UNIVERSITY OF CALIFORNIA
RIVERSIDE

Information Weighted Consensus for Distributed Estimation in Vision Networks

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Ahmed Tashrif Kamal

August 2013

Dissertation Committee:

Dr. Amit K. Roy-Chowdhury, Chairperson
Dr. Jay A. Farrell
Dr. Ertem Tuncel

The Dissertation of Ahmed Tashrif Kamal is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

In the past few years of my academic endeavor, I had an amazing journey through the vast ocean of knowledge with some of the most amazing people of my time. I would like to start by expressing my deepest gratitude to my advisor Dr. Amit Roy-Chowdhury and co-advisor Dr. Jay Farrell. I was introduced to my research topic by Dr. Roy-Chowdhury. He was there for me in every ups and downs of my academic endeavor throughout my graduate life. I was able to achieve my academic goals due to all the support and mentoring I received from both my advisors.

I am also grateful to my committee member Dr. Ertem Tuncel. He helped me out with different conceptual issues that arose throughout different graduate courses and my PhD research. I learnt most of the background materials I needed for my research from the courses taught by committee members Dr. Roy-Chowdhury, Dr. Farrell and Dr. Tuncel who are also amazing teachers.

I would also like to express my gratitude to my colleagues Dr. Bi Song and Chong Ding. I did learn a lot by collaborating with them at the foundation stage of my PhD research. Due to all the brain-storming with them and my supervisors, I was able to grasp a good understanding of the challenges in the problem and was able to successfully formulate it. I would also like to thank my colleague Anirban Chakraborty for his help in various experiments and brainstorming sessions.

I will be ever grateful to all my teachers throughout my earlier academic life. Especially, I would like to thank Mr. Mahmud for inspiring me in science and Mr. Nimai Das for inspiring me in mathematics. I would also like to thank Mr. Delowar Hossain who was an amazing teacher to me.

Words cannot express the level of my gratitude to my mother Tahmina Begum

and my father Dr. Kamaluddin Ahmed for all their support, guidance, teaching and unconditional love. I am also ever grateful to my wife Mehnaz Mahbub for all her love and support that made the tough barriers easy to pass throughout my graduate years. I would also like to thank my sister Kashfia Mehrin for her love and support.

I would also like to thank the Office of Naval Research for their grant (N000140910666) supporting my research.

To my parents and my wife for all the support.

ABSTRACT OF THE DISSERTATION

Information Weighted Consensus for Distributed Estimation in Vision Networks

by

Ahmed Tashrif Kamal

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, August 2013
Dr. Amit K. Roy-Chowdhury, Chairperson

Due to their high fault-tolerance, ease of installation and scalability to large networks, distributed algorithms have recently gained immense popularity in the sensor networks community, especially in computer vision. Multi-target tracking in a camera network is one of the fundamental problems in this domain. Distributed estimation algorithms work by exchanging information between sensors that are communication neighbors. Since most cameras are directional sensors, it is often the case that neighboring sensors may not be sensing the same target. Such sensors that do not have information about a target are termed as "naive" with respect to that target. State-of-the-art distributed state estimation algorithms (e.g., the Kalman Consensus Filter (KCF)) in the sensor networks community are not directly applicable to tracking applications in camera networks mainly due to this naivety issue. In our work, we propose generalized distributed algorithms for state estimation in a sensor network taking the naivety issue into account.

For multi-target tracking, along with the tracking framework, a data association step is necessary where the measurements in each camera's view are associated with the appropriate targets' tracks. At first, under the assumption that the data association is given, we develop distributed state estimation algorithms addressing the naivety is-

sue. In this process, first, we propose the Generalized Kalman Consensus Filter (GKCF) where an information-weighting scheme is utilized to account for the naivety issue. Next, we propose the Information-weighted Consensus Filter (ICF) which can achieve optimal centralized performance while also accounting for naivety. This is the core contribution of this thesis. Next, we introduce the aspect of multi-target tracking where a probabilistic data association scheme is incorporated in the distributed tracking scheme resulting the Multi-Target Information Consensus (MTIC) algorithm. The incorporation of the probabilistic data association mechanism makes the MTIC algorithm very robust to false measurements/clutter.

The aforementioned algorithms are derived under the assumption that the measurements are related to the state variables using a linear relationship. However, in general, this is not true for many sensors including cameras. Thus, to account for the non-linearity in the observation model, we propose non-linear extensions of the previous algorithms which we denote as the Extended ICF (EICF) and the Extended MTIC (EMTIC) algorithms. In-depth theoretical and experimental analysis are provided to compare these algorithms with existing ones.

Contents

List of Figures

List of Algorithms

Important Notations

Notation	Dimension	Description
i, i'	scalar	sensor index
j	scalar	target index
N_C	scalar	total number of sensors
N_T	scalar	total number of targets
$\mathcal{N}_i$	set	set of neighbors of node i
t	scalar	time step
k	scalar	consensus iteration no.
p	scalar	state vector length
m_i	scalar	C_i's measurement vector length
m	scalar	$\sum_{i=1}^{N_C} m_i$
$\mathbf{x}$	p	target's true state
$\hat{\mathbf{x}}_c$, $\hat{\mathbf{x}}_i$	p	centralized and C_i's estimate
$\mathbf{J}_c$, $\mathbf{J}_i$	$p \times p$	information matrix of $\mathbf{x}_c$ & $\mathbf{x}_i$
$\widehat{\boldsymbol{\mathcal{X}}}^-$	$N_C p$	stack of estimates from all nodes
$\boldsymbol{\mathcal{P}}$	$N_C p \times N_C p$	error covariance of $\widehat{\boldsymbol{\mathcal{X}}}^-$
$\boldsymbol{\mathcal{F}}$	$N_C p \times N_C p$	information matrix of $\widehat{\boldsymbol{\mathcal{X}}}^-$
$\mathbf{F}_{ii'}$	$p \times p$	$\{i, i'\}^{th}$ block element of $\boldsymbol{\mathcal{F}}$
$\mathbf{F}_i^-$	$p \times p$	$\sum_{i'=1}^{N_C} \mathbf{F}_{i'i}$
$\boldsymbol{\mathcal{H}}_I$	$N_C p \times p$	$[\mathbf{I}_p, \mathbf{I}_p, \ldots \mathbf{I}_p]^T$
$(\ )^-$, $(\ )^+$		prior and posterior information
$\boldsymbol{\Phi}$	$p \times p$	state transition matrix
$\mathbf{Q}$	$p \times p$	process noise covariance
$\mathbf{z}_i$	m_i	C_i's measurement
$\mathbf{R}_i$	$m_i \times m_i$	covariance matrix of $\mathbf{z}_i$
$\mathbf{H}_i$	$m_i \times p$	observation matrix at C_i
$\mathbf{h}_i$	function	observation function of C_i
$\boldsymbol{\mathcal{Z}}$	m	stack of all measurements
$\boldsymbol{\mathcal{R}}$	$m \times m$	covariance matrix of $\boldsymbol{\mathcal{Z}}$
$\boldsymbol{\mathcal{H}}$	$m \times p$	stack of all $\mathbf{H}_i$
ϵ	scalar	consensus speed parameter
Δ_i	scalar	degree of node connectivity
$\mathbf{v}_i[k]$	p	consensus weighted information
$\mathbf{V}_i[k], \mathbf{W}_i[k]$	$p \times p$	consensus information matrix
β^{jn}	scalar	probability that $\mathbf{z}^n$ is T^j's obs.
β^{j0}	scalar	probability that T^j was not observed
λ_f	scalar	clutter density

Chapter 1

Introduction

Due to the availability of modern low-cost sensors, large-scale camera networks are being used in applications such as wide-area surveillance, disaster response, environmental monitoring, etc. Multiple sensors can cover more area, provide views from different angles and the fusion of all their measurements may lead to robust scene understanding. Among different information fusion approaches, distributed schemes are often chosen over centralized or hierarchical approaches due to their scalability to a large number of sensors, ease of installation and high tolerance to node failure. In this article, we focus on the problem of *distributed multi-target tracking in a camera network*. However, the proposed methods are much generalized distributed state estimation schemes and their applications are not limited only to camera networks. We use the term *distributed* to mean that each camera processes its own data and arrives at a final solution through negotiations with its neighbors; there is no central processor. Note that term distributed has been also used in computer vision to refer to a camera network that is distributed over a wide area but where the processing is centralized. To motivate the core contribution of this work, we first describe the inter-relationship between distributed estimation

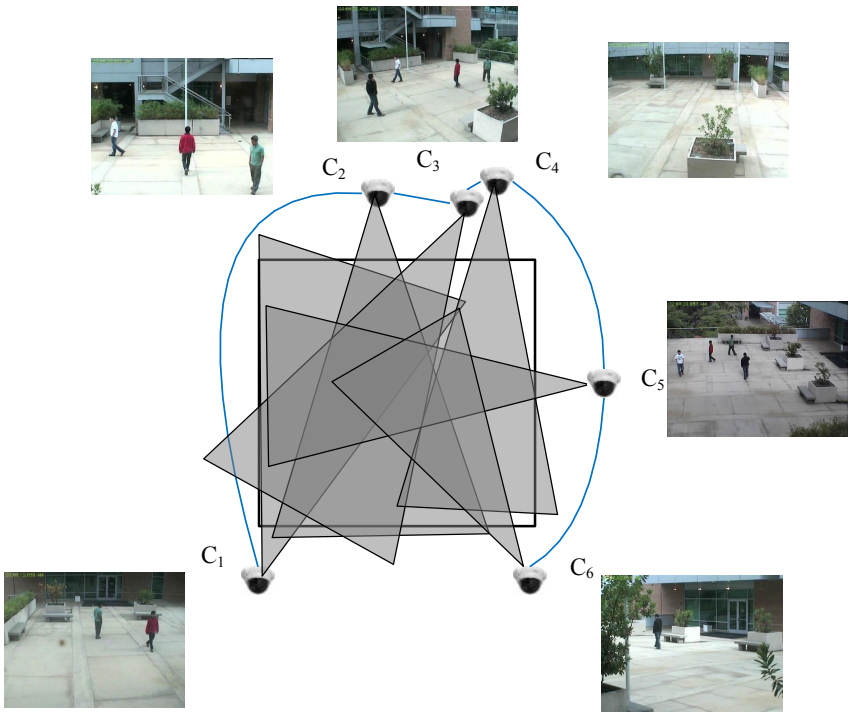


Figure 1.1: In this figure, there are six sensing nodes, $C_1, C_2, \ldots, C_6$ observing an area (black rectangle) consisting of four targets. The solid blue lines show the communication channels between different nodes. This figure also depicts the presence of "naive" nodes. For example, C_3, C_5, C_6 get direct measurements about the black target which it shares with its immediate network neighbors. However, C_1 does not have *direct* access to measurements of that target and thus is naive w.r.t. that target's state.

and camera networks.

Most of the work in distributed tracking has been in the multi-agent systems community [18]. The methods there assume that each target can be viewed by each sensor which may not be true for many application scenarios, especially for a camera network (see Fig. 1.1) where each camera can view only a limited portion of the entire area. This limits the observability of each sensor to a subset of all the targets. In this work, our goal is to design distributed tracking schemes which are suited for such sensors

with limited field-of-view (FOV).

A distributed multi-target tracking problem can be divided into three sub-problems, namely, distributed information fusion, data association (measurement to track association) and dynamic state estimation. Among many types of distributed information fusion approaches, *consensus algorithms* [18] are schemes where each node, corrects its own state using information only from its network neighbors. By iteratively doing so, each node can individually compute a global function of the information from all the nodes (e.g. average). The important fact is that consensus is reached without all-to-all communication; thus consensus based frameworks do not require any specific communication network topology and are generally applicable to any arbitrary, connected network. The consensus estimates asymptotically converge to the global result. However, in a finite time window, only a limited number of iterations can be performed due to limited bandwidth. Due to the simplicity and robustness of consensus algorithms, they have been used in many applications, including estimation problems in sensor networks (e.g., [28, 30, 31]).

In a distributed multi-target tracking scheme, each node may need to maintain a state estimate of each target even though it is not directly observing the target, since the nodes will need to collaborate with each other. Each node gets measurements of the targets and must associate the measurements to the appropriate target's track. In a consensus-based scheme, each node maintains its own copy of the state estimates of all the targets which makes consensus-based approaches inherently appropriate for our problem.

When applying these approaches to camera networks, we need to be aware of one particular issue with vision sensors. As exemplified in Fig. 1.1, many of the cameras may not see a target and it is very much possible that neighboring cameras on

the communication graph do not see the same target. We call a node '*naive*' about a target when there are no measurements of that target available in its *local neighborhood* (consisting of the node and its immediate network neighbors). In such a situation, in a consensus-based framework, due to limited local observability and limited number of consensus iterations, the naive node has access to less information about the target's state. Naivety relative to the j-th target is likely to be present in the network when nodes on the vision graph for target j are sparsely connected in the communication graph. The vision graph for target T^j is a graph where there is an edge between each pair of nodes that are observing T^j. The vision graph is usually time varying, different from the communication graph, and different for different targets.

A well-known consensus-based scheme for distributed state estimation is the Kalman Consensus Filter (KCF) [17]. The KCF algorithm was originally designed for the scenario where each node has an observation of the target. The quality of neighboring node's prior information was not taken into account in KCF. Thus, naive nodes may adversely affect the overall performance of the network. Moreover, the cross-covariance terms between the state estimates at different nodes were not incorporated in the estimation process in KCF as they are usually hard to compute in a distributed environment. Due to these reasons, the performance of KCF often suffers when applied to a camera network. Distributed algorithms that are derived under the assumption that each sensor has full state observability at each time (e.g. KCF), may be adapted to be used in the presence of naive nodes when communication protocols (e.g. [9]) for translating a known vision graph into a set of routing paths that connect nodes in the communication graph are available. However, scalable distributed vision graph discovery algorithms do not exist. Moreover, distributed algorithms that require the knowledge of the vision graphs usually require special target handoff protocols. Such issues are important for networks

with sparse communication topology. For example, camera networks are often spread over a wide area which prevents each camera from communicating directly with all other cameras. There can be many naive nodes in sparse communication topologies.

1.1 Contribution

The presence of these naive nodes motivates us to propose various novel distributed state estimation frameworks which can be utilized in many applications in sensor networks including target tracking in camera networks.

First, we propose the Generalized Kalman Consensus Filter (GKCF) [11] where each node's confidence in its state estimate is also incorporated in the distributed estimation framework. In the presence of naive nodes, the proposed GKCF algorithm outperforms the standard KCF.

The results of the KCF and the GKCF algorithms are sub-optimal i.e., they do not converge to the optimal centralized Kalman filter [10] results. Additionally, computation and communication resource constraints are important issues in a distributed estimation framework because in many application scenarios, the nodes are low powered wireless devices. Therefore, a distributed state estimation framework that can guarantee convergence to the optimal centralized estimate while maintaining low computation and communication resource requirements in the presence of naive nodes is desirable. To achieve this, next we propose the distributed state estimation framework called the Information-weighted Consensus Filter (ICF) [12, 13]. The ICF is guaranteed to converge to the optimal centralized performance under certain reasonable conditions.

The issue of naivety is handled in the ICF algorithm by proper information weighting in the estimation framework. Optimality is achieved by proper relative weight-

ing between the prior and the measurement information. Both the GKCF and ICF algorithms support multiple consensus iterations at each time step t to improve performance. In addition, the experimental results show that ICF outperforms other distributed estimation algorithms such as KCF and GKCF at any given computation and communication resource limit. The ICF algorithm (and its derivatives that we propose later) does not require handoff protocols or the knowledge of the vision graph.

The above mentioned methods assume that there is a single target, or for multiple targets, the measurement-to-track association is provided. For a multi-target tracking problem, the data association and the tracking steps are highly inter-dependent. The performance of tracking will affect the performance of data association and vice-versa. Thus, an integrated distributed tracking and data association solution is required where the uncertainty from the tracker can be incorporated in the data association process and vice-versa. Among many single-sensor multi-target data association frameworks, the Multiple Hypothesis Tracking (MHT) [23] and the Joint Probabilistic Data Association Filter JPDAF [2] are two popular schemes. MHT usually achieves higher accuracy at the cost of high computational load. On the other hand, JPDAF achieves reasonable results at much lower computation cost. As distributed solutions are usually applied to resource constrained systems, the JPDAF scheme will be utilized in the distributed multi-target tracking framework we propose in our work.

With the tight integration between the JPDAF algorithm and our previously derived ICF algorithm, we next propose the Multi-Target Information Consensus (MTIC) [14] algorithm. The estimation errors in tracking and data association, as well as the effect of naivety, are jointly addressed in the development of the MTIC algorithm. The incorporation of the probabilistic data association mechanism makes the MTIC algorithm very robust to false measurements/clutter.

The aforementioned methods are derived assuming a linear observation model. However, camera observation models are non-linear. Thus to apply these distributed tracking algorithms in a realistic camera network, we also need to extend these algorithms to handle non-linearity in the observation model. To accomplish this, we present the Extended Information Consensus Filter (EICF) and the Extended Multi-target Information Consensus (EMTIC) algorithms.

1.2 Related Work

The purely decentralized nature of the fusion algorithm differentiates it from the majority of multi-camera tracking approaches in the computer vision literature. For example, in [6], a centralized approach for tracking in a multi-camera setup was proposed where the cameras were distributed spatially over a large area. In [4], an efficiently target hand-off scheme was proposed but no multi-camera information fusion was involved. However, in this thesis, we deal with the distributed multi-target tracking problem where there is no centralized server, the processing is distributed over all the camera nodes and no target hand-off strategy is required.

Various methods for distributed multi-target tracking have been proposed in the sensor-networks literature. In [5], a solution to the distributed data association problem was proposed by means of the message passing algorithm based on graphical models in which iterative, parallel exchange of information among the nodes viewing the same target was required. However, in our proposed framework, no special communication pattern is assumed. In [22], the authors proposed a framework for querying a distributed database of video surveillance data in order to retrieve a set of likely paths of a person moving in the area under surveillance using a dynamic Bayesian Model.

However, unlike our proposed methods, this method does not deal with the distributed fusion of the information in the network.

In [21, 26, 28], the distributed multi-target tracking schemes did not account for naivety or the presence of cross-correlation between the estimates at different nodes. We propose the ICF to deal with both these issues. The MTIC algorithm, is an extension of the ICF to deal with data association between multiple targets. The EICF and EMTIC provide the ability to work with non-linear camera models. Along with proposing these novel distributed algorithms, we also provide in-depth proof and analysis of all these algorithms.

1.3 Organization

We organize this article as the following. In Chapter 2, we review the average consensus and the KCF algorithm and propose the GKCF algorithm. In Chapter 3, we propose the ICF algorithm. In Chapter 4, we propose the MTIC algorithm. Finally, in Chapter 5, we propose the non-linear extensions to the previous algorithms resulting the EICF and the EMTIC algorithms. Thorough theoretical and experimental comparisons are provided in each chapter for each method.

Chapter 2

Kalman Consensus Filter and its Extension

2.1 Introduction

In this chapter, we propose a consensus-based framework that is capable of tracking targets throughout the network in a distributed manner. The problem of tracking targets in a distributed sensor network has been studied previously. The Kalman Consensus Filter (KCF) [17] is a state-of-the-art distributed algorithm for fusing multiple measurements from different sensors. The KCF is a very appropriate framework for camera networks and has been applied in [27, 28]. However, certain issues that are specific to video sensors have not been considered in the existing solutions.

A camera is a unidirectional sensor with a limited sensing region which is called the field-of-view (FOV). Thus, in a realistic camera network, a target would usually be seen in only a few of the nodes Fig. 2.1. In a distributed decision making process, the nodes are assumed to have peer-to-peer communication channels. Thus, when a sensor gets new measurements for a target, say T^j, it shares this measurement information with

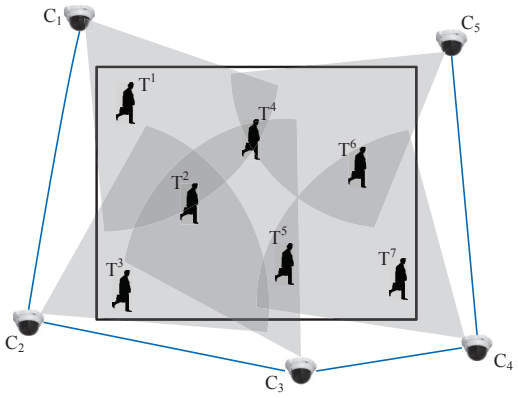


Figure 2.1: In this figure, there are five sensing nodes, $C_1, C_2, \ldots, C_5$ observing an area (black rectangle) consisting of seven targets $T^1, T^2, \ldots T^7$. The solid blue lines show the communication channels between different nodes. This figure also depicts the presence of "naive" nodes. For example, C_1 gets direct measurements about T^1 which it shares with its immediate network neighbor, C_2. However, the rest of the cameras, i.e., C_3, C_4, C_5 do not have *direct* access to measurements of T^1 and thus are naive w.r.t. T^1's state.

its network neighbors. This measurement information is used to update the estimate of T^j's state and error covariance at each node that directly observes T^j or receives measurement of T^j from their neighbor(s). At the same time, the nodes also share their previous state estimate with each other and try to compensate the difference between their state estimates of T^j using a consensus scheme. Thus, at some nodes in the network that are neither sensing T^j directly nor are neighbor to a node sensing T^j (termed as *naive nodes* for T^j), the state estimate for this target is only adjusted by the consensus scheme and its error covariance is not adjusted; therefore, the error covariance matrices of each target may diverge. Even if the consensus term maintains consistency of the state estimates, the different covariance matrices at each node can have a profound effect on the convergence transients, as each agent uses its local covariance matrix in the computation for incorporating measurement information.

Such issues are important for networks with sparse communication topology. For example, camera networks are often spread over a wide area which prevents each camera from communicating directly with all other cameras. There can be many naive nodes in sparse communication topologies. The presence of these naive nodes motivates us to propose certain modifications to the KCF framework for application in camera networks. In such scenarios, the proposed Generalized Kalman Consensus Filter (GKCF) outperforms the standard KCF. Although this article is focused on camera networks (since it is a common scenario where these constraints apply), the GKCF approach is applicable to other sensor networks that have similar characteristics. If the network is fully (or close to fully) connected, then the effect of naive nodes is usually very low and the standard KCF performs well [27, 28].

To allow for a clear discussion of the literature and our motivation, Sec. 2.2 states the problem of interest and related notation. Secs. 2.3 and 2.4 reviews the related literature, presents a technical description of our motivation and overviews the contributions of the chapter. In Sec. 2.5, we derive our Generalized Kalman Consensus Filter algorithm. Finally, Sec. 2.7, shows simulation results comparing the performance of our approach with other methods.

2.2 Problem Formulation

Consider a sensor network with N_C sensors. There are no specific assumptions on the overlap between the FOVs of the sensors. The communication in the network can be represented using an undirected connected graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$. The set $\mathcal{C} = \{C_1, C_2, \ldots, C_{N_C}\}$ contains the vertices of the graph and represents the sensor nodes. The set $\mathcal{E}$ contains the edges of the graph which represents the available communication

channels between different nodes. The set of nodes having direct communication channel with node C_i (sharing an edge with C_i) is represented by $\mathcal{N}_i$.

The target's state is represented by the vector $\mathbf{x} \in \mathcal{R}^p$. For example, for a tracking application in a camera network, $\mathbf{x}$ might be a vector containing ground plane position and velocity components. The state dynamics of the target are modeled as

$$\mathbf{x}(t+1) = \mathbf{\Phi}\mathbf{x}(t) + \boldsymbol{\gamma}(t). \tag{2.1}$$

Here $\mathbf{\Phi} \in \mathcal{R}^{p\times p}$ is the state transition matrix and the process noise $\boldsymbol{\gamma}(t)$ is modeled as $\mathcal{N}(\mathbf{0}, \mathbf{Q})$.

At time t, depending on its FOV and the location of the target, a sensor may get a measurement. Note that in this chapter, our assumption of the knowledge of data association implies that if we have a measurement, it belongs to the target. Let us denote the measurement of the target at C_i as $\mathbf{z}_i$. It is assumed that $\mathbf{z}_i$ was generated by the following observation model

$$\mathbf{z}_i = \mathbf{H}_i\mathbf{x}_i + \boldsymbol{\nu}_i. \tag{2.2}$$

Here, $\mathbf{H}_i \in \mathcal{R}^{m\times p}$ is the observation matrix for node C_i. The noise $\boldsymbol{\nu}_i \in \mathcal{R}^m$ is modeled as a zero mean Gaussian random variable with covariance $\mathbf{R}_i \in \mathcal{R}^{m\times m}$.

Each node also maintains a prior/predicted state estimate $\hat{\mathbf{x}}_i^-(t)$ (and its covariance $\mathbf{P}_i^-(t)$) for each target. Throughout this article, the inverse of the state covariance matrix (information/precision matrix) will be used and denoted as $\mathbf{J}_i = (\mathbf{P}_i)^{-1}$. We assume that the initial prior state estimate and information matrix is available to each node for the target upon its detection. *Our goal is to track the target at each node, i.e., find the state estimate for the target at each node by using the prior and measurement information available in the entire network in a distributed fashion.*

2.3 Average Consensus: Review

Average consensus [18] is a popular distributed algorithm to compute the arithmetic mean of some values $\{a_i\}_{i=1}^{N_C}$. Suppose, each node i has a quantity a_i. We are interested in computing the average value of these quantities i.e. $\frac{1}{N_C}\sum_{i=1}^{N_C} a_i$, in a distributed manner.

In average consensus algorithm, each node initializes its consensus state as $a_i[0] = a_i$ and iteratively communicates with its neighbors and updates its own state information. Throughout this article, the index inside a square bracket will indicate the consensus iteration number as opposed to the index inside a parenthesis that will typically refer to the time index. At the beginning of iteration k, a node C_i sends its previous state $a_i[k-1]$ to its immediate network neighbors $C_{i'} \in \mathcal{N}_i$ and also receives the neighbors' previous states $a_{i'}[k-1]$. Then it updates its own state information using the following equation

$$\begin{aligned} a_i[k] &= a_i[k-1] + \epsilon \sum_{i' \in \mathcal{N}_i} (a_{i'}[k-1] - a_i[k-1]) \\ &= \mathcal{A}(a_i[k-1]) \end{aligned} \tag{2.3}$$

Here $\mathcal{A}(a_i)$ is a shorthand mathematical operator for a single step of average consensus (defined as the above). By iteratively doing so, the values of the states at all the nodes converge to the average of the initial values. The rate parameter ϵ should be chosen between 0 and $\frac{1}{\Delta_{max}}$, where Δ_{max} is the maximum degree of the network graph $\mathcal{G}$. Choosing larger values of ϵ will result in faster convergence, but choosing values equal or more than Δ_{max} will render the algorithm unstable. The average consensus algorithm can be used to compute the average of vectors and matrices by applying it to their individual elements separately. Average consensus assumes all agents have an estimate for all elements of a.

