uncertain dynamics for the centralized architecture. The distributed PCRLBs with measurement origin uncertainty has not yet been considered in the literature. Extending the proposed distributed PCRLB to include the measurement origin uncertainty is another direction for future research.

6. **Consensus-Based Distributed Sensor Selection for Multi-target Tracking:** The proposed distributed sensor selection algorithms is considered for scenarios with a single target, or fixed and well-separated targets. A natural extension is the problem of distributed consensus-based sensor selection for large scale multi-target tracking applications where targets overlap and occlude each other.

7. **Reduced-Order Implementations**: In the thesis, I focused primarily on the full-order distributed particle filter implementations where the entire state vector is estimated at each node. Appendix E presents some initiative results on distributed reduced-order particle filters and the corresponding reduced-order computation of the dPCRLB. Recall in reduced-order estimation, a different subset of the state vector is estimated at the processing nodes. The overall system is divided into several coupled low-dimensional sub-systems. The particle filter implemented at one sub-system computes the marginal posterior density of the local state variables. Marginalizing a sampled representation (particle filter) has proved to be computationally straightforward [175], i.e., the marginal over a subset of state variables is represented by dropping the particles for other state components (ignoring them). This feature of the particle filters encourages further investigation of the reduced-order distributed

implementations. The key issue when distributing the particle filter for such reduced-order scenarios is to ensure that the local marginal posteriors approximate the centralized posterior in a meaningful way. If the local marginal posterior evolve independently at each sub-system, they may lose any coherence with the centralized posterior. Motivated by *non-linear* sparse and localized large-scale problems such as smart power grids [48], developing more accurate and near-optimal reduced-order distributed implementations of the particle filter is another important future research direction.

7.3 Applications of Distributed Particle Filter Implementations

The theses focused primarily on distributed tracking application based on bearing and range measurements. Other areas where distributed particle filter can be applied are outlined below.

1. **State Estimation in Power Grids**: State estimation [106–109] in electrical power grids is used to monitor the state of the grid, enable energy management, optimize power flows, and perform reliability/security assessment. State forecasts are also used to analyze contingencies and determine necessary corrective actions against possible failures in the power systems. In the electric power distribution networks, the underlying state and observation models are highly nonlinear. The observations are geographically distributed across the entire distribution grid. The large dimensionality of the estimation problem precludes the direct application of the centralized particle filter primarily due to its high computational complexity. In other words, although the centralized approach is optimal, it is neither robust nor scalable to such large-scale dynamical systems with geographical distributed observation nodes primarily because of two reasons. First, extensive computations are required at the fusion node due to the high dimensionality of the dynamical systems. Second, the centralized implementation requires a large number of information transfers to the fusion

centre thus adding considerable latency (a major drawback for real-time applications) to the estimation mechanism.

The state estimation approaches in complex electric power distribution networks, typically consider the overall system as a union of several low-dimensional subsystems. Each subsystem is a combination of multiple, geographically distributed nodes representing a variety of power devices such as generating stations, compensators, or loads. Within each subsystem, the voltage and power supplied to a feeder at the substation are usually the only real time measurements available to the system operator at the distribution control centre. More extensive real time monitoring and control are required for effective operation of the system and for good quality of service to the customer coupled with the need to prevent wide-spread power blackouts. As outlined below, there are at lease three major aspects in the power grids that directly impact state estimation approaches and motivate development of distributed estimation implementations.

(a) Monitoring the power grid over large geographical areas calls for distributed control, and hence, distributed state estimation to facilitate coordinated monitoring.

(b) More advanced measurement technologies like phasor measurement units (PMUs) have offered hope for near real-time monitoring of the power grid. However, the latency introduced by the centralized estimation architecture is a major barrier toward achieving this goal.

(c) To facilitate smart grid features such as demand response and two-way power flow, timely and accurate models and estimation approaches are required which calls for distributed on-line state estimation at the distribution level.

Application of the proposed distributed particle filter implementations to the power grid is an area of research that can be pursued in the future. Such applications would require extension of the particle filter approaches to reduced-order systems.

2. **State Estimation in Distributed Camera Networks**: Over the past decade, large-scale camera networks [110] have become increasingly popular in a wide range of applications, including: (i) Sports analysis; (ii) Security and surveillance; (iii) disaster response, and; (iv) Environmental modeling, where the objective is to follow the trajectory of a key target, e.g., a star player in a soccer game or a suspect in a surveillance environment. In many applications, bandwidth constraints, security concerns, and difficulty in storing and analyzing large amounts of image data centrally at a single location necessitate the development of distributed camera network (DCN) architectures [111]. In distributed tracking via camera networks each camera acts as a local agent and estimates certain parameters of the target using a signal processing algorithm based upon its own set of video sequences. The local estimates are then shared with the neighbouring cameras in an iterative, decentralized, gossip-type fashion, and a final estimate is computed across the network using consensus algorithms.

   Most of the recent focus on distributed tracking algorithms for DCN is devoted to developing distributed implementation of the Kalman filters [111]. Although particle filters are popular for visual tracking [112,113] in a centralized architecture, their distributed implementations are less explored for tracking in DCNs. Distributed particle filter approaches proposed in the thesis can be applied (with proper modifications) to tracking problems in DCN, which is another area of future research worth pursuing.

A Proof of the Results Reported in Chapter 3

A.1 Proof of Lemma 1

Proof. The local sufficient statistic $\mathcal{Y}^{(l)}(k) \triangleq \mathcal{T}_3(\mathbf{z}^{(l)}(k))$ exists by assumption. Using the Fisher-Neyman factorization theorem (Eq. (3.1)) and Eq. (2.14), the global likelihood $P(\mathbf{z}(k)|\mathbf{x}(k))$ can be stated as a product of a function only dependent on local observation vector $\mathbf{z}^{(l)}(k)$ and a function depending on both $\mathbf{x}(k)$ and $\{\mathcal{Y}^{(1)}(k), \dots, \mathcal{Y}^{(N)}(k)\}$ as follows

$$P(\mathbf{z}(k)|\mathbf{x}(k)) = \prod_{l=1}^{N} \mathcal{T}_1^{(l)}\left(\mathbf{z}^{(l)}(k)\right) \prod_{l=1}^{N} \mathcal{T}_2^{(l)}\left(\mathcal{Y}^{(l)}(k), \mathbf{x}(k)\right), \tag{A.1}$$

Hence, $\{\mathcal{Y}^{(1)}(k), \dots, \mathcal{Y}^{(N)}(k)\}$ are jointly sufficient for estimating the state variables $\mathbf{x}(k)$. □

A.2 Proof of Lemma 2

Proof. We start considering two observations, i.e., $\mathbf{z}(k) = [\mathbf{z}^{(i)^T}(k), \mathbf{z}^{(j)^T}(k)]^T$, where the global likelihood $P(\mathbf{z}(k)|\mathbf{x}(k))$ is factorized as follows using Eqs. (2.14) and (3.2)

$$P(\mathbf{z}(k)|\mathbf{x}(k)) = \Big(h_2(\mathbf{z}^{(i)}(k), \mathbf{x}(k))h_2(\mathbf{z}^{(j)}(k), \mathbf{x}(k))\Big)\Big(h_1(\mathbf{z}^{(i)}(k))h_1(\mathbf{z}^{(j)}(k))\Big)[h_3(\mathbf{x}(k))]^2. \tag{A.2}$$

Application of Eq. (3.3) to Eq. (A.2), yields the following result

$$P(\mathbf{z}(k)|\mathbf{x}(k)) = h_2\Big(\phi(\mathbf{z}^{(i)}(k), \mathbf{z}^{(j)}(k)), \mathbf{x}(k)\Big)\, h_1(\mathbf{z}^{(i)}(k))h_1(\mathbf{z}^{(j)}(k))[h_3(\mathbf{x}(k))]^2 h_4(\mathbf{z}^{(i)}(k), \mathbf{z}^{(j)}(k)). \tag{A.3}$$

Therefore, a sufficient statistic $\phi(\mathbf{z}^{(i)}(k), \mathbf{z}^{(j)}(k))$ is found from $\mathbf{z}^{(i)}(k)$ and $\mathbf{z}^{(j)}(k)$. By induction to any number of nodes N, we observe that there exist a function $\mathcal{S}(\cdot)$ such that the GSS equals $\mathcal{S}(\mathcal{Y}^{(1)}(k), \dots, \mathcal{Y}^{(N)}(k))$. □

A.3 Proof of Theorem 2

Proof. The true bearing to the target can be defined as follows

$$\left[X(k) - X^{(l)}\right] \cos(\mathbb{Z}_\theta^{(l)}(\mathbf{x}(k))) - \left[Y(k) - Y^{(l)}\right] \sin(\mathbb{Z}_\theta^{(l)}(\mathbf{x}(k))) = 0. \tag{A.4}$$

The bearing measurement $Z_\theta^{(l)}(k)$ observed by node l, for $(1 \le l \le N)$ is noisy. When the noisy measured bearing is used in place of the true bearing in Eq. (3.12), Reference [119] shows that the relationship in Eq. (A.4) changes to

$$\left[X(k) - X^{(l)}\right] \cos(Z_\theta^{(l)}(k)) - \left[Y(k) - Y^{(l)}\right] \sin(Z_\theta^{(l)}(k)) = \left(X^2(k) + Y^2(k)\right)^{1/2} \sin({\zeta_\theta}^{(l)}(k)), \tag{A.5}$$

which is reordered as

$$\begin{aligned} &\underbrace{Y^{(l)} \sin(Z_\theta^{(l)}(k)) - X^{(l)} \cos(Z_\theta^{(l)}(k))}_{\mathcal{Z}_\theta^{(l)}(k)} \\ &\quad = Y(k) \sin(Z_\theta^{(l)}(k)) - X(k) \cos(Z_\theta^{(l)}(k)) + \underbrace{(X^2(k) + Y^2(k))^{1/2} \sin({\zeta_\theta}^{(l)}(k))}_{\mathcal{V}_\theta^{(l)}(k)}, \end{aligned} \tag{A.6}$$

For $\zeta_\theta^{(l)}(k) \sim \mathcal{N}(0, {\sigma_\theta^{(l)}}^2(k))$, noise $\mathcal{V}_\theta^{(l)}(k)$ is zero mean with variance given by

$$R_\theta^{(l)}(k) = E\{X^2(k) + Y^2(k)\}(1 - \exp^{-2{\sigma_\theta^{(l)}}^2})/2, \tag{A.7}$$

obtained by observing that $E\{\sin^2(\zeta_\theta^{(l)}(k))\} = 1/2(1 - \exp(-2{\sigma_\theta^{(l)}}^2))$. Evaluating $R^{(l)}(k)$ requires the propagation of the second moment matrix

$$S(k) = E\{\boldsymbol{f}(\mathbf{x}(k-1))\boldsymbol{f}^T(\mathbf{x}(k-1))\} + \boldsymbol{Q}(k), \tag{A.8}$$

obtained from state equation (Eq. (2.3)), where $S(k)$ can be computed locally using particle $\mathbb{X}_i(k-1)$ and their corresponding weights $W_i(k-1)$ as follows

$$S(k)=\sum_{i=1}^{N_p} W_i(k-1)\big[f(\mathbb{X}_i(k-1))f^T(\mathbb{X}_i(k-1))\big]+Q(k), \tag{A.9}$$

and $Q(k)$ is the second moment of the state noise $\xi(k)$ in Eq. (2.3). Note that term $E\{X^2(k)+Y^2(k)\}$ in Eq. (A.7) equals the sum of the first two diagonal entries of $S(k)$. Based on Eqs. (A.6)-(A.8), the global likelihood function is then given by

$$P(\mathbf{z}_\theta(k)|\mathbf{x}(k)) = \frac{1}{C_\theta(k)}\exp\Big\{-\sum_{l=1}^{N}\frac{\big(\mathcal{Z}_\theta^{(l)}(k)-\mathcal{G}_\theta^{(l)}(\mathbf{x}(k))\big)^2}{2R_\theta^{(l)}(k)}\Big\}, \tag{A.10}$$

where $C_\theta(k) = (2\pi)^{N/2}\prod_{i=1}^{N}(R_\theta^{(l)}(k))^{1/2}$, and $\mathcal{G}_\theta^{(l)}(\mathbf{x}(k)) = Y(k)\sin(Z_\theta^{(l)}(k))-X(k)\cos(Z_\theta^{(l)}(k))$.

□

A.4 Proof of Theorem 3

Proof. First, Eq. (3.21) is rearranged as

$$\Big(\mathbb{Z}_{\mathrm{R}}^{(l)}(\mathbf{x}(k))\Big)^2 = \big(X(k)-X^{(l)}(k)\big)^2+\big(Y(k)-Y^{(l)}(k)\big)^2. \tag{A.11}$$

Eq. (A.11) is further expanded as

$$\mathbb{Z}_{\mathrm{R}}^{(l)}(\mathbf{x}(k)) = \big(X(k)-X^{(l)}(k)\big)\frac{X(k)-X^{(l)}(k)}{\mathbb{Z}_{\mathrm{R}}^{(l)}(\mathbf{x}(k))} + \big(Y(k)-Y^{(l)}(k)\big)\frac{Y(k)-Y^{(l)}(k)}{\mathbb{Z}_{\mathrm{R}}^{(l)}(\mathbf{x}(k))} \tag{A.12}$$

which is given by

$$\mathbb{Z}_{\mathrm{R}}^{(l)}(\mathbf{x}(k)) = \big(X(k)-X^{(l)}(k)\big)\sin\big(\mathbb{Z}_\theta^{(l)}(k)\big) + \big(Y(k)-Y^{(l)}(k)\big)\cos\big(\mathbb{Z}_\theta^{(l)}(k)\big). \tag{A.13}$$

The global likelihood function is then given by

$$P(\mathbf{z}_\phi(k)|\mathbf{x}(k)) = \frac{1}{C_\phi(k)}\exp\Big\{-\sum_{l=1}^{N}\frac{\big(\mathcal{Z}_\phi^{(l)}(k)-\mathcal{G}_\phi^{(l)}(\mathbf{x}(k))\big)^2}{2R_\phi^{(l)}(k)}\Big\}, \tag{A.14}$$

where $C_\phi(k) = (2\pi)^{N/2}\prod_{i=1}^{N}(R_\phi^{(l)}(k))^{1/2}$,

$$\mathcal{Z}_\phi^{(l)}(k)=Z_\phi^{(l)}(k)\cos(Z_\phi^{(l)}(k))-X^{(l)}(k)\sin(Z_\phi^{(l)}(k))\sin(Z_\theta^{(l)}(k))-Y^{(l)}(k)\sin(Z_\phi^{(l)}(k))\cos(Z_\theta^{(l)}(k)), \tag{A.15}$$

and

$$\mathcal{G}_\phi^{(l)}(\mathbf{x}(k))=Z_\phi(k)\cos(Z_\phi^{(l)}(k))-X(k)\sin(Z_\phi^{(l)}(k))\sin(Z_\theta^{(l)}(k))-Y(k)\sin(Z_\phi^{(l)}(k))\cos(Z_\theta^{(l)}(k)). \tag{A.16}$$

Finally based on [119], elevation bearing noise variance is

$$R_\phi^{(l)}(k)=E\{X^2(k)+Y^2(k)+Z^2(k)\}(1-\exp^{-4\sigma_\phi^{(l)2}})/4. \tag{A.17}$$

The global elevation bearing likelihood function can be expressed as function of ten GSSs given by

$$\begin{aligned}
G_{\phi,1}(k) &= \sum_{l=1}^{N}(\mathcal{Z}_\phi^{(l)}(k))^2/(R_\phi^{(l)}(k))\\
G_{\phi,2}(k) &= \sum_{l=1}^{N}\left((\mathcal{Z}_\phi^{(l)}(k))^2\cos^2(Z_\phi^{(l)}(k))\right)/(R_\phi^{(l)}(k))\\
G_{\phi,3}(k) &= \sum_{l=1}^{N}\frac{\mathcal{Z}_\phi^{(l)}(k)\sin(Z_\theta^{(l)}(k))\sin(Z_\phi^{(l)}(k))}{R_\phi^{(l)}(k)}\\
G_{\phi,4}(k) &= \sum_{l=1}^{N}\frac{\mathcal{Z}_\phi^{(l)}(k)\cos(Z_\theta^{(l)}(k))\sin(Z_\phi^{(l)}(k))}{R_\phi^{(l)}(k)}\\
G_{\phi,5}(k) &= \sum_{l=1}^{N}\left(\cos^2(Z_\phi^{(l)}(k))\right)/(R_\phi^{(l)}(k))\\
G_{\phi,6}(k) &= \sum_{l=1}^{N}\left(\cos(Z_\theta^{(l)}(k))\sin(Z_\phi^{(l)}(k))\right)^2/(R_\phi^{(l)}(k))\\
G_{\phi,7}(k) &= \sum_{l=1}^{N}\left(\sin(Z_\theta^{(l)}(k))\sin(Z_\phi^{(l)}(k))\right)^2/(R_\phi^{(l)}(k))\\
G_{\phi,8}(k) &= \sum_{l=1}^{N}\frac{\sin(Z_\theta^{(l)}(k))\sin(Z_\phi^{(l)}(k))\cos(Z_\phi^{(l)}(k))}{R_\phi^{(l)}(k)}
\end{aligned}$$

$$G_{\phi,9}(k) = \sum_{l=1}^{N} \frac{\cos(Z_\theta^{(l)}(k))\sin(Z_\phi^{(l)}(k))\cos(Z_\phi^{(l)}(k))}{R_\phi^{(l)}(k)}$$
$$G_{\phi,10}(k) = \sum_{l=1}^{N} \frac{\cos(Z_\theta^{(l)}(k))\sin(Z_\theta^{(l)}(k))\cos(Z_\theta^{(l)}(k))}{R_\phi^{(l)}(k)}$$

□

A.5 Proof of Theorem 4

Proof. Based on Eqs. (A.11)-(A.13), the observation model for range based tracking can be approximated as

$$Z_{\mathrm{R}}^{(l)}(k) \approx \big(X(k)-X^{(l)}(k)\big)\sin\big(Z_\theta^{(l)}(k)\big) + \big(Y(k)-Y^{(l)}(k)\big)\cos\big(Z_\theta^{(l)}(k)\big) + \zeta_{\mathrm{R}}^{(l)}(k), \tag{A.18}$$

which simplifies to

$$\underbrace{Z_{\mathrm{R}}^{(l)}(k) + X^{(l)}(k)\sin(Z_\theta^{(l)}(k)) + Y^{(l)}(k)\cos(Z_\theta^{(l)}(k))}_{\mathcal{Z}_{\mathrm{R}}^{(l)}(k)} \approx \underbrace{X(k)\sin\big(Z_\theta^{(l)}(k)\big) + Y(k)\cos\big(Z_\theta^{(l)}(k)\big)}_{\mathcal{G}_R^{(l)}(\mathbf{x}(k))} + \zeta_{\mathrm{R}}^{(l)}(k). \tag{A.19}$$

The global likelihood function is then given by

$$P(\mathbf{z}_R(k)|\mathbf{x}(k)) \propto \exp\Big\{-\sum_{l=1}^{N}\frac{\big(\mathcal{Z}_R^{(l)}(k)-\mathcal{G}_R^{(l)}(\mathbf{x}(k))\big)^2}{2R_R^{(l)}(k)}\Big\}. \tag{A.20}$$

Based on [119], the range noise variance is given by

$$R_R^{(l)}(k) = \frac{[\zeta_{\mathrm{R}}^{(l)}(k)]^2\big(1+e^{-2\zeta_o^{(l)}(k)}\big)^2}{4}. \tag{A.21}$$

By expanding Eq. (A.20), the global range likelihood function can be expressed as function of six GSSs given in Eq. (3.29). □

B Proof of the Results Reported in Chapter 4

B.1 Proof *of Theorem 5 [127]*

Proof. Applying the Bayes' rule to Eq. (4.4), the posterior distribution is given by

$$P\big(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k)\big) \propto P\big(\mathbf{z}(k)|\mathbf{x}(k)\big)P\big(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k-1)\big). \tag{B.1}$$

Now, using the Markovian property of the state variables, Eq. (B.1) becomes

$$P\big(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k)\big) \propto P\big(\mathbf{z}(k)|\mathbf{x}(k)\big) \times P\big(\mathbf{x}(k)|\mathbf{x}(k-1)\big)P\big(\mathbf{x}(0\!:\!k-1)|\mathbf{z}(1\!:\!k-1)\big). \tag{B.2}$$

Assuming that the local observations made at two sensor nodes conditioned on the state variables are independent of each other Eq. (B.2) becomes

$$P\big(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k)\big) \propto \left(\prod_{l=1}^{N} P\big(\mathbf{z}^{(l)}(k)|\mathbf{x}(k)\big)\right) \times P\big(\mathbf{x}(k)|\mathbf{x}(k-1)\big)P\big(\mathbf{x}(0\!:\!k-1)|\mathbf{z}(1\!:\!k-1)\big). \tag{B.3}$$

Using the Bays' rule, the local likelihood function $P\left(\mathbf{z}^{(l)}(k)|\mathbf{x}(k)\right)$ at node l, for $(1 \le l \le N)$ is

$$P\left(\mathbf{z}^{(l)}(k)|\mathbf{x}(k)\right) = \frac{P\left(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k)\right)}{P\left(\mathbf{x}(k)|\mathbf{z}^{(l)}(1\!:\!k-1)\right)} P\left(\mathbf{z}^{(l)}(k)|\mathbf{z}^{(l)}(1\!:\!k-1)\right). \tag{B.4}$$

Finally, the result (Eq. (4.4)) is provided by substituting Eq. (B.4) in Eq. (B.3). □

Table B.1: Comparison of the Computational Complexity.

		UKF/FF Complexity	Particle Filter Complexity	Consensus Step
Centralized		$\max(\mathrm{O}(n_x^3), \mathrm{O}(N^3), \mathrm{O}(n_x N^2))$	$\mathrm{O}\big((n_x^2 + N)N_s\big)$	–
UCD/DPF	Per node	$\mathrm{O}(n_x^3)$	$\mathrm{O}(N_{\mathrm{UPF}} n_x^2)$	$\mathrm{O}(n_x^2 \Delta_{\mathcal{G}} N_c(\boldsymbol{U}))$
	Total	$\mathrm{O}(N n_x^3)$	$\mathrm{O}(N N_{\mathrm{UPF}} n_x^2)$	$\mathrm{O}(N n_x^2 \Delta_{\mathcal{G}} N_c(\boldsymbol{U}))$
CF/DPF	Per node	$\mathrm{O}(N_{\mathrm{FF}} n_x^2)$	$\mathrm{O}(N_{\mathrm{LF}} n_x^2)$	$\mathrm{O}(n_x^2 \Delta_{\mathcal{G}} N_c(\boldsymbol{U}))$
	Total	$\mathrm{O}(N N_{\mathrm{FF}} n_x^2)$	$\mathrm{O}(N N_{\mathrm{LF}} n_x^2)$	$\mathrm{O}(N n_x^2 \Delta_{\mathcal{G}} N_c(\boldsymbol{U}))$

B.2 Proof *of Theorem 6*

Proof. Following the approach in the proof of Theorem 5 (Appendix B.1), we first write the posterior density at iteration $k+m$ as

$$P\left(\mathbf{x}(0\!:\!k+m)|\mathbf{z}(1\!:\!k+m)\right) \propto \frac{\prod_{l=1}^{N} P\left(\mathbf{x}(k+m)|\mathbf{z}^{(l)}(1\!:\!k+m)\right)}{\prod_{l=1}^{N} P\left(\mathbf{x}(k+m)|\mathbf{z}^{(l)}(1\!:\!k+m-1)\right)} P\left(\mathbf{x}(0\!:\!k+m)|\mathbf{z}(1\!:\!k+m-1)\right) \quad \text{(B.5)}$$

Then the last term is factorized as follows

$$P\left(\mathbf{x}(0\!:\!k+m)|\mathbf{z}(1\!:\!k+m-1)\right) = P\left(\mathbf{x}(k+m)|\mathbf{x}(k+m-1)\right) P\left(\mathbf{x}(0\!:\!k+m-1)|\mathbf{z}(1\!:\!k+m-1)\right). \quad \text{(B.6)}$$

As in Eq. (B.5), we continue to expand $P(\mathbf{x}(0 : k+m-1)|\mathbf{z}(1 : k+m-1))$ (i.e., the posterior distribution at iteration $k+m-1$) all the way back to iteration $k+1$ to prove Eq. (4.27). □

B.3 Computational Complexity of The CF/DPF and UCD/DPF

In this section, I provide a rough comparison of the computational complexity of the UCD/DPF and CF/DPF versus that of the centralized implementation. Because of the non-linear dynamics of the particle filter, it is somewhat difficult to drive a generalized expression for its computational complexity. There are steps that can not be easily evaluated in the complexity computation of the particle filter such as the cost of evaluating a non-linear function (as is the case for the state and observation models) [131]. Below the simplified case of a linear state model with Gaussian excitation and observation noise is considered. Further, the observations are assumed to be uncorrelated.

Following the approach proposed in [131], the computational complexity of different implementations of the particle filter is expressed in terms of flops, where a flop is defined as addition, subtraction, multiplication or division of two floating point numbers. In the analysis, I take into account the number n_x of states, which are at times ignored in the computational complexity of the particle filter. Note that the computational complexity of multiplication or inversion of $(n_x \times n_x)$ matrices is of $\mathrm{O}\left(n_x^3\right)$, and multiplication of $(n_x \times n_x)$ matrix with an $(n_x \times 1)$ vector is of $\mathrm{O}\left(n_x^2\right)$. As such, the total equivalent flop computational complexity [131] of the centralized particle filter for N-node network with N_s particles is derived as follows:

1. State Update (based on Eq. (2.3)): $\mathrm{O}\left(n_x^2 N_s\right)$ considering a linear state model.
2. Evaluation of Weights (based on Eq. (2.79)): $\mathrm{O}\left(N N_s\right)$ assuming uncorrelated observations with Gaussian distributions.
3. Resampling (if needed): $\mathrm{O}\left(N_s\right)$ (a direct implementation of the resampling procedure has a complexity of $\mathrm{O}(N_s \log(N_s))$ [43], however, there are several alternative approaches including systematic resampling [43] which has a complexity of $\mathrm{O}\left(N_s\right)$).

The computational complexity of the centralized particle filter is given by $\mathrm{O}\left((n_x^2 + N)N_s\right)$, which includes the dependence on the number n_x of states. Table B.1 compares the computational complexity of the centralized implementation versus its distributed counterparts: the UCD/DPF and CF/DPF. The CF/DPF runs two particle filters (local filter and fusion filter) at each node,

N_{LF} denotes the number of particles used by the local filter and N_{FF} denotes the number of particles used by the fusion filter. The number of particles used by the UCD/DPF implementation is denoted by N_{UPF}. The derivation of the expressions listed in Table B.1 is described below.

The centralized implementation is based on an unscented particle filter [44], which uses an additional step of the unscented Kalman filter (UKF). The computational complexity of the UKF component is given by $\max(\mathrm{O}(n_x^3), \mathrm{O}(N^3), \mathrm{O}(n_x N^2))$, or, $\mathrm{O}(N^3)$, for $n_x << N$. The overall computational complexity of the centralized particle filter is, therefore, of $\mathrm{O}(N^3 + NN_s)$.

The first distributed implementation based on the UCD/DPF runs a particle filter at each observation node. The individual particle filter is similar in complexity to the centralized particle filter (without the UKF) except that the observation (target's bearing at each node) is a scalar. Setting $N = 1$, the computational complexity of the UCD/DPF is of $\mathrm{O}\left(n_x^2 N_{\text{UPF}} + N_{\text{UPF}}\right)$ or $\mathrm{O}\left(n_x^2 N_{\text{UPF}}\right)$ per node, where N_{UPF} is the number of particles at each sensor node in the UCD/DPF. The overall computational complexity of UCD/DPF is, therefore, of $\mathrm{O}\left(N n_x^2 N_{\text{UPF}}\right)$. There are two additional components to the UCD/DPF. First, the unscented Kalman filter in the UCD/DPF has an overall computational complexity of $\mathrm{O}(N n_x^3)$. Second, the distributed implementations (UCD/DPF and CF/DPF) introduce an additional consensus step, whose complexity is derived as a function of the maximum degree $\Delta_{\mathcal{G}}$ of the network and the total number of consensus iterations $N_c(\boldsymbol{U})$ required to reach a global consensus. The computational complexity of the consensus step at each node is at most of $\mathrm{O}(n_x^2 \Delta_{\mathcal{G}})$ per iteration times total number of consensus iterations $N_c(\boldsymbol{U})$, therefore, the consensus step has a computational complexity of $\mathrm{O}(n_x^2 \Delta_{\mathcal{G}} N_c(\boldsymbol{U}))$. The associated convergence time $N_c(\boldsymbol{U}) = 1/\log(1/r_{\text{asym}}(\boldsymbol{U}))$, which provides the asymptotic number of consensus iterations (required for the error to decrease by the factor of $1/e$) can be computed using the asymptotic convergence rate (Eq. (2.119)). According to Theorem 1, $N_c(\boldsymbol{U}) = -1/\max_{2 \leq i \leq N} \log(|\lambda_i(\boldsymbol{U})|)$. The computational complexity of the consensus step is, therefore, related to the properties of the communication network and the consensus matrix $\boldsymbol{U}$. Based on the aforementioned derivation, the computational complexity of the UCD/DPF is given by $\max\left\{\mathrm{O}(Nn_x^2N_{\mathrm{UPF}}, Nn_x^3, n_x^2\Delta_{\mathcal{G}}N_c(\boldsymbol{U}))\right\}$.

The computational complexity of the CF/DPF is similarly derived and listed in Table B.1. The CF/DPF does not use the UKF instead it uses the fusion filter, which has complexity similar to the distributed particle filter as shown in column 2 of Table B.1. The computational complexity of the CF/DPF is, therefore, given by $\max\left\{\mathrm{O}(Nn_x^2N_{\mathrm{LF}}, NN_{\mathrm{FF}}n_x^2, n_x^2\Delta_{\mathcal{G}}N_c(\boldsymbol{U}))\right\}$.

Since the computational complexity of the three implementations involve different variables, it is difficult to compare them subjectively. In the simulations, the value of the variables are as follows: $n_x = 4$, $N = 20$, $N_s = 10,000$, $N_{\mathrm{UPF}} = N_{\mathrm{LF}} = N_{\mathrm{FF}} = 500$, and $N_c(\boldsymbol{U}) = 8$ which results in the following rough computational counts for the three implementations: Centralized implementation: 3.6×10^5, CF/DPF: 3.4×10^5, and UCD/DPF: 1.8×10^5 computational counts. This means that the three implementations have roughly the same computational complexity for the simulation. Note that the computational burden is distributed evenly across the nodes in the CF/DPF and UCD/DPF, while the fusion center performs most of the computations in the centralized particle filter. This places an additional power energy constraint on the fusion center causing the system to fail if the power in the fusion center drains out. Finally, I note that the UCD/DPF and CF/DPF require a higher number of information transfers but the goal here is to implement a distributed system without the fusion center.

C Proof of the Results Reported in Chapter 5

C.1 Proof *of Proposition 2*

Proof. The proof of Proposition 2 uses the Markovian property of the state variables and is based on the following factorization of the joint prediction distribution

$$P\big(\mathbf{x}(0:k+1)|\mathbf{z}(1:k)\big) = P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)P\big(\mathbf{x}(0:k)|\mathbf{z}(1:k)\big).$$

The steps involved are similar to the proof of Theorem 7 included below and not repeated here. □

C.2 Proof *of Theorem 7*

Proof. The proof for Theorem 7 is based on the following nonlinear Bayesian fusion rule [127] (Lemma 6), which expresses the global posterior density as a function of local filtering and prediction densities.

Lemma 6. *Assuming that the observations conditioned on the state variables made at node l are independent of the observations made at a different node j, ($j \neq l$), the global posterior for a N-sensor network is*

$$P\left(\mathbf{x}(0:k+1)|\mathbf{z}(1:k+1)\right) \propto \\ \frac{\prod_{l=1}^{N} P(\mathbf{x}(k+1)|\mathbf{z}^{(l)}(1:k+1))}{\prod_{l=1}^{N} P\left(\mathbf{x}(k+1)|\mathbf{z}^{(l)}(1:k)\right)} P\left(\mathbf{x}(k+1)|\mathbf{x}(k)\right) P\left(\mathbf{x}(0:k)|\mathbf{z}(1:k)\right). \tag{C.1}$$

We first consider $J_{\mathrm{FO}}\big(\mathbf{x}(0\!:\!k)\big)$. Decomposing $\mathbf{x}(0\!:\!k) = [\mathbf{x}^T(0\!:\!k{-}1), \mathbf{x}^T(k)]^T$ in $J_{\mathrm{FO}}\big(\mathbf{x}(0\!:\!k)\big)$, Eq. (6.34) from Definition 3 reduces to

$$J_{\mathrm{FO}}(\mathbf{x}(0:k)) = \mathbb{E}\Big\{ - \left[\begin{array}{c:c} \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k)} \\ \hdashline \Delta_{\mathbf{x}(k)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \end{array}\right] \log P(\mathbf{x}(0:k)|\mathbf{z}(1:k))\Big\} \triangleq \begin{bmatrix} A_{\mathrm{FO}}^{11}(k) & A_{\mathrm{FO}}^{12}(k) \\ A_{\mathrm{FO}}^{21}(k) & A_{\mathrm{FO}}^{22}(k) \end{bmatrix} \tag{C.2}$$

provided that the aforementioned expectations and derivatives exist. The bottom right block (denoted by $A_{\mathrm{FO}}^{22}(k)$) on the right hand side (RHS) of Eq. (C.2) corresponds to a (2×2) block matrix, i.e., $A_{\mathrm{FO}}^{22}(k) \triangleq \mathbb{E}\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \log P(\mathbf{x}(0\!:\!k)|\mathbf{z}(1\!:\!k))\}$, and similarly for the remaining A_{FO}^{**}'s.

Following the aforementioned procedure used to derive Eq. (C.2) for $J_{\mathrm{FO}}(\mathbf{x}(0\!:\!k{+}1))$, we get

$$\begin{aligned} J_{\mathrm{FO}}\big(\mathbf{x}(0:k+1)\big) &= \mathbb{E}\left\{ - \left[\begin{array}{c:c:c} \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k)} & \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k+1)} \\ \hdashline \Delta_{\mathbf{x}(k)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} & \Delta_{\mathbf{x}(k)}^{\mathbf{x}(k+1)} \\ \hdashline \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k)} & \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \end{array}\right] \log P(\mathbf{x}(0:k+1)|\mathbf{z}(1:k+1))\right\} \\ &\triangleq \begin{bmatrix} E_{\mathrm{FO}}^{11}(k) & E_{\mathrm{FO}}^{12}(k) & E_{\mathrm{FO}}^{13}(k) \\ E_{\mathrm{FO}}^{21}(k) & E_{\mathrm{FO}}^{22}(k) & E_{\mathrm{FO}}^{23}(k) \\ E_{\mathrm{FO}}^{31}(k) & E_{\mathrm{FO}}^{32}(k) & E_{\mathrm{FO}}^{33}(k) \end{bmatrix}. \end{aligned} \tag{C.3}$$

It can be shown that $E_{\mathrm{FO}}^{11}(k) = A_{\mathrm{FO}}^{11}(k)$, $E_{\mathrm{FO}}^{12}(k) = A_{\mathrm{FO}}^{12}(k)$, $E_{\mathrm{FO}}^{13}(k) = E_{\mathrm{FO}}^{31}(k) = \mathbf{0}$, $E_{\mathrm{FO}}^{21}(k) = A_{\mathrm{FO}}^{21}(k)$, $E_{\mathrm{FO}}^{22}(k) = A_{\mathrm{FO}}^{22}(k) + C_{\mathrm{FO}}^{11}(k)$, $E_{\mathrm{FO}}^{23}(k) = C_{\mathrm{FO}}^{12}(k)$, $E_{\mathrm{FO}}^{32}(k) = C_{\mathrm{FO}}^{21}(k)$, and $E_{\mathrm{FO}}^{33}(k) = C_{\mathrm{FO}}^{22}(k)$, which leads to the following structure (similar to the one in [148])

$$J_{\mathrm{FO}}\big(\mathbf{x}(0:k+1)\big) = \left[\begin{array}{cc|c} A_{\mathrm{FO}}^{11}(k) & A_{\mathrm{FO}}^{12}(k) & \mathbf{0} \\ A_{\mathrm{FO}}^{21}(k) & A_{\mathrm{FO}}^{22}(k) + C_{\mathrm{FO}}^{11}(k) & C_{\mathrm{FO}}^{12}(k) \\ \hline \mathbf{0} & C_{\mathrm{FO}}^{21}(k) & C_{\mathrm{FO}}^{22}(k) \end{array}\right], \tag{C.4}$$

where block $\mathbf{0}$ stands for a block of all zeros with the appropriate dimension. To save on space, we only prove the equalities $E_{\mathrm{FO}}^{11}(k) = A_{\mathrm{FO}}^{11}(k)$ and $E_{\mathrm{FO}}^{33}(k) = C_{\mathrm{FO}}^{22}(k)$. The remaining entries can be proven following a similar procedure.

Case 1 (Proof for $\boldsymbol{E}^{11}_{\text{FO}}(k) = \boldsymbol{A}^{11}_{\text{FO}}(k)$): Factorizing the posterior distribution for the top left block in Eq. (C.3),

$$\begin{aligned}\Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(0:k-1)} \log P\big(\mathbf{x}(0:k+1)|\mathbf{z}(1:k+1)\big) &= \Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(0:k-1)} \log \frac{P\big(\mathbf{z}(k+1)|\mathbf{x}(k+1)\big)P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)}{P\big(\mathbf{z}(k+1)|\mathbf{z}(1:k)\big)} \\ &+ \Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(0:k-1)} \log P(\mathbf{x}(0:k)|\mathbf{z}(1:k)), \qquad \text{(C.5)}\end{aligned}$$

which leads to

$$\begin{aligned}\boldsymbol{E}^{11}_{\text{FO}}(k) &\triangleq \mathbb{E}_{P(\mathbf{x}(0:k+1),\mathbf{z}(1:k+1))}\big\{ - \Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(0:k-1)} \log P\big(\mathbf{x}(0:k+1)|\mathbf{z}(1:k+1)\big)\big\} \\ &= -\int\int \Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(0:k-1)} \log P\big(\mathbf{x}(0:k)|\mathbf{z}(1:k)\big) \\ &\times \left[\int\int P(\mathbf{x}(0:k+1),\mathbf{z}(1:k+1))d\mathbf{x}(k+1)d\mathbf{z}(k+1)\right] d\mathbf{x}(0:k)d\mathbf{z}(1:k). \text{(C.6)}\end{aligned}$$

The inner integral reduces to $P\big(\mathbf{x}(0:k),\mathbf{z}(1:k)\big)$, which gives

$$\boldsymbol{E}^{11}_{\text{FO}}(k) = -\iint \Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(0:k-1)} \log P\big(\mathbf{x}(0:k)|\mathbf{z}(1:k)\big)P\big(\mathbf{x}(0:k),\mathbf{z}(1:k)\big)d\mathbf{x}(0:k)d\mathbf{z}(1:k) = \boldsymbol{A}^{11}_{\text{FO}}(k) \text{(C.7)}$$

as per the definition of $\boldsymbol{A}^{11}_{\text{FO}}(k)$ in Eq. (C.5).