Consensus algorithms have been extended to perform various tasks in a network of agents such as linear algebraic operations like SVD, least squares, PCA, GPCA [31]. These distributed estimation frameworks have been applied in various fields including camera networks for distributed implementations of 3-D point triangulation, pose estimation [30], and action recognition [28]. The average consensus algorithm is applicable only for a static parameter estimation problem. For a dynamic state estimation problem, a predictor-corrector solution approach is needed.

2.4 Kalman Consensus Filter: Review

Now, we present the KCF algorithm [17] in Algorithm 1 and explain the motivation for proposing the GKCF. It should be noted that the KCF algorithm works under the assumption that all the sensors have sensed all the targets. The issue of limited sensing range in the distributed estimation process has been considered previously. In [20], the authors considered the case where not all sensors get measurements of the target. However, the solution was not fully distributed; rather it was a hybrid solution consisting of a distributed and a centralized scheme for information fusion. The nodes used the KCF algorithm to update their state estimates. These state estimates were sent along with the state covariance information to a *fusion center*. For larger networks a hierarchical tree structure of fusion centers was proposed, where the information of all the nodes reached a root fusion center. Throughout this article, we are interested in solving the problem using a completely distributed architecture. We will now discuss various specific conditions that require attention when the KCF is applied to sparse networks with naive nodes, and next we will propose solution strategies for each of them.

Algorithm 1 KCF at C_i at time step t

Given $\mathbf{J}_i^-(t)$, $\hat{\mathbf{x}}_i^-(t)$ and ϵ

1) Get measurement $\mathbf{z}_i$.

2) Compute information vector and matrix

$$\mathbf{u}_i \leftarrow \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{z}_i \tag{2.4}$$

$$\mathbf{U}_i \leftarrow \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{H}_i \tag{2.5}$$

3) Broadcast $\mathbf{u}_i$, $\mathbf{U}_i$, $\hat{\mathbf{x}}_i^-$ to neighbors and receive $\mathbf{u}_{i'}$, $\mathbf{U}_{i'}$, $\hat{\mathbf{x}}_{i'}^-$ from neighbors.

4) Fuse the information vectors and matrices

$$\mathbf{b}_i \leftarrow \sum_{i' \in \mathcal{N}_i \cup \{i\}} \mathbf{u}_{i'} \tag{2.6}$$

$$\mathbf{B}_i \leftarrow \sum_{i' \in \mathcal{N}_i \cup \{i\}} \mathbf{U}_{i'} \tag{2.7}$$

5) Compute Kalman Consensus estimate

$$\mathbf{M}_i \leftarrow \left(\mathbf{J}_i^- + \mathbf{B}_i\right)^{-1} \tag{2.8}$$

$$\hat{\mathbf{x}}_i^+ \leftarrow \hat{\mathbf{x}}_i^- + \mathbf{M}_i \left(\mathbf{b}_i - \mathbf{B}_i \hat{\mathbf{x}}_i^-\right) + \gamma (\mathbf{J}_i^-)^{-1} \sum_{i' \in \mathcal{N}_i} \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right) \tag{2.9}$$

$$\gamma = \epsilon / (1 + ||(\mathbf{J}_i^-)^{-1}||), \quad ||X|| = tr(X^T X)^{\frac{1}{2}} \tag{2.10}$$

6) Predict for next time step

$$\mathbf{J}_i^-(t+1) \leftarrow \left(\Phi \mathbf{M}_i \Phi^T + \mathbf{Q}\right)^{-1} \tag{2.11}$$

$$\hat{\mathbf{x}}_i^-(t+1) \leftarrow \Phi \hat{\mathbf{x}}_i^+(t) \tag{2.12}$$

1) Average vs. weighted average: The basic KCF algorithm uses *average* consensus to combine state estimates from neighboring nodes (see Eqn. (2.9)). With average consensus, the state estimates of all the nodes get the same weight in the summation. Since naive nodes do not have observations of the target, their estimates are often highly erroneous. This results in reduced performance in the presence of naive nodes.

2) Covariance/Information Matrix Propagation: The information matrix measurement update of Eqn. (2.8) considers the node's own information matrix and the local neighborhood's measurement covariance. It does not account for cross covariance between the estimates by the node and its neighbors. In the theoretical proof of optimality for KCF, the cross covariances terms between neighbors' state estimates were present [17]. It has been stated in [17] that dropping these cross covariance terms is a valid approximation when the state estimate error covariance matrices are almost equal in all the nodes.

However, when C_i is naive w.r.t. the target, $\mathbf{b}_i$ and $\mathbf{B}_i$ are both zero. Therefore, $\mathbf{M}_i = (\mathbf{J}_i^-)^{-1}$ at Eqn. (2.8). Consequently, from Eqn. (2.11) it can be seen that the diagonal elements of $\mathbf{J}_i^-$ tend to zero at each time update as long as C_i remains naive with respect to the target. This makes the covariance matrix diverge. From this, it can be clearly seen that omitting the cross covariances in the covariance update equation is not valid for sparse networks with naive agents. The correlation between the two dependent variables is the unknown parameter making this computation difficult. There has been some work, e.g. [24] and [1], where the authors incorporated cross covariance information, which should lead to the optimum result. But, no method for computing these terms were provided and predefined fixed values were used instead.

3) Over-correction of the states: The measurement update term and consensus term in Eqn. (2.9) are both functions of the prior state estimate $\hat{\mathbf{x}}_i^-$. Both terms apply

corrections to the prior state estimate, from different information sources. Thus the state estimate might get over-corrected. This is usually not a big issue in sensor networks without naive nodes because every node's state estimate will be close to the consensus. In sparse networks, the estimates of naive nodes may lag behind by a significant time. This happens because naive nodes do not have direct access to new observation of a target, the only way they can get updated information about a target is through a neighbor's state estimate which was updated in the previous iteration. Thus a naive node might be multiple iterations away from getting new information about a target. This information imbalance can cause large oscillations. In the KCF algorithms this effect can be decreased by choosing a smaller rate parameter ϵ. However, decreasing ϵ yields slower convergence of the naive node's state estimate.

The above issues can be problematic for tracking applications involving a camera network with naive nodes. A naive node may associate an observation to a wrong target. This can affect the tracking performance of nodes that are actually observing the target by influencing them to drift away from their estimates. Since KCF is a very appropriate framework to build a distributed tracker in a camera network, we propose some changes to address the above challenges leading to a Generalized Kalman Consensus Filter. The following are the main proposed modifications.

1) The consensus portion of the GKCF correction step at each node will take into account the state covariances of neighbors. The nodes will then converge towards the weighted mean, instead the unweighted mean.

2) Each node and its neighbors' state covariance matrices will be used jointly at consensus step to update that node's error covariance matrix. This will prevent the state covariance of the naive nodes from monotonically increasing.

3) Weighted average consensus will correct the prior estimate towards the weighted

mean. Then the measurement information in the local neighborhood will be utilized to update this consensus state and covariance, thus preventing the overcorrection issue mentioned above.

2.5 Generalized Kalman Consensus Filter

Now, we will present the GKCF algorithm which is summarized in Algorithm 2. To derive the GKCF algorithm, we first introduce the weighted average consensus. Next, we show how to incorporate this consensus scheme into our framework.

2.5.1 Weighted Average Consensus

Let the initial state estimate of all N_C agents be $\hat{\mathbf{x}}_i^-$ with information matrix $\mathbf{J}_i^-$. As we use this information matrix term as weights in the weighted average consensus algorithm, the terms *weight* and *information matrix* will be used interchangeably. So, the centralized weighted average of the initial states is

$$\bar{\mathbf{x}}_c = \left(\sum_{i=1}^{N_C} \mathbf{J}_i^-\right)^{-1} \sum_{i=1}^{N_C} \mathbf{J}_i^- \hat{\mathbf{x}}_i^-. \tag{2.13}$$

Define the initial weighted state and weight of each agent as

$$\mathbf{v}_i[0] = \mathbf{J}_i^- \hat{\mathbf{x}}_i^-, \tag{2.14}$$

$$\mathbf{V}_i[0] = \mathbf{J}_i^-. \tag{2.15}$$

Weighted average consensus [18] states that if the iterative update in Eqns. (2.16) and (2.17)

$$\mathbf{v}_i[k] = \mathbf{v}_i[k-1] + \epsilon \sum_{i' \in \mathcal{N}_i} (\mathbf{v}_{i'}[k-1] - \mathbf{v}_i[k-1]) \tag{2.16}$$

$$\mathbf{V}_i[k] = \mathbf{V}_i[k-1] + \epsilon \sum_{i' \in \mathcal{N}_i} (\mathbf{V}_{i'}[k-1] - \mathbf{V}_i[k-1]) \tag{2.17}$$

is performed for all $i = 1, \ldots, N_C$, then each of the terms $\mathbf{V}_i[K]^{-1}\mathbf{v}_i[K]$ tends to the weighted average $\bar{\mathbf{x}}_c$ as $K \to \infty$. As a by-product, the weights also converge to the average of the initial weights i.e., $\mathbf{V}_i[K] \to \frac{1}{N_C}\sum_{i=1}^{N_C}\mathbf{J}_i^-$. Both these properties of the weighted average consensus will be utilized in our approach.

2.5.2 Covariance/Information Matrix Propagation

After communicating with its neighbors and prior to using measurement information, the optimal (in the local neighborhood) state estimate at C_i is a linear combination of the information from C_i and its neighbors. Since these variables are not independent, optimal estimation would require knowledge of the cross correlation structure between each pair of these random variables. Since, it is usually quite difficult to compute this cross correlation, we need some other way to approximate the covariance or in this case the information matrix. The update operation of the information matrix $\mathbf{V}_i[k]$ in Eqn. (2.17) can be used as an approximation of the information matrix due to the incoming information from the neighbors' states. A property of the weighted average consensus is that the weights also converge to the average of the weights as the state estimates converge towards the weighted average. Thus, this kind of covariance/weight propagation enables the weights to be updated accordingly when informative state estimates arrive at a naive node.

After computing the state and weight estimates with all the available information, we need to propagate the weight and state in time. One should note that instead of propagating the state estimate, we have to propagate the weighted state estimate as necessitated by the weighted average consensus equations. Thus the weight propagation equation takes the form of Eqn. (2.28).

Algorithm 2 GKCF on sensor C_i a time-step t

Given $\mathbf{J}_i^-$, $\hat{\mathbf{x}}_i^-$, ϵ and K. Also let,

1) Get measurements $\mathbf{z}_i$

2) Compute information vector and matrix

$$\mathbf{u}_i \leftarrow \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{z}_i \tag{2.18}$$

$$\mathbf{U}_i \leftarrow \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{H}_i \tag{2.19}$$

3) Broadcast $\mathbf{u}_i, \mathbf{U}_i$ to neighbors and receive $\mathbf{u}_{i'}, \mathbf{U}_{i'}$ from neighbors

4) Fuse the information matrices and vectors in the local neighborhood

$$\mathbf{b}_i \leftarrow \sum_{i' \in \mathcal{N}_i \cup \{i\}} \mathbf{u}_{i'} \tag{2.20}$$

$$\mathbf{B}_i \leftarrow \sum_{i' \in \mathcal{N}_i \cup \{i\}} \mathbf{U}_{i'} \tag{2.21}$$

5) Initialize consensus variables

$$\mathbf{v}_i[0] \leftarrow \mathbf{J}_i^- \hat{\mathbf{x}}_i^- \tag{2.22}$$

$$\mathbf{V}_i[0] \leftarrow \mathbf{J}_i^- \tag{2.23}$$

6) For K iterations run average consensus on $\mathbf{v}_i[k]$ and $\mathbf{V}_i[k]$ and then compute updated estimate

$$\bar{\mathbf{x}}_i^- \leftarrow \mathbf{V}_i[K]^{-1} \mathbf{v}_i[K] \tag{2.24}$$

$$\bar{\mathbf{J}}_i^- \leftarrow \mathbf{V}_i[K] \tag{2.25}$$

7) Compute GKCF estimate

$$\mathbf{J}_i^+ \leftarrow \bar{\mathbf{J}}_i^- + \mathbf{B}_i \tag{2.26}$$

$$\hat{\mathbf{x}}_i^+ \leftarrow \bar{\mathbf{x}}_i^- + (\mathbf{J}_i^+)^{-1} \left(\mathbf{b}_i - \mathbf{B}_i \bar{\mathbf{x}}_i^- \right) \tag{2.27}$$

8) Propagate for next time step

$$\mathbf{J}_i^-(t+1) \leftarrow \left(\Phi \mathbf{J}_i^+(t)^{-1} \Phi^T + \mathbf{Q} \right)^{-1} \tag{2.28}$$

$$\hat{\mathbf{x}}_i^-(t+1) \leftarrow \Phi \hat{\mathbf{x}}_i^+(t) \tag{2.29}$$

2.5.3 Two-stage Update

To resolve the issue of overcorrection of the states, we divide the estimation process in two stages. First, as mentioned above, C_i updates its state and information matrix using its neighbors' states and information matrices. Next, we further update our state and information matrix with current measurement information, which we explain below.

Consider that a node that has completed Step 2 in Algorithm 2. If it did not have any observation, then $\mathbf{z}_i$ and $\mathbf{R}_i^{-1}$ were set to zero. Using the fused information vector and matrix and the updated prior weight and state estimate (from the weighted average consensus step of Eqns. (2.24) and (2.25)), we get the final state and weight estimate at time t. Thus, using Eqns. (2.26) and (2.27) we can estimate the state and weight which includes the properly weighted innovations from the measurements and the state estimates of the neighbors. Note that GKCF is a more general algorithm that at the expense of additional communications can converge even closer to the global weighted average (by virtue of the weighted average consensus steps).

2.6 Theoretical Comparison

The KCF algorithm in [17] was originally presented using a single consensus step. This section presents the state estimation step of the GKCF and KCF algorithm in an equivalent form for a single consensus step (i.e., $K = 1$) and compares the differences between the algorithms theoretically. The derivation of the following results are presented in the appendix.

GKCF (see Proposition 5 in Appendix)

$$\hat{\mathbf{x}}_i^+ = \hat{\mathbf{x}}_i^- + \left(\mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i\right)^{-1} \left(\mathbf{b}_i - \mathbf{B}_i \hat{\mathbf{x}}_i^- + \epsilon \sum_{i' \in \mathcal{N}_i} \mathbf{J}_{i'}^- \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^- \right) \right) \tag{2.30}$$

$$\mathbf{J}_i^+ = \mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i \tag{2.31}$$

KCF

$$\begin{aligned} \hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathbf{J}_i^- + \mathbf{B}_i\right)^{-1} \left(\mathbf{b}_i - \mathbf{B}_i \hat{\mathbf{x}}_i^-\right) \\ &\quad + \frac{\epsilon}{1 + ||(\mathbf{J}_i^-)^{-1}||} (\mathbf{J}_i^-)^{-1} \sum_{i' \in \mathcal{N}_i} \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^- \right) \end{aligned} \tag{2.32}$$

$$\mathbf{J}_i^+ = \mathbf{J}_i^- + \mathbf{B}_i \tag{2.33}$$

The following points of comparison are important:

1) In the third term of Eqn. (2.32), KCF gives equal weight to all the neighbors' priors. In the presence naivety, this has detrimental effect as the information at different nodes are different and need to be weighted by their information matrices. This is properly accounted for in the GKCF algorithm as the innovation from each neighbor's prior state $\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^- \right)$ is properly weighted by their information matrix $\mathbf{J}_{i'}^-$ in Eqn. (2.30).

2) In Eqn. (2.30), GKCF uses $\left(\mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i\right)^{-1}$ to normalize both the innovation from the measurements and the innovation from the state priors. Whereas, in Eqn. (2.32), the normalizing terms are not balanced because KCF uses $\left(\mathbf{J}_i^- + \mathbf{B}_i\right)^{-1}$ to normalize the measurement innovation and $(\mathbf{J}_i^-)^{-1}$ to normalize the innovation from the priors.

3) In Eqn. (2.32), the normalizing term $1 + ||(\mathbf{J}_i^-)^{-1}||$ is a design choice [17] to maintain stability of the algorithm. However, it does not guarantee optimality of KCF. Although GKCF is also not guaranteed to converge to the optimal centralized estimate, it does not bear such a design choice.

2.7 Experimental Evaluation

In this section, we evaluate the performance of the proposed GKCF algorithm in a simulated environment and compare it with other methods: the Centralized Kalman Filter (CKF) and the Kalman Consensus Filter (KCF) [17]. Comparison in a simulated environment allows an in-depth analysis of the proposed algorithms as parameters are varied to measure performance under different conditions.

We simulate a camera network containing N_T targets randomly moving (with a fixed model) within a 500×500 space. Each target's initial state vector is random. A set of N_C camera sensors monitor the space (we consider that each communication node consists of one camera). In each experiment, the cameras are randomly distributed in the space with random orientations resulting in overlapping field-of-views (FOVs).

Target State Parameters

Each target was initialized at the center of the simulation grid. The target's state vector was a $4D$ vector, with the $2D$ position and $2D$ velocity components. The initial speed was set to 2 units per time step and with a random direction uniformly chosen from 0 to 2π. The targets evolved for 40 time steps using the target dynamical model of Eqn. (2.1). Only the targets which remained in the simulation grid for the 40 time steps were considered. The process covariance $\mathbf{Q}$ is set to $diag(10, 10, 1, 1)$.

For the target state-estimation model, the dynamical model of Eqn. (2.1) was also used with the same $\mathbf{Q}$ defined above. The initial prior state $\hat{\mathbf{x}}_i^-(1)$ and prior covariance $\mathbf{P}_i^-(1)$ are set equal at each node. A diagonal matrix is used for $\mathbf{P}_i^-(1)$ with the main diagonal elements as $\{100, 100, 10, 10\}$. The initial prior state $\hat{\mathbf{x}}_i^-(1)$ is generated by adding Gaussian noise of covariance $\mathbf{P}_i^-(1)$ to the ground truth state.

Sensor Parameters

A total of $N_C = 15$ nodes were generated at uniformly chosen random locations on the simulation area. The measurement vector length for each sensor is 2. The FOV was chosen to be equilateral triangles. We define the sensing range, SR, for each sensor to be the height of this equilateral triangle. SR was chosen to be 300 units for all the sensors. A sensor can have an observation of a target only if the ground truth position of the target is within the sensor's FOV and in that case, a measurement $\mathbf{z}_i$ was generated using the linear observation model Eqn. (2.2) with noise covariance $\mathbf{R}_i = 100\mathbf{I}_2$. The observation matrix $\mathbf{H}_i$ and state transition matrix $\mathbf{\Phi}$ is given below.

$$\mathbf{H}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \qquad \mathbf{\Phi} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Network Topology Parameters

A circulant graph was chosen as the communication topology for the sensor network to highlight the different issues associated with the sparsity of the communication network. The circulant graph is Δ-regular where the degree of each node is the same, say Δ. The adjacency matrix of a circulant graph is a circulant matrix. So, we denote Δ to be the degree of the communication graph $\mathcal{G}$. In the experiments, $\Delta = 2$ was used, unless stated otherwise. Note that, in this scenario, the maximum degree $\Delta_{max} = \Delta$ because the degree of all the nodes are same.

Consensus Parameters

In Algorithm 2, each sensor communicates its measurement information to its neighbors iteratively. The maximum number of consensus iterations K was set to 5 unless stated otherwise. The consensus speed parameter was chosen to be $\epsilon = 0.65/\Delta_{max} = 0.65/\Delta$.

Experimental Description

The parameters that were varied in the experiments are, the maximum number of consensus iterations K, degree of the communication network Δ, sensing range SR and the number of cameras N_C. For each experiment, only one parameter was varied while the others were kept constant. As a measure of performance, we computed the estimation error, e, defined as the Euclidean distance between the ground truth position and the estimated posterior position. An example of the simulation framework is shown in Fig. 2.2.

For each experiment, 20 different simulation environments differing in camera poses were randomly generated using the method discussed above. For each environment, 20 different randomly generated target tracks were used. Thus, for each experiment, the estimation errors e, were averaged over $20 \times 20 = 400$ random scenarios, 40 time steps and over all sensors N_C. The mean errors for different methods are shown in the following graphs as the results of different experiments. In the graphs, each line (of a unique color) corresponds to the mean error $\bar{e}$ for one estimation method.

The KCF algorithm as originally proposed in [17] uses a single consensus step per measurement update. To compare it with GKCF, which supports multiple iterations, we extend KCF for multiple iterations. For this, at each time step, the measurement

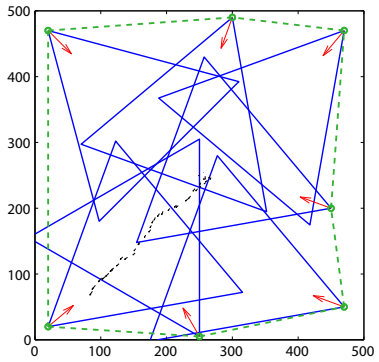


Figure 2.2: In this image of an example simulation environment, the red arrows indicate the locations and orientations of the cameras. The camera FOVs are shown in blue triangles. There are 7 cameras in this example. The green dotted lines represent the network connectivity. Each black arrows depict the actual trajectory of a target moving on the grid.

innovation component is set to zero for $k > 1$ and we consider only the new information provided by the neighboring nodes' estimates.

2.7.1 Experiment 1: Varying K

The objective of this experiment is to compare the performance of different estimation algorithms for different K. Here, K was varied from 1 to 20 at increments of 1. The other parameters were kept constant at their default values. The priors were chosen to be equal.

The results of this experiment are shown in Fig. 2.3. The graph shows that for $K = 1$, GKCF performs much better than KCF close to CKF. As, the number of iterations K is increased, the mean error for GKCF decreases. The main reason for this difference in performance between KCF and GKCF is that KCF does not account for the different information content in different nodes' prior information which unlike KCF, GKCF does. In this simulation all the initial priors were equal.

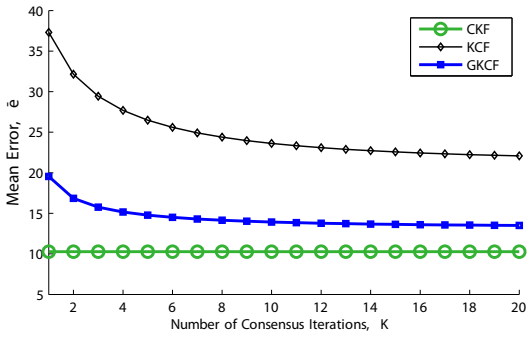


Figure 2.3: Varying K with equal initial priors

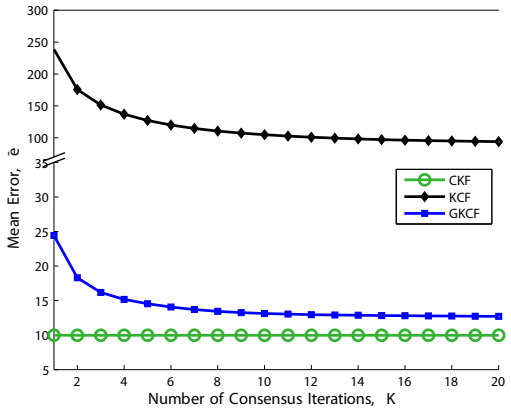


Figure 2.4: Varying K with unequal initial priors

To show the robustness of the GKCF approach, we conducted the same experiment with unequal and uncorrelated priors (Fig. 2.4) using Algorithm 2. The results show that the performance of KCF is severely affected by this whereas GKCF is very robust to unequal initial priors.

2.7.2 Experiment 2: Varying Δ

The objective of this experiment is to compare the performance of different approaches for different values of the degree Δ (i.e., vary the sparsity of the network from sparse connectivity to dense connectivity). For $N_C = 15$, the maximum possible degree can be $\Delta = 14$ at full mesh connectivity. In this experiment, Δ was varied from 2 to 14 at increments of 2.

The results are shown in Fig. 2.5, where the total number of consensus iterations K was set to 5. It can be seen that the GKCF performs much better than KCF at all Δ even with very sparse connectivity. For full connectivity, i.e. $\Delta = 14$, where $\mathcal{G}$ is a complete graph, all the distributed methods achieve centralized performance.

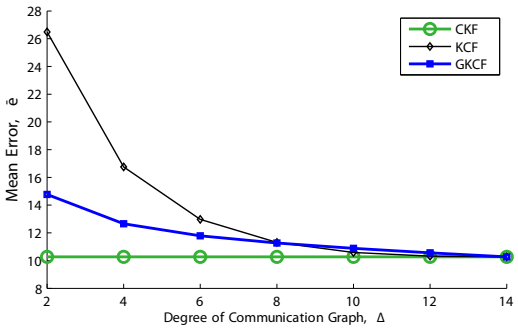


Figure 2.5: Varying the density of connectivity of the network graph.

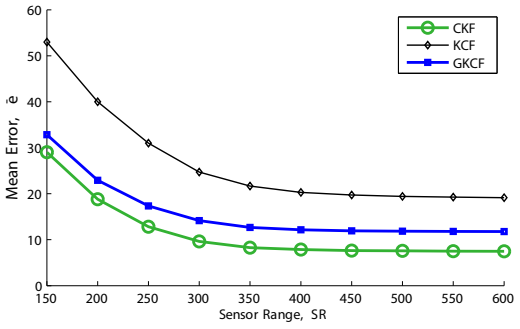


Figure 2.6: Varying the sensing range of the sensors.

2.7.3 Experiment 3: Varying SR

The objective of this experiment is to compare the performance of different approaches as a function of the sensor range SR (i.e., varying the area of coverage of each sensor.)

Fig. 2.6 shows the performance of each approach as SR is varied and the total number of iterations K was set to 5. It can be seen that GKCF performed better than KCF at each sensor range.

2.7.4 Experiment 4: Varying N_C

The objective of this experiment is to compare the performance of different approaches as a function of the total number of sensors, N_C, as it was varied from 5 to 31 at increments of 2. Values less than 5 were not used because at such low number of sensors, even the centralized algorithm cannot perform well due to the high number of time instants at which the number of measurements is insufficient for the state vector to be observable, due to the low percentage coverage of the environment.