Case 2 (Proof for $\boldsymbol{E}^{33}_{\text{FO}}(k) = \boldsymbol{C}^{22}_{\text{FO}}(k)$): Based on Eq. (C.1), term $\log(P(\mathbf{x}(0:k+1)|\mathbf{z}(1:k+1)))$ is

$$\begin{aligned}\log P\big(\mathbf{x}(0:k+1)|\mathbf{z}(1:k+1)\big) &= \sum_{l=1}^{N}\log\big(P(\mathbf{x}(k+1)|\mathbf{z}^{(l)}(1:k+1))\big) - \sum_{l=1}^{N}\log\Big(P(\mathbf{x}(k+1)|\mathbf{z}^{(l)}(1:k))\Big) \\ &+ \log\big(P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big) + \log\big(P\left(\mathbf{x}(0:k)|\mathbf{z}(1:k)\right)\big). \qquad \text{(C.8)}\end{aligned}$$

Substituting (C.8) in the definition of $\boldsymbol{E}^{33}_{\text{FO}}(k)$ (Eq. (C.3)), we get

$$\begin{aligned}\boldsymbol{E}^{33}_{\text{FO}}(k) &\triangleq \mathbb{E}\Big\{-\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)} \log\left(P\left(\mathbf{x}(k+1)|\mathbf{x}(k)\right)\right)\Big\} \\ &+ \sum_{l=1}^{N}\mathbb{E}\Big\{-\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)} \log\Big(P(\mathbf{x}(k+1)|\mathbf{z}^{(l)}(1:k+1))\Big)\Big\} \\ &- \sum_{l=1}^{N}\mathbb{E}\Big\{-\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)} \log\Big(P(\mathbf{x}(k+1)|\mathbf{z}^{(l)}(1:k))\Big)\Big\}, \qquad \text{(C.9)}\end{aligned}$$

which equals $\boldsymbol{C}^{22}_{\text{FO}}(k)$ based on Eq. (5.28).

Going back to complete the proof of Theorem 7, we note that the information sub-matrix $\boldsymbol{J}_{\mathrm{FO}}\big(\mathbf{x}(k{+}1)\big)$ is given by the inverse of the right bottom $(n_x \times n_x)$ block corresponding to $\boldsymbol{C}^{22}_{\mathrm{FO}}(k)$ in Eq. (C.4), i.e.,

$$\begin{aligned}\boldsymbol{J}_{\mathrm{FO}}\big(\mathbf{x}(k{+}1)\big) &= \boldsymbol{C}^{22}_{\mathrm{FO}}(k) - \big[\mathbf{0}\ \boldsymbol{C}^{21}_{\mathrm{FO}}(k)\big] \times \begin{bmatrix} \boldsymbol{A}^{11}_{\mathrm{FO}}(k) & \boldsymbol{A}^{12}_{\mathrm{FO}}(k) \\ \boldsymbol{A}^{21}_{\mathrm{FO}}(k) & \boldsymbol{A}^{22}_{\mathrm{FO}}(k){+}\boldsymbol{C}^{11}_{\mathrm{FO}}(k) \end{bmatrix}^{-1} \times \begin{bmatrix} \mathbf{0} \\ \boldsymbol{C}^{12}_{\mathrm{FO}}(k) \end{bmatrix} \\ &= \boldsymbol{C}^{22}_{\mathrm{FO}}(k) - \boldsymbol{C}^{21}_{\mathrm{FO}}(k)\Big(\boldsymbol{A}^{22}_{\mathrm{FO}}(k){-}\boldsymbol{A}^{21}_{\mathrm{FO}}(k)\big[\boldsymbol{A}^{11}_{\mathrm{FO}}(k)\big]^{-1}\boldsymbol{A}^{12}_{\mathrm{FO}}(k){+}\boldsymbol{C}^{11}_{\mathrm{FO}}(k)\Big)^{-1}\boldsymbol{C}^{12}_{\mathrm{FO}}(k).\end{aligned}\tag{C.10}$$

Further, Term $\boldsymbol{J}_{\mathrm{FO}}(\mathbf{x}(k))$, defined as the information submatrix for estimating $\mathbf{x}(k)$, is given by the inverse of the $(n_x \times n_x)$ right-lower block of $\big[\boldsymbol{J}_{\mathrm{FO}}\big(\mathbf{x}(0:k)\big)\big]^{-1}$ in Eq. (C.2). Based on the matrix inversion Lemma [152], the middle term in Eq. (C.10) reduces to

$$\boldsymbol{A}^{22}_{\mathrm{FO}}(k) - \boldsymbol{A}^{21}_{\mathrm{FO}}(k)\big[\boldsymbol{A}^{11}_{\mathrm{FO}}(k)\big]^{-1}\boldsymbol{A}^{12}_{\mathrm{FO}}(k) = \boldsymbol{J}_{\mathrm{FO}}(\mathbf{x}(k)). \tag{C.11}$$

Substituting Eq. (C.11) in Eq. (C.10) proves Theorem 7. □

C.3 Proof *of Corollary 1*

Proof. The proofs for Eqs. (5.30) and (5.31) are similar to that for Theorem 7 with the posterior factorization of $P(\mathbf{x}(0:k+1)|\mathbf{z}(1:k+1))$ defined in Lemma 7, [127], below.

Lemma 7. *Assuming that the observations conditioned on the state variables made at node l are independent of the observations made at a different node j, $(j \neq l)$, the global posterior for a N-sensor network is*

$$P\left(\mathbf{x}(0{:}k{+}1)|\mathbf{z}(1{:}k{+}1)\right) \propto \frac{\prod_{l=1}^{N} P\big(\mathbf{x}(k{+}1)|\mathbf{z}^{(l)}(k{+}1), \mathbf{z}(1{:}k)\big)}{\prod_{l=1}^{N} P\big(\mathbf{x}(k{+}1)|\mathbf{z}(1{:}k)\big)} P\left(\mathbf{x}(k{+}1)|\mathbf{x}(k)\right) P\left(\mathbf{x}(0{:}k)|\mathbf{z}(1{:}k)\right). \tag{C.12}$$

The change in $P(\mathbf{x}(0\!:\!k\!+\!1)|\mathbf{z}(1\!:\!k\!+\!1))$ is due to the setup used in Scenario 2, where both current local observation and previous global observations are used in the current state estimate. □

C.4 Proof *of Lemma 9*

Proof. Using the Markovian property

$$P\big(\mathbf{x}(0:k+1),\mathbf{z}(k+1)|\mathbf{z}(1:k)\big) = P\big(\mathbf{z}(k+1)|\mathbf{x}(k+1)\big)P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)P\big(\mathbf{x}(0:k)|\mathbf{z}(1:k)\big). \tag{C.13}$$

Considering independent observations given the state variables, the first term on the right hand side (RHS) of Eq. (C.13) is

$$P\big(\mathbf{z}(k+1)|\mathbf{x}(k+1)\big) = \prod_{l=1}^{N} P\big(\mathbf{z}^{(l)}(k+1)|\mathbf{x}(k+1)\big) = \prod_{l=1}^{N} \frac{P\left(\mathbf{x}(k+1),\mathbf{z}^{(l)}(k+1)|\mathbf{z}^{(l)}(1:k)\right)}{P\left(\mathbf{x}(k+1)|\mathbf{z}^{(l)}(1:k)\right)}. \tag{C.14}$$

Using the Chong-Mori-Chang track-fusion theorem [127], the third term on the RHS of Eq. (C.13) is factorized as follows

$$P\left(\mathbf{x}(0:k)|\mathbf{z}(1:k)\right) \propto \frac{\prod_{l=1}^{N} P(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k))}{\prod_{l=1}^{N} P\left(\mathbf{x}(k)|\mathbf{z}^{(l)}(1:k-1)\right)} P\left(\mathbf{x}(k)|\mathbf{x}(k-1)\right) P\left(\mathbf{x}(0:k-1)|\mathbf{z}(1:k-1)\right). \tag{C.15}$$

Finally, substituting (C.14) and (C.15) in (C.13), we get (5.71). □

C.5 Proof *of Theorem 8*

Proof. Decomposing $\mathbf{x}(0:k+1) = [\mathbf{x}^T(0:k-1), \mathbf{x}^T(k), \mathbf{x}^T(k+1)]^T$, Eq. (6.34) for iteration $k+1$ reduces to

$$\boldsymbol{I}(0:k+1) = \mathbb{E}\left\{ - \left[\begin{array}{c:c:c} \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k)} & \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k+1)} \\ \hdashline \Delta_{\mathbf{x}(k)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} & \Delta_{\mathbf{x}(k)}^{\mathbf{x}(k+1)} \\ \hdashline \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k)} & \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \end{array}\right] \log P_c(k+1) \right\} \tag{C.16}$$

$$\triangleq \begin{bmatrix} \boldsymbol{A}_{\mathrm{FO}}^{11}(k) & \boldsymbol{A}_{\mathrm{FO}}^{12}(k) & \mathbf{0} \\ \boldsymbol{A}_{\mathrm{FO}}^{21}(k) & \boldsymbol{A}_{\mathrm{FO}}^{22}(k) + \boldsymbol{C}_{\mathrm{FO}}^{11}(k) & \boldsymbol{C}_{\mathrm{FO}}^{12}(k) \\ \mathbf{0} & \boldsymbol{C}_{\mathrm{FO}}^{21}(k) & \boldsymbol{C}_{\mathrm{FO}}^{22}(k) \end{bmatrix}. \tag{C.17}$$

Block $\mathbf{0}$ stands for a block of all zeros. Terms $\boldsymbol{C}_{\mathrm{FO}}^{11}(k)$, $\boldsymbol{C}_{\mathrm{FO}}^{12}(k)$ and $\boldsymbol{C}_{\mathrm{FO}}^{21}(k)$ are defined as in Eqs. (5.68)-(5.69). Terms $\boldsymbol{A}_{\mathrm{FO}}^{11}(k)$, $\boldsymbol{A}_{\mathrm{FO}}^{12}(k)$, $\boldsymbol{A}_{\mathrm{FO}}^{21}(k)$, and $\boldsymbol{A}_{\mathrm{FO}}^{22}(k)$ are derived as follows

$$\begin{bmatrix} \boldsymbol{A}_{\mathrm{FO}}^{11}(k) & \boldsymbol{A}_{\mathrm{FO}}^{12}(k) \\ \boldsymbol{A}_{\mathrm{FO}}^{21}(k) & \boldsymbol{A}_{\mathrm{FO}}^{22}(k) \end{bmatrix} = \mathbb{E}\left\{ - \left[\begin{array}{c:c} \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k)} \\ \hdashline \Delta_{\mathbf{x}(k)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \end{array}\right] \log P_a(k) \right\} \tag{C.18}$$

where $P_a(k) \triangleq P(\mathbf{x}(0:k)|\mathbf{z}(1:k))$. Term $\boldsymbol{J}_{\mathrm{FO,AUX}}(k)$ is the inverse of the $(n_x \times n_x)$ right-lower block of Eq. (C.18), i.e.,

$$\boldsymbol{J}_{\mathrm{FO,AUX}}(k) = \boldsymbol{A}_{\mathrm{FO}}^{22}(k) - \boldsymbol{A}_{\mathrm{FO}}^{21}(k)\left[\boldsymbol{A}_{\mathrm{FO}}^{11}(k)\right]^{-1}\boldsymbol{A}_{\mathrm{FO}}^{12}(k). \tag{C.19}$$

Term $\boldsymbol{C}_{\mathrm{FO}}^{22}(k) = \mathbb{E}\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P_c(k+1)\}$ is simplified as

$$\begin{aligned} \boldsymbol{C}_{\mathrm{FO}}^{22}(k) &= \mathbb{E}_{P_c(k+1)}\left\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log\left(P\left(\mathbf{x}(k+1)|\mathbf{x}(k)\right)\right)\right\} \\ &+ \sum_{l=1}^{N} \mathbb{E}_{P_c(k+1)}\left\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log\left(P(\mathbf{x}(k+1), \mathbf{z}^{(l)}(k+1))|\mathbf{z}^{(l)}(1:k)\right)\right\} \\ &- \sum_{l=1}^{N} \mathbb{E}_{P_c(k+1)}\left\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log\left(P(\mathbf{x}(k+1)|\mathbf{z}^{(l)}(1:k))\right)\right\}. \end{aligned} \tag{C.20}$$

Finally, using Eq. (C.20) and definitions (5.65)-(5.66), term $\boldsymbol{C}_{\mathrm{FO}}^{22}(k)$ reduces to Eq (5.70). The information sub-matrix $\boldsymbol{L}_{\mathrm{FO}}(\mathbf{x}(k+1))$ can be calculated as the inverse of the right lower $(n_x \times n_x)$

sub-matrix of $[\boldsymbol{I}(\mathbf{x}(0:k+1))_{\mathrm{FO}}]^{-1}$ and Eq. (C.19) as follows

$$\begin{aligned}
\boldsymbol{L}_{\mathrm{FO}}(\mathbf{x}(k+1)) &= \boldsymbol{C}_{\mathrm{FO}}^{22}(k) - \begin{bmatrix}\mathbf{0} & \boldsymbol{C}_{\mathrm{FO}}^{21}(k)\end{bmatrix} \begin{bmatrix} \boldsymbol{A}_{\mathrm{FO}}^{11}(k) & \boldsymbol{A}_{\mathrm{FO}}^{12}(k) \\ \boldsymbol{A}_{\mathrm{FO}}^{21}(k) & \boldsymbol{A}_{\mathrm{FO}}^{22}(k) + \boldsymbol{C}_{\mathrm{FO}}^{11}(k) \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{C}_{\mathrm{FO}}^{12}(k) \end{bmatrix} \\
&= \boldsymbol{C}_{\mathrm{FO}}^{22}(k) - \boldsymbol{C}_{\mathrm{FO}}^{21}(k)\big(\boldsymbol{J}_{\mathrm{FO,AUX}}(\mathbf{x}(k)) + \boldsymbol{C}_{\mathrm{FO}}^{11}(k)\big)^{-1}\boldsymbol{C}_{\mathrm{FO}}^{12}(k). \qquad \text{(C.21)}
\end{aligned}$$

□

D Proof of the Results Reported in Chapter 6

D.1 Local Conditional FIM

Below, we highlight the relationship between the local accumulated conditional FIM $\boldsymbol{I}^{(l)}(0\colon k+1)$ and local instantaneous conditional FIM $\boldsymbol{L}^{(l)}(k+1)$. The local instantaneous conditional FIM $\boldsymbol{L}^{(l)}(k+1)$ is computed using either of the following three approaches: (i) Directly by inverting large matrix $\boldsymbol{I}^{(l)}(0\colon k+1)$; (ii) Recursively as a function of the previous local instantaneous auxiliary FIM $\boldsymbol{J}^{(l)}_{\text{AUX}}(k)$ [55], and; (iii) Recursively as a function of the previous local instantaneous conditional FIM $\boldsymbol{L}^{(l)}(k)$ presented in Result 1. In approach (i), first the local accumulated conditional FIM $\boldsymbol{I}^{(l)}(0\colon k+1)$ is factorized as follows

$$\boldsymbol{I}^{(l)}(0\colon k+1) = \begin{bmatrix} [A^{11}(k+1)]^{(l)} & [A^{12}(k+1)]^{(l)} \\ [A^{21}(k+1)]^{(l)} & [A^{22}(k+1)]^{(l)} \end{bmatrix} = \mathbb{E}\left\{-\left[\begin{array}{c:c} \Delta^{\mathbf{x}(0:k)}_{\mathbf{x}(0:k)} & \Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(0:k)} \\ \hdashline \Delta^{\mathbf{x}(0:k)}_{\mathbf{x}(k+1)} & \Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)} \end{array}\right] \log P^{(l)}_c(k+1))\right\}. \quad \text{(D.1)}$$

Then, the local instantaneous conditional FIM $\boldsymbol{L}^{(l)}(k+1)$ associated with the estimate $\hat{\mathbf{x}}(k+1)$ is obtained by taking the inverse of the $(n_x \times n_x)$ right-lower square block of $[\boldsymbol{I}^{(l)}(0\colon k+1)]^{-1}$ by applying the following matrix inversion Lemma [152].

Lemma 8. *Matrix inversion Lemma:*

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{\Omega}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{\Phi}^{-1} \\ -\boldsymbol{\Phi}^{-1}\boldsymbol{B}^T\boldsymbol{A}^{-1} & \boldsymbol{\Phi}^{-1} \end{bmatrix}, \quad \text{(D.2)}$$

where subblocks $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}$ *have conformable dimensions,* $\boldsymbol{\Omega} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{B}^T$*, and* $\boldsymbol{\Phi} = \boldsymbol{C} - \boldsymbol{B}^T\boldsymbol{A}^{-1}\boldsymbol{B}$.

Based on Lemma 8, the local instantaneous conditional FIM is given by

$$\boldsymbol{L}^{(l)}(k{+}1) = [\boldsymbol{A}^{22}(k{+}1)]^{(l)} - [\boldsymbol{A}^{21}(k{+}1)]^{(l)}[\boldsymbol{A}^{22}(k{+}1)]^{(l)^{-1}}[\boldsymbol{A}^{12}(k{+}1)]^{(l)}. \tag{D.3}$$

which requires inversion of large matrix $[A^{11}(k{+}1)]^{(l)}$.

D.2 Proof of Result 1

Here Result 1 is derived. We also show that under a minor constraint, the result in [55] reduces to Result 1, which is equivalent to replacing the local instantaneous auxiliary FIM $[\boldsymbol{J}_{\mathrm{AUX}}(\mathbf{x}(k))]^{(l)}$ by the local instantaneous conditional FIM $\boldsymbol{L}^{(l)}(k)$. The rational for the approximation is included after the proof.