Fig. 2.7 shows results for variable N_C and fixed $K = 20$. As the number of sensors is increased, the observability of the targets by the network increases, but

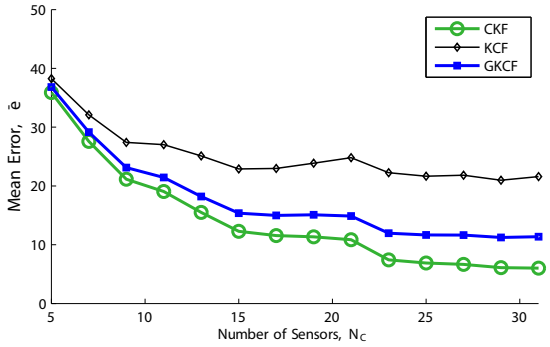


Figure 2.7: Performance comparison of different approaches by varying total number of sensors.

the number of naive nodes also increases. Due to the amount of information about the targets increasing, the performance of all the approaches (including centralized) improves. For a small N_C, all the distributed methods performed close to the centralized method, because the number of naive nodes is small and the network distance from a naive node to an informed node is small. As the number of nodes increases, the number of consensus iterations required to reach consensus also increases. Thus we can see that the performance of all distributed algorithms (both GKCF and KCF) deteriorates more from the centralized performance for high number of sensors. For all N_C, GKCF outperformed KCF.

2.8 Conclusion

In this chapter, we introduced a novel method for distributed state estimation algorithm, the Generalized Kalman Consensus Filter (GKCF). We discussed under what circumstances the assumptions of KCF are not valid and hence modifications are necessary. This is especially true in camera networks where each sensor has a limited FOV and they are geographically separated by distances that do not allow full communication. Then we proposed a generalized framework, Generalized KCF, which outper-

formed the KCF approach under such conditions. We showed the theoretical derivation of our framework and also showed simulation results to compare the performance of our algorithm with other approaches.

Chapter 3

Information-weighted Consensus Filter

3.1 Introduction

Among many types of distributed estimation schemes, *consensus algorithms* [18] are schemes where each node, by iteratively communicating with its network neighbors, can compute a function of the measurements at each node (e.g. average). The consensus estimates asymptotically converge to the global result. In practice, only a limited number of iterations can be performed due to limited bandwidth and target dynamics. Thus, true convergence may not be always reached. In the presence of state dynamics, usually a predictor-corrector model is used for state estimation, where a state prediction is made from the prior information and corrected using new measurements. The Kalman Consensus Filter (KCF) [17] is a popular distributed state estimation framework based on the average consensus algorithm. KCF works well in situations where each node gets a measurement of the target.

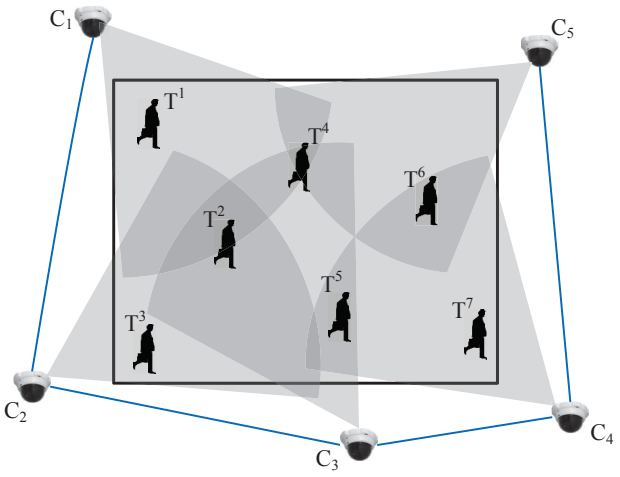


Figure 3.1: In this figure, there are five sensing nodes, $C_1, C_2, \ldots, C_5$ observing an area (black rectangle) consisting of seven targets $T^1, T^2, \ldots T^7$. The solid blue lines show the communication channels between different nodes. This figure also depicts the presence of "naive" nodes. For example, C_1 gets direct measurements about T^1 which it shares with its immediate network neighbor, C_2. However, the rest of the cameras, i.e., C_3, C_4, C_5 do not have *direct* access to measurements of T^1 and thus are naive w.r.t. T^1's state.

In a sensor network, a node might have limited observability when it does not have any measurement of a target available in its *local neighborhood* (consisting of the node and its immediate network neighbors). Due to limited observability and limited number of iterations, the node becomes *naive* about the target's state. A naive node (see Fig. 3.1) contains less information about the state. If a naive node's estimate is given an equal weight in the information fusion scheme (as in KCF), the performance of the overall state estimation framework may decrease. The effect of naivety is severe in sparse networks where the total number of edges is much smaller than the maximum possible number of edges. The Generalized Kalman Consensus Filter (GKCF), was discussed in the previous chapter which overcomes this issue by utilizing a weighted-averaging consensus scheme where the priors of each node were weighted by their covariance matrices.

Both KCF and GKCF estimates are suboptimal and the main reason is that the cross-covariances between the priors across different nodes are not incorporated in the estimation framework. As the consensus progresses, the errors in the information at each node become highly correlated with each other. Thus, to compute the optimal state estimate, the error cross-covariances cannot be neglected. However, it is difficult to compute the cross-covariance in a distributed framework. We note that in a consensus-based framework, the state estimates at different nodes achieve reasonable convergence over multiple iterations. At this point, each node contains almost identical/redundant information. This fact can be utilized to compute the optimal estimate in a distributed framework without explicitly computing the cross-covariances.

Motivated by this idea, we propose an information-weighted consensus algorithm for distributed state estimation which is guaranteed to converge to the optimal centralized estimates as the prior state estimates become equal at different nodes i.e., the total number of iterations approach to infinity at the previous time step. We also show experimentally that even with limited number of iterations, the proposed algorithm achieves near-optimal performance. The issue of naivety and optimality is handled by proper information weighting of the prior and measurement information. The communication bandwidth requirement is also low for the proposed method.

3.1.1 Contributions

The KCF algorithm weights all its neighbors' prior states $\hat{\mathbf{x}}_{i'}^{-}$'s equally which causes high estimation error when naive nodes are present. An initial approach to resolve this issue was proposed in [11]. There the Generalized Kalman Consensus Filter (GKCF) algorithm was proposed where the neighbors' prior $\hat{\mathbf{x}}_{i'}^{-}$'s were weighted by their covariance matrices $\mathbf{P}_{i'}^{-}$'s. The GKCF algorithm outperforms the KCF in the

presence of naive nodes. However, the effect of correlation between errors of the nodes' prior estimates was not brought into account in any of the prior methods because it is usually extremely hard to estimate the cross-covariance across all the nodes in a distributed framework. Mainly due to this reason, these distributed estimation schemes are suboptimal.

Naivety relative to the j-th target is likely to be present in the network when nodes on the vision graph for target j are sparsely connected in the communication graph. The vision graph (see [30]) for target T^j is a graph where there is an edge between each pair of nodes that are observing T^j (see Fig. 3.1). The vision graph is usually time varying, different from the communication graph, and different for different targets. Distributed algorithms that are derived under the assumption that each sensor has full state observability at each time (e.g. KCF), may be adapted to be used in the presence of naive nodes when communication protocols (e.g. [9]) for translating a known vision graph into a set of routing paths that connect nodes in the communication graph are available. However, scalable distributed vision graph discovery algorithms do not exist. Moreover, distributed algorithms that require the knowledge of the vision graphs usually require special target handoff protocols.

Additionally, computation and communication resource constraints are important issues in a distributed estimation framework because in many application scenarios, the nodes are low powered wireless devices. Therefore, a distributed state estimation framework that can guarantee convergence to the optimal centralized estimate while maintaining low computation and communication resource requirements in the presence of naive nodes is desirable. In this chapter, we propose such a distributed state estimation framework called the Information-weighted Consensus Filter (ICF). The ICF is guaranteed to converge to the optimal centralized performance under certain rea-

sonable conditions. We also show experimentally that in other conditions it achieves near-optimal performance.

The issue of naivety is handled in the ICF algorithm by proper information weighting in the estimation framework. Optimality is achieved by proper relative weighting between the prior and the measurement information. ICF also supports multiple consensus iterations at each time step t to improve performance. In addition, the experimental results show that ICF outperforms other distributed estimation algorithms at any given computation and communication resource limit. The ICF algorithm does not require handoff protocols or the knowledge of the vision graph.

3.1.2 Related work

Recently, distributed decision and control frameworks have gained immense popularity. *Consensus algorithms* [18] are one of the many types of distributed schemes used for collective decision making. Consensus algorithms are protocols that are run individually by each agent where each agent communicates with just its network neighbors and corrects its own information iteratively using the information sent by its neighbors. The protocol, over multiple iterations, ensures the convergence of all the agents in the network to a single consensus. The consensus they reach is a predefined function of all the information available in the network. It is important to note that this consensus is reached just by peer-to-peer communication without requiring a central fusion node. For example, the item being estimated may be the arithmetic mean (average consensus) [18] or the geometric mean [19] of the initial values. The simplicity and scalability of consensus algorithms makes them extremely useful in distributed estimation tasks in sensor networks.

Consensus algorithms have been extended to perform various linear algebraic operations such as SVD, least squares, PCA, GPCA in a network of agents [30] It also has been utilized in distributed state estimation framework such as the Kalman Consensus Filter (KCF) [17] and the Generalized Kalman Consensus Filter (GKCF) [11] that we discussed in the previous chapter. The KCF algorithm is a popular distributed estimation framework and has reasonable performance in networks where the entire state is individually observable by each node. However, its performance deteriorates in the presence of naivety [11]. The idea of information-weighted consensus was introduced in [13]; however, that article did not provide the detailed theoretical analysis of its properties, comparisons with other approaches, or the implications for its application in wide-area camera networks. These items are each provided herein. A survey on distributed estimation and control applications using linear consensus algorithms can be found in [8]. Here the authors show how a weighted average can be computed in a distributed framework using two parallel average consensus schemes. Distributed state and parameter estimation frameworks have been applied in various fields including camera networks for distributed implementations of 3-D point triangulation, pose estimation [30], tracking [28], action recognition [28, 15], collaborative tracking and camera control [27, 7], camera calibration [29, 3] etc.

3.2 Problem Formulation

The communication network is represented by the undirected connected graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$. The set $\mathcal{C} = \{C_1, ..., C_{N_C}\}$ contains the vertices of the graph and represents the communication nodes. The set $\mathcal{E}$ contains the edges of the graph, which represent the available communication channels between different nodes. Also, let $\mathcal{N}_i$ be the set of

nodes that have a direct communication channel with node C_i (i.e. shares an edge with C_i). We call the nodes in $\mathcal{N}_i$, the neighbors of C_i. The number of neighbors (degree) of C_i is represented by Δ_i. For simplicity, we will drop the time step index t in places where the time step is not important to explain the issue under consideration. There are N_T targets in the area and the length of their individual state vectors is q. Thus, for $\mathbf{x} \in \mathcal{R}^p$, p would be equal to qN_T. The number of targets can be time-varying, but that will not be a focus of the thesis, as it would distract from the main topic.

A data association protocol is required for a multi-target estimation framework. However, in this chapter, to analyze the performance of the state estimation approaches independent of a data association method, we assume that the data association is given and without errors. Data association will be brought into account in the following chapters.

At each time step t, a node C_i may get some measurements from its sensors. For example, for a target state estimation task in a camera network, a measurement might be the projection of the position of some target onto a camera's pixel coordinate system. We denote the measurement at C_i as $\mathbf{z}_i \in \mathcal{R}^{m_i}$. As, the sensors can be heterogeneous and the number of sensors can be different at each communication node, the length of the measurement vector m_i can vary across different nodes. The measurement $\mathbf{z}_i$ is modeled as

$$\mathbf{z}_i = \mathbf{H}_i\mathbf{x} + \boldsymbol{\nu}_i. \tag{3.1}$$

Here $\mathbf{H}_i \in \mathcal{R}^{m_i \times p}$ is the observation matrix for node C_i. The matrix $\mathbf{H}_i$ can be time-varying. The noise $\boldsymbol{\nu}_i \in \mathcal{R}^{m_i}$ in the measurement of C_i is modeled as a zero mean Gaussian random variable $\mathcal{N}(\mathbf{0}, \mathbf{R}_i)$ where $\mathbf{R}_i \in \mathcal{R}^{m_i \times m_i}$. As we will derive the estimator in the information form, we will generally end up with equation with inverse covariances

(information matrix/ precision matrix). If a node C_i does not have a measurement, we will use $(\mathbf{R}_i)^{-1} = \mathbf{0} \in \mathcal{R}^{m_i \times m_i}$ and $\mathbf{z}_i = 0 \in \mathcal{R}^{m_i}$. Note that $\mathbf{H}_i$ is typically not full column rank ($m_i < p$). In fact, we are not assuming that the state $\mathbf{x}$ is observable from $\mathbf{z}_i$. We do assume that $\mathbf{x}$ is observable from $\{\mathbf{z}_i\}_{i=1}^{N_C}$. The collection of all measurements from all sensors can be expressed as,

$$\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{H}}\mathbf{x} + \boldsymbol{\nu}. \tag{3.2}$$

Here, $\boldsymbol{\mathcal{Z}} = [\mathbf{z}_1^T, \mathbf{z}_2^T, \ldots \mathbf{z}_N^T]^T \in \mathcal{R}^m$ is the concatenation of all measurements in the network where $m = \sum_{i=1}^{N} m_i$ and $\boldsymbol{\mathcal{H}} = [\mathbf{H}_1^T, \mathbf{H}_2^T, \ldots \mathbf{H}_N^T]^T \in \mathcal{R}^{m \times p}$ is the stack of all the observation matrices. For the measurement noise vector, $\boldsymbol{\nu} = [\boldsymbol{\nu}_1^T, \boldsymbol{\nu}_2^T, \ldots \boldsymbol{\nu}_N^T]^T \in \mathcal{R}^m$, we denote its covariance as $\boldsymbol{\mathcal{R}} \in \mathcal{R}^{m \times m}$. We assume the measurement noise to be uncorrelated across nodes. Thus, the measurement covariance matrix is $\boldsymbol{\mathcal{R}} = diag(\mathbf{R}_1, \mathbf{R}_2, \ldots \mathbf{R}_{N_C})$.

3.3 Centralized MAP Estimation

Before considering the decentralized solution for the estimation of $\mathbf{x}$ it is informative to review the centralized solution, i.e., the centralized maximum a posteriori (MAP) estimator. The centralized prior state estimate of $\mathbf{x}$ is denoted as $\hat{\mathbf{x}}_c^- \in \mathcal{R}^p$ where the error in the prior estimate is $\boldsymbol{\eta}_c = \hat{\mathbf{x}}_c^- - \mathbf{x}$ with $cov(\boldsymbol{\eta}_c) = \mathbf{P}_c^- \in \mathcal{R}^{p \times p}$, which is assumed to be nonsingular. The observation model in Eqn. (3.2) and prior state $\hat{\mathbf{x}}_c^-$ can be combined into one equation as

$$\begin{bmatrix} \hat{\mathbf{x}}_c^- \\ \\ \boldsymbol{\mathcal{Z}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_p \\ \\ \boldsymbol{\mathcal{H}} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \boldsymbol{\eta}_c \\ \\ \boldsymbol{\nu} \end{bmatrix}. \tag{3.3}$$

Here $\mathbf{I}_p$ is the $p \times p$ identity matrix.

Letting, $\boldsymbol{\mathcal{Y}} = \begin{bmatrix} \hat{\mathbf{x}}_c^- \\ \boldsymbol{\mathcal{Z}} \end{bmatrix}$, $\boldsymbol{\mathcal{H}}_c = \begin{bmatrix} \mathbf{I}_p \\ \boldsymbol{\mathcal{H}} \end{bmatrix}$ and $\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\eta}_c \\ \boldsymbol{\nu} \end{bmatrix}$, we have $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}}_c\mathbf{x} + \boldsymbol{\beta}$, where $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\mathcal{C}})$. We assume that the error in the prior state estimate is uncorrelated to the measurement noise. Thus, we have the block diagonal covariance matrix $\boldsymbol{\mathcal{C}} = diag(\mathbf{P}_c^-, \boldsymbol{\mathcal{R}})$. Let us define the prior information matrix to be $\mathbf{J}_c^- = (\mathbf{P}_c^-)^{-1} \in \mathcal{R}^{p\times p}$. Thus, defining $\boldsymbol{\mathcal{J}} = \boldsymbol{\mathcal{C}}^{-1}$ we have $\boldsymbol{\mathcal{J}} = diag(\mathbf{J}_c^-, \boldsymbol{\mathcal{R}}^{-1})$. The centralized maximum a posteriori (MAP) estimate [16] of the state $\mathbf{x}$ is

$$\begin{aligned} \mathbf{x}_c^+ &= (\boldsymbol{\mathcal{H}}_c^T \boldsymbol{\mathcal{J}} \boldsymbol{\mathcal{H}}_c)^{-1} (\boldsymbol{\mathcal{H}}_c^T \boldsymbol{\mathcal{J}} \boldsymbol{\mathcal{Y}}) \\ &= \left(\mathbf{J}_c^- + \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{H}}\right)^{-1} (\mathbf{J}_c^- \mathbf{x}_c^- + \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{Z}}), \qquad (3.4) \\ \mathbf{J}_c^+ &= \left(\mathbf{J}_c^- + \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{H}}\right). \qquad (3.5) \end{aligned}$$

where $\mathbf{J}_c^+ = (cov(\hat{\mathbf{x}}_c^+))^{-1}$ quantifies the information about $\mathbf{x}$ in $\hat{\mathbf{x}}_c^+$. Eqns. (3.4-3.5) are useful in the discussion of the physical interpretations of the alternative decentralized algorithms.

3.4 Information Consensus based Distributed MAP Estimation (IC-MAP)

In a distributed estimation framework, each node C_i possesses a prior estimate of the state vector that we denote as $\hat{\mathbf{x}}_i^- \in \mathcal{R}^p$ (unlike a single prior state estimate in a centralized solution). The objective of the network is to use distributed computations across the network such that the posterior state estimate $\hat{\mathbf{x}}_i^+ \in \mathcal{R}^p$ at each node converges to the centralized estimate $\hat{\mathbf{x}}_c^+$. However, due to resource constraints, this convergence may not be fully achieved at any given time. Therefore, if consensus were performed directly on the priors, then at the k-th consensus iteration, the estimate of the i-th node

can be modeled as having three components

$$\hat{\mathbf{x}}_i^-[k] = \mathbf{x} + \boldsymbol{\eta}_c + \boldsymbol{\delta}_i[k],$$

where $\hat{\mathbf{x}}_c^- = \mathbf{x} + \boldsymbol{\eta}_c$ with the quantities $\mathbf{x}$ and $\boldsymbol{\eta}_c$ having been previously defined, and the consensus algorithm ensures that $\|\boldsymbol{\delta}_i[k]\|$ approaches zero as k approaches infinity for all i. The index in the parenthesis after a variable will refer to the time-index and the index in the square brackets will refer to the consensus iteration index.

The error covariance in the prior estimate at C_i is $\mathbf{P}_{ii}^-[k] = E[\boldsymbol{\eta}_i[k]\,(\boldsymbol{\eta}_i[k])^T] \in \mathcal{R}^{p\times p}$, where $\boldsymbol{\eta}_i[k] = \boldsymbol{\eta}_c + \boldsymbol{\delta}_i[k]$. Similarly, the error cross-covariance between C_i and $C_{i'}$'s prior estimates is $\mathbf{P}_{ii'}^-[k] = E[\boldsymbol{\eta}_i[k]\,(\boldsymbol{\eta}_{i'}[k])^T] \in \mathcal{R}^{p\times p}$. As $k \to \infty$, at each node, consensus forces $\hat{\mathbf{x}}_i^-[k] \to \hat{\mathbf{x}}_c^-$. Therefore, for any $\{i, i'\}$, $\hat{\mathbf{x}}_i^-[k]$ and $\hat{\mathbf{x}}_{i'}^-[k]$ becomes correlated as $\hat{\mathbf{x}}_i^-[k] \to \hat{\mathbf{x}}_{i'}^-[k]$ resulting in $\mathbf{P}_{ii'}^-[k] \neq 0$. In fact, it is straightforward to show that as the number of consensus iterations k approaches infinity, $\mathbf{P}_{ii'}^-[k]$ converges to $\mathbf{P}_c^-$ for all $\{i, i'\}$.

We drop the k and denote the collection of all the state priors from all the nodes as $\widehat{\boldsymbol{\mathcal{X}}}^- = [(\hat{\mathbf{x}}_1^-)^T, (\hat{\mathbf{x}}_2^-)^T, \ldots, (\hat{\mathbf{x}}_{N_C}^-)^T]^T \in \mathcal{R}^{Np}$. The relationship between the state, the priors and the prior errors can be summarized as,

$$\widehat{\boldsymbol{\mathcal{X}}}^- = \boldsymbol{\mathcal{H}}_I \mathbf{x} + \boldsymbol{\eta}. \tag{3.6}$$

Here, $\mathbf{x}$ is the true state of the targets, $\boldsymbol{\eta} = [\boldsymbol{\eta}_1^T, \boldsymbol{\eta}_2^T, \ldots \boldsymbol{\eta}_{N_C}^T]^T \in \mathcal{R}^{N_C p}$ is the error vector, and $\boldsymbol{\mathcal{H}}_I = [\mathbf{I}_p, \mathbf{I}_p, \ldots, \mathbf{I}_p]^T \in \mathcal{R}^{N_C p\times p}$.

Combining the measurements with the result of a *finite number* k of steps of consensus on the priors yields

$$\begin{bmatrix} \widehat{\boldsymbol{\mathcal{X}}}^- \\ \boldsymbol{\mathcal{Z}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mathcal{H}}_I \\ \boldsymbol{\mathcal{H}} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \boldsymbol{\eta} \\ \boldsymbol{\nu} \end{bmatrix}. \tag{3.7}$$

Letting, $\boldsymbol{\mathcal{Y}'} = \begin{bmatrix} \widehat{\boldsymbol{\mathcal{X}}}^{-} \\ \boldsymbol{\mathcal{Z}} \end{bmatrix}$, $\boldsymbol{\mathcal{H}'}_c = \begin{bmatrix} \boldsymbol{\mathcal{H}}_I \\ \boldsymbol{\mathcal{H}} \end{bmatrix}$ and $\boldsymbol{\beta'} = \begin{bmatrix} \boldsymbol{\eta} \\ \boldsymbol{\nu} \end{bmatrix}$, we have $\boldsymbol{\mathcal{Y}'} = \boldsymbol{\mathcal{H}'}_c\mathbf{x} + \boldsymbol{\beta'}$, where $\boldsymbol{\beta'} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\mathcal{C}'})$. The noise term $\boldsymbol{\beta'}$ is Gaussian because it is accumulated through one or more consensus iterations (which are linear operations) performed on Gaussian random variables.

Let us denote the prior information matrix $\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{P}}^{-1}$ where these two matrices can be expressed as $(p \times p)$ blocks,

$$\boldsymbol{\mathcal{P}} = \begin{bmatrix} \mathbf{P}_{11}^{-} & \mathbf{P}_{12}^{-} & \dots & \mathbf{P}_{1N_C}^{-} \\ \mathbf{P}_{21}^{-} & \mathbf{P}_{22}^{-} & & \vdots \\ \vdots & & \ddots & \\ \mathbf{P}_{N_C1}^{-} & \dots & & \mathbf{P}_{N_CN_C}^{-} \end{bmatrix}, \; \boldsymbol{\mathcal{F}} = \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} & \dots & \mathbf{F}_{1N_C} \\ \mathbf{F}_{21} & \mathbf{F}_{22} & & \vdots \\ \vdots & & \ddots & \\ \mathbf{F}_{N_C1} & \dots & & \mathbf{F}_{NN} \end{bmatrix}. \tag{3.8}$$

Let us define the information matrix of the prior of node i as

$$\mathbf{J}_i^{-} = (\mathbf{P}_{ii}^{-})^{-1} \tag{3.9}$$

Here, $\mathbf{J}_i^{-} \in \mathcal{R}^{p \times p}$ and in general, $\mathbf{J}_i^{-} \neq \mathbf{F}_{ii}$. Assuming that the error in the prior state estimates are uncorrelated to the noise in the new measurements, we have the block diagonal covariance matrix $\boldsymbol{\mathcal{C}'} = diag(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{R}})$. Thus, we get its inverse as $\boldsymbol{\mathcal{J}'} = diag(\boldsymbol{\mathcal{F}}, \boldsymbol{\mathcal{R}}^{-1})$.

The centralized maximum a posteriori (MAP) estimate of the state $\mathbf{x}$ is

$$\begin{aligned} \hat{\mathbf{x}}_c^{+} &= (\boldsymbol{\mathcal{H}'}_c^T \boldsymbol{\mathcal{J}'} \boldsymbol{\mathcal{H}'}_c)^{-1} (\boldsymbol{\mathcal{H}'}_c^T \boldsymbol{\mathcal{J}'} \boldsymbol{\mathcal{Y}'}) \\ &= \left(\boldsymbol{\mathcal{H}}_I^T \boldsymbol{\mathcal{F}} \boldsymbol{\mathcal{H}}_I + \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{H}}\right)^{-1} (\boldsymbol{\mathcal{H}}_I^T \boldsymbol{\mathcal{F}} \widehat{\boldsymbol{\mathcal{X}}}^{-} + \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{Z}}), \qquad (3.10) \\ \mathbf{J}_c^{+} &= \boldsymbol{\mathcal{H}}_I^T \boldsymbol{\mathcal{F}} \boldsymbol{\mathcal{H}}_I + \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{H}}. \qquad (3.11) \end{aligned}$$

Defining

$$\mathbf{F}_i^{-} = \sum_{i'=1}^{N_C} \mathbf{F}_{i'i}, \tag{3.12}$$

we have

$$\boldsymbol{\mathcal{H}}_I^T \boldsymbol{\mathcal{F}} \boldsymbol{\mathcal{H}}_I = \sum_{i=1}^{N_C} \mathbf{F}_i^- \quad \text{and} \quad \boldsymbol{\mathcal{H}}_I^T \boldsymbol{\mathcal{F}} \widehat{\boldsymbol{\mathcal{X}}}^- = \sum_{i=1}^{N_C} \mathbf{F}_i^- \mathbf{x}_i^-. \tag{3.13}$$

Let us define $\mathbf{U}_i = \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{H}_i$ and $\mathbf{u}_i = \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{z}_i$. Due to the block diagonal structure of $\boldsymbol{\mathcal{R}}^{-1}$, we get

$$\boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{H}} = \sum_{i=1}^{N_C} \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{H}_i = \sum_{i=1}^{N_C} \mathbf{U}_i, \tag{3.14}$$

$$\boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{Z}} = \sum_{i=1}^{N_C} \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{z}_i = \sum_{i=1}^{N_C} \mathbf{u}_i. \tag{3.15}$$

Thus, from Eqns. (3.10) and (3.11), we get

$$\hat{\mathbf{x}}_c^+ = \left(\sum_{i=1}^{N_C} (\mathbf{F}_i^- + \mathbf{U}_i) \right)^{-1} \sum_{i=1}^{N_C} (\mathbf{F}_i^- \mathbf{x}_i^- + \mathbf{u}_i), \tag{3.16}$$

$$\mathbf{J}_c^+ = \sum_{i=1}^{N_C} (\mathbf{F}_i^- + \mathbf{U}_i). \tag{3.17}$$

This is the centralized solution that fully accounts for the presence of different priors at each agent, and the cross-correlated errors in the priors between agents, which develop naturally due to consensus, but which are not known in a decentralized implementation. In the following, we show how Eqns. (3.16) and (3.17) can be computed in a distributed manner.