Proof. The conditional FIM given observations up to and including time $k-1$ is factorized as follows

$$\boldsymbol{I}^{(l)}(0\colon k) = \begin{bmatrix} [\boldsymbol{A}^{11}_{\mathrm{FO}}(k)]^{(l)} & [\boldsymbol{A}^{12}_{\mathrm{FO}}(k)]^{(l)} \\ [\boldsymbol{A}^{21}_{\mathrm{FO}}(k)]^{(l)} & [\boldsymbol{A}^{22}_{\mathrm{FO}}(k)]^{(l)} \end{bmatrix} = \mathbb{E}\left\{-\left[\begin{array}{c:c} \Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(0:k-1)} & \Delta^{\mathbf{x}(k)}_{\mathbf{x}(0:k-1)} \\ \hdashline \Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(k)} & \Delta^{\mathbf{x}(k)}_{\mathbf{x}(k)} \end{array}\right]\log P^{(l)}_c(k)\right\}, \tag{D.4}$$

where $P^{(l)}_c(k) = P(\mathbf{x}(0\colon k), \mathbf{z}^{(l)}(k)|\mathbf{z}^{(l)}(1\colon k-1))$. Term $\boldsymbol{L}^{(l)}(k)$ is the inverse of the right lower block of $[\boldsymbol{I}^{(l)}(0\colon k)]^{-1}$ which is given by (using the matrix inversion lemma)

$$\boldsymbol{L}^{(l)}(k) = [\boldsymbol{A}^{11}_{\mathrm{FO}}(k)]^{(l)} - [\boldsymbol{A}^{21}_{\mathrm{FO}}(k)]^{(l)}[\boldsymbol{A}^{11}_{\mathrm{FO}}(k)]^{(l)^{-1}}[\boldsymbol{A}^{12}_{\mathrm{FO}}(k)]^{(l)}. \tag{D.5}$$

For next iteration $k{+}1$, we have

$$\boldsymbol{I}^{(l)}(0\colon k{+}1) = \mathbb{E}\left\{-\left[\begin{array}{c:c:c} \Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(0:k-1)} & \Delta^{\mathbf{x}(k)}_{\mathbf{x}(0:k-1)} & \Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(0:k-1)} \\ \hdashline \Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(k)} & \Delta^{\mathbf{x}(k)}_{\mathbf{x}(k)} & \Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k)} \\ \hdashline \Delta^{\mathbf{x}(0:k-1)}_{\mathbf{x}(k+1)} & \Delta^{\mathbf{x}(k)}_{\mathbf{x}(k+1)} & \Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)} \end{array}\right]\log P^{(l)}_c(k{+}1)\right\}, \tag{D.6}$$

where $P_c^{(l)}(k+1) = P(\mathbf{x}(0\colon k+1), \mathbf{z}^{(l)}(k+1)|\mathbf{z}^{(l)}(1\colon k))$ which can be factorized as follows

$$.P\big(\mathbf{x}(0\colon k+1), \mathbf{z}^{(l)}(k+1)|\mathbf{z}^{(l)}(1\colon k)\big) = P\big(\mathbf{z}^{(l)}(k+1)|\mathbf{x}(k+1)\big) \times P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\frac{P\big(\mathbf{x}(0\colon k), \mathbf{z}^{(l)}(k)|\mathbf{z}^{(l)}(1\colon k-1)\big)}{P\big(\mathbf{z}^{(l)}(k)|\mathbf{z}^{(l)}(1\colon k-1)\big)}. \tag{D.7}$$

Taking logarithm of Eq. (D.7).

$$\log P_c^{(l)}(k+1) = \log P\big(\mathbf{z}^{(l)}(k+1)|\mathbf{x}(k+1)\big) + \log P_c^{(l)}(k) + \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big) - \log P\big(\mathbf{z}^{(l)}(k)|\mathbf{z}^{(l)}(1\colon k-1)\big).$$

Therefore, Eq. (D.6) reduces to

$$\boldsymbol{I}^{(l)}(0\colon k+1) = \left[\begin{array}{c:c:c} -\mathbb{E}_{P_c^{(l)}(k+1)}\Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(0:k-1)}\log P_c^{(l)}(k) & -\mathbb{E}_{P_c^{(l)}(k+1)}\Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k)}\log P_c^{(l)}(k) & \mathbf{0} \\ \hdashline -\mathbb{E}_{P_c^{(l)}(k+1)}\Delta_{\mathbf{x}(k)}^{\mathbf{x}(0:k-1)}\log P_c^{(l)}(k) & -\mathbb{E}_{P_c^{(l)}(k+1)}\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)}\log P_c^{(l)}(k) + [\boldsymbol{B}^{11}(k)]^{(l)} & [\boldsymbol{B}^{12}(k)]^{(l)} \\ \hdashline \mathbf{0} & [\boldsymbol{B}^{21}(k)]^{(l)} & [\boldsymbol{B}^{22}(k)]^{(l)} \end{array}\right], \tag{D.8}$$

where $P_c^{(l)}(k) = P(\mathbf{x}(0\colon k), \mathbf{z}^{(l)}(k)|\mathbf{z}^{(l)}(1\colon k\text{–}1))$, $[\boldsymbol{B}^{11}(k)]^{(l)}$, $[\boldsymbol{B}^{12}(k)]^{(l)}$, $[\boldsymbol{B}^{21}(k)]^{(l)}$, and $[\boldsymbol{B}^{22}(k)]^{(l)}$ are given by Eqs. (6.38)-(6.40). The four blocks on the top left sub-matrix of Eq. (D.8) are functions of $\mathbf{z}^{(l)}(k)$ which make them different from $[\boldsymbol{A}^{**}(k)]^{(l)}$ in Eq. (D.4). In order to recursively compute $\boldsymbol{L}^{(l)}(k+1)$ from $\boldsymbol{L}^{(l)}(k)$, these four terms are approximated by their expectations with respect to $P(\mathbf{z}^{(l)}(k)|\mathbf{z}^{(l)}(1\colon k-1))$, i.e.,

$$-\mathbb{E}_{P_c^{(l)}(k+1)}\Big\{\Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(0:k-1)}\log P_c^{(l)}(k)\Big\} \approx -\mathbb{E}_{P(\mathbf{z}^{(l)}(k)|\mathbf{z}^{(l)}(1\colon k-1))}\Big\{\mathbb{E}_{P_c^{(l)}(k+1)}\Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(0:k-1)}\log P_c^{(l)}(k)\Big\}$$
$$-\int P(\mathbf{z}^{(l)}(\cancel{k})|\mathbf{z}^{(l)}(1\colon k-1))P_c^{(l)}(k+1)$$
$$\Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(0:k-1)}\log\cancel{x}P_c^{(l)}(k)d\mathbf{x}(0\colon k+1)d\mathbf{z}^{(l)}(k+1)d\mathbf{z}^{(l)}(k) = [\boldsymbol{A}^{11}(k)]^{(l)}. \tag{D.9}$$

Similarly, it can be shown that

$$-\mathbb{E}_{P_c^{(l)}(k+1)}\left\{\Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k)}\log P_c^{(l)}(k)\right\} \approx [\boldsymbol{A}^{12}(k)]^{(l)}. \tag{D.10}$$

$$-\mathbb{E}_{P_c^{(l)}(k+1)}\left\{\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k:k-1)}\log P_c^{(l)}(k)\right\} \approx [\boldsymbol{A}^{21}(k)]^{(l)}. \tag{D.11}$$

$$-\mathbb{E}_{P_c^{(l)}(k+1)}\left\{\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)}\log P_c^{(l)}(k)\right\} \approx [\boldsymbol{A}^{22}(k)]^{(l)}. \tag{D.12}$$

Finally, Eq. (D.8) can be approximated as follows

$$\boldsymbol{I}^{(l)}(0:k+1) \approx \left[\begin{array}{c:c:c} [\boldsymbol{A}^{11}(k)]^{(l)} & [\boldsymbol{A}^{12}(k)]^{(l)} & \mathbf{0} \\ \hdashline [\boldsymbol{A}^{21}(k)]^{(l)} & [\boldsymbol{A}^{22}(k)]^{(l)} + [\boldsymbol{B}^{11}(k)]^{(l)} & [\boldsymbol{B}^{12}(k)]^{(l)} \\ \hdashline \mathbf{0} & [\boldsymbol{B}^{21}(k)]^{(l)} & [\boldsymbol{B}^{22}(k)]^{(l)} \end{array}\right].$$

Going back to complete the proof, we note that the information sub-matrix $\boldsymbol{L}^{(l)}(k+1)$ is given by the inverse of the right bottom $(n_x \times n_x)$ block of $[\boldsymbol{I}^{(l)}(0:k)]^{-1}$ (corresponding to $[\boldsymbol{B}^{22}(k)]^{(l)}$ in Eq. (D.13)), i.e.,

$$\boldsymbol{L}^{(l)}(k+1) = [\boldsymbol{B}^{22}(k)]^{(l)} - \left[\mathbf{0}\ [\boldsymbol{B}^{21}(k)]^{(l)}\right] \begin{bmatrix} [\boldsymbol{A}^{11}(k)]^{(l)} & [\boldsymbol{A}^{12}(k)]^{(l)} \\ [\boldsymbol{A}^{21}(k)]^{(l)} & [\boldsymbol{A}^{22}(k)]^{(l)}+[\boldsymbol{B}^{11}(k)]^{(l)} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ [\boldsymbol{B}^{12}(k)]^{(l)} \end{bmatrix}, \tag{D.13}$$

which results in the following equation

$$\begin{aligned} \boldsymbol{L}^{(l)}(k+1) &= [\boldsymbol{B}^{22}(k)]^{(l)} \\ &- \ [\boldsymbol{B}^{21}(k)]^{(l)}\Big([\boldsymbol{A}^{22}(k)]^{(l)} - [\boldsymbol{A}^{21}(k)]^{(l)}[\boldsymbol{A}^{11}(k)]^{(l)^{-1}}[\boldsymbol{A}^{12}(k)]^{(l)}+[\boldsymbol{B}^{11}(k)]^{(l)}\Big)^{-1}[\boldsymbol{B}^{12}(k)]^{(l)}] \end{aligned} \tag{D.14}$$

Based on Eq. (D.5), the middle term in Eq. (D.14) reduces to $\boldsymbol{L}^{(l)}(k)+[\boldsymbol{B}^{11}(k)]^{(l)}$ which by substituting in Eq. (D.14) proves Result 1. $\square$

Finally we note that Result 1 is valid with the following approximation:

The top left four blocks of the accumulated conditional FIM given by Eq. (D.8) are replaced by their expectations with respect to $P(\mathbf{z}^{(l)}(k)|\mathbf{z}^{(l)}(1:k-1))$.

As shown above, this leads to Eqs. (6.37)-(6.40) of Result 1. Comparing Eqs. (6.37)-(6.40) with our earlier result [55], we note that the instantaneous auxiliary FIM $\boldsymbol{J}^{(l)}_{\text{AUX}}(k)$ is replaced with the instantaneous conditional FIM $\boldsymbol{L}^{(l)}(k)$. Consequently, the CQ/dPCRLB updates the conditional dPCRLB directly without the need of computing the auxiliary FIM leading to significant communication savings (by a factor of 2).

Finally, we note that the centralized conditional PCRLB [152] our earlier result [55] (distributed counterpart of [152]) and Result 1 use approximations at each iteration with the possibility that the error due to approximations accumulates over time [153]. It is difficult to perform an exact error comparison between the result in [55] and the proposed Result 1. Intuitively speaking, the approximation in [55] is only applied to the top left block of the auxiliary FIM, while in Result 1 the approximation is applied to all four blocks of the conditional FIM. Note however that the approximated block in [55] is involved in three inversions to complete the update at each iteration, which propagates the approximation to all the elements of the conditional PCRLB. As such, both approximations have comparable error. This explains why the gap between the two corresponding bounds is negligible as shown by simulations.

D.3 Proof of Result 2

Below, Result 2 is proved. First, we derive Lemma 9 which provides a factorization of the global quantized conditional posterior distribution $P_{Q,c}(k+1)$ at iteration $k+1$ as a function of the local quantized conditional posterior distribution $P^{(l)}_{Q,c}(k+1)$ at iteration $k+1$ and the global quantized conditional posterior distribution $P_{Q,c}(k)$ at iteration k.

Lemma 9. *Assuming that the quantized observations conditioned on the state variables are in-*

dependent, the global posterior for a network with N_f processing nodes is factorized as follows

$$P_{Q,c}(k+1) \triangleq P(\mathbf{x}(0:k+1), Y(k+1)|Y(1:k)) \propto \frac{\prod_{l=1}^{N_f} P_{Q,c}^{(l)}(k+1)}{\prod_{l=1}^{N_f} P\big(\mathbf{x}(k+1)|Y^{(l)}(1:k)\big)} P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big) P_{Q,c}(k), \tag{D.15}$$

where

$$P_{Q,c}(k) \triangleq P(\mathbf{x}(0:k), Y(k)|Y(1:k-1)),$$

and

$$P_{Q,c}^{(l)}(k+1) \triangleq P(\mathbf{x}(0:k+1), Y^{(l)}(k+1)|Y^{(l)}(1:k)).$$

Proof of Lemma 9. Using the Markovian property

$$P_{Q,c}(k+1) = P\big(Y(k+1)|\mathbf{x}(k+1)\big) P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big) P\big(\mathbf{x}(0:k)|Y(1:k)\big). \tag{D.16}$$

Comparing Eq. (D.15) with (D.16), we need to prove: (i) $P\big(Y(k+1)|\mathbf{x}(k+1)\big) \propto \prod_{l=1}^{N_f} P_{Q,c}^{(l)}(k+1)/P\big(\mathbf{x}(k+1)|Y^{(l)}(1:k)\big)$, and; (ii) $P_{Q,c}(k) \propto P\big(\mathbf{x}(0:k)|Y(1:k)\big)$.

Relationship (i): Given the state variables, the observations are assumed to be independent as is the case in most Bayesian estimators. Then, the first term on the right hand side (RHS) of (D.16) is given by

$$P\big(Y(k+1)|\mathbf{x}(k+1)\big) = \prod_{l=1}^{N_f} P\big(Y^{(l)}(k+1)|\mathbf{x}(k+1)\big). \tag{D.17}$$

We also factorize the local conditional distribution at node l, for $(1 \le l \le N_f)$, as follows

$$P\big(\mathbf{x}(k+1), Y^{(l)}(k+1)|Y^{(l)}(1:k)\big) = P\big(Y^{(l)}(k+1)|\mathbf{x}(k+1)\big) P\big(\mathbf{x}(k+1)|Y^{(l)}(1:k)\big). \tag{D.18}$$

In terms of the local likelihood $P(Y^{(l)}(k+1)|\mathbf{x}(k+1))$, Eq. (D.18) can be expressed as follows

$$P\big(Y^{(l)}(k+1)|\mathbf{x}(k+1)\big) = \frac{P\big(\mathbf{x}(k+1), Y^{(l)}(k+1)|Y^{(l)}(1:k)\big)}{P\big(\mathbf{x}(k+1)|Y^{(l)}(1:k)\big)}. \tag{D.19}$$

Substituting Eq. (D.19) in Eq. (D.17), we have

$$P\big(Y(k+1)|\mathbf{x}(k+1)\big) = \prod_{l=1}^{N_f} \frac{P\left(\mathbf{x}(k+1), Y^{(l)}(k+1)|Y^{(l)}(1:k)\right)}{P\left(\mathbf{x}(k+1)|Y^{(l)}(1:k)\right)},$$

which proves Relation (i).

Relationship (ii): Term $P_{Q,c}(k)$ can be factorized as follows

$$P_{Q,c}(k) = P\big(\mathbf{x}(0:k)|Y(1:k)\big)P\big(Y(k)|Y(1:k-1)\big). \tag{D.20}$$

Since $P\big(Y(k)|Y(1:k-1)\big)$ is independent of the state variables, Eq. (D.20) can be expressed as follows

$$P_{Q,c}(k) \propto P\big(\mathbf{x}(0:k)|Y(1:k)\big), \tag{D.21}$$

which proves Relation (ii).

This completes the proof for Lemma 1. $\square$

Proof of Result 2. Given the quantized observations up to and including time k, the global accumulated conditional FIM can be decomposed as follows

$$\boldsymbol{I}_{\mathrm{Q}}^{(\mathrm{G})}(0:k) = \mathbb{E}\left\{-\left[\begin{array}{c:c} \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k)} \\ \hdashline \Delta_{\mathbf{x}(k)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \end{array}\right] \log P_{Q,c}(k)\right\} \triangleq \left[\begin{array}{cc} \boldsymbol{E}_{\mathrm{FO}}^{11}(k) & \boldsymbol{E}_{\mathrm{FO}}^{12}(k) \\ \boldsymbol{E}_{\mathrm{FO}}^{21}(k) & \boldsymbol{E}_{\mathrm{FO}}^{22}(k) \end{array}\right]. \tag{D.22}$$

As stated previously in Appendix A, the instantaneous conditional FIM $\boldsymbol{L}_{\mathrm{Q}}^{(\mathrm{G})}(k)$ is obtained by taking the inverse of the right lower block of $[\boldsymbol{I}_{\mathrm{Q}}^{(\mathrm{G})}(0:k)]^{-1}$. Using Lemma 8 we get

$$\boldsymbol{L}_{\mathrm{Q}}^{(\mathrm{G})}(k) = \boldsymbol{E}_{\mathrm{FO}}^{11}(k) - \boldsymbol{E}_{\mathrm{FO}}^{21}(k)[\boldsymbol{E}_{\mathrm{FO}}^{11}(k)]^{-1}\boldsymbol{E}_{\mathrm{FO}}^{12}(k). \tag{D.23}$$

For iteration $k+1$, we decompose $\mathbf{x}(0:k+1) = [\mathbf{x}^T(0:k-1), \mathbf{x}^T(k), \mathbf{x}^T(k+1)]^T$. As for Eq. (D.22), the global accumulated conditional FIM for iteration $k+1$ is then given by

$$\boldsymbol{I}_{\mathrm{Q}}^{(\mathrm{G})}(0:k+1) = \mathbb{E}\left\{-\left[\begin{array}{c:c:c} \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k)} & \Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k+1)} \\ \hdashline \Delta_{\mathbf{x}(k)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} & \Delta_{\mathbf{x}(k)}^{\mathbf{x}(k+1)} \\ \hdashline \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(0:k-1)} & \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k)} & \Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \end{array}\right] \log P_{Q,c}(k+1)\right\}. \tag{D.24}$$

Using Lemma 9, Eq. (D.24) reduces to

$$I_{\mathrm{Q}}^{(\mathrm{G})}(0\!:k\!+\!1)=\left[\begin{array}{c:c:c} -\mathbb{E}_{P_{Q,c}(k+1)}\Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(0:k-1)}\log P_{Q,c}(k) & -\mathbb{E}_{P_{Q,c}(k+1)}\Delta_{\mathbf{x}(0:k-1)}^{\mathbf{x}(k)}\log P_{Q,c}(k) & \mathbf{0} \\ \hdashline -\mathbb{E}_{P_{Q,c}(k+1)}\Delta_{\mathbf{x}(k)}^{\mathbf{x}(0:k-1)}\log P_{Q,c}(k) & -\mathbb{E}_{P_{Q,c}(k+1)}\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)}\log P_{Q,c}(k)+C_{\mathrm{FO}}^{11}(k) & C_{\mathrm{FO}}^{12}(k) \\ \hdashline \mathbf{0} & C_{\mathrm{FO}}^{21}(k) & C_{\mathrm{FO}}^{22}(k) \end{array}\right] \tag{D.25}$$

where $P_{Q,c}(k+1) \triangleq P(\mathbf{x}(0:k+1), Y(k+1)|Y(1:k))$. Similar to our discussion in Appendix B, the four blocks on the top left sub-matrix of Eq. (D.25) are functions of $Y(k)$, which make them different from $E^{**}(k)$ in Eq. (D.22). In order to recursively compute $L_{\mathrm{Q}}^{(\mathrm{G})}(k+1)$ from $L_{\mathrm{Q}}^{(\mathrm{G})}(k)$, these four blocks are approximated by taking their expectations with respect to $P(Y(k)|Y(1:k\!-\!1))$ resulting in

$$I_{\mathrm{Q}}^{(\mathrm{G})}(0:k+1)\approx\begin{bmatrix} E_{\mathrm{FO}}^{11}(k) & E_{\mathrm{FO}}^{12}(k) & \mathbf{0} \\ E_{\mathrm{FO}}^{21}(k) & E_{\mathrm{FO}}^{22}(k)+C_{Q}^{11}(k) & C_{Q}^{12}(k) \\ \mathbf{0} & C_{Q}^{21}(k) & C_{Q}^{22}(k) \end{bmatrix}, \tag{D.26}$$

where block $\mathbf{0}$ denotes a block of all zeros. Terms $C_{Q}^{11}(k)$, $C_{Q}^{12}(k)$ and $C_{Q}^{21}(k)$ were defined previously in Eqs. (6.52)-(6.53). Next, using Lemma 9, term $C_{Q}^{22}(k)=\mathbb{E}\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log P_{Q,c}(k+1)\}$ in Eq. (D.26) is expressed as

$$\begin{aligned} C_{Q}^{22}(k) &= \mathbb{E}_{P_{Q,c}(k+1)}\Big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log\big(P(\mathbf{x}(k+1)|\mathbf{x}(k))\big)\Big\} \\ &+ \sum_{l=1}^{N_f}\mathbb{E}_{P_{Q,c}(k+1)}\Big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log\big(P(\mathbf{x}(k\!+\!1), Y^{(l)}(k\!+\!1))|Y^{(l)}(1\!:\!k))\big)\Big\} \\ &- \sum_{l=1}^{N_f}\mathbb{E}_{P_{Q,c}(k+1)}\Big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log\big(P(\mathbf{x}(k+1)|Y^{(l)}(1:k))\big)\Big\}. \end{aligned} \tag{D.27}$$

Finally, we note that the two summation terms in Eq. (D.27) are individual sums of the local instantaneous conditional FIMs at iteration $k+1$, i.e.,

$$\sum_{l=1}^{N_f}\mathbb{E}_{P_{Q,c}(k+1)}\Big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)}\log\big(P(\mathbf{x}(k\!+\!1), Y^{(l)}(k\!+\!1))|Y^{(l)}(1\!:\!k))\big)\Big\} \approx \sum_{l=1}^{N_f} L_{\mathrm{Q}}^{(l)}(k+1) \tag{D.28}$$

and

$$\sum_{l=1}^{N_f} \mathbb{E}_{P_{Q,c}(k+1)}\Big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log\big(P(\mathbf{x}(k+1)|Y^{(l)}(1{:}k))\big)\Big\} \approx \sum_{l=1}^{N_f} \boldsymbol{L}_{\mathrm{Q}}^{(l)}(k+1|k). \qquad \text{(D.29)}$$

Term $\boldsymbol{C}_Q^{22}(k)$ in Eq. (D.27), therefore, reduces to

$$\boldsymbol{C}_{\mathrm{Q}}^{22}(k) \approx \sum_{l=1}^{N_f} \boldsymbol{L}_{\mathrm{Q}}^{(l)}(\mathbf{x}(k+1)) - \sum_{l=1}^{N_f} \boldsymbol{L}_{\mathrm{Q}}^{(l)}(k+1|k) + \mathbb{E}\big\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\big\}.$$

The information sub-matrix $\boldsymbol{L}_{\mathrm{Q}}^{(\mathrm{G})}(k+1)$ can then be calculated as the inverse of the right lower $(n_x \times n_x)$ sub-matrix of $[\boldsymbol{I}_{\mathrm{Q}}^{(G)}(0{:}k+1)]^{-1}$ (Eq. (D.26)) as follows

$$\boldsymbol{L}_{\mathrm{Q}}^{(\mathrm{G})}(k+1) \approx \boldsymbol{C}_Q^{22}(k) - \begin{bmatrix}\mathbf{0} & \boldsymbol{C}_Q^{21}(k)\end{bmatrix} \begin{bmatrix} \boldsymbol{E}_{\mathrm{FO}}^{11}(k) & \boldsymbol{E}_{\mathrm{FO}}^{12}(k) \\ \boldsymbol{E}_{\mathrm{FO}}^{21}(k) & \boldsymbol{E}_{\mathrm{FO}}^{22}(k) + \boldsymbol{C}_Q^{11}(k) \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{C}_Q^{12}(k) \end{bmatrix}. \qquad \text{(D.30)}$$

Simplifying Eq. (D.30), we get

$$\boldsymbol{L}_{\mathrm{Q}}^{(\mathrm{G})}(k+1) \approx \boldsymbol{C}_Q^{22}(k) - \boldsymbol{C}_Q^{21}(k)\big(\boldsymbol{L}_{\mathrm{Q}}^{(\mathrm{G})}(k) + \boldsymbol{C}_Q^{11}(k)\big)^{-1}\boldsymbol{C}_Q^{12}(k),$$

where Eq. (D.23) has been used to obtain the final result. This completes the proof for Result 2.