3.4.1 Distributed Implementation

To implement Eqns. (3.16-3.17) in a distributed fashion, define, $\mathbf{V}_i[0] = \mathbf{F}_i^- + \mathbf{U}_i$ and $\mathbf{v}_i[0] = \mathbf{F}_i^- \hat{\mathbf{x}}_i^- + \mathbf{u}_i$, so that

$$\hat{\mathbf{x}}_c^+ = \left(\sum_{i=1}^{N_C} \mathbf{V}_i[0] \right)^{-1} \sum_{i=1}^{N_C} \mathbf{v}_i[0], \tag{3.18}$$

$$\mathbf{J}_c^+ = \sum_{i=1}^{N_C} \mathbf{V}_i[0]. \tag{3.19}$$

Under the assumption that a node C_i has information about $\mathbf{F}_i^-$ (methods for computing $\mathbf{F}_i^-$ will be discussed later), C_i could compute $\mathbf{V}_i[0] = \mathbf{F}_i^- + \mathbf{U}_i$ and $\mathbf{v}_i[0] = \mathbf{F}_i^- \hat{\mathbf{x}}_i^- + \mathbf{u}_i$ from $\mathbf{F}_i^-$, its own state prior $\hat{\mathbf{x}}_i^-$, measurement $\mathbf{z}_i$, measurement information matrix $\mathbf{R}_i^{-1}$ and measurement model parameter $\mathbf{H}_i$. Then, each node transmits to its neighbors its own information matrix $\mathbf{V}_i[k] \in \mathcal{R}^{p\times p}$ and information vector $\mathbf{v}_i[k] \in \mathcal{R}^p$, receives its neighbors' information, and uses the average consensus algorithm as described in Sec. 2.3 to converge toward the global averages of these two quantities. Therefore, from Eqns. (3.18-3.19) we get

$$\hat{\mathbf{x}}_c^+ = \lim_{k\to\infty} (N_C \mathbf{V}_i[k])^{-1} (N_C \mathbf{v}_i[k]) = \lim_{k\to\infty} (\mathbf{V}_i[k])^{-1} \mathbf{v}_i[k] \tag{3.20}$$

$$\mathbf{J}_c^+ = \lim_{k\to\infty} N_C \mathbf{V}_i[k] \tag{3.21}$$

From the discussion above, given $\mathbf{F}_i^-$, it is clear that the centralized MAP estimate in Eqns. (3.18-3.19) is achieved using a distributed scheme as $k \to \infty$. We call this distributed approach of computing the MAP estimate, the Information Consensus based MAP (IC-MAP) estimation framework.

3.4.2 Computation of $\mathbf{F}_i^-$

To compute the distributed MAP estimate, node C_i needs to have knowledge of $\mathbf{F}_i^-$. In general, computation of $\mathbf{F}_i^-$ requires the knowledge of the entire covariance matrix $\boldsymbol{\mathcal{P}}$ defined in Eqn. (3.8) (i.e. the prior covariances ($\mathbf{P}_{ii}^-$'s) of each node and the prior cross-covariances ($\mathbf{P}_{ii'}^-$'s) between each pair of nodes). However, computing $\boldsymbol{\mathcal{P}}$ at every time step at each node in a distributed framework is unrealistic as it would require too much information exchange among the nodes. However, in the following, we show that for two special cases (which are of great practical importance), $\mathbf{F}_i^-$ can be computed at each node using only a node's own prior covariance matrix $\mathbf{P}_{ii}^-$ (or $\mathbf{J}_i^-$ in the

information form). The first case is for converged priors which is an important scenario because in a consensus-based framework, with high-enough number of consensus iterations, the prior state information at all the nodes ultimately converge to the same value. The second case is when the prior state estimates across the nodes are uncorrelated to each other. This is generally true at the early steps of consensus when the nodes had no prior information about the target and initialized their prior information with random quantities.

The proposed distributed state estimation framework in Sec. 3.5 is based on these two special cases. The practical significance of these two cases can be seen from the experimental results in Sec. 3.6 which implies that the proposed algorithm (derived from these two special cases) is robust even when the assumptions of neither of these two cases are met.

3.4.2.1 Case 1: Converged Priors

Here we will discuss the case where the estimate of the state vector at each node has converged to the centralized estimate at the previous time step $t-1$. Thus at time t, the prior information at each node is the same and equal to the prior information of a centralized framework (i.e., k sufficiently large such that $\|\boldsymbol{\delta}_i[k]\| = 0$ for all i). This case will be of great significance when we will incorporate target dynamics and additional measurement steps to our framework. From Eqns. (3.5) and (3.17) we have

$$\mathbf{J}_c^+ = \mathbf{J}_c^- + \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}} = \sum_{i=1}^{N_C}(\mathbf{F}_i^- + \mathbf{U}_i). \tag{3.22}$$

as from Eqn. (3.14), $\boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}} = \sum_{i=1}^{N_C}\mathbf{U}_i$. Thus, from Eqn. (3.22),

$$\sum_{i=1}^{N_C}\mathbf{F}_i^- = \mathbf{J}_c^- = \sum_{i=1}^{N_C}\frac{\mathbf{J}_c^-}{N_C}. \tag{3.23}$$

Now, for converged priors, for all i we have $\mathbf{J}_c^- = \mathbf{J}_i^-$. Using this in Eqn. (3.23), we have

$$\sum_{i=1}^{N_C} \mathbf{F}_i^- = \sum_{i=1}^{N_C} \frac{\mathbf{J}_i^-}{N_C}. \tag{3.24}$$

Using this in Eqns. (3.16) and (3.17) and using the fact that $\hat{\mathbf{x}}_i^- = \hat{\mathbf{x}}_c^-$ for converged priors, we have

$$\hat{\mathbf{x}}_c^+ = \left(\sum_{i=1}^{N_C} \left(\frac{\mathbf{J}_i^-}{N_C} + \mathbf{U}_i \right) \right)^{-1} \sum_{i=1}^{N_C} \left(\frac{\mathbf{J}_i^-}{N_C} \hat{\mathbf{x}}_i^- + \mathbf{u}_i \right), \tag{3.25}$$

$$\mathbf{J}_c^+ = \sum_{i=1}^{N_C} \left(\frac{\mathbf{J}_i^-}{N_C} + \mathbf{U}_i \right). \tag{3.26}$$

This can be computed in a distributed manner by initializing $\mathbf{V}_i[0] = \frac{1}{N_C}\mathbf{J}_i^- + \mathbf{U}_i$ and $\mathbf{v}_i[0] = \frac{1}{N_C}\mathbf{J}_i^- \hat{\mathbf{x}}_i^- + \mathbf{u}_i$. This is equivalent to using $\frac{1}{N_C}\mathbf{J}_i^-$ instead of $\mathbf{F}_i^-$ in Eqns. (3.16-3.17).

The intuition behind this is as follows. After convergence, given the prior at one node, the other priors do not contain any new information. Upon convergence in the previous time step, the prior information matrix at each node $\mathbf{J}_i^-$ is equal to $\mathbf{J}_c^-$: Each agent has an identical copy ($\hat{\mathbf{x}}_i^- = \hat{\mathbf{x}}_c^-$) and amount of information ($\mathbf{J}_i^- = \mathbf{J}_c^-$). Thus at the current time step, the prior information matrix $\mathbf{J}_i^-$ should be divided by N_C as shown in the formulation of Eqns. (3.25-3.26), so that the effective total weight of all the priors in the estimation scheme remains as $\mathbf{J}_c^-$.

3.4.2.2 Case 2: Uncorrelated Prior Errors

Now, we consider the case where the errors in the priors are uncorrelated with each other across different nodes. This case can be used at the initial time step if it is known that the prior errors are uncorrelated across different nodes.

When the prior state errors are uncorrelated, the covariance matrix $\boldsymbol{\mathcal{P}}$ will be block diagonal. So, its inverse $\boldsymbol{\mathcal{F}}$ will also be block diagonal as, $\boldsymbol{\mathcal{F}} = diag(\mathbf{F}_{11}$, $\mathbf{F}_{22}$, $\ldots$ $\mathbf{F}_{N_C N_C}) = diag\left((\mathbf{P}_{11}^{-})^{-1}, (\mathbf{P}_{22}^{-})^{-1}, \ldots (\mathbf{P}_{N_C N_C}^{-})^{-1}\right)$. In this special case, with $\mathbf{J}_i^-$ defined in Eqn. (3.9), we have

$$\mathbf{F}_{ii} = (\mathbf{P}_{ii}^{-})^{-1} = \mathbf{J}_i^-. \tag{3.27}$$

As the off-diagonal block elements of $\boldsymbol{\mathcal{F}}$ are zero, from the definition of $\mathbf{F}_i^-$ in Eqn. (3.12), we have

$$\mathbf{F}_i^- = \sum_{i'=1}^{N_C} \mathbf{F}_{i'i} = \mathbf{F}_{ii} = \mathbf{J}_i^-. \tag{3.28}$$

Thus, when the prior errors are uncorrelated across the different nodes, using $\mathbf{F}_i^- = \mathbf{J}_i^-$ (in Eqns. (3.16-3.17)) and average consensus, we can compute the centralized MAP estimate in a distributed framework.

3.5 Information-weighted Consensus Filter

In the previous section, we derived the Information Consensus based MAP (IC-MAP) estimation framework and proved its optimality for two important scenarios. In this section, we will consider a dynamic model in state space form and extend the IC-MAP framework in Sec. 3.4 for distributed state estimation. We call this distributed state estimation framework, the *Information-weighted Consensus Filter (ICF)*. We prove theoretically that as the number of consensus iteration $k \rightarrow \infty$, the ICF estimates converge to the estimates of the centralized Kalman filter.

Let us consider the following linear dynamical model

$$\mathbf{x}(t+1) = \boldsymbol{\Phi}\mathbf{x}(t) + \boldsymbol{\gamma}(t). \tag{3.29}$$

Here $\boldsymbol{\Phi} \in \mathcal{R}^{p \times p}$ is the state transition matrix and the process noise is $\boldsymbol{\gamma}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$.

For this model, for the centralized case, we have the following state prediction step [16],

$$\mathbf{J}_c^-(t+1) = \left(\mathbf{\Phi}\mathbf{J}_c^+(t)^{-1}\mathbf{\Phi}^T + \mathbf{Q}\right)^{-1}, \tag{3.30}$$

$$\hat{\mathbf{x}}_c^-(t+1) = \mathbf{\Phi}\hat{\mathbf{x}}_c^+(t). \tag{3.31}$$

Combining this with the IC-MAP estimation framework proposed in Sec. 3.4, we get the Information-weighted Consensus Filter (ICF) in Algorithm 3.

Algorithm 3 ICF at node C_i at time step t

Input: prior state estimate $\hat{\mathbf{x}}_i^-(t)$, prior information matrix $\mathbf{J}_i^-(t)$, observation matrix $\mathbf{H}_i$, consensus speed factor ϵ and total number of consensus iterations K.

1) Get measurement $\mathbf{z}_i$ and measurement information matrix $\mathbf{R}_i^{-1}$

2) Compute consensus proposals,

$$\mathbf{V}_i[0] \leftarrow \frac{1}{N_C}\mathbf{J}_i^-(t) + \mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i \tag{3.32}$$

$$\mathbf{v}_i[0] \leftarrow \frac{1}{N_C}\mathbf{J}_i^-(t)\hat{\mathbf{x}}_i^-(t) + \mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{z}_i \tag{3.33}$$

3) Perform average consensus (Sec. 2.3) on $\mathbf{v}_i[0]$ and $\mathbf{V}_i[0]$ independently for K iterations.

4) Compute a posteriori state estimate and information matrix for time t

$$\hat{\mathbf{x}}_i^+(t) \leftarrow (\mathbf{V}_i[K])^{-1}\mathbf{v}_i[K] \tag{3.34}$$

$$\mathbf{J}_i^+(t) \leftarrow N_C\mathbf{V}_i[K] \tag{3.35}$$

5) Predict for next time step $(t+1)$

$$\mathbf{J}_i^-(t+1) \leftarrow \left(\mathbf{\Phi}(\mathbf{J}_i^+(t))^{-1}\mathbf{\Phi}^T + \mathbf{Q}\right)^{-1} \tag{3.36}$$

$$\hat{\mathbf{x}}_i^-(t+1) \leftarrow \mathbf{\Phi}\hat{\mathbf{x}}_i^+(t) \tag{3.37}$$

Output: ICF estimate $\hat{\mathbf{x}}_i^+(t)$ and information matrix $\mathbf{J}_i^+(t)$.

Note that in Eqns. (3.32) and (3.33), we have used the results for the converged prior case (Sec. 3.4.2.1), i.e., using $\frac{1}{N_C}\mathbf{J}_i^-$ instead of $\mathbf{F}_i^-$. Theoretically, at each time step, if $k \to \infty$, the IC-MAP estimator guarantees that the priors for the next time step at each node will be equal to the optimal centralized one. This convergence further guarantees that the optimal centralized estimate will be reached at the next time steps if $\mathbf{F}_i^- = \frac{1}{N_C}\mathbf{J}_i^-$ is used. This guarantees the optimality of Algorithm 3 with $k \to \infty$ at each time step.

In practice, due to the fact that the total number of iterations, k, is finite, convergence will not be achieved fully. Therefore, the distributed implementation will not perform quite as well as the centralized solution. The simulation results in Sec. 3.6 will demonstrate that the ICF has near-optimal performance as either k increases or t increases for a fixed k.

While Eqn. (3.20) shows that inclusion of N_C is inconsequential in the computation of $\mathbf{x}$ in the CKF, the role of N_C is critical in the distributed implementation. At Eqns. (3.32) and (3.33), due to prior consensus steps, all nodes have the same estimate with the same information $\mathbf{J}_i^-$. If the $\frac{1}{N_C}$ is neglected, then the measurement information receives too little relative weighting by a factor of N_C.

3.5.1 Initialization and Special Situations

In a practical implementation scenario, at system startup or for the first few iterations in a naive node, $\mathbf{V}_i[0]$ in Eqn. (3.34) can be $\mathbf{0}$, if there is no prior or measurement information available in the local neighborhood. In this situation, a node will not perform step 4 and 5 in Algorithm 3 until it receives non-zero information from its neighbors (through step 3) or gets a measurement itself (through step 1) yielding $\mathbf{V}_i[0]$ to be non-zero. In the target tracking application, this situation occurs when a new

target is detected by one or more, but not all of the sensors.

In some situations prior information may be present at system startup. If the priors are known to be equal, then using the standard ICF algorithm as in Algorithm 3, should give optimal results. However, if the priors across the nodes are known to be uncorrelated at the initial time step ($t = 1$), the results for the uncorrelated case (Sec.3.4.2.2) should be used instead in the first time step. To do this, only at $t = 1$, instead of Eqns. (3.32) and (3.33) in Algorithm 3, the following initializations should be used,

$$\mathbf{V}_i[0] = \mathbf{J}_i^- + \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{H}_i \tag{3.38}$$

$$\mathbf{v}_i[0] = \mathbf{J}_i^- \hat{\mathbf{x}}_i^- + \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{z}_i. \tag{3.39}$$

Thus, at $t = 1$, with $k \to \infty$, the states would converge to the optimal centralized estimate. Then for $t > 1$, using Algorithm 3 would guarantee convergence to the optimal centralized estimate for the following time steps (i.e. for $t > 1$).

An example of the estimation results of different methods is shown in Fig. 3.2. The example includes $N_C = 15$ nodes and a single target. The target state contains the two dimensional position and velocity. Each line represents the state estimate at one node after each iteration. For clarity, in each experiment only the first component of the state vector for the first 3 node's is shown. State estimation is performed for 3 time steps. At each time step, 30 consensus iterations were performed. The priors were initially uncorrelated and different at each node. For ICF, at $t = 1$, the uncorrelated initializations Eqns. (3.38-3.39) were used. This example shows that the ICF converges to the centralized estimate at each time step after several iterations. The reason that ICF performs better than KCF is because KCF's performance deteriorates in the presence of naivety and the cross-covariances between the priors are implicitly considered in ICF.

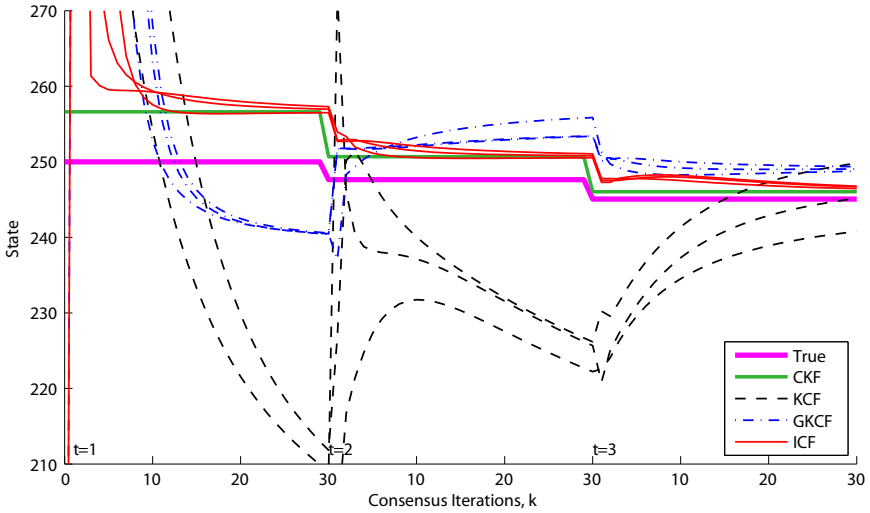


Figure 3.2: An example showing the convergence of different algorithms with multiple consensus iterations at different time steps.

3.5.2 ICF, GKCF and KCF Comparison

The KCF algorithm in [17] was originally presented using a single consensus step. This section presents the state estimation step of the ICF, GKCF and KCF algorithm in an equivalent form for a single consensus step (i.e., $K = 1$) and compares the differences between the algorithms theoretically. The derivation of the following results are presented in the appendix.

ICF (Asymptotically optimal with equal initial priors) (see Proposition 1 in Appendix)

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \\ &\quad \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \epsilon \sum_{i' \in \mathcal{N}_i} \frac{\mathbf{J}_{i'}^-}{N_C}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \qquad (3.40)\end{aligned}$$

$$\mathbf{J}_i^+ = N_C\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right) \qquad (3.41)$$

GKCF (Suboptimal) (see Proposition 5 in Appendix)

$$\hat{\mathbf{x}}_i^+ = \hat{\mathbf{x}}_i^- + \left(\mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i\right)^{-1} \left(\mathbf{b}_i - \mathbf{B}_i\hat{\mathbf{x}}_i^- + \epsilon \sum_{i' \in \mathcal{N}_i} \mathbf{J}_{i'}^- \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \tag{3.42}$$

$$\mathbf{J}_i^+ = \mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i \tag{3.43}$$

KCF (Suboptimal)

$$\hat{\mathbf{x}}_i^+ = \hat{\mathbf{x}}_i^- + \left(\mathbf{J}_i^- + \mathbf{B}_i\right)^{-1} \left(\mathbf{b}_i - \mathbf{B}_i\hat{\mathbf{x}}_i^-\right) + \frac{\epsilon}{1 + ||(\mathbf{J}_i^-)^{-1}||} (\mathbf{J}_i^-)^{-1} \sum_{i' \in \mathcal{N}_i} \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right) \tag{3.44}$$

$$\mathbf{J}_i^+ = \mathbf{J}_i^- + \mathbf{B}_i \tag{3.45}$$

The following points of comparison are important:

1) In GKCF and KCF, the terms $\mathbf{B}_i$ and $\mathbf{b}_i$ are the summation of the measurement information and the weighted measurements in the local neighborhood. This fusion mechanism, unlike average consensus, does not guarantee a proper global convergence. However, in ICF, $\mathcal{A}(\mathbf{U}_i)$ and $\mathcal{A}(\mathbf{u}_i)$ are used which guarantee convergence to the global average values.

2) In GKCF and KCF, $\mathbf{J}_i^-$ is used instead of $\frac{\mathbf{J}_i^-}{N_C}$ as in ICF. If the priors are uncorrelated, using $\mathbf{J}_i^-$ is appropriate for a single time step. But, as the nodes converge, which is the goal of consensus, the information becomes redundant at each node and thus dividing the prior information matrices by N_C is required to match the centralized solution.

3) In the information matrix update equation of the ICF, there is a multiplication factor of N_C. The total information in an estimate is the sum of the information matrices of the priors and the measurements. However, as the average consensus scheme gives us the *average* information in the priors and measurements, this should be multiplied by N_C to get the exact measure of the *total* information.

4) In the third term of Eqn. (3.44), KCF gives equal weight to all the neighbors' priors. In the presence naivety, this has detrimental effect as the information at different nodes are different and need to be weighted by their information matrices.

5) In Eqn. (3.40), ICF uses $\left(\frac{1}{N_C}\mathcal{A}(\mathbf{J}_i^-) + \mathcal{A}(\mathbf{U}_i)\right)^{-1}$ to normalize both the innovation from the measurements and the innovation from the state priors. Whereas, in Eqn. (3.44), the normalizing terms are not balanced because KCF uses $\left(\mathbf{J}_i^- + \mathbf{B}_i\right)^{-1}$ to normalize the measurement innovation and $(\mathbf{J}_i^-)^{-1}$ to normalize the innovation from the priors.

6) In Eqn. (3.44), the normalizing term $1 + ||(\mathbf{J}_i^-)^{-1}||$ is a design choice [17] to maintain stability of the algorithm. However, it does not guarantee optimality of KCF.

3.6 Experimental Evaluation

In this section, we evaluate the performance of the proposed ICF algorithm in a simulated environment and compare it with other methods: the Centralized Kalman Filter (CKF), the Kalman Consensus Filter (KCF) [17] and the Generalized Kalman Consensus Filter (GKCF) [11]. Comparison in a simulated environment allows an in-depth analysis of the proposed algorithms as parameters are varied to measure performance under different conditions.

We simulate a camera network containing N_T targets randomly moving (with a fixed model) within a 500×500 space. Each target's initial state vector is random. A set of N_C camera sensors monitor the space (we consider that each communication node consists of one camera). In each experiment, the cameras are randomly distributed in the space with random orientations resulting in overlapping field-of-views (FOVs).

Target State Parameters

Each target was initialized at the center of the simulation grid. The target's state vector was a $4D$ vector, with the $2D$ position and $2D$ velocity components. The initial speed was set to 2 units per time step and with a random direction uniformly chosen from 0 to 2π. The targets evolved for 40 time steps using the target dynamical model of Eqn. (3.29). Only the targets which remained in the simulation grid for the 40 time steps were considered. The process covariance $\mathbf{Q}$ is set to $diag(10, 10, 1, 1)$.

For the target state-estimation model, the dynamical model of Eqn. (3.29) was also used with the same $\mathbf{\Phi}$ and $\mathbf{Q}$ defined above. The initial prior state $\hat{\mathbf{x}}_i^-(1)$ and prior covariance $\mathbf{P}_i^-(1)$ is set equal at each node. A diagonal matrix is used for $\mathbf{P}_i^-(1)$ with the main diagonal elements as $\{100, 100, 10, 10\}$. The initial prior state $\hat{\mathbf{x}}_i^-(1)$ is generated by adding zero-mean Gaussian noise of covariance $\mathbf{P}_i^-(1)$ to the ground truth state.

Sensor Parameters

A total of $N_C = 15$ nodes were generated at uniformly chosen random locations on the simulation area. The measurement vector length for each sensor is $m_i = 2$. The FOV was chosen to be equilateral triangles. We define the sensing range, SR, for each sensor to be the height of this equilateral triangle. SR was chosen to be 300 units for all the sensors. A sensor can have an observation of a target only if the ground truth position of the target is within the sensor's FOV and in that case, a measurement $\mathbf{z}_i$ was generated using the linear observation model Eqn. (3.1) with noise covariance

$\mathbf{R}_i = 100\mathbf{I}_2$. The observation matrix $\mathbf{H}_i$ and state transition matrix $\mathbf{\Phi}$ is given below.

$$\mathbf{H}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \qquad \mathbf{\Phi} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Network Topology Parameters

A circulant graph was chosen as the communication topology for the sensor network to highlight the different issues associated with the sparsity of the communication network. The circulant graph is Δ-regular where the degree of each node is the same, say Δ. The adjacency matrix of a circulant graph is a circulant matrix. So, we denote Δ to be the degree of the communication graph $\mathcal{G}$. In the experiments, $\Delta = 2$ was used, unless stated otherwise. Note that, in this scenario, the maximum degree $\Delta_{max} = \Delta$ because the degree of all the nodes are same.

Consensus Parameters

At step 3 of Algorithm 3, each sensor communicates its measurement information to its neighbors iteratively. The maximum number of consensus iterations K was set to 5 unless stated otherwise. The consensus speed parameter was chosen to be $\epsilon = 0.65/\Delta_{max} = 0.65/\Delta$.

Experimental Description

The parameters that were varied in the experiments are, the maximum number of consensus iterations K, communication bandwidth μ, computation unit τ, degree of the communication network Δ, sensing range SR and the number of cameras N_C. For

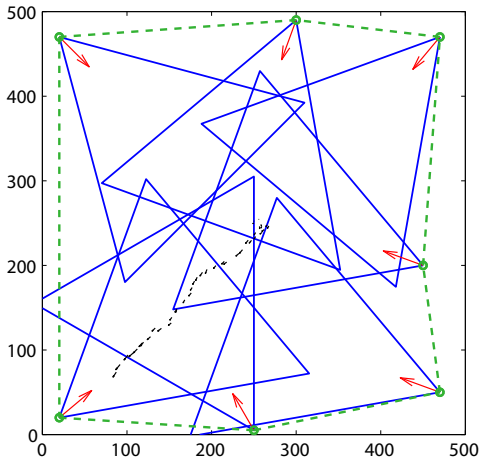


Figure 3.3: In this image of an example simulation environment, the red arrows indicate the locations and orientations of the cameras. The camera FOVs are shown in blue triangles. There are 7 cameras in this example. The green dotted lines represent the network connectivity. Each black arrows depict the actual trajectory of a target moving on the grid.

each experiment, only one parameter was varied while the others were kept constant. As a measure of performance, we computed the estimation error, e, defined as the Euclidean distance between the ground truth position and the estimated posterior position. An example of the simulation framework is shown in Fig. 3.3.

For each experiment, 20 different simulation environments differing in camera poses were randomly generated using the method discussed above. For each environment, 20 different randomly generated target tracks were used. Thus, for each experiment, the estimation errors e, were averaged over $20 \times 20 = 400$ random scenarios, 40 time steps and over all sensors N_C. The mean errors for different methods are shown in the following graphs as the results of different experiments. In the graphs, each line (of a unique color) corresponds to the mean error $\bar{e}$ for one estimation method.

The KCF algorithm as originally proposed in [17] uses a single consensus step per measurement update. To compare it with ICF, which supports multiple iterations,

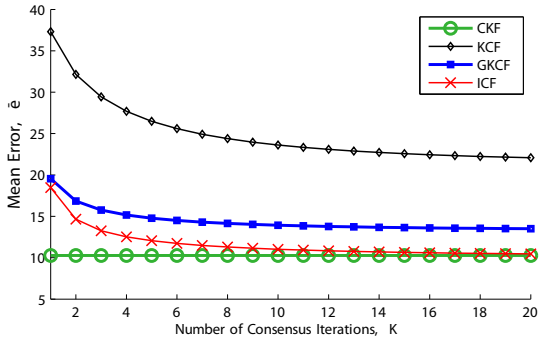


Figure 3.4: ICF performance comparison varying K (equal priors)

we extend KCF for multiple iterations. For this, at each time step, the measurement innovation component is set to zero for $k > 1$ and we consider only the new information provided by the neighboring nodes' estimates.

3.6.1 Experiment 1: Varying K

The objective of this experiment is to compare the performance of different estimation algorithms for different K. Here, K was varied from 1 to 20 at increments of 1. The other parameters were kept constant at their default values. The priors were chosen to be equal.

The results of this experiment are shown in Fig. 3.4. The graph shows that for $K = 1$, ICF performs much better than KCF and GKCF performs close to ICF. As, the number of iterations K is increased, the mean error for ICF decreases and approaches the performance of the centralized method at about $K = 10$. The main reason for this difference in performance is that KCF and GKCF do not account for the redundancy in the prior information across different nodes, but ICF does. In this simulation all the initial priors were equal, thus the information in the priors were completely redundant. KCF and GKCF by not taking into account redundancy in priors, gives more weight to prior information and less weight to the measurement information than the optimal

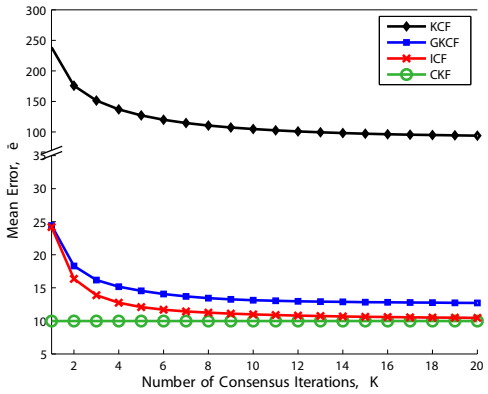


Figure 3.5: Uncorrelated Priors

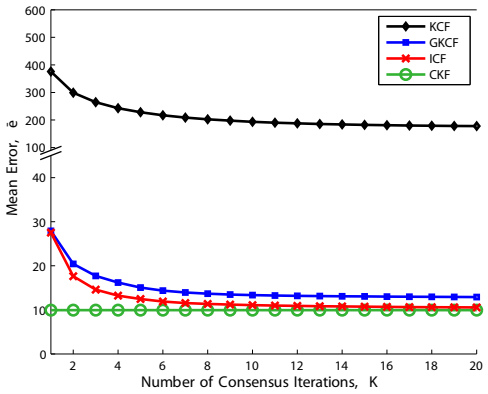


Figure 3.6: Correlated Priors ($\rho = .5$)

weights.

ICF is guaranteed to converge to the optimal centralized estimated if the initial priors are equal. However, to show the robustness of the ICF approach, we conducted the same experiment with unequal and uncorrelated priors (Fig. 3.5) and with unequal and correlated (with $\rho = 0.5$ correlation-coefficient between the priors across nodes) (Fig. 3.6) using Algorithm 3. The results show that ICF achieves near-optimal performance even when the optimality constraints are not met. This is because ICF is a consensus based approach and irrespective of the initial condition, after several time steps or consensus iterations, the priors converge. ICF was proved to be optimal with converged priors. Thus, after a few time steps it achieves near-optimal performance as the system approaches the optimality conditions.

3.6.2 Experiment 2: Varying μ and τ

The objective of this experiment is to compare the performance of different algorithms at different amounts of communication bandwidth and computational resources. We define the bandwidth, μ, of each method to be the total number of scalars sent at each consensus iteration from a node to each neighbor. As, covari-

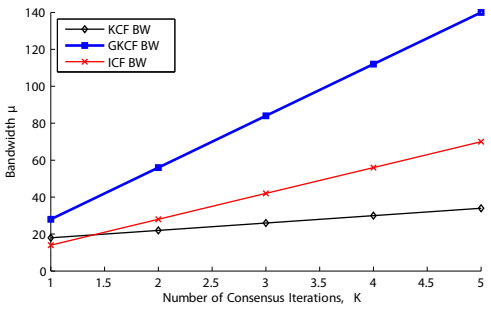


Figure 3.7: Bandwidth μ vs. number of consensus iterations K for different approaches.

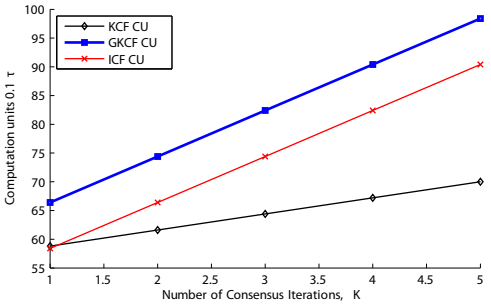


Figure 3.8: Computation unit τ vs. number of consensus iterations K for different approaches.

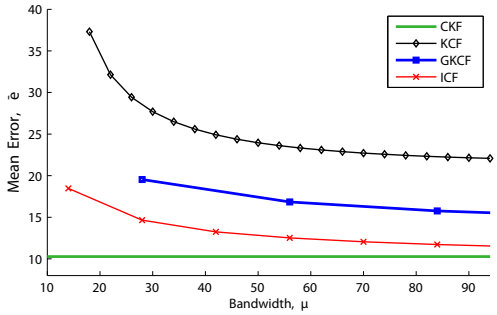


Figure 3.9: Performance comparison between different approaches as a function of the bandwidth μ, showing that the ICF performs better than other distributed approaches for each μ.

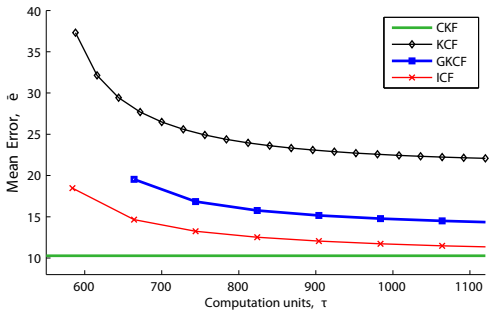


Figure 3.10: Performance comparison between different approaches as a function of the computational unit τ, showing that the ICF performs better than other distributed approaches at each τ.

ance/information matrices are symmetric, only sending the upper/lower triangular portion of the matrix suffices. Using standard convention, the approximate communication data requirement μ and computation cycle requirement τ was computed for $p = 4$ and $m_i = 2$.

Figs. 3.7 and 3.8 show the bandwidth and computational requirements for different algorithms for different numbers of consensus iterations K. For any given K,

ICF always requires half the bandwidth of GKCF. The KCF has lower communication and computational requirements.

Figs. 3.9 and 3.10 show the performance achieved by each method for different amounts of bandwidth and computational requirements. At any given bandwidth μ or computation unit τ, the ICF performs better than the other distributed methods and the performance approaches that of the CKF with increased resources.

3.6.3 Experiment 3: Varying Δ

The objective of this experiment is to compare the performance of different approaches for different values of the degree Δ (i.e., vary the sparsity of the network from sparse connectivity to dense connectivity). For $N_C = 15$, the maximum possible degree can be $\Delta = 14$ at full mesh connectivity. In this experiment, Δ was varied from 2 to 14 at increments of 2.

The results are shown in Fig 3.11, where the total number of consensus iterations K was set to 5. It can be seen that the ICF performs better than the other distributed algorithms at all Δ and almost approaches the centralized performance at $\Delta = 4$. For full connectivity, i.e. $\Delta = 14$, where $\mathcal{G}$ is a complete graph, all the distributed methods achieve centralized performance.

3.6.4 Experiment 4: Varying SR

The objective of this experiment is to compare the performance of different approaches as a function of the sensor range SR (i.e., varying the area of coverage of each sensor.)

Fig. 3.12 shows the performance of each approach as SR is varied and the

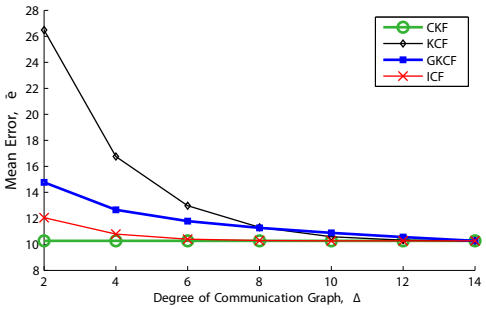


Figure 3.11: Performance comparison of different approaches as a function of the degree of communication graph, Δ. Increase in network connectivity increases the performance of all methods.

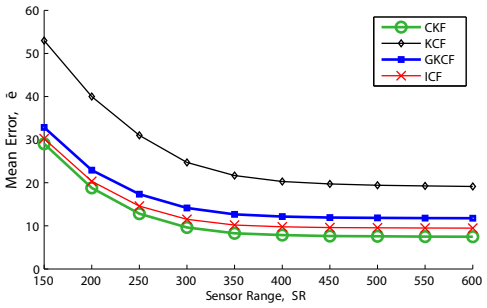


Figure 3.12: Performance comparison of different approaches by varying sensor range, SR. As the sensor range increases, the network gets more measurement information and has fewer naive nodes increasing overall performance of all methods (including centralized).

total number of iterations K was set to 5. It can be seen that ICF performed better than the other distributed algorithms at each sensor range.

3.6.5 Experiment 5: Varying N_C

The objective of this experiment is to compare the performance of different approaches as a function of the total number of sensors, N_C, as it was varied from 5 to 31 at increments of 2. Values less than 5 were not used because at such low number of sensors, even the centralized algorithm cannot perform well due to the high number of time instants at which the number of measurements is insufficient for the state vector to be observable, due to the low percentage coverage of the environment.

Fig. 3.13 shows results for variable N_C and fixed $K = 20$. As the number of sensors is increased, the observability of the targets by the network increases, but the number of naive nodes also increases. Due to the amount of information about

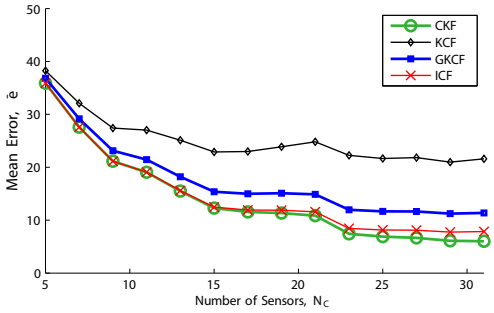


Figure 3.13: Performance comparison of different approaches by varying total number of sensors, N_C.

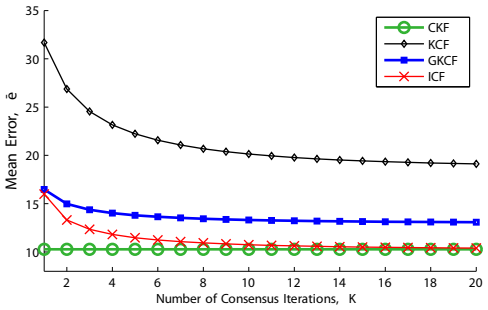


Figure 3.14: Performance comparison for imbalanced communication graph.

the targets increasing, the performance of all the approaches (including centralized) improves. For a small N_C, all the distributed methods performed close to the centralized method, because the number of naive nodes is small and the network distance from a naive node to an informed node is small. As the number of nodes increases, the number of consensus iterations required to reach consensus also increases. Thus we can see that the performance of ICF deviates from the centralized performance for high number of sensors. For all N_C, ICF outperformed the alternative distributed methods.

3.6.6 Experiment 6: Arbitrary Communication Graph

In the previous experiments, for the ease of varying different parameters, we assumed the communication graph to be balanced. To show that ICF is applicable for any connected graph, we conduct experiments on random connected graphs for $N_C = 15$. Edges were added between five random pairs of nodes on the balanced graph which was used on the previous examples. The result of this experiment is shown in Fig. 3.14. It depicts the fact that as $K \to \infty$, for any random connected graph, ICF converges to the centralized solution.

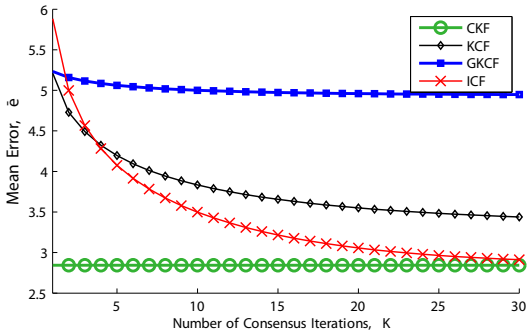


Figure 3.15: Performance comparison under unlimited observability.

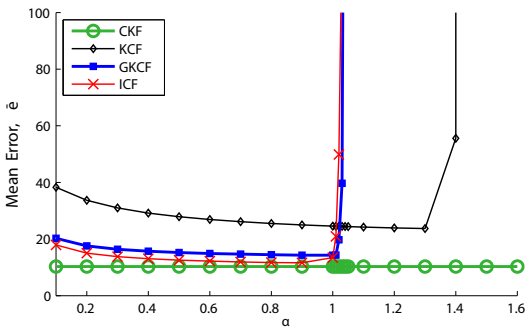


Figure 3.16: Stability analysis.

3.6.7 Experiment 7: Full Observability

Limited local observability (i.e., naive nodes) was one of the motivations for the derivation of the ICF algorithm. To show that ICF is generally applicable even in scenarios with without locally unobservable states, in this experiment, we assume that each node can observe all the targets at all times. From the results in Fig. 3.15 it can be seen that still ICF outperforms the other methods for most cases and achieves the optimal performance as $K \to \infty$.

3.6.8 Experiment 8: Stability Analysis

In this experiment, we experimentally verify the stability of the ICF approach. In Sec. 2.3, we have mentioned that the average consensus algorithm is stable if the consensus rate parameter ϵ is chosen between 0 and $\frac{1}{\Delta_{max}}$. The stability of the ICF algorithm is directly related to the stability of the average consensus algorithm. We set $\epsilon = \frac{\alpha}{\Delta_{max}}$ and vary α to perform the stability analysis. From the results in Fig. 3.16, it can be seen that ICF is stable for $0 < \alpha < 1$ (i.e., $0 < \epsilon < \frac{1}{\Delta_{max}}$), which is the same stability conditions for average consensus algorithm.

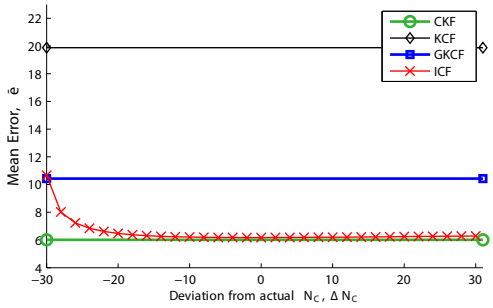


Figure 3.17: Robustness to inaccurate knowledge of N_C.

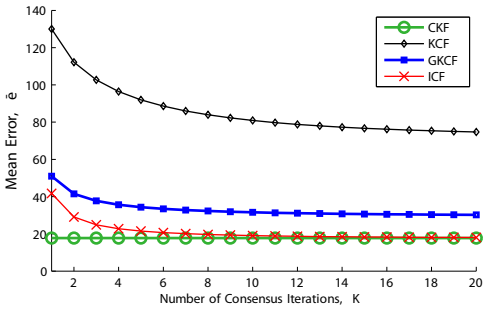


Figure 3.18: Robustness to model assumption error.

3.6.9 Experiment 9: Robustness to inaccurate knowledge of N_C

Unlike CKF, KCF and GKCF; ICF requires the knowledge of the total number of nodes N_C. Total number of nodes N_C, can be computed in a distributed framework [8]. In case of node failure or inclusion of new nodes to the system, each agent might have a wrong estimate of N_C, say $N_C + \Delta N_C$. In this experiment, to test the sensitivity of ICF to the error in the value of N_C, ΔN_C is varied from $-N_C + 1$ to N_C. The actual value of N_C for this experiment is 31. Total number of iterations, K was set to 100. The results are shown in Fig. 3.17. It shows that ICF is highly tolerant to discrepancies between the actual N_C and the estimated N_C and performs better than other methods for most cases.

3.6.10 Experiment 10: Robustness to non-linear state propagation

Finally, it is natural to ask whether the performance demonstrated in the previous experiments was highly dependent on the fact that the estimator incorporated the correct model.

In this final experiment, the ground truth tracks were generated for $t = 1$ to

40 using the following non-linear model of Eqn. (3.46).

$$\mathbf{x}(t) = 200e^{-\lambda(t-1)} \begin{bmatrix} \cos(\omega t) \\ \sin(\omega t) \\ -\lambda\cos(\omega t) - \omega\cos(\omega t) \\ -\lambda\cos(\omega t) + \omega\cos(\omega t) \end{bmatrix} + \begin{bmatrix} 250 \\ 250 \\ 0 \\ 0 \end{bmatrix} \tag{3.46}$$

A total of 20 tracks were generated by varying $\omega = \frac{\pi}{80}$ to $\frac{4\pi}{80}$ with increments of $\frac{\pi}{80}$ and $\lambda = 0.02$ to 0.1 with increments of 0.02. The rest of the simulation settings were kept unchanged (similar to that of Experiment 1). Thus, in the estimation step, the linear dynamical model of Eqn. (3.29) was used. Fig. 3.18 shows the performance of different algorithms for this simulation. Comparing with Fig. 3.4 it can be seen that the performance of all the algorithms deteriorated (including CKF) as one would expect. However, the relative performance of the different algorithms are similar. This shows that the ICF and the performance comparison results are quite robust, even when the assumed dynamical model does not match the true state propagation.

3.7 Conclusion

In this chapter we have presented a novel distributed state estimation framework called the Information-weighted Consensus Filter (ICF) which is generally applicable to almost any connected sensor network, converges to the optimal centralized estimate under reasonable conditions, does not require target hand-off protocols, and requires computation and communication resource similar to or less than alternative algorithms. The development of this algorithm was motivated by applications in camera networks wherein the observability properties of the state by the nodes is distinct across the nodes. We compared ICF with the state-of-the-art distributed estimation methods

such as KCF and GKCF, both theoretically and through simulations. Our simulation results show that ICF outperforms these methods. The results also indicate that ICF is robust to situations where optimality conditions are not met. In the next chapters we will present novel algorithms incorporating data association schemes to ICF to handle multiple targets.

Chapter 4

Multi-Target Information Consensus

4.1 Introduction

In this chapter, our goal is to design a distributed multi-target tracking scheme which is suited for sensors with limited field-of-view (FOV), like cameras. A distributed multi-target tracking problem can be divided into three sub-problems, namely, distributed information fusion, data association (measurement to track association) and dynamic state estimation.

The Kalman Consensus Filter (KCF) [17] is a popular distributed dynamic state estimation framework based on the average consensus algorithm. KCF works well in situations where each node gets a measurement of the target. However, we have discussed in the previous chapters that due the issue with naivety, the performance of KCF deteriorates when there are sensor with limited FOVs. In the previous chapter, the Information-weighted Consensus Filter (ICF) was proposed which addresses this issue with naivety and is also guaranteed to converge to the optimal centralized Kalman filter

estimate.

The above mentioned methods assume that there is a single target, or for multiple targets, the measurement-to-track association is provided. For a multi-target tracking problem, the data association and the tracking steps are highly inter-dependent. The performance of tracking will affect the performance of data association and vice-versa. Thus, an integrated distributed tracking and data association solution is required where the uncertainty from the tracker can be incorporated in the data association process and vice-versa. Among many single-sensor multi-target data association frameworks, the Multiple Hypothesis Tracking (MHT) [23] and the Joint Probabilistic Data Association Filter JPDAF [2] are two popular schemes. MHT usually achieves higher accuracy at the cost of high computational load. On the other hand, JPDAF achieves reasonable results at much lower computation cost. As distributed solutions are usually applied to low-power wireless sensor networks where the computational and communication power is limited, the JPDAF scheme will be utilized in the proposed distributed multi-target tracking framework.

The **main contribution** of this chapter is the tight integration of data association with state-of-the-art distributed single target tracking methods, taking special care of the issue of naivety, and demonstration of its performance in the case of a camera network. In Sec. 4.2 the problem formulation is provided, along with a review of different consensus-based estimation methods. In Sec. 4.3, the JPDAF approach is reviewed and extended to a multi-sensor framework. In Sec. 4.5, the Multi Target Information Consensus (MTIC) tracker is proposed. Finally, in Sec. 4.6, the proposed method is compared against others experimentally.

4.1.1 Related Work

The purely decentralized nature of the fusion algorithm differentiates it from the majority of multi-camera tracking approaches in the computer vision literature. For example, in [6], a centralized approach for tracking in a multi-camera setup was proposed where the cameras were distributed spatially over a large area. In [4], an efficient target hand-off scheme was proposed but no multi-camera information fusion was involved. However, in this article, we deal with the distributed multi-target tracking problem where there is no centralized server, the processing is distributed over all the camera nodes and no target hand-off strategy is required. Various methods for distributed multi-target tracking have been proposed in the sensor-networks literature. In [5], a solution to the distributed data association problem was proposed by means of the message passing algorithm based on graphical models in which iterative, parallel exchange of information among the nodes viewing the same target was required. However, in our proposed framework, no special communication pattern is assumed. In [21, 26, 28], the distributed multi-target tracking schemes did not account for naivety or the presence of cross-correlation between the estimates at different nodes. The method proposed herein is based on the properties of the ICF [13] that was proposed in the previous chapter, which deals with both these issues.

4.2 Problem Formulation

Consider a sensor network with N_C sensors. There are no specific assumptions on the overlap between the FOVs of the sensors. The communication in the network can be represented using an undirected connected graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$. The set $\mathcal{C} = \{C_1, C_2, \ldots, C_{N_C}\}$ contains the vertices of the graph and represents the sensor nodes.

The set $\mathcal{E}$ contains the edges of the graph which represents the available communication channels between different nodes. The set of nodes having direct communication channel with node C_i (sharing an edge with C_i) is represented by $\mathcal{N}_i$. There are N_T targets ($\{T^1, T^2, \ldots T^{N_T}\}$) in the area viewed by the sensors. It is assumed that N_C and N_T is known to each sensor.

The state of the j^{th} target is represented by the vector $\mathbf{x}^j \in \mathcal{R}^p$. For example, for a tracking application in a camera network, $\mathbf{x}^j$ might be a vector containing ground plane position and velocity components. The state dynamics of target T^j are modeled as

$$\mathbf{x}^j(t+1) = \mathbf{\Phi}\mathbf{x}^j(t) + \boldsymbol{\gamma}^j(t). \tag{4.1}$$

Here $\mathbf{\Phi} \in \mathcal{R}^{p\times p}$ is the state transition matrix and the process noise $\boldsymbol{\gamma}^j(t)$ is modeled as $\mathcal{N}(\mathbf{0}, \mathbf{Q}^j)$.

At time t, each sensor C_i, depending on its FOV and the location of the targets, gets $l_i(t)$ measurements denoted as $\{\mathbf{z}_i^n\}_{n=1}^{l_i(t)}$. The sensors do not know a priori, which measurement was generated from which target. Under the hypothesis that the observation $\mathbf{z}_i^n$ is generated from T^j, it is assumed that $\mathbf{z}_i^n$ was generated by the following observation model

$$\mathbf{z}_i^n = \mathbf{H}_i^j \mathbf{x}_i^j + \boldsymbol{\nu}_i^j. \tag{4.2}$$

Here, $\mathbf{H}_i^j \in \mathcal{R}^{m\times p}$ is the observation matrix for node C_i for T^j. The noise $\boldsymbol{\nu}_i^j \in \mathcal{R}^m$ is modeled as a zero mean Gaussian random variable with covariance $\mathbf{R}_i^j \in \mathcal{R}^{m\times m}$.

Each node also maintains a prior/predicted state estimate $\hat{\mathbf{x}}_i^{j-}(t)$ (and its covariance $\mathbf{P}_i^{j-}(t)$) for each target. The inverse of the state covariance matrix (information/precision matrix) will be denoted as $\mathbf{J}_i^j = (\mathbf{P}_i^j)^{-1}$. We assume that the initial prior state estimate and information matrix is available to each node for each target upon its

detection. *Our goal is to track each target at each node, i.e., find the state estimate for each target at each node by using the prior and measurement information available in the entire network in a distributed fashion.* A critical step in this process is association of measurements with targets, which is the topic of this chapter.

4.3 Joint Probabilistic Data Association Filter: Review

The KCF, GKCF and ICF algorithms discussed in the previous chapters assume that the data association (which measurement belongs to which target) is known. For a realistic multi-target state estimation problem, solving data association is itself a challenging problem even in the centralized case. Here we briefly review the Joint Probabilistic Data Association Filter (JPDAF) [2] algorithm which is the starting point of the proposed multi-sensor multi-target distributed tracking algorithm.