□

E Reduced order Distributed Particle Filter

The UCD/DPF (Section 3.4) , the CSS/DPF (Section 3.1), and the CF/DPF (Chapter 4) implementations are all full-order distributed estimation algorithms (Section 2.1.2.1) where all the state variables are estimated at each node. In this section, I propose a reduced-order distributed implementation of the particle filter which is more suitable for large scale dynamical systems where the dimension of the state vector is relatively large and observations are localized.

As previously stated in Section 2.1.2.2, reduced-order state estimation algorithms [84–86], decompose the large-scale system into smaller subsystems with only a subset of n_x state variables estimated at each subsystem. Such methods are more efficient than full-order distributed implementations both in terms of the computational complexity and the number of transmissions (information transfers) between neighbouring nodes. Most of the existing reduced-order distributed estimation approaches have been developed for *linear dynamical systems* [84], while their *nonlinear* counterparts [85, 86] decouple the subsystem dynamics from each other. In other words, the state model in the subsystems have no or little interaction between themselves.

Motivated by the *nonlinear*, large-scale estimation problems as in smart grids [48], I propose a fusion-based reduced order, distributed implementation of the particle filter (FR/DPF). The FR/DPF partitions the overall system and implements a reduced order, localized particle filter at each lower dimensional subsystem. Unlike the existing *nonlinear* reduced-order tracking approaches [85, 86] that decouple the subsystems from each other, the state dynamics of the subsystems overlap in the FR/DPF, i.e., they share common states and are coupled through local system interactions. The FR/DPF ensures the consistency of its localized marginal filtering distributions with those of its centralized counterpart by introducing state and observation fusion between neighbouring subsystems.

Based on Eqs. (2.18) and (2.19), each subsystem runs a local particle filter and represents its marginalized filtering distribution with its own local particles $\mathbb{X}_i^{(l)}(k-1)$ and their associated weights $W_i^{(l)}(k{-}1)$. Iteration k of the FR/DPF consists of the following three steps (Section E.0.1-E.0.3).

E.0.1 Local Particle Filters (Observation Fusion)

Updating the particles $\mathbb{X}_i^{(l)}(k{-}1)$ at each subsystem is implemented in pretty much the usual way (Eq. (2.76)) but based on localized process models (Eq. (2.19)). In each subsystem, the particle update includes forcing terms $\boldsymbol{d}^{(l)}(k{-}1)$, which are obtained in Section E.0.3, described later. The critical computation step in the local filters is the update of the particle weights $W_i^{(l)}(k-1)$. The weight update (Eq. (2.77)) requires calculation of the likelihood function, $P(\mathbf{z}(k)|\mathbf{x}(k))$ derived from the global observation model. Subsystem S_l, therefore, needs observations, local particles, and their associated weights from all other subsystems S_m, $m \neq l$, for $(1 \leq m \leq N)$. Alternatively, the weight update equation (Eq. (2.77)) at subsystem S_l can be expressed in terms of the local state estimates instead of the particles for states not being estimated at Subsystem S_l. The approximated expression is given by

$$W_i^{(l)}(k) \propto W_i^{(l)}(k-1)P\Big(\mathbf{z}(k)|\mathbb{X}_i^{(l)}(k),\hat{\mathbf{x}}^{(\neq l)}(k|k-1)\Big)\frac{P\Big(\mathbb{X}_i^{(l)}(k)|\mathbb{X}_i^{(l)}(k-1),\hat{\mathbf{x}}^{(\neq l)}(k-1)\Big)}{q\Big(\mathbb{X}_i^{(l)}(k)|\mathbb{X}_i^{(l)}(k-1),\hat{\mathbf{x}}^{(\neq l)}(k-1),\mathbf{z}(k)\Big)} \quad \text{(E.1)}$$

where $\hat{\mathbf{x}}^{(\neq l)}(\cdot)$ are estimates of the state variables *not* included in the local state vector $\mathbf{x}^{(l)}$ for subsystem S_l. Note that Eq. (E.1) for Subsystem S_l still requires all observations from the entire network. Clearly, such an approach is impractical. A further approximation is to limit the observation fusion to the neighbouring nodes $\mathcal{G}^{(l)}$, which have shared states with Subsystem S_l. This also restricts the required non-local state estimates $\mathbf{x}^{\neq l}(\cdot)$ to only those from $\mathcal{G}^{(l)}$. Estimates $\hat{\mathbf{x}}^{\neq l}(k-1)$ are available at the neighbouring nodes in $\mathcal{G}^{(l)}$ from the previous iteration. The predicted state variables $\hat{\mathbf{x}}^{(\neq l)}(k|k-1)$ are computed from particles $\mathbb{X}^{(\neq l)}(k|k-1)$ of the neighbouring nodes.

In the context of the reduced-order illustrative example included in Section 2.1.2.2, Subsystem S_1 updates vector particles $\mathbb{X}_i^{(1)}(k-1) = [\mathbb{X}_{1,i}^{(1)}(k-1), \mathbb{X}_{2,i}^{(1)}(k-1), \mathbb{X}_{3,i}^{(1)}(k-1)]$ based on the reduced-order process model defined in Eq. (2.29). For subsystem S_1, $\hat{\mathbf{x}}^{(\neq 1)}(k) = [\hat{X}_4(k), \hat{X}_5(k)]$ and Eq. (E.1) reduces to

$$\begin{aligned} W_i^{(1)}(k) &\propto W_i^{(1)}(k-1) P\Big(\mathbf{z}(k)|\mathbb{X}_i^{(1)}(k), \hat{X}_4(k|k-1), \hat{X}_5(k|k-1)\Big) \\ &\times \frac{P\Big(\mathbb{X}_i^{(1)}(k)|\mathbb{X}_i^{(1)}(k-1), \hat{X}_4(k-1), \hat{X}_5(k-1)\Big)}{q\Big(\mathbb{X}_i^{(1)}(k)|\mathbb{X}_i^{(1)}(k-1), \hat{X}_4(k-1), \hat{X}_5(k-1), \mathbf{z}(k)\Big)}. \end{aligned} \tag{E.2}$$

Limiting the observation $\mathbf{z}(k)$ to $\mathbf{z}^{(1)}(k)$ and those at the neighbouring nodes $\mathcal{G}^{(1)} = \{S_2\}$, (i.e., $\mathbf{z}^{(2)}(k)$), Eq. (E.2) reduces to

$$\begin{aligned} W_i^{(1)}(k) &\propto W_i^{(1)}(k-1) P\Big(\mathbf{z}^{(1)}(k), \mathbf{z}^{(2)}(k)|\mathbb{X}_i^{(1)}(k), \hat{X}_4(k|k-1)\Big) \\ &\times \frac{P\Big(\mathbb{X}_i^{(1)}(k)|\mathbb{X}_i^{(1)}(k-1), \hat{X}_4(k-1)\Big)}{q\Big(\mathbb{X}_i^{(1)}(k)|\mathbb{X}_i^{(1)}(k-1), \hat{X}_4(k-1), \mathbf{z}^{(1)}(k), \mathbf{z}^{(2)}(k)\Big)}, \end{aligned} \tag{E.3}$$

where $\hat{X}_4(k|k-1) = \sum_{i=1}^{N_s} W_i^{(2)}(k)\mathbb{X}_{4,i}^{(2)}(k|k-1)$ is computed from the updated particles at Subsystem S_2. Note that Eq. (E.3) restricts $\hat{\mathbf{x}}^{(\neq l)}$ to estimates of the state variables at the neighbouring nodes. Intuitively speaking, this approximation works well because of the localized nature of the observations. The approach of restricting observations to their immediate neighborhoods is similar to the distributed estimation methodology used in linear systems [84]. Subsystems S_2 and S_3 also update their particles and weights using a similar localization approach.

E.0.2 Reduced-order State Fusion

The FR/DPF based distributed implementation introduces different estimates of shared states across the network. For example, X_2 and X_3 are both shared between S_1 and S_2 with their own particle sets resulting in different local estimates. For each state variable X_n, $(1 \leq n \leq n_x)$, we define a different state-based neighbourhood $\mathcal{G}_n$ which includes subsystems having X_n in their local state vector. If $\mathcal{G}_n$ contains more than one subsystem, there are multiple estimates of X_n available. Fig. 2.4 lists state neighbourhood $\mathcal{G}_n$ and subsystem neighbourhood $\mathcal{G}^{(l)}$ for system shown in Fig. 2.3.

Fusing the estimated values is considered to provide consistency across the network. Two issues related to state fusion are observed: (i) In order to perform state fusion, the common information between the subsystems sharing the same state variable must be compensated for, or, instead, a conservative fusion rule should be used; (ii) Transferring particle sets corresponding to the shared sates is not practical due to an impractically large number of information transfers. I choose to use a conservative fusion rule and perform the fusion without sending complete set of particles for the shared states. For each shared state $X_n(k)$, Subsystem $S_l \in \mathcal{G}_n$ computes the minimum mean square error (MMSE) estimate $\mu_n^{(l)}(k)$ and its corresponding error covariance matrix $P_n^{(l)}(k)$. The fusion criterion used to merge is the following parallel estimation fusion rule [84]

$$\hat{X}_n^{(\text{fused})}(k) = \Big(\sum_{l\in\mathcal{G}_n}\Big[P_n^{(l)}(k)\Big]^{-1}\Big)^{-1}\Big(\sum_{l\in\mathcal{G}_n}\Big[P_n^{(l)}(k)\Big]^{-1}\mu_n^{(l)}(k)\Big), \tag{E.4}$$

with error covariance $\hat{P}_n^{(\text{fused})}(k) = \sum_{l\in\mathcal{G}_n}[P_n^{(l)}(k)]^{-1}$. The summation terms in Eq. (E.4) are calculated using average consensus algorithms. Once the state fusion process for state $X_n(k)$ is complete, Subsystem $S_l \in \mathcal{G}_n$ updates its local particles for state X_n by generating particles from $\mathcal{N}(X_n^{(\text{fused})}(k), P_n^{(\text{fused})}(k))$.

In the context of the reduced-order illustrative example included in Section 2.1.2.2, I have

$\mathcal{G}_1 = \{S_1\}$ for state $X_1(k)$ implying X_1 is only observed at S_1 and no fusion is needed. For state $X_2(k)$, $\mathcal{G}_2 = \{S_1, S_2\}$. Its fused estimate is

$$\hat{X}_2^{(\text{fused})}(k) = \left(\left[P_2^{(1)}(k)\right]^{-1} + \left[P_2^{(2)}(k)\right]^{-1}\right)^{-1} \left(\left[P_2^{(1)}(k)\right]^{-1}\mu_2^{(1)}(k) + \left[P_2^{(2)}(k)\right]^{-1}\mu_2^{(2)}(k)\right). \quad \text{(E.5)}$$

The process is repeated for all remaining states S_3, S_4, and S_5.

E.0.3 Computing Forcing Terms

The final step is to compute $\boldsymbol{d}^{(l)}(k)$ and $\hat{\mathbf{x}}^{(\neq l)}(k)$ to be used in the next iteration $(k+1)$. At this stage, all subsystems have consistent estimates for their shared states. Subsystem S_l requests the required forcing term $\boldsymbol{d}^{(l)}(k)$ from its neighbours $S_j \in \mathcal{G}^{(l)}$. Subsystem S_j computes $\boldsymbol{d}^{(l)}(k)$ by taking a weighted combination of the particles $\mathbb{X}_i^{(j)}(k)$ corresponding to states included in $\boldsymbol{d}^{(l)}(k)$. Term $\hat{\mathbf{x}}^{(l)}(k)$ is computed the same way as for $\boldsymbol{d}^{(l)}(k)$. In our running example, the forcing term required by Subsystem S_1 is $\boldsymbol{d}^{(1)}(k) = [X_4(k)]$. Subsystem S_2 computes $\boldsymbol{d}^{(1)}(k) = \sum_{i=1}^{N_s} W_i^{(2)}(k) X_{4,i}^{(2)}(k)$, which is then transferred to S_1. Similarly, for the forcing terms at other subsystems S_2 and S_3.

E.0.4 Computational Complexity

Following the approach suggested by Karlsson [122], the computational complexity of the particle filter for n_x state variables and N_s vector particles with $(n_x \times 1)$ dimension, is approximately given by $\mathrm{O}(n_x^2 N_s)$ floating point operations (flops). By partitioning the overall system into N localized subsystems, the number of state variables per subsystem is roughly n_x/N. If N_s vector particles for each reduced state is maintained at each subsystem and taking the extreme case with no state variables shared between neighbouring subsystems, the computational complexity of the FR/DPF is $N \times \mathrm{O}((\frac{n_x}{N})^2 N_s) \approx \mathrm{O}(n_x^2 N_s / N)$. In other words, the FR/DPF provides a computational savings of up to a factor of N over its centralized counterpart. Note that the above is a lower bound as some states will always be shared.

E.1 PCRLB for Reduced-order Distributed Estimation

In this section, I derive the recursive expression for computing the dPCRLB for reduced-order configured systems. The problem I wish to solve is to express the global information sub-matrix, denoted by $\boldsymbol{J}_{\mathrm{RO}}\big(\mathbf{x}(k+1)\big)$, in terms of its previous iterate $\boldsymbol{J}_{\mathrm{RO}}\big(\mathbf{x}(k)\big)$, local FIMs $\boldsymbol{J}_{\mathrm{RO}}\big(\mathbf{x}^{(l)}(k+1)\big)$, and local prediction FIMs $\boldsymbol{J}_{\mathrm{RO}}\big(\mathbf{x}^{(l)}(k+1|k)\big)$, for $1 \leq l \leq N$.

Definition 9. *Term $\boldsymbol{J}_{RO}\big(\mathbf{x}^{(l)}(0:k)\big)$, for $1 \leq l \leq N$, denotes the local FIM corresponding to the local estimate of $\mathbf{x}^{(l)}(0:k)$ derived from the local posterior density $P(\mathbf{x}^{(l)}(0:k)|\mathbf{z}^{(l)}(1:k))$. We define $\boldsymbol{J}_{RO}(\mathbf{x}^{(l)}(k))$ as the FIM submatrix for estimating $\mathbf{x}^{(l)}(k)$ given $\mathbf{z}^{(l)}(1:k)$.*

Definition 10. *Term $\boldsymbol{J}_{RO}\big(\mathbf{x}^{(l)}(0:k+1|k)\big)$ denotes the local FIM corresponding to the local prediction estimate of $\mathbf{x}^{(l)}(0:k+1)$ derived from the local prediction density $P(\mathbf{x}^{(l)}(0:k+1)|\mathbf{z}^{(l)}(1:k))$. Term $\boldsymbol{J}_{RO}(\mathbf{x}^{(l)}(k+1|k))$ is defined as the FIM submatrix for estimating $\mathbf{x}^{(l)}(k+1)$ given $\mathbf{z}^{(l)}(1:k)$.*

As for the full-order system, the inverse of the local filtering FIM, i.e., $[\boldsymbol{J}_{\mathrm{RO}}(\mathbf{x}^{(l)}(k))]^{-1}$ is equal to the $n_{x^{(l)}} \times n_{x^{(l)}}$ right-lower block of $[\boldsymbol{J}_{\mathrm{RO}}(\mathbf{x}^{(l)}(0:k))]^{-1}$. In deriving the recursive expression for computing the reduced-order dPCRLB, I encounter a second form of the reduced-order local FIM (denoted by $\tilde{\boldsymbol{J}}_{\mathrm{RO}}(\mathbf{x}^{(l)}(k))$) as the bound on the local filtering distribution $P(\mathbf{x}^{(l)}(k)|\mathbf{z}^{(l)}(1:k))$, i.e.,

$$\tilde{\boldsymbol{J}}_{\mathrm{RO}}\big(\mathbf{x}^{(l)}(k)\big) = \mathbb{E}\big\{ -\Delta_{\mathbf{x}^{(l)}(k)}^{\mathbf{x}^{(l)}(k)} \log P\big(\mathbf{x}^{(l)}(k)|\mathbf{z}^{(l)}(1:k)\big)\big\}. \tag{E.6}$$

The inverse of the prediction FIM $\boldsymbol{J}_{\mathrm{RO}}\big(\mathbf{x}^{(l)}(k+1|k)\big)$ is given by the inverse of the $n_{x^{(l)}} \times n_{x^{(l)}}$

right-lower block of $\left[J_{\text{RO}}\left(\mathbf{x}^{(l)}(0\!:\!k\!+\!1|k)\right)\right]^{-1}$. The bound on the local prediction is

$$\tilde{J}_{\text{RO}}\left(\mathbf{x}^{(l)}(k+1|k)\right) = \mathbb{E}\left\{-\Delta_{\mathbf{x}^{(l)}(k)}^{\mathbf{x}^{(l)}(k)} \log P\left(\mathbf{x}^{(l)}(k+1)|\mathbf{z}^{(l)}(1\!:\!k)\right)\right\}. \tag{E.7}$$

Next, I present Theorem 9 that forms the basis of the optimal recursive algorithm for updating $J_{\text{RO}}(\mathbf{x}(k))$.