The JPDAF is a single sensor algorithm, thus the sensor index i is unnecessary and will be dropped. A double superscript is required for the hypothesis that measurement $\mathbf{z}^n$ is associated with target T^j. At time t, the measurement innovation $\tilde{\mathbf{z}}^{jn}$ and the innovation covariance $\mathbf{S}^j$ of measurement $\mathbf{z}^n$ for target T^j is computed as,

$$\tilde{\mathbf{z}}^{jn} = \mathbf{z}^n - \mathbf{H}^j \hat{\mathbf{x}}^{j-} \tag{4.3}$$

$$\mathbf{S}^j = \mathbf{H}^j \mathbf{P}^{j-} \mathbf{H}^{jT} + \mathbf{R}^j \tag{4.4}$$

The probability that T^j is the correct target to associate with $\mathbf{z}^n$ is β^{jn} and the probability that none of the measurements belong to T^j is β^{j0}. See [2] for details about computing these probabilities.

The Kalman gain $\mathbf{K}^j$, mean measurement $\mathbf{y}^j$ and mean measurement innova-

tion $\tilde{\mathbf{y}}^j$ for target T^j are defined as

$$\mathbf{K}^j = \mathbf{P}^{j-}\mathbf{H}^{jT}(\mathbf{S}^j)^{-1}, \tag{4.5}$$

$$\mathbf{y}^j = \sum_{n=1}^{l} \beta^{jn}\mathbf{z}^n, \tag{4.6}$$

$$\tilde{\mathbf{y}}^j = \sum_{n=1}^{l} \beta^{jn}\tilde{\mathbf{z}}^{jn} = \mathbf{y}^j - (1-\beta^{j0})\mathbf{H}^j\hat{\mathbf{x}}^{j-}. \tag{4.7}$$

The state and its covariance estimate for JPDAF is given as

$$\hat{\mathbf{x}}^{j+}(t) = \hat{\mathbf{x}}^{j-}(t) + \mathbf{K}^j\tilde{\mathbf{y}}^j \tag{4.8}$$

$$\mathbf{P}^{j+}(t) = \mathbf{P}^{j-}(t) - (1-\beta^{j0})\mathbf{K}^j\mathbf{S}^j\left(\mathbf{K}^j\right)^T + \mathbf{K}^j\tilde{\mathbf{P}}^j\left(\mathbf{K}^j\right)^T \tag{4.9}$$

where,

$$\tilde{\mathbf{P}}^j = \left(\sum_{n=1}^{l} \beta^{jn}\tilde{\mathbf{z}}^{jn}\left(\tilde{\mathbf{z}}^{jn}\right)^T\right) - \tilde{\mathbf{y}}^j\left(\tilde{\mathbf{y}}^j\right)^T. \tag{4.10}$$

4.4 Data Association: Information Form

In the following, we first express the JPDAF algorithm in the information form, from which we will extend it to the multiple sensor case. This will then be used in the next section to derive the distributed multi-target tracking algorithm.

The JPDAF estimation Eqns. (4.8-4.9) can be written in the information form as the following (see Appendix B):

$$\hat{\mathbf{x}}^{j+} = \left(\mathbf{J}^{j-} + \mathbf{U}^j\right)^{-1}\left(\mathbf{J}^{j-}\hat{\mathbf{x}}^{j-} + \mathbf{u}^j + \beta^{j0}\mathbf{U}^j\hat{\mathbf{x}}^{j-}\right) \tag{4.11}$$

$$\mathbf{J}^{j+} = \mathbf{J}^{j-} + \mathbf{G}^j \tag{4.12}$$

where,

$$\mathbf{G}^j = \mathbf{J}^{j-}\mathbf{K}^j\left(\left(\mathbf{C}^j\right)^{-1} - \mathbf{K}^{jT}\mathbf{J}^{j-}\mathbf{K}^j\right)^{-1}\mathbf{K}^{jT}\mathbf{J}^{j-} \tag{4.13}$$

$$\mathbf{C}^j = (1-\beta^{j0})\mathbf{S}^j - \tilde{\mathbf{P}}^j \tag{4.14}$$

$$\mathbf{u}^j = \mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{y}^j \quad \text{and} \quad \mathbf{U}^j = \mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j. \tag{4.15}$$

In Eqn. (4.11), $\mathbf{J}^{j-}\hat{\mathbf{x}}^{j-}$ is the weighted prior information and $\mathbf{u}^j + \beta^{j0}\mathbf{U}^j\hat{\mathbf{x}}^{j-}$ is the weighted measurement information (taking data association uncertainty β^{j0} into account). The sum of these two terms represents the total information available to us if we have a single sensor. To incorporate measurement information from an additional sensor, the weighted measurement information from that sensor has to be added to this summation. This is a property of estimators in the information form for combining measurements from multiple sensors, when noise in those measurements is uncorrelated with each other, which we assume in this work. In a similar fashion, the information matrices ($\mathbf{U}_i^j$ and $\mathbf{G}_i^j$) from additional sensors should also be added to the appropriate terms. This gives us the multi-sensor centralized estimate in the information form as the following:

$$\hat{\mathbf{x}}^{j+} = \left(\mathbf{J}^{j-} + \sum_{i=1}^{N_C}\mathbf{U}_i^j\right)^{-1}\left(\mathbf{J}^{j-}\hat{\mathbf{x}}^{j-} + \sum_{i=1}^{N_C}\left(\mathbf{u}_i^j + \beta_i^{j0}\mathbf{U}_i^j\hat{\mathbf{x}}^{j-}\right)\right), \tag{4.16}$$

$$\mathbf{J}^{j+} = \mathbf{J}^{j-} + \sum_{i=1}^{N_C}\mathbf{G}_i^j. \tag{4.17}$$

4.5 Multi-Target Information Consensus

Based on the data association results derived in the previous section and the ICF, we will now derive a distributed multi-target tracking algorithm. We shall call this as the Multi Target Information Consensus (MTIC) tracker. As we have multiple sensors, we will bring back the sensor index in the subscripts.

Now, in a distributed system, each node will have its own prior information $\{\hat{\mathbf{x}}_i^{j-}, \mathbf{J}_i^{j-}\}$. However, consensus guarantees that the information at all nodes converge to the same value. This is an important point that was utilized in the ICF framework and similarly, it will be utilized here. Assuming that consensus was reached at the

Algorithm 4 MTIC for target T^j at node C_i at time step t

Input: $\hat{\mathbf{x}}_i^{j-}(t)$, $\mathbf{J}_i^{j-}(t)$, $\mathbf{H}_i^j$, $\mathbf{R}_i^j$.

1) Get measurements: $\{\mathbf{z}_i^n\}_{n=1}^{l_i(t)}$

2) Compute $\mathbf{S}_i^j$, $\mathbf{y}_i^j$, β_i^{j0}, $\mathbf{K}_i^j$ and $\mathbf{C}_i^j$

3) Compute information vector and matrices:

$$\mathbf{u}_i^j \leftarrow \mathbf{H}_i^{jT}\mathbf{R}_i^{j\,-1}\mathbf{y}_i^j \tag{4.18}$$

$$\mathbf{U}_i^j \leftarrow \mathbf{H}_i^{jT}\mathbf{R}_i^{j\,-1}\mathbf{H}_i^j \tag{4.19}$$

$$\mathbf{G}_i^j \leftarrow \mathbf{J}_i^{j-}\mathbf{K}_i^j\left(\mathbf{C}_i^{j\,-1} - \mathbf{K}_i^{j\,T}\mathbf{J}_i^{j-}\mathbf{K}_i^j\right)^{-1}\mathbf{K}_i^{j\,T}\mathbf{J}_i^{j-} \tag{4.20}$$

4) Initialize consensus data

$$\mathbf{v}_i^j[0] \leftarrow \mathbf{u}_i^j + \left(\frac{\mathbf{J}_i^{j-}}{N_C} + \beta_i^{j0}\mathbf{U}_i^j\right)\hat{\mathbf{x}}_i^{j-} \tag{4.21}$$

$$\mathbf{V}_i^j[0] \leftarrow \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{U}_i^j \tag{4.22}$$

$$\mathbf{W}_i^j[0] \leftarrow \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{G}_i^j \tag{4.23}$$

5) Perform average consensus (Sec. 2.3) on $\mathbf{v}_i^j[0]$, $\mathbf{V}_i^j[0]$ and $\mathbf{W}_i^j[0]$ independently for K iterations.

6) Estimate:

$$\hat{\mathbf{x}}_i^{j+} \leftarrow \left(\mathbf{V}_i^j[K]\right)^{-1}\mathbf{v}_i^j[K] \tag{4.24}$$

$$\mathbf{J}_i^{j+} \leftarrow N_C\mathbf{W}_i^j[K] \tag{4.25}$$

7) Predict:

$$\hat{\mathbf{x}}_i^{j-}(t+1) \leftarrow \mathbf{\Phi}\hat{\mathbf{x}}_i^{j+}(t) \tag{4.26}$$

$$\mathbf{J}_i^{j-}(t+1) \leftarrow \left(\mathbf{\Phi}\left(\mathbf{J}_i^{j+}(t)\right)^{-1}\mathbf{\Phi}^T + \mathbf{Q}\right)^{-1} \tag{4.27}$$

Output: $\hat{\mathbf{x}}_i^{j+}(t)$, $\mathbf{J}_i^{j+}(t)$, $\hat{\mathbf{x}}_i^{j-}(t+1)$, $\mathbf{J}_i^{j-}(t+1)$.

previous time step, the prior information at each node will be equal, i.e., $\mathbf{J}_i^{j-} = \mathbf{J}_c^{j-}$ and $\hat{\mathbf{x}}_i^{j-} = \hat{\mathbf{x}}_c^{j-}\ \forall i, j$. From this, we can rewrite Eqns. (4.16) and (4.17) as follows:

$$\begin{aligned}\hat{\mathbf{x}}_i^{j+} &= \left(\sum_{i=1}^{N_C}\left(\frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{U}_i^j\right)\right)^{-1}\sum_{i=1}^{N_C}\left(\mathbf{u}_i^j + \left(\frac{\mathbf{J}_i^{j-}}{N_C} + \beta_i^{j0}\mathbf{U}_i^j\right)\hat{\mathbf{x}}_i^{j-}\right) \\ &= \left(\sum_{i=1}^{N_C}\mathbf{V}_i^j\right)^{-1}\sum_{i=1}^{N_C}\mathbf{v}_i^j = \left(\frac{\sum_{i=1}^{N_C}\mathbf{V}_i^j}{N_C}\right)^{-1}\frac{\sum_{i=1}^{N_C}\mathbf{v}_i^j}{N_C} \qquad (4.28)\end{aligned}$$

$$\mathbf{J}_i^{j+} = \sum_{i=1}^{N_C}\left(\frac{\mathbf{J}^{j-}}{N_C} + \mathbf{G}_i^j\right) = \sum_{i=1}^{N_C}\mathbf{W}_i^j = N_C\frac{\sum_{i=1}^{N_C}\mathbf{W}_i^j}{N_C} \qquad (4.29)$$

where,

$$\begin{aligned}&\mathbf{V}_i^j = \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{U}_i^j, \qquad \mathbf{W}_i^j = \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{G}_i^j \\ &\text{and} \qquad \mathbf{v}_i^j = \mathbf{u}_i^j + \left(\frac{\mathbf{J}_i^{j-}}{N_C} + \beta_i^{j0}\mathbf{U}_i^j\right)\hat{\mathbf{x}}_i^{j-} \qquad (4.30)\end{aligned}$$

The three averaging terms in Eqns. (4.28) and (4.29) can be computed in a distributed manner using the average consensus algorithm [18]. The algorithm is summarized in Algorithm 4.

Note that if a sensor does not get any measurement for T^j, i.e., $\beta_i^{j0} = 1$, $\mathbf{u}_i^j$, $\mathbf{U}_i^j$ and $\mathbf{G}_i^j$ are set to zero vectors and matrices (due to no measurement information content).

4.6 Experiments

In this section, first we evaluate the performance of the proposed MTIC algorithm in a simulated environment and compare it with other methods: JPDA-KCF, ICF with ground truth data association (ICF-GT) and centralized Kalman Filter with ground truth data association (CKF-GT). ICF-GT converges to CKF-GT in several iterations, thus ICF-GT will provide a performance bound for the other iterative approaches. Note that ICF-GT and CKF-GT requires the knowledge of the ground truth

data association, whereas MTIC and JPDA-KCF do not.

We simulate a camera network with $N_C = 15$ cameras monitoring an area containing $N_T = 3$ targets roaming randomly in a 500×500 area. Each camera has a rectangular FOV of 200×200 and they are randomly placed in such a way that together they cover the entire area. A circulant network topology with a degree of 2 (at each node) was chosen for the network connectivity. Each target was randomly initialized at a different location with random velocity. The target's state vector was a $4D$ vector, with the $2D$ position and $2D$ velocity components. The targets evolved for 40 time steps using the target dynamical model of Eqn. (4.1). The state transition matrix (used both in track generation and estimation) $\mathbf{\Phi}$ and process covariance $\mathbf{Q}$ were chosen as

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{Q} = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The initial prior covariance $\mathbf{P}_i^{j-}(1) = diag(100, 100, 10, 10)$ was used at each node for each target. The initial prior state $\hat{\mathbf{x}}_i^{j-}(1)$ was generated by adding zero-mean Gaussian noise of covariance $\mathbf{P}_i^{-}(1)$ to initial the ground truth state. The observations were generated using Eqn. (4.2). The observation matrix $\mathbf{H}_i^j$ was set as

$$\mathbf{H}_i^j = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

If the ground truth state was within the FOV of a sensor, a measurement was generated from the ground truth track using the measurement model Eqn. (4.2) with $\mathbf{R}_i = 100\mathbf{I}_2$. The consensus rate parameter ϵ was set to $0.65/\Delta_{max}$ where $\Delta_{max} = 2$, as each node was connected to two other nodes. Total number of consensus iterations per measurement step, K, was set to 20. The parameters for computing the association probabilities,

β_i^{jn}'s, were set as follows (see [2] for details). False measurements (clutter) were generated at each node at each measurement step using a Poisson process with $\lambda = \frac{1}{32}$. Here, λ is the average number of false measurements per sensor per measurement step. Gate probability P_G was set to 0.99. The probability of detecting a target in each camera, P_D was computed by integrating the probability density function of the predicted measurement, (i.e., $\mathcal{N}(\mathbf{H}_i^j \hat{\mathbf{x}}_i^{j-}, \mathbf{S}_i^j)$) over the area visible to the camera.

To measure the performance of different approaches, one of the parameters was varied while keeping the others to their aforementioned values. As a measure of performance, we computed the estimation error, e, defined as the Euclidean distance between the ground truth position and the estimated posterior position. The simulation results were averaged over multiple simulation runs with 100 randomly generated sets of tracks. The mean (μ_e) of the errors for different methods are shown in the following graphs as the results of different experiments.

First, the amount of clutter was varied and the results are shown in Fig. 4.1. The average amount of clutter per sensor per measurement step, λ, was varied from $\frac{1}{256}$ to 8. From the figure it can be seen that both MTIC and JPDA-KCF is very robust even to a very high amount of clutter. The amount of clutter was kept at $\lambda = \frac{1}{32}$ for the other experiments.

Fig. 4.2a shows the performance of different approaches where the proximity of the tracks were varied. The proximity was defined in terms of average number of overlaps across all pairs of tracks present in a simulation run. Two tracks were assumed to be overlapping if the Euclidean distance between their ground truth states was below 50 units at the same time step. From Fig. 4.2a, it can be seen that as the overlap increases, the performance of different approaches deteriorated. However, MTIC performs better than JPDA-KCF. It can be seen that for a high overlap, the error did not increase.

This is mainly due to the reason that most of the tracks with high overlap were close to each other after they separated. Thus, although the data association failed, the tracking error did not grow much.

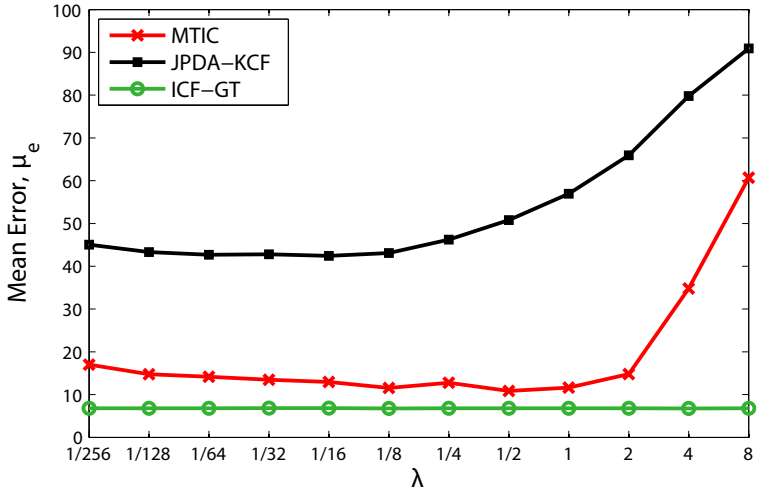
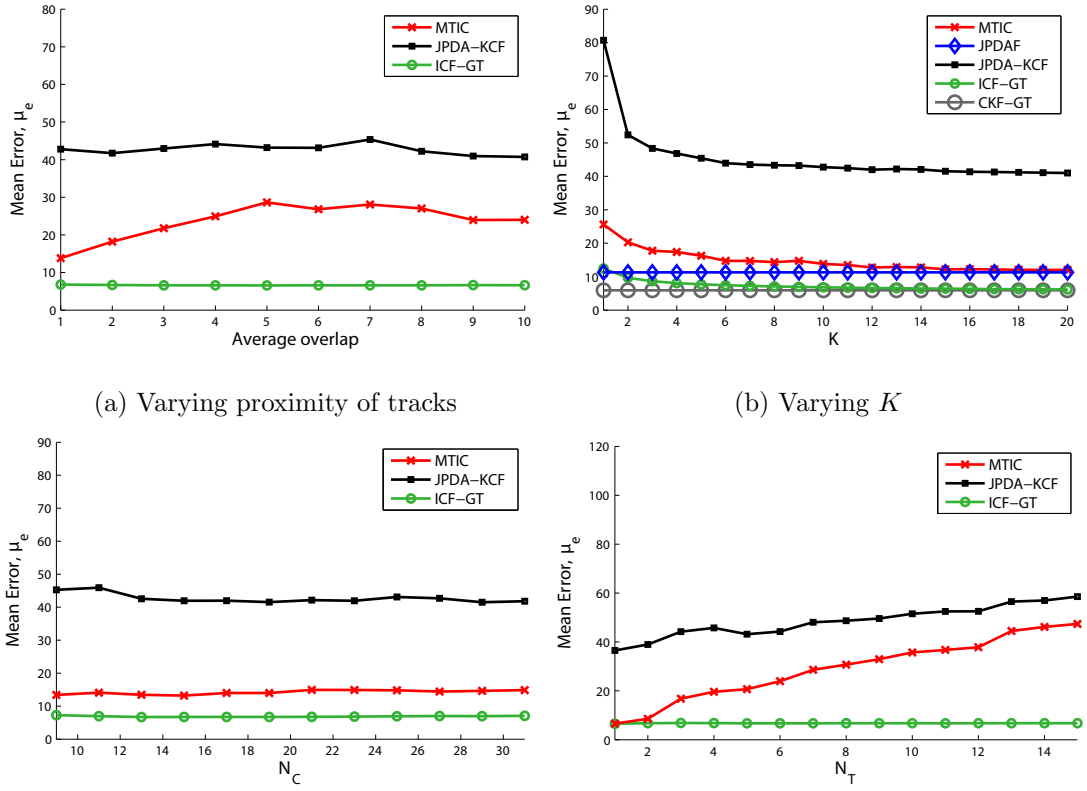


Figure 4.1: Performance comparison by varying amount of clutter.

To show the convergence of the different methods, the total number of iterations per measurement step, K was varied. It can be seen from Fig. 4.2b that with an increased number of iteration, ICF-GT approached the centralized method CKF-GT. It can also be seen that MTIC outperforms JPDA-KCF for any given K.

Next, the total number of sensors N_C and total number of targets N_T were varied and the results are shown in Figs. 4.2c and 4.2d. With more sensors, the total number of available measurements increases which should increase estimation performance. However, with an increase in the number of sensors, the total number of false measurements also increases which can adversely affect the performance. Due to these two contradictory issues, the performance remained almost constant with different number of sensors. With the increase in the number of targets, the problem of data association became more challenging which had an adverse effect in the performance of the

different algorithms as can be seen in Fig. 4.2d.



(a) Varying proximity of tracks

(b) Varying K

(c) Varying N_C

(d) Varying N_T

Figure 4.2: MTIC Performance comparison by varying different parameters.

4.7 Conclusion

In this chapter, we have proposed the Multi Target Information Consensus (MTIC) algorithm, which is a generalized consensus-based distributed multi-target tracking scheme applicable to a wide-variety of sensor networks. MTIC handles the issues with naivety which makes it applicable to sensor networks where the sensors may have limited FOV (which is the case for a camera network). The estimation errors in tracking and data association, as well as the effect of naivety, are integrated into a single efficient algorithm. This makes MTIC very robust to false measurements/clutter. Experimental

analysis shows the strength of the proposed method over existing ones. In the next chapter we will extend this algorithm to handle non-linearity in the observation model.

Chapter 5

Information-weighted Consensus with non-linear models

5.1 Introduction

In the previous chapters, we assumed a linear relationship between the observation and the target state. In many application scenarios, the observation is a non-linear function of the target state. This is especially true for cameras because in the perspective camera observation model, the position of a point in the camera's pixel coordinate system is non-linearly related to the position of the point in the world coordinate system.

Thus, for the algorithms derived in the previous sections to be applicable in such scenarios, we need to re-derive them to support non-linearity in the observation model. In this chapter, we propose non-linear extensions to the ICF and the MTIC algorithms proposed in the previous chapters. We call these algorithms, the Extended ICF (EICF) and the Extended MTIC (EMTIC). We will also show comparisons of these in a simulated framework along with real-life experiments.

5.2 Camera Observation Model (Non-linear)

Let us denote the orientation and position of the camera C_i w.r.t. the world coordinate system as ${}^{i}_{w}\mathbf{R}$ and ${}^{i}_{w}\mathbf{p}$. The number of pixels per unit length is k and f_i is the focal length of C_i. The center of the image plane in the pixel coordinate system is $\{o_x, o_y\}$. Say, in world and camera coordinate the system, the position of a point j is ${}^{j}_{w}\mathbf{p} = [x_w, y_w, z_w]^T$ and ${}^{j}_{i}\mathbf{p} = [x_i, y_i, z_i]^T$ respectively. Note that for ground plane tracking, $z_w = 0$. The target state can be $\mathbf{x}^j = [x_w, y_w, u_w, v_w]^T$ i.e., it may consist of the position and velocity components of a target in the ground plane. Now, ${}^{j}_{w}\mathbf{p}$ and $\mathbf{x}^j$ are related as,

$$ {}^{j}_{w}\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}^j \tag{5.1} $$

From ${}^{j}_{w}\mathbf{p}$, we can compute ${}^{j}_{i}\mathbf{p}$ as,

$$ {}^{j}_{i}\mathbf{p} = {}^{i}_{w}\mathbf{R}({}^{j}_{w}\mathbf{p} - {}^{i}_{w}\mathbf{p}) \tag{5.2} $$

Finally, from From ${}^{j}_{i}\mathbf{p}$ the position of that point in C_i's pixel coordinate system $\mathbf{z}^j_i$ can be written using the camera perspective projection model as,

$$ \mathbf{z}^j_i = \begin{bmatrix} kf_i \dfrac{x_i}{z_i} + o_x \\ kf_i \dfrac{y_i}{z_i} + o_y \end{bmatrix} \tag{5.3} $$

Using the above three equations, we can see that $\mathbf{z}^j_i$ is a non-linear function of $\mathbf{x}^j$ which can be represented as $\mathbf{z}^j_i = \mathbf{h}_i(\mathbf{x}^j)$. Using first-order Taylor series approximation, the linearized observation matrix can be written as,

$$ \mathbf{H}^j_i = \nabla_{\mathbf{x^j}} \mathbf{h}_i(\mathbf{x}^j)|_{\mathbf{x}^j = \hat{\mathbf{x}}^{j-}_i} \tag{5.4} $$

Next, we will extend the ICF and MTIC algorithm for non-linear observation models where both $\mathbf{H}^j_i$ and $\mathbf{h}_i()$ will be utilized.

5.3 Extended Information-weighted Consensus Filter

In this section, we will derive the Extended Information Consensus Filter (EICF) which we will prove to converge to the centralized Extended Kalman Filter (EKF) results. Considering the data association is given, the measurement in each sensor can be expressed using the non-linear relation as,

$$\mathbf{z}_i = \mathbf{h}_i(\mathbf{x}) + \boldsymbol{\nu}_i. \tag{5.5}$$

The collection of all measurements from all sensors can be expressed as,

$$\boldsymbol{\mathcal{Z}} = \mathbf{h}_c(\mathbf{x}) + \boldsymbol{\nu}. \tag{5.6}$$

Here, $\boldsymbol{\mathcal{Z}} = [\mathbf{z}_1^T, \mathbf{z}_2^T, \dots \mathbf{z}_N^T]^T \in \mathcal{R}^m$ is the concatenation of all measurements in the network where $m = \sum_{i=1}^{N} m_i$. The central non-linear observation function $\mathbf{h}_c()$ is a stack of all the functions from individual sensors such that $\mathbf{h}_c(\mathbf{x}) = \{\mathbf{h}_1(\mathbf{x})^T, \mathbf{h}_2(\mathbf{x})^T, \dots \mathbf{h}_{N_C}(\mathbf{x})^T\}^T$. We will represent the stack of linearized observation matrices as $\boldsymbol{\mathcal{H}} = [\mathbf{H}_1^T, \mathbf{H}_2^T, \dots \mathbf{H}_N^T]^T \in \mathcal{R}^{m \times p}$. For the measurement noise vector, $\boldsymbol{\nu} = [\boldsymbol{\nu}_1^T, \boldsymbol{\nu}_2^T, \dots \boldsymbol{\nu}_N^T]^T \in \mathcal{R}^m$, we denote its covariance as $\boldsymbol{\mathcal{R}} \in \mathcal{R}^{m \times m}$. We assume the measurement noise to be uncorrelated across nodes. Thus, the measurement covariance matrix is $\boldsymbol{\mathcal{R}} = diag(\mathbf{R}_1, \mathbf{R}_2, \dots \mathbf{R}_N)$. Let us denote $\mathbf{U}_c = \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{H}}$ and $\mathbf{u}_c = \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{Z}}$. The subscript c stands for "centralized". Let us also denote the centralized predicted measurement, $\mathbf{h}_c(\hat{\mathbf{x}}_c^-) = \{\mathbf{h}_1(\hat{\mathbf{x}}_c^-)^T, \mathbf{h}_2(\hat{\mathbf{x}}_c^-)^T, \dots \mathbf{h}_{N_C}(\hat{\mathbf{x}}_c^-)^T\}^T$.

The state estimation equations from the centralized Extended Kalman Filter (EKF) algorithm can be written as (see Appendix A),

$$\hat{\mathbf{x}}_c^+ = \left(\mathbf{J}_c^- + \mathbf{U}_c\right)^{-1} \left(\mathbf{J}_c^- \hat{\mathbf{x}}_c^- + \mathbf{u}_c + \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} (\boldsymbol{\mathcal{H}} \hat{\mathbf{x}}_c^- - \mathbf{h}_c(\hat{\mathbf{x}}_c^-))\right) \tag{5.7}$$

$$\mathbf{J}_c^+ = \left(\mathbf{J}_c^- + \mathbf{U}_c\right) \tag{5.8}$$

Given that all the nodes have reached consensus on the previous time step, we have, $\hat{\mathbf{x}}_i^- = \hat{\mathbf{x}}_c^-$ and $\mathbf{J}_i^- = \mathbf{J}_c^-$ for all i. This implies, $\mathbf{J}_c^- = \sum_{i=1}^{N_C} \frac{\mathbf{J}_i^-}{N_C}$ and $\mathbf{J}_c^- \hat{\mathbf{x}}_c^- = \sum_{i=1}^{N_C} \frac{\mathbf{J}_i^-}{N_C} \hat{\mathbf{x}}_i^-$. Also, as $\mathbf{R}_c$ is block diagonal, we have, $\mathbf{U}_c = \sum_{i=1}^{N_C} \mathbf{U}_i$, $\mathbf{u}_c = \sum_{i=1}^{N_C} \mathbf{u}_i$ and $\boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} (\boldsymbol{\mathcal{H}} \hat{\mathbf{x}}_c^- - \mathbf{h}_c(\hat{\mathbf{x}}_c^-)) = \sum_{i=1}^{N_C} \mathbf{H}_i^T \mathbf{R}_i^{-1} (\mathbf{H}_i \hat{\mathbf{x}}_i^- - \mathbf{h}_i(\hat{\mathbf{x}}_i^-))$.

Thus, from Eqns. (5.7) and (5.8) we get,

$$\hat{\mathbf{x}}_c^+ = \left(\sum_{i=1}^{N_C} \left(\frac{\mathbf{J}_i^-}{N_C} + \mathbf{U}_i \right) \right)^{-1} \sum_{i=1}^{N_C} \left(\mathbf{J}_i^- \hat{\mathbf{x}}_i^- + \mathbf{u}_i + \mathbf{H}_i^T \mathbf{R}_i^{-1} \left(\mathbf{H}_i \hat{\mathbf{x}}_i{}^- - \mathbf{h}_i(\hat{\mathbf{x}}_i^-) \right) \right) \quad (5.9)$$

$$\mathbf{J}_c^+ = \sum_{i=1}^{N_C} \left(\frac{\mathbf{J}_i^-}{N_C} + \mathbf{U}_i \right) \quad (5.10)$$

We will call the above equations the Extended Joint Probabilistic Data Association Filter (EJPDAF). This can be computed in a distributed manner by initializing $\mathbf{v}_i[0]$ and $\mathbf{V}_i[0]$ as the following at each node and running average consensus algorithm on them.

$$\mathbf{v}_i[0] = \frac{\mathbf{J}_i^-}{N_C} \hat{\mathbf{x}}_i^- + \mathbf{u}_i + \mathbf{H}_i^T \mathbf{R}_i^{-1} (\mathbf{H}_i \hat{\mathbf{x}}_i^- - \mathbf{h}_i(\hat{\mathbf{x}}_i^-)) \quad (5.11)$$

$$\mathbf{V}_i[0] = \frac{\mathbf{J}_i^-}{N_C} + \mathbf{U}_i \quad (5.12)$$

Note that for linear observation model, $\mathbf{h}_i(\hat{\mathbf{x}}_i^-) = \mathbf{H}_i \hat{\mathbf{x}}_i^-$. Using this relation, the EICF equations reduce to the original ICF (derived for linear model) equations as the following,

$$\mathbf{v}_i[0] = \frac{\mathbf{J}_i^-}{N_C} \hat{\mathbf{x}}_i^- + \mathbf{u}_i \quad (5.13)$$

$$\mathbf{V}_i[0] = \frac{\mathbf{J}_i^-}{N_C} + \mathbf{U}_i \quad (5.14)$$

Algorithm 5 EICF at node C_i at time step t

Input: prior state estimate $\hat{\mathbf{x}}_i^-(t)$, prior information matrix $\mathbf{J}_i^-(t)$, consensus speed factor ϵ and total number of consensus iterations K.

1) Linearize $\mathbf{h}_i$ at $\hat{\mathbf{x}}_i^-(t)$ to compute $\mathbf{H}_i$

2) Get measurement $\mathbf{z}_i$ with covariance $\mathbf{R}_i$

3) Compute consensus proposals,

$$\mathbf{V}_i[0] \leftarrow \frac{1}{N_C}\mathbf{J}_i^-(t) + \mathbf{U}_i \tag{5.15}$$

$$\mathbf{v}_i[0] \leftarrow \frac{1}{N_C}\mathbf{J}_i^-(t)\hat{\mathbf{x}}_i^-(t) + \mathbf{u}_i + \mathbf{H}_i^T\mathbf{R}_i^{-1}(\mathbf{H}_i\hat{\mathbf{x}}_i^- - \mathbf{h}_i(\hat{\mathbf{x}}_i^-)) \tag{5.16}$$

4) Perform average consensus (Sec. 2.3) on $\mathbf{v}_i[0]$, $\mathbf{V}_i[0]$ independently for K iterations.

5) Compute a posteriori state estimate and information matrix for time t

$$\hat{\mathbf{x}}_i^+(t) \leftarrow (\mathbf{V}_i[K])^{-1}\mathbf{v}_i[K] \tag{5.17}$$

$$\mathbf{J}_i^+(t) \leftarrow N_C\mathbf{V}_i[K] \tag{5.18}$$

6) Predict for next time step $(t+1)$

$$\mathbf{J}_i^-(t+1) \leftarrow \left(\mathbf{\Phi}(\mathbf{J}_i^+(t))^{-1}\mathbf{\Phi}^T + \mathbf{Q}\right)^{-1} \tag{5.19}$$

$$\hat{\mathbf{x}}_i^-(t+1) \leftarrow \mathbf{\Phi}\hat{\mathbf{x}}_i^+(t) \tag{5.20}$$

Output: EICF estimate $\hat{\mathbf{x}}_i^+(t)$, $\mathbf{J}_i^+(t)$, $\hat{\mathbf{x}}_i^-(t+1)$, $\mathbf{J}_i^-(t+1)$.

The EICF algorithm is summarized in Algorithm 5. To highlight the similar type of difference between ICF and EICF; the centralized Kalman filter and EKF estimate (see appendix) in information form is given in below,

Kalman filter:

$$\hat{\mathbf{x}}_c^+ = \left(\mathbf{J}_c^- + \mathbf{U}_c\right)^{-1}\left(\mathbf{J}_c^-\hat{\mathbf{x}}_c^- + \mathbf{u}_c\right) \tag{5.21}$$

EKF:

$$\hat{\mathbf{x}}_c^+ = \left(\mathbf{J}_c^- + \mathbf{U}_c\right)^{-1}\left(\mathbf{J}_c^-\hat{\mathbf{x}}_c^- + \mathbf{u}_c + \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\left(\boldsymbol{\mathcal{H}}\hat{\mathbf{x}}_c^- - \mathbf{h}_c\left(\hat{\mathbf{x}}_c^-\right)\right)\right) \tag{5.22}$$

Note that the EKF has the additional term $\boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\left(\boldsymbol{\mathcal{H}}\hat{\mathbf{x}}_c^- - \mathbf{h}_c\left(\hat{\mathbf{x}}_c^-\right)\right)$ which is due to the non-linearity in the observation model. The same difference is observed in ICF Eqn. (3.33) and EICF Eqn. (5.11).

5.4 Extended Multi-target Information Consensus

In this section, we will extend the MTIC algorithm to handle non-linear sensing models. For the linear case, the measurement innovation in the JPDAF algorithm is as the following,

$$\tilde{\mathbf{y}}^j = \mathbf{y}^j - (1-\beta^{j0})\mathbf{H}^j\hat{\mathbf{x}}^{j-} \tag{5.23}$$

However, for the non-linear sensing model, $\mathbf{h}()$, the measurement innovation term becomes,

$$\tilde{\mathbf{y}}^j = \mathbf{y}^j - (1-\beta^{j0})\mathbf{h}(\hat{\mathbf{x}}^{j-}) \tag{5.24}$$

Using this, the state estimation equations can be written as the following (see Appendix C)

$$\begin{aligned}\hat{\mathbf{x}}_c^{j+} &= \left(\mathbf{J}_c^{j-} + \sum_{i=1}^{N_C}\mathbf{U}_i^j\right)^{-1} \\ &\quad \left(\mathbf{J}_i^{j-}\hat{\mathbf{x}}_i^{j-} + \sum_{i=1}^{N_C}\left(\mathbf{u}_i^j + \mathbf{H}_i^{jT}\mathbf{R}_i^{j-1}(\mathbf{H}_i^j\hat{\mathbf{x}}_i^{j-} - (1-\beta_i^{j0})\mathbf{h}_i(\hat{\mathbf{x}}_i^{j-}))\right)\right) \qquad (5.25)\\ \mathbf{J}_c^{j+} &= \mathbf{J}_i^{j-} + \sum_{i=1}^{N_C}\mathbf{G}_i^j \qquad (5.26)\end{aligned}$$

Given that all the nodes have reached consensus on the previous time step, we have, $\hat{\mathbf{x}}_i^{j-} = \hat{\mathbf{x}}_c^{j-}$ and $\mathbf{J}_i^{j-} = \mathbf{J}_c^{j-}$ for all i. This implies, $\mathbf{J}_c^{j-} = \sum_{i=1}^{N_C} \frac{\mathbf{J}_i^{j-}}{N_C}$ and $\mathbf{J}_c^{j-}\hat{\mathbf{x}}_c^{j-} = \sum_{i=1}^{N_C} \frac{\mathbf{J}_i^{j-}}{N_C}\hat{\mathbf{x}}_i^{j-}$. Thus we can write,

$$\begin{aligned}
\hat{\mathbf{x}}_c^{j+} &= \left(\sum_{i=1}^{N_C}\left(\frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{U}_i^j\right)\right)^{-1} \\
&\quad \sum_{i=1}^{N_C}\left(\frac{\mathbf{J}_i^{j-}}{N_C}\hat{\mathbf{x}}_i^{j-} + \mathbf{u}_i^j + {\mathbf{H}_i^j}^T {\mathbf{R}_i^j}^{-1}(\mathbf{H}_i^j\hat{\mathbf{x}}_i^{j-} - (1-\beta_i^{j0})\mathbf{h}_i(\hat{\mathbf{x}}_i^{j-}))\right) \quad (5.27)\\
\mathbf{J}_c^{j+} &= \sum_{i=1}^{N_C}\left(\frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{G}_i^j\right) \quad (5.28)
\end{aligned}$$

Thus, for EMTIC, the consensus variables are initialized as,

$$\begin{aligned}
\mathbf{v}_i^j[0] &= \frac{\mathbf{J}_i^{j-}}{N_C}\hat{\mathbf{x}}_i^{j-} + \mathbf{u}_i^j + {\mathbf{H}_i^j}^T {\mathbf{R}_i^j}^{-1}\left(\mathbf{H}_i^j\hat{\mathbf{x}}_i^{j-} - (1-\beta_i^{j0})\mathbf{h}_i(\hat{\mathbf{x}}_i^{j-})\right) \quad (5.29)\\
\mathbf{V}_i^j[0] &= \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{U}_i^j \quad (5.30)\\
\mathbf{W}_i^j[0] &= \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{G}_i^j \quad (5.31)
\end{aligned}$$

Note that for linear observation model, $\mathbf{h}_i(\hat{\mathbf{x}}_i^{j-}) = \mathbf{H}_i^j\hat{\mathbf{x}}_i^{j-}$. Using this relation, the non-linear MTIC equations become the original MTIC (derived for linear model) equations,

$$\begin{aligned}
\mathbf{v}_i^j[0] &= \frac{\mathbf{J}_i^{j-}}{N_C}\hat{\mathbf{x}}_i^{j-} + \mathbf{u}_i^j + \beta_i^{j0}\mathbf{U}_i^j\hat{\mathbf{x}}_i^{j-} \quad (5.32)\\
\mathbf{V}_i^j[0] &= \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{U}_i^j \quad (5.33)\\
\mathbf{W}_i^j[0] &= \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{G}_i^j \quad (5.34)
\end{aligned}$$

The EMTIC algorithm is summarized is Algorithm 6.

Algorithm 6 EMTIC for target T^j at node C_i at time step t

Input: $\hat{\mathbf{x}}_i^{j-}(t)$, $\mathbf{J}_i^{j-}(t)$, $\mathbf{h}_i$, $\mathbf{R}_i^j$.

1) Linearize $\mathbf{h}_i$ at $\hat{\mathbf{x}}_i^{j-}(t)$ to compute $\mathbf{H}_i^j$

2) Get measurements: $\{\mathbf{z}_i^n\}_{n=1}^{l_i(t)}$

3) Compute $\mathbf{S}_i^j$, $\mathbf{y}_i^j$, β_i^{j0}, $\mathbf{K}_i^j$ and $\mathbf{C}_i^j$

4) Compute information vector and matrices:

$$\mathbf{u}_i^j \leftarrow \mathbf{H}_i^{jT}\mathbf{R}_i^{j\,-1}\mathbf{y}_i^j \tag{5.35}$$

$$\mathbf{U}_i^j \leftarrow \mathbf{H}_i^{jT}\mathbf{R}_i^{j\,-1}\mathbf{H}_i^j \tag{5.36}$$

$$\mathbf{G}_i^j \leftarrow \mathbf{J}_i^{j-}\mathbf{K}_i^j\left(\mathbf{C}_i^{j\,-1} - \mathbf{K}_i^{j\,T}\mathbf{J}_i^{j-}\mathbf{K}_i^j\right)^{-1}\mathbf{K}_i^{j\,T}\mathbf{J}_i^{j-} \tag{5.37}$$

5) Initialize consensus data

$$\mathbf{v}_i^j[0] \leftarrow \frac{\mathbf{J}_i^{j-}}{N_C}\hat{\mathbf{x}}_i^{j-} + \mathbf{u}_i^j + \mathbf{H}_i^{j\,T}\mathbf{R}_i^{j\,-1}\left(\mathbf{H}_i^j\hat{\mathbf{x}}_i^{j-} - (1-\beta_i^{j0})\mathbf{h}_i(\hat{\mathbf{x}}_i^{j-})\right) \tag{5.38}$$

$$\mathbf{V}_i^j[0] \leftarrow \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{U}_i^j \tag{5.39}$$

$$\mathbf{W}_i^j[0] \leftarrow \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{G}_i^j \tag{5.40}$$

6) Perform average consensus (Sec. 2.3) on $\mathbf{v}_i^j[0]$, $\mathbf{V}_i^j[0]$ and $\mathbf{W}_i^j[0]$ independently for K iterations.

7) Estimate:

$$\hat{\mathbf{x}}_i^{j+} \leftarrow \left(\mathbf{V}_i^j[K]\right)^{-1}\mathbf{v}_i^j[K] \tag{5.41}$$

$$\mathbf{J}_i^{j+} \leftarrow N_C\mathbf{W}_i^j[K] \tag{5.42}$$

8) Predict:

$$\hat{\mathbf{x}}_i^{j-}(t+1) \leftarrow \mathbf{\Phi}\hat{\mathbf{x}}_i^{j+}(t) \tag{5.43}$$

$$\mathbf{J}_i^{j-}(t+1) \leftarrow \left(\mathbf{\Phi}\left(\mathbf{J}_i^{j+}(t)\right)^{-1}\mathbf{\Phi}^T + \mathbf{Q}^j\right)^{-1} \tag{5.44}$$

Output: $\hat{\mathbf{x}}_i^{j+}(t)$, $\mathbf{J}_i^{j+}(t)$, $\hat{\mathbf{x}}_i^{j-}(t+1)$, $\mathbf{J}_i^{j-}(t+1)$.

5.5 Comparison of KCF, ICF, MTIC, EKCF, EICF and EMTIC

We now compare the state estimation equations of KCF, ICF, MTIC, EKCF, EICF and EMTIC for one particular target and a single consensus iteration step. The derivation of these particular forms are shown in Appendix D.

KCF: (see [17])

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathbf{J}_i^- + \mathbf{B}_i\right)^{-1}\left(\mathbf{b}_i - \mathbf{B}_i\hat{\mathbf{x}}_i^-\right) \\ &\quad + \frac{\epsilon}{1+||(\mathbf{J}_i^-)^{-1}||}(\mathbf{J}_i^-)^{-1}\sum_{i'\in\mathcal{N}_i}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right) \qquad (5.45)\\ \mathbf{J}_i^+ &= \mathbf{J}_i^- + \mathbf{B}_i \qquad (5.46)\end{aligned}$$

ICF: (see Proposition 1 in Appendix D)

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \\ &\quad \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \epsilon\sum_{i'\in\mathcal{N}_i}\frac{\mathbf{J}_{i'}^-}{N_C}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \qquad (5.47)\\ \mathbf{J}_i^+ &= N_C\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right) \qquad (5.48)\end{aligned}$$

MTIC: (see Proposition 2 in Appendix D)

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}(\frac{\mathbf{J}_i^-}{N_C}) + \mathcal{A}(\mathbf{U}_i)\right)^{-1}\left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^-\right. \\ &\quad \left.+\mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i^- - (1-\beta_{i0})\mathbf{U}_i\hat{\mathbf{x}}_i^-) + \epsilon\sum_{i'\in\mathcal{N}_i}\frac{\mathbf{J}_{i'}^-}{N_C}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \qquad (5.49)\\ \mathbf{J}_i^+ &= N_C\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{G}_i)\right) \qquad (5.50)\end{aligned}$$

EKCF: (see [25])

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathbf{J}_i^- + \mathbf{B}_i\right)^{-1} \\ &\quad \left(\mathbf{b}_i - \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{h}_i(\hat{\mathbf{x}}_i^-) + \gamma \sum_{i' \in \mathcal{N}_i} \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \qquad (5.51) \\ \mathbf{J}_i^+ &= \mathbf{J}_i^- + \mathbf{B}_i \qquad (5.52)\end{aligned}$$

where $\gamma = \frac{\epsilon}{1 + ||(\mathbf{J}_i^- + \mathbf{B}_i)^{-1}||}$

EICF: (see Proposition 3 in Appendix D)

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- \right. \\ &\quad \left. + \mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i^- - \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{h}_i(\hat{\mathbf{x}}_i^-)) + \epsilon \sum_{i' \in \mathcal{N}_i} \frac{\mathbf{J}_{i'}^-}{N_C} \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \qquad (5.53) \\ \mathbf{J}_i^+ &= N_C \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right) \qquad (5.54)\end{aligned}$$

EMTIC: (see Proposition 4 in Appendix D)

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- \right. \\ &\quad \left. + \mathcal{A}\left(\mathbf{U}_i\hat{\mathbf{x}}_i^- - (1 - \beta_i^0)\mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{h}_i(\hat{\mathbf{x}}_i^-)\right) + \epsilon \sum_{i' \in \mathcal{N}_i} \frac{\mathbf{J}_{i'}^-}{N_C} \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \qquad (5.55) \\ \mathbf{J}_i^+ &= N_C \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{G}_i)\right) \qquad (5.56)\end{aligned}$$

Note that the differences in the prior states between the neighboring nodes, $\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-$ are weighted by the corresponding neighbor's prior information matrix $\mathbf{J}_{i'}^-$ in ICF, MTIC, EICF and EMTIC. This handles the issue with naivety as the innovation from a naive neighbor's prior state will be given less weight. However, in KCF and EKCF, the innovation from each neighbor's prior is given equal weight which may deteriorate the performance of KCF and EKCF in the presence of naive nodes.

The term $\mathbf{u}_i$, in Eqns. (5.47) and (5.49) are not exactly the same, as ICF assumes perfect data association and computes $\mathbf{u}_i$ from the appropriate measurement

$\mathbf{z}_i^j$. Whereas, in MTIC, $\mathbf{u}_i$ is computed from the mean measurement $\mathbf{y}_i^j$. The same argument holds for EICF Eqn. (5.53) and EMTIC Eqn. (5.55).

In MTIC Eqn. (5.49), the $\mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i^- - (1-\beta_{i0})\mathbf{U}_i\hat{\mathbf{x}}_i^-)$ term comes due to the data association error. In EICF Eqn. (5.53) the $\mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i^- - \mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{h}_i(\hat{\mathbf{x}}_i^-))$ term comes from non-linearity in the observation model. In a similar fashion, in EMTIC Eqn. (5.55), the $\mathcal{A}\left(\mathbf{U}_i\hat{\mathbf{x}}_i^- - (1-\beta_i^0)\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{h}_i(\hat{\mathbf{x}}_i^-)\right)$ term is a result of both data association error and non-linearity in the observation model. It can be seen that, with perfect data association and linear observation model, this term equals to zero as it is not present in the ICF Eqn. (5.47) equation.

The information matrix update equations, i.e., Eqns. (5.48) and (5.50), are different for ICF and MTIC as the data association uncertainty is incorporated in $\mathbf{G}_i$ for MTIC. This shows the tight integration of the data association and tracking steps in MTIC, as the uncertainty of one step is considered in the other. The same argument holds for EICF Eqn. (5.54) and EMTIC Eqn. (5.56).

5.6 Experiments

In this section, first we evaluate the performance of the proposed EICF algorithm and compare it with the Extended Kalman Consensus Filter (EKCF) (distributed) [25] and the Extended Kalman Filter (EKF) (centralized) algorithms. Next, we evaluate the performance of the proposed EMTIC algorithm and compare it with the Extended Joint Probabilistic Data Association Filter (EJPDAF) (centralized) (see Eqns. (5.9)-(5.10)), the EICF (with ground truth data association) algorithm and the EKF (with ground truth data association) algorithm. Finally we provide real-life experiments to show the performance of the EMTIC algorithm and compare it with the EICF (with

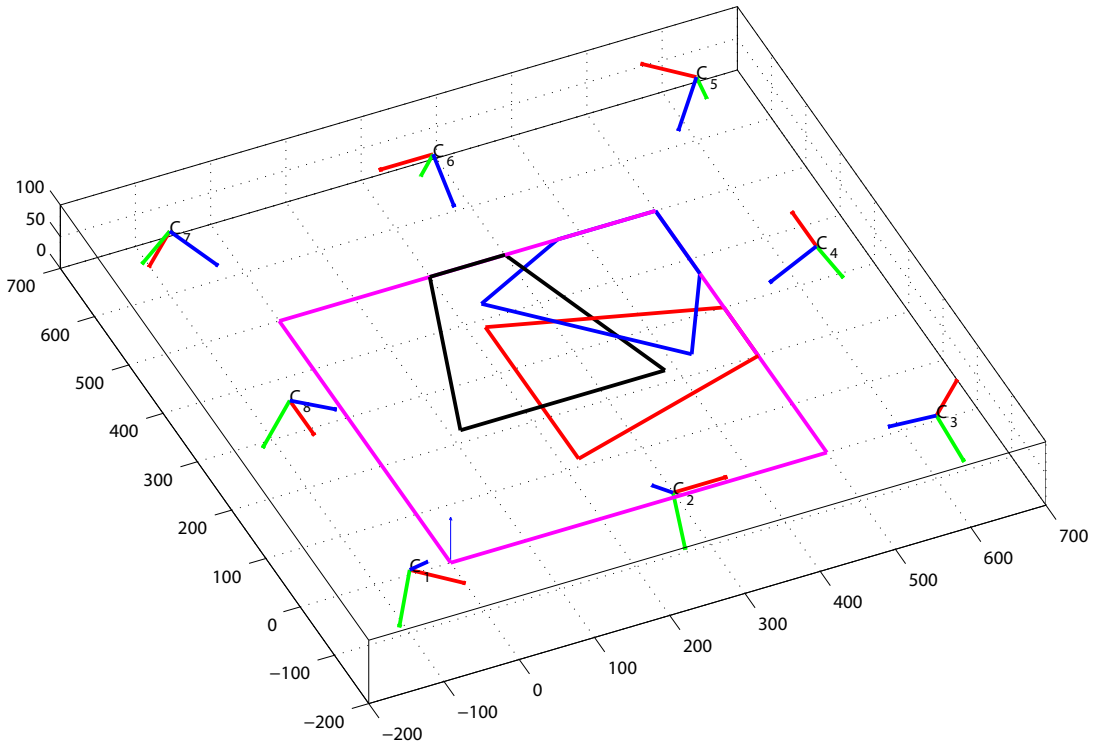


Figure 5.1: EICF and EMTIC Simulation setup

ground truth data association) algorithm.

5.6.1 Simulation Experiments

We simulate a camera network with $N_C = 8$ cameras monitoring an area with targets randomly roaming in its 500×500 area. The simulation setup is shown in Fig. 5.1. Each of the cameras can view approximately 20% of the entire area with some overlap with other cameras. The FOV of three cameras are shown in the figure. Together they cover the entire area. A circulant network topology with a degree of 2 (at each node) was chosen for the network connectivity. Each target was randomly initialized at a different location with random velocity. The target's state vector was a $4D$ vector, with the $2D$ position and $2D$ velocity components. The targets evolved for 40 time steps using the target dynamical model of Eqn. (4.1). The state transition matrix (used both in track generation and estimation) $\mathbf{\Phi}$ and process covariance $\mathbf{Q}$ were

chosen as

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{Q} = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The initial prior covariance $\mathbf{P}_i^{j-}(1) = diag(100, 100, 10, 10)$ was used at each node for each target. The initial prior state $\hat{\mathbf{x}}_i^{j-}(1)$ was generated by adding zero-mean Gaussian noise of covariance $\mathbf{P}_i^{-}(1)$ to initial the ground truth state. The observations were generated using Eqn. (5.5). The linearized observation matrix $\mathbf{H}_i^j$ was computed for each target at each camera at each time step using the prior state estimate $\hat{\mathbf{x}}_i^j(t)$ in Eqn. (5.4).