Theorem 9. *The reduced-order FIM* $\{J_{RO}(\mathbf{x}(k))\}$ *for the filtering estimate* $\hat{\mathbf{x}}(k)$ *follows the recursion*

$$\begin{aligned}
J_{RO}(\mathbf{x}(k+1)) &= C_{RO}^{22}(k) - C_{RO}^{21}(k)\left(J_{RO}(\mathbf{x}(k)) + C_{RO}^{11}(k)\right)^{-1} C_{RO}^{12}(k) && \text{(E.8)}\\
C_{RO}^{11}(k) &= \mathbb{E}\left\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k)} \log P\left(\mathbf{x}(k\!+\!1)|\mathbf{x}(k)\right)\right\}, && \text{(E.9)}\\
C_{RO}^{12}(k) &= \left[C_{RO}^{21}(k)\right]^{T} = \mathbb{E}\left\{-\Delta_{\mathbf{x}(k)}^{\mathbf{x}(k+1)} \log P\left(\mathbf{x}(k\!+\!1)|\mathbf{x}(k)\right)\right\}, && \text{(E.10)}\\
\textit{and } C_{RO}^{22}(k\!+\!1) &= \mathbb{E}\left\{-\Delta_{\mathbf{x}(k+1)}^{\mathbf{x}(k+1)} \log P\left(\mathbf{x}(k\!+\!1)|\mathbf{x}(k)\right)\right\}\\
&\quad + \sum_{l=1}^{N}\left([T^{(l)}(k)]^{+}[\tilde{J}_{RO}(\mathbf{x}^{(l)}(k\!+\!1)) - \tilde{J}_{RO}(\mathbf{x}^{(l)}(k\!+\!1|k))][T^{(l)}(k)]^{+^{T}}\right). && \text{(E.11)}
\end{aligned}$$

Derived for reduced-order estimation, Theorem 9 is similar in nature to Theorem 7 for the full-order dPCRLB (Eqs. (5.26)-(5.27)) except for $C_{\text{RO}}^{22}(k)$ which involves local reduced-order FIMs $J_{\text{RO}}(\mathbf{x}^{(l)}(k))$ and $J_{\text{RO}}(\mathbf{x}^{(l)}(k\!+\!1|k))$. Terms $C_{\text{RO}}^{11}(k)$, $C_{\text{RO}}^{12}(k)$ and $C_{\text{RO}}^{21}(k)$ are the same as their counterparts and still based on the overall state model. As for full-order systems, terms $\tilde{J}_{\text{RO}}(\mathbf{x}^{(l)}(k\!+\!1))$ and $\tilde{J}_{\text{RO}}(\mathbf{x}^{(l)}(k\!+\!1|k))$ are approximated by their counterparts $J_{\text{RO}}(\mathbf{x}^{(l)}(k\!+\!1))$ and $J_{\text{RO}}(\mathbf{x}^{(l)}(k\!+\!1|k))$. Later in this section, I investigate how to compute these terms locally within each reduced-order subsystem. Theorem 9 (Eqs. (E.8)-(E.11)) provides the optimal recursive expression for computing the global FIM in terms of of local reduced-order FIMs, when the spatial decomposition of the system maintains the structure of the overall process model. The proof of Theorem 9 is provided below.

Proof of Theorem 9. To prove Theorem 9, we use a different factorization of the posterior, which expresses the global posterior distribution $P(\mathbf{x}(0:k)|\mathbf{z}(1:k))$ as a function of local reduced-order filtering distributions $P(\mathbf{x}^{(l)}(k)|\mathbf{z}^{(l)}(1:k))$. Lemma 10 [176] describes the nonlinear fusion rule.

Lemma 10. *Assuming that the observations conditioned on the state variables made at node l are independent of the observations made at node j, $(j \neq l)$, the global posterior for a reduced-order estimation model is given by*

$$P(\mathbf{x}(0:k)|\mathbf{z}(1:k)) \propto \frac{\prod_{l=1}^{N} P\left(\mathbf{x}^{(l)}(k)|\mathbf{z}^{(l)}(1:k)\right)}{\prod_{l=1}^{N} P\left(\mathbf{x}^{(l)}(k)|\mathbf{z}^{(l)}(1:k-1)\right)} P(\mathbf{x}(k)|\mathbf{x}(k-1))\, P(\mathbf{x}(0:k-1)|\mathbf{z}(1:k-1)). \tag{E.12}$$

Due to limited space, we only highlight the main steps of the proof. The FIM $\boldsymbol{J}_{\mathrm{RO}}(\mathbf{x}(0:k+1))$ and the associated notation $\boldsymbol{E}^{**}_{**}(k)$ for the reduced-order is similar in structure to Eq. (C.3) except the subscript 'FO' is replaced by 'RO'. Using factorization (E.12) in the first term on RHS of Eq. (C.3) for $\boldsymbol{J}_{\mathrm{RO}}(\mathbf{x}(0:k+1))$ and simplifying

$$\boldsymbol{J}_{\mathrm{RO}}\big(\mathbf{x}(0:k+1)\big) = \left[\begin{array}{cc:c} \boldsymbol{A}^{11}_{\mathrm{RO}}(k) & \boldsymbol{A}^{12}_{\mathrm{RO}}(k) & \mathbf{0} \\ \boldsymbol{A}^{21}_{\mathrm{RO}}(k) & \boldsymbol{A}^{22}_{\mathrm{RO}}(k) + \boldsymbol{C}^{11}_{\mathrm{RO}}(k) & \boldsymbol{C}^{12}_{\mathrm{RO}}(k) \\ \hdashline \mathbf{0} & \boldsymbol{D}^{21}_{\mathrm{RO}}(k) & \boldsymbol{C}^{22}_{\mathrm{RO}}(k) \end{array}\right], \tag{E.13}$$

where terms $\boldsymbol{A}^{11}_{\mathrm{RO}}(k)$, $\boldsymbol{A}^{12}_{\mathrm{RO}}(k)$, $\boldsymbol{A}^{21}_{\mathrm{RO}}(k)$ and $\boldsymbol{A}^{22}_{\mathrm{RO}}(k)$ are the same as their full-order counterparts (i.e., $\boldsymbol{A}^{**}_{\mathrm{RO}}(k) = \boldsymbol{A}^{**}_{\mathrm{FO}}(k)$) as defined in Eq. (C.2) and $\boldsymbol{C}^{11}_{\mathrm{RO}}(k)$, $\boldsymbol{C}^{12}_{\mathrm{RO}}(k)$, $\boldsymbol{C}^{21}_{\mathrm{RO}}(k)$, and $\boldsymbol{C}^{22}_{\mathrm{RO}}(k)$ are expressed in Eqs. (E.9)-(E.11). Note that the derivation of Eq. (E.13) is similar to the derivation of (C.4) included in the proof of Theorem 7. The information sub-matrix $\boldsymbol{J}_{\mathrm{RO}}(\mathbf{x}(k+1))$ is calculated as the inverse of the right lower $(n_x \times n_x)$ sub-matrix of $[\boldsymbol{J}_{\mathrm{RO}}(\mathbf{x}(0:k+1))]^{-1}$ in Eq. (E.13) which is given by Eq. (E.8). □

E.1.1 Reduced-order Computation of RO/dPCRLB

In order to compute the RO/dPCRLB, one approach is to follow the steps listed for the full-order scenario in Section 5.2.2. This will result in the global FIM at each node. In a reduced-order system, the processing nodes do not have access to the global model nor estimates for all states, therefore, such an approach is impractical. Instead, I propose computation of a block of FIM that corresponds to the states local at a node. In my approach, subsystem l computes the diagonal block $\boldsymbol{J}_{\text{RO}}^{\text{Global}}(\mathbf{x}^{(l)}(k+1))$ of the FIM $\boldsymbol{J}_{\text{RO}}(\mathbf{x}(k+1))$ corresponding to its local sates $\mathbf{x}^{(l)}(k)$. The FIM block for $\mathbf{x}^{(l)}(k)$ is

$$\boldsymbol{J}_{\text{RO}}^{\text{Global}}(\mathbf{x}^{(l)}(k+1)) = \boldsymbol{T}^{(l)}(k)\boldsymbol{J}_{\text{RO}}(\mathbf{x}(k+1))[\boldsymbol{T}^{(l)}(k)]^T, \tag{E.14}$$

where $\boldsymbol{T}^{(l)}(k)$ denotes the $(n_{x^{(l)}} \times n_x)$ transformation matrix. Exploiting the block banded structure of the global FIM, the dPCRLB for the local states is then computed from the local FIM block and the adjacent blocks obtained from the neighbouring nodes. This is explained later in Step 3.

I first outline the procedure for updating FIM block $\boldsymbol{J}_{\text{RO}}^{\text{Global}}(\mathbf{x}^{(l)}(k+1))$ at node l. Using Theorem 9, Eq. (E.14) is expanded as follows

$$\begin{aligned}\boldsymbol{J}_{\text{RO}}^{\text{Global}}\big(\mathbf{x}^{(l)}(k+1)\big) &= [\boldsymbol{C}_{\text{RO}}^{22}(k)]^{(l)} \\ &- \boldsymbol{T}^{(l)}(k)\boldsymbol{C}_{\text{RO}}^{21}(k)\underbrace{\big(\boldsymbol{J}_{\text{RO}}^{\text{Global}}(\mathbf{x}(k)) + \boldsymbol{C}_{\text{RO}}^{11}(k)\big)^{-1}}_{\boldsymbol{S}(k)}[\boldsymbol{T}^{(l)}(k)\boldsymbol{C}_{\text{RO}}^{21}(k)]^T\end{aligned} \tag{E.15}$$

where $[\boldsymbol{C}_{\text{RO}}^{22}(k)]^{(l)} = \boldsymbol{T}^{(l)}(k)\boldsymbol{C}_{\text{RO}}^{22}(k)[\boldsymbol{T}^{(l)}(k)]^T$. Next I describe the steps required to compute Eq. (E.15) in a distributed reduced-order fashion.

Step 1: In order to compute $[\boldsymbol{C}_{\text{RO}}^{22}(k)]^{(l)}$, node l, for $1 \leq l \leq N$, needs to compute local FIM blocks $\boldsymbol{J}_{\text{RO}}(\mathbf{x}^{(l)}(k+1))$ and $\boldsymbol{J}_{\text{RO}}(\mathbf{x}^{(l)}(k+1|k))$. Based on Proposition 1 (following the procedure for derivation of Eq. (5.38)), I get

$$\boldsymbol{J}_{\mathrm{RO}}\big(\mathbf{x}^{(l)}(k+1)\big) = [\boldsymbol{D}^{22}_{\mathrm{RO}}(k)]^{(l)} - [\boldsymbol{D}^{21}_{\mathrm{RO}}(k)]^{(l)}\Big(\boldsymbol{J}_{\mathrm{RO}}\big(\mathbf{x}^{(l)}(k)\big) + [\boldsymbol{D}^{11}_{\mathrm{RO}}(k)]^{(l)}\Big)^{-1}[\boldsymbol{D}^{12}_{\mathrm{RO}}(k)]^{(l)} \tag{E.16}$$

with

$$\begin{aligned}
[\boldsymbol{D}^{11}_{\mathrm{RO}}(k)]^{(l)} &= \mathbb{E}\Big[-\Delta^{\mathbf{x}^{(l)}_{(k)}}_{\mathbf{x}^{(l)}_{(k)}}\log P\big(\mathbf{x}^{(l)}(k{+}1)|\mathbf{x}^{(l)}(k), \boldsymbol{d}^{(l)}(k)\big)\Big], && \text{(E.17)}\\
[\boldsymbol{D}^{12}_{\mathrm{RO}}(k)]^{(l)} &= \mathbb{E}\Big[-\Delta^{\mathbf{x}^{(l)}(k+1)}_{\mathbf{x}^{(l)}(k)}\log P\big(\mathbf{x}^{(l)}(k{+}1)|\mathbf{x}^{(l)}(k), \boldsymbol{d}^{(l)}(k)\big)\Big] && \text{(E.18)}\\
\text{and }\ [\boldsymbol{D}^{22}_{\mathrm{RO}}(k)]^{(l)} &= \mathbb{E}\Big[-\Delta^{\mathbf{x}^{(l)}_{(k+1)}}_{\mathbf{x}^{(l)}_{(k+1)}}\log P\big(\mathbf{x}^{(l)}(k{+}1)|\mathbf{x}^{(l)}(k), \boldsymbol{d}^{(l)}(k)\big)\Big]\\
&\quad - \mathbb{E}\Big[\Delta^{\mathbf{x}^{(l)}_{(k+1)}}_{\mathbf{x}^{(l)}_{(k+1)}}\log P\big(\mathbf{z}^{(l)}(k{+}1)|\mathbf{x}^{(l)}(k{+}1)\big)\Big]. && \text{(E.19)}
\end{aligned}$$

The local predictive FIM is similarly derived from Eq. (5.44) and is given by

$$\boldsymbol{J}_{\mathrm{RO}}(\mathbf{x}^{(l)}(k{+}1|k)) = [\boldsymbol{B}^{22}_{\mathrm{RO}}(k)]^{(l)} - [\boldsymbol{D}^{21}_{\mathrm{RO}}(k)]^{(l)}(\boldsymbol{J}_{\mathrm{RO}}(\mathbf{x}^{(l)}(k)) + [\boldsymbol{D}^{11}_{\mathrm{RO}}(k)]^{(l)})^{-1}[\boldsymbol{D}^{12}_{\mathrm{RO}}(k)]^{(l)} \tag{E.20}$$

where

$$[\boldsymbol{B}^{22}_{\mathrm{RO}}(k)]^{(l)} = \mathbb{E}\{-\Delta^{\mathbf{x}(k+1)}_{\mathbf{x}(k+1)}\log P(\mathbf{x}^{(l)}(k{+}1)|\mathbf{x}^{(l)}(k), \boldsymbol{d}^{(l)}(k))\}. \tag{E.21}$$

Note that terms $[\boldsymbol{D}^{11}_{\mathrm{RO}}(k)]^{(l)}$, $[\boldsymbol{D}^{12}_{\mathrm{RO}}(k)]^{(l)}$, $[\boldsymbol{D}^{22}_{\mathrm{RO}}(k)]^{(l)}$, and $[\boldsymbol{B}^{22}_{\mathrm{RO}}(k)]^{(l)}$ are based on reduced-order models and can be computed locally.

Step 2: Having computed the local FIMs $\boldsymbol{J}_{\mathrm{RO}}(\mathbf{x}^{(l)}(k+1))$ and $\boldsymbol{J}_{\mathrm{RO}}(\mathbf{x}^{(l)}(k{+}1|k))$, node l computes $[\boldsymbol{C}^{22}_{\mathrm{RO}}(k)]^{(l)}$ with a modified version of Eq. (E.11) where the summation is limited to neighbouring nodes of node l with which it has shared states. Due to the sparse and localized nature of the process model, only the neighbouring nodes of subsystem l have shared states with node l. Therefore, the communication and computational overheads for the distributed computation of $[\boldsymbol{C}^{22}_{\mathrm{RO}}(k)]^{(l)}$ is limited to its local neighbourhoods.

Step 3: The next step is to compute the second term on the RHS of Eq. (E.15). (i) First, note that because the local state model at node l only includes a subset of state variables, $\mathbf{x}^{(l)}(\cdot)$ and $\boldsymbol{d}^{(l)}(\cdot)$, derivations with respect to $\mathbf{x}(\cdot)$ will result in a block of zero terms corresponding to the states not present in the local state model. Therefore, $\boldsymbol{T}^{(l)}(k)\boldsymbol{C}^{21}_{\mathrm{RO}}(k)$ is partitioned as $\left[[\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l)} \,\vdots\, [\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l,d)} \,\vdots\, \mathbf{0}\right]$, with

$$[\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l)} = \boldsymbol{T}^{(l)}(k)\boldsymbol{C}^{21}_{\mathrm{RO}}(k)[\boldsymbol{T}^{(l)}(k)]^T = \mathbb{E}\{-\Delta^{\mathbf{x}^{(l)}(k+1)}_{\mathbf{x}^{(l)}(k)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\}, \tag{E.22}$$

and

$$[\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l,d)} = \boldsymbol{T}^{(l)}(k)\boldsymbol{C}^{21}_{\mathrm{RO}}(k)[\boldsymbol{T}^{(l,d)}(k)]^T = \mathbb{E}\{-\Delta^{\boldsymbol{d}^{(l)}(k+1)}_{\mathbf{x}^{(l)}(k)} \log P\big(\mathbf{x}(k+1)|\mathbf{x}(k)\big)\}. \tag{E.23}$$

Matrix $\boldsymbol{T}^{(l,d)}(k)$ denotes the $n_{d^{(l)}} \times n_{x^{(l)}}$ nodal transformation matrix corresponding to the $n_{d^{(l)}}$ required forcing terms $\boldsymbol{d}^{(l)}(k)$ at node l. (ii) Second, based on the above partitioning, a subdivision of matrix $\boldsymbol{S}(k)$ is constructed as follows

$$\left[\begin{array}{c:c} \boldsymbol{S}^{(l)}(k) & \boldsymbol{S}^{(l,d)}(k) \\ \hdashline [\boldsymbol{S}^{(l,d)}(k)]^T & \boldsymbol{S}^{(d,d)}(k) \end{array}\right] = \left[\begin{array}{c:c} \boldsymbol{T}^{(l)}(k)\boldsymbol{S}(k)[\boldsymbol{T}^{(l)}(k)]^T & \boldsymbol{T}^{(l)}(k)\boldsymbol{S}(k)[\boldsymbol{T}^{(l,d)}(k)]^T \\ \hdashline [\boldsymbol{T}^{(l)}(k)\boldsymbol{S}(k)[\boldsymbol{T}^{(l,d)}(k)]^T]^T & \boldsymbol{T}^{(l,d)}(k)\boldsymbol{S}(k)[\boldsymbol{T}^{(l,d)}(k)]^T \end{array}\right]. \tag{E.24}$$

Note that, $\boldsymbol{T}^{(l)}(k)\boldsymbol{S}(k)[\boldsymbol{T}^{(l)}(k)]^T$ is $(n_{x^{(l)}} \times n_{x^{(l)}})$, sub-block of $\boldsymbol{S}(k)$. (iii) Finally, the RHS of Eq. (E.15) is expanded as follows

$$\begin{aligned} &\boldsymbol{T}^{(l)}(k)\boldsymbol{C}^{21}_{\mathrm{RO}}(k)\boldsymbol{S}(k)[\boldsymbol{T}^{(l)}(k)\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^T \\ &= \quad [\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l)}\boldsymbol{S}^{(l)}(k)[\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l)^T} + [\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l)}\boldsymbol{S}^{(l,d)}(k)[\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l,d)^T} \\ &+ \quad \big([\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l)}\boldsymbol{S}^{(l,d)}(k)[\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l,d)^T}\big)^T + [\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l,d)}\boldsymbol{S}^{(d,d)}(k)[\boldsymbol{C}^{21}_{\mathrm{RO}}(k)]^{(l,d)^T}. \end{aligned} \tag{E.25}$$

Two issues need to be addressed at this step. First, although matrix $\boldsymbol{S}(k)$ is inverse of a large $(n_x \times n_x)$ matrix $(\boldsymbol{J}_{\mathrm{RO}}(\mathbf{x}(k)) + \boldsymbol{C}^{11}_{\mathrm{RO}}(k))$, it is not computed directly. Instead the four blocks defined in Eq. (E.24) are computed using block with dimension $(n_{x^{(l)}} \times n_{x^{(l)}})$ at the most and without taking the inverse of large matrix. This can be accomplished using distributed iterate-collapse-inversion-overrelaxation (DICI-OR) algorithm [84]. The DICI-RO is an iterative distributed algorithm used for computing the inverse of the symmetric positive definite banded matrix $\boldsymbol{S}(k)$ defined in Eq. (E.15), when its submatrices in the banded area are distributed among different local nodes. The DICI-RO is a 2-step algorithm with an *iterate step* and a *collapse step.* The iterate step is implemented to compute the corresponding (banded) elements of the inverse of $\boldsymbol{S}(k)$. A nonlinear collapse step is then employed to compute the non-banded elements of the inverse of $\boldsymbol{S}(k)$ from already computed banded elements of the inverse of $\boldsymbol{S}(k)$. Please refer to [84] for further details. In our problem, we need to compute the inverse of $\boldsymbol{S}(k)$ from diagonal blocks distributed across the network at local subsystems. Matrix $\boldsymbol{S}(k) = \boldsymbol{J}_{\text{RO}}^{\text{Global}}(\mathbf{x}(k)) + \boldsymbol{C}_{\text{RO}}^{11}(k)$ is assumed block-banded as only diagonal blocks corresponding to the local subsystems are computed in our algorithm. Matrix $\boldsymbol{C}_{\text{RO}}^{11}(k)$ is also banded because of the localized and sparse nature of the state model. Instead of using the global FIM $\boldsymbol{J}_{\text{RO}}(\mathbf{x}(k))$ and $\boldsymbol{C}_{\text{RO}}^{11}(k)$, the DICI-OR algorithm [84] computes $\boldsymbol{S}^{(l)}(k)$, $\boldsymbol{S}^{(l,d)}(k)$, $\boldsymbol{S}^{(d,d)}(k)$ based on the local FIMs $\boldsymbol{J}_{\text{RO}}^{\text{Global}}(\mathbf{x}^{(m)}(k))$ and $[\boldsymbol{C}_{\text{RO}}^{11}(k)]^{(m)}$ of the neighbouring nodes $m \in \aleph^{(l)}$ of node l. Second, term $[\boldsymbol{C}_{\text{RO}}^{21}(k)]^{(l)}$ can be approximated by its local counterparts, i.e., $[\boldsymbol{C}_{\text{RO}}^{21}(k)]^{(l)} \approx [\boldsymbol{D}_{\text{RO}}^{21}(k)]^{(l)}$ and term $[\boldsymbol{C}_{\text{RO}}^{21}(k)]^{(l,d)}$ is obtained from the local matrix $[\boldsymbol{D}_{\text{RO}}^{21}(k)]^{(m)}$ of neighbouring node m of node l's which has $\boldsymbol{d}^{(l)}$ in its local state vector.

Step 4: Finally, Eq. (E.15) is used to update $\boldsymbol{J}_{\text{RO}}^{\text{Global}}(\mathbf{x}^{(l)}(k+1))$ at node l, for $1 \leq l \leq N$. The convergence of the proposed computational algorithm for estimating a sub-block of the global FIM corresponding to the local state subset is guaranteed by the convergence properties of the DICI-OR algorithm. See [84] for details.

E.1.2 Computing the RO/dPCRLB from localized FIM

The inversion algorithm for block banded matrices can be used to compute the RO/dPCRLB (i.e., to compute inverse of the FIM). One such approach, referred to as the DICI-RO. Note that the FIM is a full matrix and the RO/dPCRLB approach suggested in Section E.1.1 updates only its diagonal block entries. This may result in some variation in the RO/dPCRLB as compared to the approach suggested in Section 5.2.2. The accuracy of the block-banded FIM approach can be improved by computing the off-diagonal blocks, which will additional more computation overhead. In this appendix, I limit myself to obtaining the RO/dPCRLB from the diagonal blocks of the FIM.

E.1.3 Particle Filter Realization for reduced-order dPCRLB

The particle-based computation of the dPCRLB equations for the reduced-order systems is similar to the full-order scenario (Section E.1.3) except for the following differences. At subsystem l, derivations in Eq. (5.48)-(5.53) are now based on the local state vector $\mathbf{x}^{(l)}(\cdot)$. A reduced-order distributed implementation of the particle filter is employed to compute the required particle set $\{\mathbb{X}_i^{(l,\text{FF})}(k), W_i^{(l,\text{FF})}\}$. For example, Eq. (E.17) can be represented in terms of the reduced-order particle sets as

$$[\hat{\boldsymbol{D}}_{\text{RO}}^{11}(k)]^{(l)} \approx -\sum_{i=1}^{N_p} W_i^{(l,\text{FF})}(k)\Big(\Delta_{\mathbf{x}^{(l)}(k)}^{\mathbf{x}^{(l)}(k)} \log P\big(\mathbf{x}^{(l)}(k+1)|\mathbf{x}^{(l)}(k)\big)]\Big)\Big|_{\mathbf{x}^{(l)}(k)=\mathbb{X}_i^{(l,\text{FF})}(k)}. \tag{E.26}$$

For the additive Gaussian forcing terms, the above equation reduces to

$$[\hat{\boldsymbol{D}}_{\text{RO}}^{11}(k)]^{(l)} \approx \sum_{i=1}^{N_p} W_i^{(l,\text{FF})}(k)\Big([\nabla_{\mathbf{x}^{(l)}(k)}\boldsymbol{f}^T(k)]\boldsymbol{Q}^{-1}(k)[\nabla_{\mathbf{x}(k)}\boldsymbol{f}^T(k)]\Big)\Big|_{\mathbf{x}(k)=\mathbb{X}_i^{(l,\text{FF})}(k)}. \tag{E.27}$$

As a final note to the dPCRLB implementations, I note the differences between Theorem 7 (the dPCRLB algorithm for full-order systems) and Theorem 9 (the dPCRLB algorithm for reduced-order systems). Theorem 7 is applicable when the estimates of the entire state vector is available locally at each node. In reduced-order estimation, a different subset of the state vector is estimated at the local nodes. Eq. (5.28) included in Theorem 7 cannot be implemented in the reduced-order systems and is replaced by Eq. (E.11) which allows for reduced-order FIMs corresponding to different subsets of the state vector to be fused to determine the overall FIM. In the reduced-order format, Theorem 9 includes Eqs. (E.9)-(E.10) which are similar to Eqs. (5.26)-(5.27). In reality, reduced-order systems can not compute Eqs. (E.9)-(E.10) directly which requires the entire state vector to be known at each node. In Section E.1.1, I discussed how Eqs. (E.9)-(E.10) in Theorem 9 are computed in a reduced-order fashion.

Bibliography

[1] O. Hlinka, F. Hlawatsch, P.M. Djuric, "Distributed Particle Filtering in Agent Networks: A Survey, Classification, and Comparison," IEEE Signal Processing Magazine, vol. 30, no. 1, pp. 61-81, 2013.

[2] P.K. Varshney, "Distributed Detection and Data Fusion", *New York: Springer-Verlag*,1996.

[3] Y. Bar-Shalom and X. Li, *Multitarget-multisensor tracking: principles and techniques*, 1995.

[4] Y. Huang, W. Liang, H. Yu, and Y. Xiao, "Target tracking based on a distributed particle filter in underwater sensor networks ", In *Wireless Communication and Mobile Computing*, pp. 1023–1033, 2008.

[5] G.G. Rigatos, "Distributed particle filtering over sensor networks for autonomous navigation of UAVs," in *Robot Manipulators*, SciYo Publications, Croatia, 2010.

[6] S. Thrun, W. Burgard, and D. Fox, "Probabilistic Robotics," *The MIT Press,* 2005.

[7] A. Simonetto, T. Keviczky, R. Babuska, "Distributed Non-linear Estimation for Robot Localization using Weighted Consensus ", In *IEEE Inter. Con. on Robotics and Automation*, pp.3026–3031, 2010.

[8] H. Aghajan and A. Cavallaro, "Multi-Camera Networks: Principles and Applications," *New York: Academic*, 2009.

[9] Y. Bar-Shalom, X.R. Li, T. and Kirubarajan, "Estimation with Applications to Tracking and Navigation: Theory, Algorithms, and Software," *New York: Wiley*, 2001.

[10] M. Gaynor, S.L. Moulton, M.Welsh, E. LaCombe, A. Romwan, and J Wynne, "Intergrating wireless sensor networks with the grid," *IEEE Internet Computing*, vol. 8, no. 4, pp. 32-39, 2004.

[11] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore, "Environmental wireless sensor networks," *Proc. IEEE*, vol. 98, pp. 1903-1917, 2010.

[12] J. G. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh, "Wireless sensor networks for healthcare," *Proc. IEEE*, vol. 98, pp. 1947-1960, 2010.

[13] T. Zhao and A. Nehorai, newblock "Distributed sequential Bayesian estimation of a diffusive source in wireless sensor networks," *IEEE Trans. Signal Processing*, vol. 55, pp. 15111524, 2007.

[14] F. Zhao and L. J. Guibas, "Wireless Sensor Networks: An Information Processing Approach," *Amsterdam, The Netherlands: Morgan Kaufmann*, 2004.

[15] R. Tharmarasa, T. Kirubarajan, A. Sinha, and T. Lang, "Decentralized Sensor Selection for Large-Scale Multisensor-Multitarget Tracking," *IEEE Trans. Aerospace and Electronic Systems,* vol. 47, no. 2, pp. 1307-1324, 2011.

[16] M. Coates, "Distributed particle filters for sensor networks," *ISPN Sensor Networks*, pp. 99–107, 2004.

[17] X. Sheng, Y. Hu, and P. Ramanathan, "GMM approximation for multiple targets localization and tracking in wireless sensor network," in *Fourth International Symposium on Information Processing in Sensor Networks*, pp. 181–188, 2005.

[18] D. Gu, "Distributed particle filter for target tracking," in *IEEE Con. on Rob. & Automation*, pp. 3856–3861, 2007.

[19] D. Ustebay, M. Coates, and M. Rabbat, "Distributed auxiliary particle filters using selective gossip," In *IEEE Inter. Conf. on Acoustics, Speech and Signal Proc.*, pp. 3296–3299, 2011.

[20] S. Farahmand, S. I. Roumeliotis, and G. B. Giannakis, "Set-membership constrained particle filter: Distributed adaptation for sensor networks, *IEEE Trans. on Signal Processing*, vol. 59, no. 9, pp. 4122-4138, Sept. 2011.

[21] O. Hlinka, O. Sluciak, F. Hlawatsch, P.M. Djuric, M. Rupp, "Likelihood consensus: Principles and application to distributed particle filtering," In *IEEE Asilomar Conference on Signals, Systems and Computers*, pp.349–353, 2010.

[22] O. Hlinka, O. Slucciak, F. Hlawatsch, P.M. Djuric, and M. Rupp, "Likelihood Consensus and Its Application to Distributed Particle Filtering," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4334-4349, Aug. 2012.

[23] D. Gu, S. Junxi, H. Zhen and L. Hongzuo, "Consensus based distributed particle filter in sensor networks," in *IEEE Int. Con, on Information and Automation*, pp. 302–307, 2008.

[24] M.J. Coates and B.N. Oreshkin, "Asynchronous distributed particle filter via decentralized evaluation of gaussian products," in *Proc. ISIF Int. Conf. Information Fusion*, Edinburgh Scotland, 2010.

[25] L.-L. Ong B. Upcroft T. Bailey M. Ridley S. Sukkarieh and H. Durrant-Whyte, "A decentralised particle filtering algorithm for multitarget tracking across multiple flight vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4539–4544, 2006.

[26] L. L. Ong B. Upcroft M. Ridley T. Bailey S. Sukkarieh and H. Durrant-Whyte, "Decentralised data fusion with particles," in *Australasian Con. on Robotics and Automation*, 2004.

[27] L.L. Ong, T. Bailey, B. Upcroft, and H. Durrant-Whyte, "Decentralised particle filtering for multiple target tracking in wireless sensor networks," in *11th Int. Conference on Information Fusion*, 2008.

[28] S. Lee and M. West, "Markov chain distributed particle filters (MCDPF)," in *IEEE Conf. Decision and Control, Shanghai*, 2009.

[29] B. Balasingam, M. Bolic, P.M. Djuric, and J. Miguez "Efficient distributed resampling for particle filters," In *IEEE Int. Con. on Acoustics, Speech and Sig. Proc.*, pp. 3772–3775, 2011.

[30] M. Bolic, P.M. Djuric, and Sangjin Hong, "Resampling algorithms and architectures for distributed particle filters," In *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2442–2450, 2005.

[31] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," In *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 33–56, 2009.

[32] R. Olfati-Saber, A. Fax and R.M. Murry, "Consensus and coopration in networked multi-agent systems," in *Proceedings of the IEEE*, 2007.

[33] A.G. Dimakis, S. Kar, J.M.F. Moura, M.G. Rabbat and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, pp. 1847–1864, 2010.

[34] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in *46th IEEE Conference on Decision and Control*, pp. 5492–5498, 2007.

[35] D. Bauso, L. Giarre and R. Pesenti, "Non-linear protocols for optimal distributed consensus in networks of dynamic agents," *Systems and Control Letters*, vol. 55, pp. 918–928, 2006.

[36] M.A. Paskin and C.E. Guestrin, "Robust probabilistic inference in distributed systems," *Uncertainty in Articial Intelligence*, 2004.

[37] M. Buehner, P. Malanotte-Rizzoli, A. Busalacchi, and T. Inui, "Estimation of the tropical Atlantic circulation from altimetry data using a reduced-rank stationary Kalman filter," *Elsevier Oceanographic Series*, vol. 68, no. 9, pp. 49–92, 2003.

[38] F. Khellah, P. Fieguth, M. Murray, and M. Allen, "Statistical processing of large image sequences," *IEEE Trans. Image Proc.*, vol. 14, 2005.

[39] G. Rigatos, P. Siano, and N. Zervos, "A Distributed State Estimation Approach to Condition Monitoring of Nonlinear Electric Power Systems," *Asian Journal of Control*, 2012.

[40] M. Ilic, E. Allen, J. Chapman, C. King, J. Lang, and E. Litvinov, "Preventing future blackouts by means of enhanced electric power systems control: From complexity to order," *Proceedings of IEEE*, vol. 93, no. 11, pp. 1920–1941, 2005.

[41] T.C. Henderson and H.F. Durrant-Whyte, *Multisensor Data Fusion*, Springer Hand. of Robotics, 2008.

[42] S. Grime and H. F. Durrant-Whyte, "Data fusion in decentralized sensor networks," in *Control Engineering Practice*, vol. 2, no. 5, pp. 849–863, 1994.

[43] N. Gordon, M. Śanjeev, S. Maskell, and T. Clapp, "A tutorial on particle filters for online non-linear/non-gaussian bayesian tracking," *IEEE Trans. on Sig. Proc.*, vol. 50, pp. 174–187, 2002.

[44] R. Van der Merwe, A. Doucet, N. de Freitas and E. Wan, "The unscented particle filter," Tech. Rep. CUED/F-INFENG/TR 380, Cambridge University, 2000.

[45] N. J. Gordon, D. Salmond, and A. Smith, "Novel approach to non-linear/non-gaussian bayesian state estimation," *IEE Proceedings*, vol. 140(2), pp. 107–113, 1993.

[46] S. Challa, M. Palaniswami and A. Shilton, "Distributed data fusion using support vector machines," in *Int. Con. on Information Fusion, (FUSION)*, vol. 2, p. 881-885, 2002.

[47] M. Rosencrantz, G. Gordon, and S. Thrun, "Decentralized data fusion with distributed particle filters," in *Conference on Uncertainty in AI (UAI)*, 2003.

[48] G.G. Rigatos and P. Siano, "Distributed state estimation for condition monitoring of non-linear electric power systems," In *IEEE Int. Sym. on Industrial Electronics*, pp. 1703–1708, 2011.

[49] A. Mohammadi and A. Asif, "A Constraint Sufficient Statistics based Distributed Particle Filter for Bearing Only Tracking", *IEEE International Conference on Communication (ICC)*, pp. 3670-3675, 2012.

[50] A. Mohammadi and A.Asif, "Consensus-based Particle Filter Implementations for Distributed Non-linear Systems", chapter 9 in *Nonlinear Est. & Applications to Industrial Sys. Control*, Editor G. Rigatos, 2011.

[51] A. Mohammadi and A.Asif, "Distributed Particle Filter Implementation with Intermittent/Irregular Consensus Convergence," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2572-2587, May15, 2013.

[52] A. Mohammadi and A.Asif, "Full Order Nonlinear Distributed Estimation in Intermittently Connected Networks", in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

[53] A. Mohammadi and A.Asif, "Posterior Cramer-Rao Bounds for Full and Reduced-order Distributed Bayesian Estimation", submitted to *IEEE Trans. on Aerospace & Electronic Systems*, 2013.

[54] A. Mohammadi and A. Asif, "Theoretical Performance Bounds for Reduced-order Linear and Nonlinear Distributed Estimation," *IEEE Global Communications Conference (GLOBECOM)*, pp. 3929-3935, 2012.

[55] A. Mohammadi and A.Asif, "Decentralized Conditional Posterior Cramér-Rao Lower Bound for Nonlinear Distributed Estimation," *IEEE Signal Processing Letters*, vol. 20, no. 2, pp. 165–68, Feb. 2013.

[56] A. Mohammadi and A.Asif, "Decentralized Computation of the Conditional Posterior Cramer-Rao Lower Bound: Application to Adaptive Sensor Selection", in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

[57] A. Mohammadi and A. Asif, "Application of Constraint Sufficient Statistics to Distributed Particle Filters: Bearing/Range Tracking," submitted to *IEEE Transactions on Signal Processing*, 2013.

[58] X. Zhong, A. Mohammadi, A. B. Premkumar and A. Asif, "A Distributed Unscented Particle Filtering Approach for Multiple Acoustic Source Tracking Using an Acoustic Vector Sensor Network," submitted to *Elsevier Signal Processing*, 2013.

[59] A. Mohammadi and A. Asif, "Consensus-Based Distributed Unscented Particle Filter," *IEEE Statistical Sig. Proc. (SSP)*, pp. 237–240, 2011.

[60] X. Zhong, A. Mohammadi, W. Wang, A.B. Premkumar, and A. Asif, "Acoustic Source Tracking in a Reverberant Environment Using a Pairwise Synchronous Microphone Network," in *IEEE International Conference on Information Fusion*, 2013.

[61] A. Mohammadi and A. Asif, "Distributed State Estimation for Large-scale Nonlinear Systems: A Reduced Order Particle Filter Implementation," *IEEE Statistical Sig. Proc. (SSP)*, pp. 249-252, 2012.

[62] A. Mohammadi and A. Asif, "Distributed Particle Filters for Intermittent Connections: Feedback Between Fusion and Local Filters improves Performance," *IEEE Statistical Sig. Proc. (SSP)*, pp. 524-527, 2012.

[63] A. Mohammadi and A. Asif, "A Consensus/Fusion based Distributed Implementation of the Particle Filter" in *IEEE Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 285-288, 2011.

[64] A. Mohammadi and A. Asif, "Distributed Posterior Cramer-Rao Lower Bound for Nonlinear Sequential Bayesian Estimation," *IEEE Sensor Array and Multichannel Signal Processing (SAM)*, pp. 509-512, 2012.

[65] A. Mohammadi, A. Asif, X. Zhong, and A. B. Premkumar, "Distributed Computation of the Conditional PCRLB for Quantized Decentralized Particle Filters", submitted to *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, available online at: *arXiv:1307.5435*, 2014.

[66] A. Mohammadi, A. Asif, and A. Asif, X. Zhong, and A.B. Premkumar, "Decentralized Bayesian Estimation with Quantized Observations: Theoretical Performance Bounds," in *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp.149-156, May 2013.

[67] A. Mohammadi and A. Asif, "Decentralized Sensor Selection based on the Distributed Posterior Cramér-Rao Lower Bound," *IEEE International Conference on Information Fusion*, pp. 1668-1675, 2012.

[68] A. Mohammadi, A. Asif, and S. Saxena, "Reduced Order Distributed Particle Filter Estimation in Nonlinear Electric Power Grid," submitted to *IEEE Journal of Selected Topics in Signal Processing,* Special Issue on *Signal Processing in Smart Electric Power Grid*, 2013.

[69] A. Mohammadi and A. Asif, "Distributed particle filtering for large scale dynamical systems," In *IEEE 13th Inter. Multitopic Conference*, pp.1–5, 2009.

[70] C.Y. Chong, "Hierarchical estimation," in *2nd MIT/ONR Workshop on C3*, 1979.

[71] Y. Bar-Shalom, "On the track-to-track correlation problem," *IEEE Trans. Automat. Contr.*, vol. 15, pp. 571–572, 1981.

[72] H. Hashemipour, S. Roy and A. Laub, "Decentralized structures for parallel kalman filtering," *IEEE Trans. Automatic Control*, vol. 33, pp. 88–93, 1988.

[73] B.S. Rao and H.F. Durrant-Whyte, "Fully decentralised algorithm for multisensor kalman filtering," *IEE Proceedings*, vol. 138, no. 5, pp. 413–420, 1991.

[74] S. Utete H. F. Durrant-Whyte, *Network Management in Decentralised Sensing Systems*, Ph.D. thesis, University of Oxford, 1994.

[75] M.S. Mahmoud, H.M. Khalid, "Distributed Kalman filtering: a bibliographic review," *IET Control Theory & Applications,* vol. 7, no. 4, pp. 483-501, 2013.

[76] S. L. Sun and Zi-Li Deng, "Multi-sensor optimal information fusion Kalman filters with applications," *Automatica*, vol. 40, no. 6, pp. 57–62, 2004.

[77] R. Olfati-Saber, "Kalman-consensus filter: Optimality, stability, and performance," in *48th IEEE Conference on Decision and Control*, pp.

[78] R. Olfati-Saber, "Distributed kalman filter with embedded consensus filters," in *44th IEEE Conference on Decision and Control*, pp. 8179–8184, 2005.

[79] R. Olfati-Saber and N.F. Sandell., "Distributed tracking in sensor networks with limited sensing range," in *IEEE American Control Conference.* 2008, p. 3157–3162.

[80] P. Alriksson and A. Rantzer, "Distributed kalman filtering using weighted averaging," in *Int. Symp. on Mathematical Theory of Networs and Systems*, 2006, pp. 8179–8184.

[81] S. Kar, B. Sinopoli, and J. Moura, "Kalman Filtering with Intermittent Observations: Weak Convergence to a Stationary Distribution," to appear in *IEEE Transactions on Automatic Control.*

[82] D.J. Lee, "Nonlinear Estimation and Multiple Sensor Fusion Using Unscented Information Filtering," *IEEE Signal Processing Letters*, vol. 15, pp. 861-864, 2008.

[83] L. Wenling and J. Yingmin Jia "Consensus-Based Distributed Multiple Model UKF for Jump Markov Nonlinear Systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 227-233, 2012.

[84] U.A. Khan and J.M.F. Moura, "Distributing the Kalman filter for large-scale systems," *IEEE Trans. on Signal Processing*, vol. 56, no.10, pp. 4919–4935, 2008.

[85] M. Briers, A. Doucet and S. S. Singh, "Sequential auxiliary particle belief propagation", *Int. Conf. on Inf. Fusion*, pp. 705–711, 2005.

[86] M.F. Bugallo, Ting Lu and P.M. Djuric, "Target Tracking by Multiple Particle Filtering," *Aerospace Conference*, pp.1–7, 2007.

[87] T.M. Berg and H.F. Durrant-Whyte, "Model distribution in decentralized multi-sensor data fusion," *American Cont. Con.*, pp. 2292-2293, 1991.

[88] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, pp.2508–2530, 2006.

[89] A. Papoulis and S.U. Pillai, *Distributed algorithms*, Morgan Kaufmann, 1997.

[90] J. Tsitsiklis, *Problems in decentralized decision making and computation*, Ph.D. thesis, MIT, 1984.

[91] A. Jadbabaie, J. Lin, and A.S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[92] J.A. Fax and R.M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[93] L. Xiao, S. Boyd and S. Lall, "A scheme for asynchronous distributed sensor fusion based on average consensus," in *Int. Symp. on Info. Proc. in Sensor Networks*, 2005.

[94] S. Kar, S.A. Aldosari, and J.M.F. Moura, "Topology for distributed inference on graphs," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2609–2613, 2008.

[95] S. Kar and J.M.F. Moura, "Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise," *IEEE Tran. on Sig. Proc.*, vol. 57, pp. 355–369, 2009.

[96] R. Rajagopal and M. J. Wainwright, "Network-based consensus averaging with general noisy channels," in *Allerton Conference on Communication, Control, and Computing*, 2007.

[97] A. Ozdaglar A. Nedic, A. Olshevsky and J. Tsitsiklis, "On distributed averaging algorithms and quantization effects," Tech. Rep., Massachusetts Institute of Technology, 2007.

[98] Khan, U.A.; Kar, S.; Moura, J.M.F.; , "Distributed average consensus: Beyond the realm of linearity," In *Asilomar Conference on Sig., Systems and Computers*, pp. 1337–1342, 2009.

[99] L. Georgopoulos and M. Hasler, "Non-linear average consensus, *in Proc. NOLTA-09.* pp. 10–14, 2009.

[100] L. Xiao and S. Boyd., "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2004.

[101] R.W. Wei Ren and Beard, D.B. Kingston, "Multi-agent Kalman consensus with relative uncertainty," In *Proceeding of American Control Conference*, vol. 3, pp. 1865–1870, 2005.

[102] B. Ristic and M.S. Arulampalam, "Tracking a manoeuvring target using angle-only measurements: algorithms and performance," *IEEE Trans. on Sig. Proc.*, vol. 83, pp. 1223–1238, 2003.

[103] M.S. Arulampalam, B. Ristic, N. Gordon, and T. Mansell, "Bearings-only tracking of maneuvering targets using particle filters," *Appl. Sig. Proc.*, vol. 15, pp. 2351–2365, 2004.

[104] A. Farina, "Target tracking with bearings-only measurements," *Signal Processing*, vol. 78, no. 1, pp. 61-78, 1999.

[105] X. Zhong and A. B. Premkumar, "Particle filtering approaches for multiple acoustic source detection and 2-D direction of arrival estimation using a single acoustic vector sensor," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4719 –4733, 2012.

[106] Yih-Fang Huang, S. Werner, J. Huang; N. Kashyap, V. Gupta, "State Estimation in Electric Power Grids: Meeting New Challenges Presented by the Requirements of the Future Grid," *IEEE Signal Processing Magazine,* vol. 29, no. 5, pp. 33-43, 2012.

[107] S. Wang, W. Gao, A.P.S. Meliopoulos, "An Alternative Method for Power System Dynamic State Estimation Based on Unscented Transform," *IEEE Transactions on Power Systems*, vol. 27, no. 2, pp. 942-950, May 2012.

[108] G. Valverde, V. Terzija, "Unscented kalman filter for power system dynamic state estimation," *IET Generation, Transmission & Distribution*, vol. 5, no. 1, pp. 29-37, Jan. 2011.

[109] E. Ghahremani, I. Kamwa, "Online State Estimation of a Synchronous Generator Using Unscented Kalman Filter From Phasor Measurements Units," *IEEE Transactions on Energy Conversion*, vol. 26, no. 4, pp. 1099-1108, Dec. 2011.

[110] B. Song, C. Ding, A.T. Kamal, J.A. Farrell, A.K. Roy-Chowdhury, "Distributed Camera Networks," *IEEE Sig. Proc. Magazine*, vol. 28, no. 3, pp. 20-31, 2011.

[111] M. Taj, A. Cavallaro, "Distributed and Decentralized Multicamera Tracking," *IEEE Sig. Proc. Magazine*, vol. 28, no. 3, pp. 46-58, 2011.

[112] P. Perez, J. Vermaak, A. Blake, "Data fusion for visual tracking with particles," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 495-513, 2004.

[113] G.Mallikarjuna Rao and Ch.Satyanarayana, "Visual Object Target Tracking Using Particle Filter: A Survey," *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 5, no. 6, 2013.

[114] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Distributed Kalman filtering based on consensus strategies," *IEEE Journal on Selected Areas in Communications*, vol.26, no.4, pp.622–633, May 2008.

[115] K. Zhou, S.I. Roumeliotis, "Multirobot Active Target Tracking With Combinations of Relative Observations," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 678-695, 2011.

[116] R. Viswanathan, "A note on distributed estimation and sufficiency," in*IEEE Tran. on Inf. Theory*, vol. 39, no. 5, pp. 1765–1767, 1993.

[117] S. M. Kay, "Fundamentals of Statistical Signal Processing: Estimation Theory", Upper Saddle River, NJ, Prentice-Hall, 1993.

[118] X.R. Li, V.P. Jilkov, "Survey of maneuvering target tracking. Part I. Dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333-1364, 2003.

[119] T. L. Song, J. Ahn, and C. Park. "Suboptimal Filter Design with Pseudo-measurements for Target Tracking," in *IEEE Trans. Aerospace and Electronic Systems*, pp. 28–39, 1988.

[120] Y. Mo and B. Sinopoli, "Communication Complexity and Energy Efficient Consensus Algorithm," in *Workshop on Estimation and Control of Networked Systems*, 2010.

[121] U.A. Khan, S. Kar, J.M.F. Moura, "Higher Dimensional Consensus: Learning in Large-Scale Networks," *IEEE Transactions on Signal Processing,* vol. 58, no. 5, pp. 2836–2849, 2010.

[122] R. Karlsson, T. Schon, and F. Gustafsson, "Complexity Analysis of the Marginalized Particle Filter," *IEEE Trans. Signal Processing*, vol. 53, no. 11, pp. 4408-4411, 2005.

[123] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations", *IEEE Trans. Automat. Contr.*, Vol. 49, No. 9, pp. 1453–1464, 2004.

[124] S. Kar, B. Sinopoli, and J. Moura, "Kalman Filtering with Intermittent Observations: Weak Convergence to a Stationary Distribution," to appear in *IEEE Transactions on Automatic Control.*

[125] T. Schon, F. Gustafsson, P.J. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models," *IEEE Trans. on Signal Proc.*, vol. 53, no. 7, pp. 2279-2289, 2005.

[126] Y. Derek, J.P. Reilly, T. Kirubarajan, K. Punithakumar, "Approximate Conditional Mean Particle Filtering for Linear/Nonlinear Dynamic State Space Models," *IEEE Trans. on Sig. Proc.*, vol. 56, no. 12, pp. 5790-5803, 2008.

[127] C.Y. Chong, S. Mori, and K.C. Chang, *Multi-target Multi-sensor Tracking*, chapter Distributed Multi-target Multi-sensor Tracking, Artech House, pp. 248–295, 1990.

[128] Y. Zhu, Z. You, J. Zhao, K. Zhang, and X. Li, "The optimality for the distributed Kalman filtering fusion with feedback", *Automatica*, vol. 37, no. 9, pp. 1489–1493, 2001.

[129] M. Gales and S. Airey, "Product of Gaussians for speech recognition," *Comp. Speech & Lang.*, vol. 20, pp. 22-40, 2006.

[130] H.Q. Liu, H.C. So, F.K.W. Chan, and K.W.K. Lui, "Distributed particle filter for target tracking in sensor networks," *Progress In Electromagnetics Research C*, Vol. 11, Non. pp. 171–182, 2009.

[131] R. Karlsson, T. Schon ,and F. Gustafsson, "Complexity analysis of the marginalized particle filter," *IEEE Trans. Signal Processing*, vol. 53(11), pp. 4408–4411, 2005.

[132] H. L. van Trees, "Detection, Estimation and Modulation Theory," New York, Wiley, 1968.

[133] H. Rowaihy, S. Eswaran, M. Johnson, D. Verma, A. Bar-Noy, T. Brown, and T. L. Porta, "A survey of sensor selection schemes in wireless sensor networks, *in Proc. SPIE*, 2007.

[134] C. Kreucher, K. Kastella, and A. Hero III, "Sensor management using an active sensing approach," *Signal Processing*, vol. 85, no. 3, pp. 607-624, 2005.

[135] F. Zhao, 1. Shin, and J. Reich, "Information-driven dynamic sensor collaboration," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 61-72, 2002.

[136] M. I. Smith, C. R. Angell, M. L. Hernandez, and W. J. Oxford, "Improved data fusion through intelligent sensor management," in *Proceedings of the SPIE*, vol. 5096, 2003.

[137] D. Smith and S.Singh, "Approaches to multisensor data fusion in target tracking: A survey". *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 1696–1710, 2006.

[138] G. M. Hoffmann and C. 1. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 32-47, 2010.

[139] A.S. Chhetri, D. Morrell, and A.P. Suppappola, "The use of particle filter with the unscented transform to schedule sensors," *ICASSP*, 2004.

[140] D. Guo and X. Wang, "Dynamic sensor collaboration via sequential monte carlo," *IEEE J. Sel. Areas in Comm.*, vol. 22, pp. 1037–1047,2004.

[141] L. Zuo, R. Niu, and P. K. Varshney, "Posterior CRLB based sensor selection for target tracking in sensor networks," *Proc. ICASSP*, vol. 2, pp. 1041–1044, 2007.

[142] F. Ghassemi and V. Krishnamurthy, "Separable Approximation for Solving the Sensor Subset Selection Problem," *IEEE Trans. on Aer. & Elec. Sys.*, vol. 47, no. 1, pp. 557-568, 2011.

[143] L.M. Kaplan, "Global node selection for localization in a distributed sensor network," *IEEE Trans. on Aer. & Elec. Sys.*, vol. 42, no. 1, pp. 113-135, 2006.

[144] K. Punithakumar, T. Kirubarajan, M. Hernandez, "Multisensor deployment using PCRLBS, incorporating sensor deployment and motion uncertainties," *IEEE Trans. on Aer. & Elec. Sys.*, vol. 42, no. 4, pp. 1474-1485, 2006.

[145] R. Tharmarasa, T. Kirubarajan, P. Jiming, and T. Lang, "Optimization-Based Dynamic Sensor Management for Distributed Multitarget Tracking," *IEEE Trans. Sys., Man, & Cybernetics*, vol. 39, pp. 534–546, 2009.

[146] M.L. Hernandez, T. Kirubarajan, Y. Bar-Shalom, "Multisensor resource deployment using posterior Cramér-Rao bounds," *IEEE Trans. Aerospace & Elec. Sys.,* vol. 40, no. 2, pp. 399–416, 2004.

[147] R. Tharmarasa, "PCRLB based multisensor array management for multitarget tracking." *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, pp. 539–555, 2007.

[148] P. Tichavsky, C.H. Muravchik, and A. Nehorai, "Posterior Cramér-Rao bounds for discrete-time nonlinear filtering," *IEEE Trans. Sig. Proc.,* vol. 46, no. 5, pp. 1386–1396, 1998.

[149] M.L. Hernandez, A.D. Marrs, N.J. Gordon, S.R. Maskell, and C.M. Reed, "Cramér-Rao bounds for non-linear filtering with measurement origin uncertainty," *Int. Conf. Inf. Fusion*, vol. 1, pp. 18–25, 2002.

[150] M.L. Hernandez, A. Farina, B. Ristic, "PCRLB for tracking in cluttered environments: measurement sequence conditioning approach,," *IEEE Trans. on Aer. & Elec. Sys.*, vol. 42, no. 2, pp. 680-704, 2006.

[151] L. Ming, B.J. Van Wyk, Q. Yong "Online Estimation of the Approximate Posterior Cramer-Rao Lower Bound for Discrete-Time Nonlinear Filtering," *IEEE Trans. on Aer. & Elec. Sys.*, vol. 47, no. 1, pp. 37-57, 2011.

[152] L. Zuo, R. Niu, and P.K. Varshney, "Conditional Posterior Cramér-Rao Lower Bounds for Nonlinear Sequential Bayesian Estimation," *IEEE Trans. Sig. Proc.,* vol. 59, no. 1, 2011.

[153] Y. Zheng, O. Ozdemir, R. Niu, and P.K. Varshney, "New Conditional Posterior Cramer-Rao Lower Bounds for Nonlinear Sequential Bayesian Estimation," *IEEE Trans. Sig. Proc.,* vol. 60, no. 10, pp. 5549-5556, 2012.

[154] T. Brehard and J.R. Le Cadre, "Closed-form posterior Cramer-Rao bounds for bearings-only tracking," *IEEE Trans. on Aer. & Elec. Sys.*, vol. 42, no. 4, pp. 1198-1223, 2006.

[155] C. Hue, J.P. Le Cadre, P. Perez, "Posterior Cramer-Rao bounds for multi-target tracking," *IEEE Trans. on Aer. & Elec. Sys.*, vol. 42, no. 1, pp. 37-49, 2006.

[156] M. Simandl, J. Kralovec, P. Tichavsky, "Filtering, predictive, and smoothing Cramér-Rao bounds for discrete-time nonlinear dynamic systems," *Automatica*, vol. 37, no. 11, pp. 1703-1716, 2001.

[157] L. Ming, P. Del Moral, C. Baehr, "Error analysis of approximated PCRLBs for nonlinear dynamics," *ICCA*, pp. 1988–1993, 2010.

[158] P.M. Djuric, J.H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. f. Bugallo and J. Miguez, "Particle Filtering", *IEEE Sig. Proc. Mag.*, vol. 20, no. 5, pp. 19–38, 2003.

[159] B. Ristic, S. Arulampalam, and N. Gordon, "Beyond the Kalman Filter. Particle Filters for Tracking Applications," Norwood, MA: Artech House, 2004.

[160] X. Zhang, and P.K. Willett, "Cramér-Rao bounds for discrete time linear filtering with measurement origin uncertainty" *Workshop on Estimation, Track. & Fusion: A Tribute to Yaakov Bar-Shalom*, 2001.

[161] M.M. Olama, S.M. Djouadi, C.D. Charalambous, and I.G. Papageorgiou, "Position and velocity tracking in mobile networks using particle and Kalman filtering with comparison", *IEEE Tran. on Vehicle Tech.*, Vol. 57, No. 2, pp. 1001–101, 2008.

[162] C. Chong, F. Zhao, S. Mori and S. Kumar, "Distributed tracking in wire-less ad hoc sensor networks", *Int. Conf. Info. Fusion*, pp. 431-438, 2003.

[163] C.H. Papadimitriou and K. Steiglitz, "Combinatorial Optimization Algorithms and Complexity" Mineola, NY: Dover Publication, 1998.

[164] Y. Ruan and P. Willett, "A quantization architecture for track fusion," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 2, pp.671-681, 2005.

[165] X. Rong Li and V.P. Jilkov, "Survey of maneuvering target tracking. Part V. Multiple-model methods," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1255–1321, 2005.

[166] T.H. Chung, J.W. Burdick, and R.M. Murray, "Decentralized motion control of mobile sensing agents in a network," *IEEE Conf. on Decision and Control*, 2006.

[167] Y. Boers, J.N. Driessen, "Interacting multiple model particle filter," IEEE Radar, Sonar and Navigation, vol. 150, no. 5, pp. 344-349, 2003.

[168] S. Kar, J.M.F. Moura, "Consensus + innovations distributed inference over networks: cooperation and sensing in networked systems," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 99-109, 2013.

[169] U.A. Khan and A. Jadbabaie, "Networked estimation under information constraints," *arXiv preprint arXiv:1111.4580*, 2011.

[170] V. Vapnik and S. Mukherjee, "Suppoert vector method for multivariate density estimation", Advances in Neural information Processing systems, Vol. 12, MIT press, 2000.

[171] C. Jauffret, and Y. Bar-Shalom, "Track Formation With Bearings and Frequency Measurements in Clutter," *IEEE Trans. on Aero. & Elec. Sys.*, vol. 26, no. 6, pp. 999–1010, 1990.

[172] T. Kirubarajan, and Y. Bar-Shalom, “Low Observable Target Motion Analysis Using Amplitude Information,” *IEEE Trans. on Aero. & Elec. Sys.*, vol. 32, no. 4, pp. 1367–1384, 1996.

[173] R. Nui, P.K. Willett, and Y. Bar-Shalom, “Matrix CRLB Scaling Due to Measurements of Uncertain Origin,” in *IEEE Trans. on Sig. Proc.*, Vol. 49, No. 7, pp. 1325–1335, 2001.

[174] Z. Minghui, S. Martinez, “On Distributed Convex Optimization Under Inequality and Equality Constraints,” *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151-164, 2012.

[175] D.A. Forsyth and J. Ponce. “Computer Vision : A Modern Approach”, *Prentice Hall*, 2003.

[176] A. T. Alonani. “Nonlinear data fusion,” *Proc. Conference on Decision and Control*, pp. 569–572, 1989.