If the ground truth state was within the FOV of a sensor, a measurement was generated from the ground truth track using the measurement model Eqn. (5.5) with $\mathbf{R}_i = 100\mathbf{I}_2$. The consensus rate parameter ϵ was set to $0.65/\Delta_{max}$ where $\Delta_{max} = 2$, as each node was connected to two other nodes. Total number of consensus iterations per measurement step, K, was varied in the different simulations. The parameters for computing the association probabilities, β_i^{jn}'s, were set as follows (see [2] for details). False measurements (clutter) were generated at each node at each measurement step using a Poisson process with $\lambda = \frac{1}{32}$. Here, λ is the average number of false measurements per sensor per measurement step. Gate probability P_G was set to 0.99. The probability of detecting a target in each camera, P_D was computed by integrating the probability density function of the predicted measurement, (i.e., $\mathcal{N}(\mathbf{h}_i(\hat{\mathbf{x}}_i^{j-}), \mathbf{S}_i^j)$) over the area visible to the camera.

As a measure of performance, we computed the estimation error, e, defined as the Euclidean distance between the ground truth position and the estimated posterior

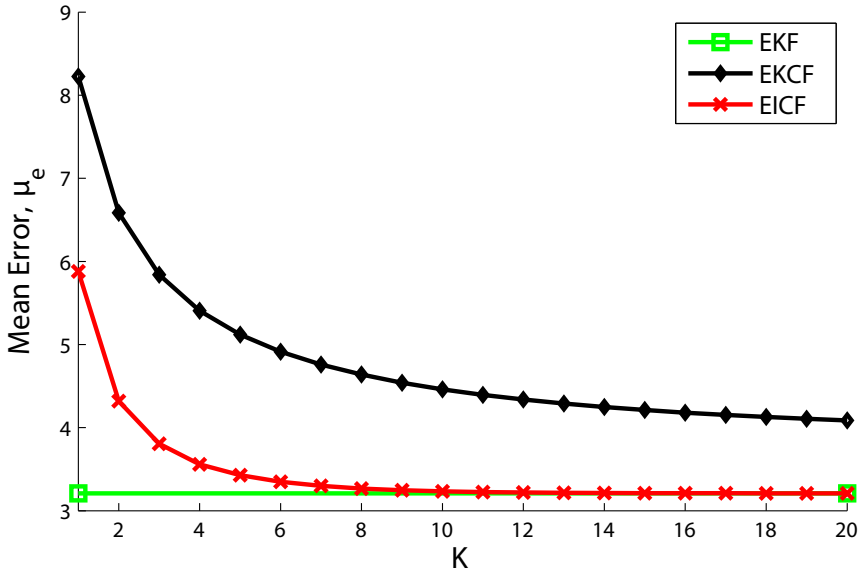


Figure 5.2: EICF performance comparison varying K

position. The simulation results were averaged over multiple simulation runs with 100 randomly generated sets of tracks. The mean (μ_e) of the errors for different methods are shown in the following graphs as the results of different experiments.

The performance of the EICF algorithm is shown in Fig. 5.2. Here, the total number of iterations, K was varied from 1 to 20. The simulation was run on 100 randomly generated tracks. It can be seen that EICF performed better than EKCF for any K. The performance of the EICF algorithm also matched the centralized performance (i.e., of EKF) as K was increased. The reason that the performance of EKCF was deteriorated was mainly due to the naivety issue as discussed earlier.

Next, the performance of the EMTIC algorithm is shown in Fig. 5.3. Here the total number of iterations K, was varied from 1 to 10. It was run over 100 randomly generated sets of tracks. At each time a set of 3 tracks were generated. As shown theoretically in this chapter, as K increased, the performance of the EMTIC algorithm matched its centralized counter-part i.e., the EJPDAF algorithm's performance. In the

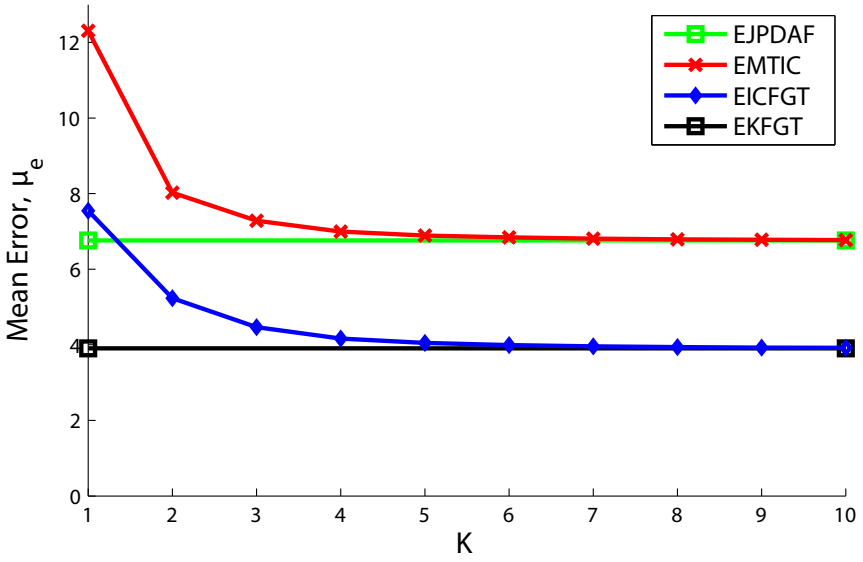


Figure 5.3: EMTIC performance comparison varying K

same figure, we also show the performance of the EICF and EKF algorithms where the ground truth data association was provided. It can be seen that the EMTIC algorithm compares well with the other algorithms, where the other algorithm (EJPDAF, EICF, EKF) either are centralized or require the knowledge of the data association where the EMTIC solves both the tracking and data association in a completely distributed manner.

5.6.2 Real-life Experiments

Next, we show the performance of the EMTIC algorithm on real-life data. Six cameras with partially overlapping FOV were chosen for the experiment. The FOV of the cameras are shown in Fig. 5.4. As the cameras were not programmable, the algorithms were run on a post-processing basis on a collected data-set. The camera network topology was simulated in the experiment. The topology used in the experiment is shown in Fig. 5.4. The cameras were calibrated w.r.t. the ground plane. There were 4 persons roaming in the area. A parts-based (utilizing PHOG features) person detector

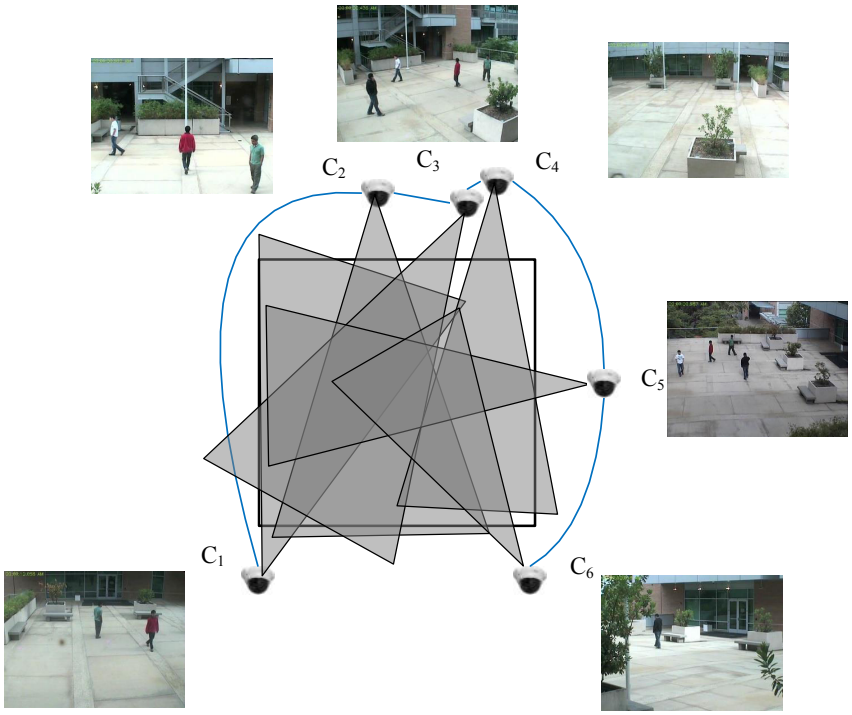


Figure 5.4: In this figure, there are six sensing nodes, $C_1, C_2, \ldots, C_6$ observing an area (black rectangle) consisting of four targets. The solid blue lines show the communication channels between different nodes. This figure also depicts the presence of "naive" nodes. For example, C_3, C_5, C_6 get direct measurements about the black target which it shares with its immediate network neighbors. However, C_1 does not have *direct* access to measurements of that target and thus is naive w.r.t. that target's state.

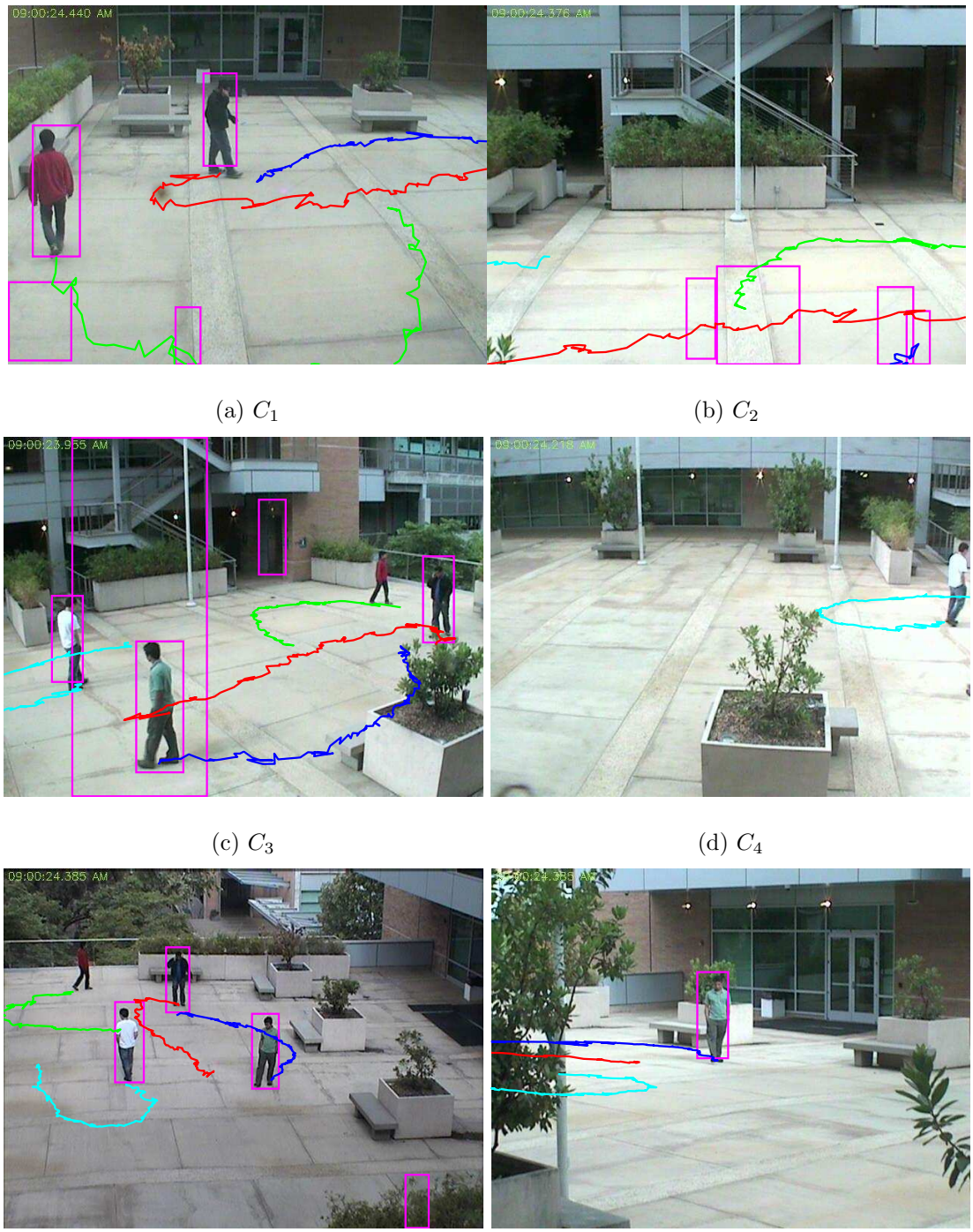
was used to detect the persons in each frame. Figs. 5.5 and 5.6 shows a snapshot of the real-life experiment (after 145 time-steps).

In Fig. 5.5, the detections are shown in magenta bounding boxes. The center point of the bottom of the bounding boxes were used as observations. There are many false measurements and also missing detections.

Fig. 5.6 shows the ground plane tracking results in C_1 for the four targets. The total number of consensus iterations, K was set to 2. In Fig. 5.6a the tracking results of the EMTIC algorithm is shown. In Fig. 5.6b, the tracking results of the EICF algorithm using the ground truth data association and in Fig. 5.6c, the tracking results of the EKF algorithm using ground truth data association information are shown. It can be seen that the EMTIC algorithm performed well compared to the EICF and EKF algorithm. It can be seen that the EMTIC tracking results are excellent although as opposed to the other methods, it is not a centralized solution or it does not require ground truth data association information. The ground plane tracks of the EMTIC algorithm are also projected and plotted on each camera's image plane for viewing purpose in Fig 5.5. In Fig. 5.7, the error from the EKF estimates for the EMTIC and the EICF algorithms are shown. It can be seen that the errors are very close to the EICF algorithm for which the ground truth data association was provided.

5.7 Conclusion

In this chapter we have extended the ICF and the MTIC algorithms to handle non-linearity in the observation model. In this chapter, we also theoretically compared all the algorithms derived in this article. Next, we showed simulation results of the EICF



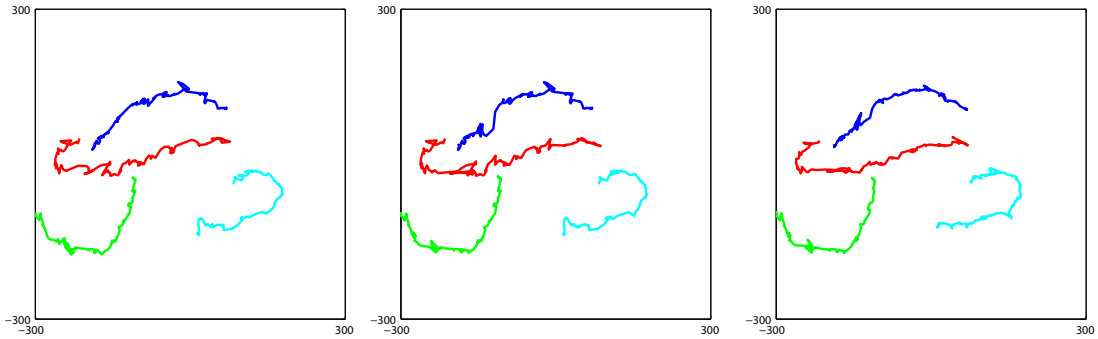
(a) C_1 (b) C_2

(c) C_3 (d) C_4

(e) C_5 (f) C_6

Figure 5.5: EMTIC Real-life experiments.

and EMTIC algorithm and compared them with other algorithms. We also showed real-life applications of the EICF and EMTIC algorithm.



(a) EMTIC tracking results (b) EICF-GT tracking results (c) EKF tracking results

Figure 5.6: EMTIC, EICF (with ground truth data association) and EKF (with ground truth data association) tracking results in the ground plane.

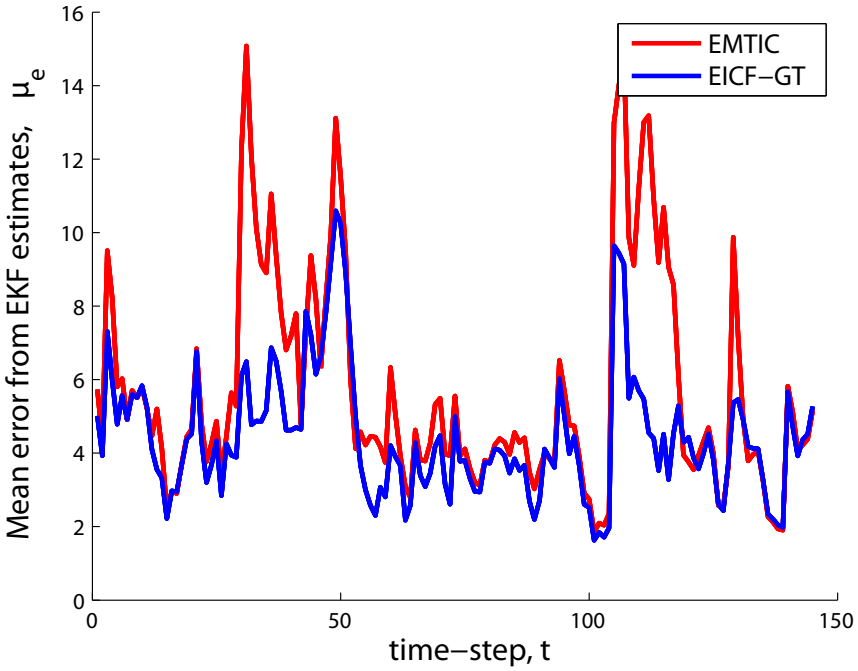


Figure 5.7: Comparing mean error from EKF

Chapter 6

Conclusions and Future Work

In this thesis, we have proposed novel algorithms for distributed state estimation in sensor networks. The core idea is in the proper information-weighting of the states which enables the very simple and well-understood distributed averaging algorithm known as the Average consensus algorithm to carry out complex state estimation and data association tasks in a sensor network. The contributions can be broken down in the following manner.

Generalized Kalman Consensus Filter

In chapter 2, we introduced a novel method for distributed state estimation algorithm, the Generalized Kalman Consensus Filter (GKCF). We discussed under what circumstances the assumptions of KCF are not valid and hence modifications are necessary. This is especially true in camera networks where each sensor has a limited FOV and they are geographically separated by distances that do not allow full communication. Then we proposed a generalized framework, Generalized KCF, which outperformed the KCF approach under such conditions. We showed the theoretical derivation of our framework and also showed simulation results to compare the performance of our algo-

rithm with other approaches.

Information-weighted Consensus Filter

In chapter 3, we presented a novel distributed state estimation framework called the Information-weighted Consensus Filter (ICF) which is generally applicable to almost any connected sensor network, converges to the optimal centralized estimate under reasonable conditions, does not require target hand-off protocols, and requires computation and communication resource similar to or less than alternative algorithms. The development of this algorithm was motivated by applications in camera networks wherein the observability properties of the state by the nodes is distinct across the nodes. We compared ICF with the state-of-the-art distributed estimation methods such as KCF and GKCF, both theoretically and through simulations. Our simulation results show that ICF outperforms these methods. The results also indicate that ICF is robust to situations where optimality conditions are not met.

Multi-Target Information Consensus

In chapter 4, we proposed the Multi Target Information Consensus (MTIC) algorithm, which is a generalized consensus-based distributed multi-target tracking scheme applicable to a wide-variety of sensor networks. MTIC handles the issues with naivety which makes it applicable to sensor networks where the sensors may have limited FOV (which is the case for a camera network). The estimation errors in tracking and data association, as well as the effect of naivety, are integrated into a single efficient algorithm. This makes MTIC very robust to false measurements/clutter. Experimental analysis shows the strength of the proposed method over existing ones.

Extension to Non-linear Models

In chapter 5, we have extended the ICF and the MTIC algorithms to handle non-linearity in the observation model of the camera. In that chapter, we also theoretically compared all the algorithms derived in this article. Next, we showed simulation results of the EICF and EMTIC algorithm and compared them with other algorithms. We also showed real-life applications of the EICF and EMTIC algorithm.

6.1 Future Work

Some of the possible future directions are discussed below.

Generalization to non-Gaussian Bayesian Estimation

The proposed algorithms currently are based on the first and second order statistics of the states by assuming that the noise and estimation errors follow Gaussian distribution. It would be interesting to derive the algorithms to be applicable to non-Gaussian Bayesian estimation frameworks. This will enable the utilization of a particle-filter-based framework without requiring an assumption on the statistical behavior of the target or the observation.

Utilization of Communication and Vision Graph Information

In many situations, one might have the knowledge of the communication and/or the vision graph of sensor network. It will be interesting to derive approaches which can utilize that information in the proposed algorithms to minimize the communication and computation resource consumption. That will be very useful for sensor networks with low-powered devices.

6.1.1 Mobile Platforms

Extending the proposed algorithms to perform distributed tracking and path planning in a network of mobile platforms would be another really interesting problem. The proposed distributed tracking algorithms work under the assumption that the camera calibration is available. For dynamic cameras, one might need to dynamically update the calibration information.

6.1.2 Hardware Implementation

In the real-life experiments presented in this article, real-life data was collected and processed offline where the communication network was simulated. Implementing the algorithms on actual smart cameras while using actual camera network for communication would be another interesting research problem. Issues related to communication such as packet loss, latency, changes in network topology will pose interesting additional challenges to the problem.

All the algorithms discussed in this thesis, at their current state require the knowledge of the total number of targets and a reasonable initialization and termination of the tracks (in case of the arrival or departure of targets from the scene). Fusion of a distributed approach to perform these tasks automatically could be another interesting problem from an implementation perspective.

Bibliography

[1] M. Alighanbari and J. P. How. An unbiased Kalman consensus algorithm. In *American Control Conference*, 2006. 16

[2] Y. Bar-Shalom, F. Daum, and J. Huang. The probabilistic data association filter. *IEEE Control Systems*, 29(6):82 –100, Dec. 2009. 6, 67, 70, 76, 92

[3] D. Borra, E. Lovisari, R. Carli, F. Fagnani, and S. Zampieri. Autonomous calibration algorithms for networks of cameras. In *American Control Conference*, pages 5126 –5131, June 2012. 36

[4] Q. Cai and J. K. Aggarwal. Tracking human motion in structured environments using a distributed-camera system. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(11):1241–1247, Nov. 1999. 7, 68

[5] M. Cetin, L. Chen, J. W. F. Iii, E. T. Ihler, O. L. Moses, M. J. Wainwright, and A. S. Willsky. Distributed fusion in sensor networks. *IEEE Signal Processing Magazine*, 23:42–55, July 2006. 7, 68

[6] R.T. Collins, A.J. Lipton, H. Fujiyoshi, and T. Kanade. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE*, 89(10):1456–1477, 2001. 7, 68

[7] C. Ding, B. Song, A. A. Morye, J. A. Farrell, and A. K. Roy-Chowdhury. Collaborative sensing in a distributed PTZ camera network. *IEEE Trans. on Image Processing*, 21(7):3282–3295, 2012. 36

[8] F. Garin and L. Schenato. A survey on distributed estimation and control applications using linear consensus algorithms. In *Networked Control Systems*, volume 406 of *Lecture Notes in Control and Information Sciences*, pages 75–107. Springer, 2011. 36, 63

[9] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *ACM Conference on Embedded Networked Sensor Systems*, 2009. 4, 34

[10] R. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME - Journal of Basic Engineering*, 82(Series D):35–45, 1960. 5

[11] A. T. Kamal, C. Ding, B. Song, J. A. Farrell, and A. K. Roy-Chowdhury. A generalized Kalman consensus filter for wide-area video networks. In *IEEE Conf. on Decision and Control*, 2011. 5, 33, 36, 52

[12] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury. Information weighted consensus filters and their application in distributed camera networks. *IEEE Trans. Automatic Control*. 5

[13] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury. Information weighted consensus. In *IEEE Conf. on Decision and Control*, 2012. 5, 36, 68

[14] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury. Information consensus for distributed multi-target tracking. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2013. 6

[15] A. T. Kamal, B. Song, and A. K. Roy-Chowdhury. Belief consensus for distributed action recognition. In *Intl. Conf. on Image Processing*, pages 141 –144, Sept. 2011. 36

[16] S. M. Kay. *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. 39, 47

[17] R. Olfati-Saber. Kalman-consensus filter: Optimality, stability, and performance. In *IEEE Conf. on Decision and Control*, 2009. 4, 9, 14, 16, 21, 22, 23, 25, 31, 36, 50, 52, 55, 66, 88

[18] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, Jan. 2007. 2, 3, 13, 18, 31, 35, 74

[19] R. Olfati-Saber, E. Franco, E. Frazzoli, and J. S. Shamma. Belief consensus and distributed hypothesis testing in sensor networks. In *Network Embedded Sensing and Control*, pages 169–182. Springer Verlag, 2006. 35

[20] R. Olfati-Saber and N. F. Sandell. Distributed tracking in sensor networks with limited sensing range. In *Proceedings of the American Control Conference*, pages 3157 – 3162, Seattle, WA, USA, June 2008. 14

[21] T. Onel, C. Ersoy, and H. Delic. On collaboration in a distributed multi-target tracking framework. In *IEEE Intl. Conf. on Communications*, June 2007. 8, 68

[22] L. L. Presti, S. Sclaroff, and M. L. Cascia. Path modeling and retrieval in distributed video surveillance databases. *IEEE Trans. on Multimedia*, 14(2):346–360, 2012. 7

[23] D. Reid. An algorithm for tracking multiple targets. *IEEE Trans. Automatic Control*, 24(6):843–854, 1979. 6, 67

[24] W. Ren, A. W. Beard, and D. B. Kingston. Multi-agent Kalman consensus with relative uncertainty. In *American Control Conference*, 2005. 16

[25] A. K. Roy-Chowdhury and B. Song. *Camera Networks: The Acquisition and Analysis of Videos over Wide Areas*. Synthesis Lectures on Computer Vision. Morgan & Claypool Publishers, 2012. 89, 90

[26] N. F. Sandell and R. Olfati-Saber. Distributed data association for multi-target tracking in sensor networks. In *IEEE Conf. on Decision and Control*, 2008. 8, 68

[27] B. Song, C. Ding, A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury. Distributed camera networks: Integrated sensing and analysis for wide area scene understanding. *IEEE Signal Processing Magazine*, 3:20–31, May 2011. 9, 11, 36

[28] B. Song, A. T. Kamal, C. Soto, C. Ding, J. A. Farrell, and A. K. Roy-Chowdhury. Tracking and activity recognition through consensus in distributed camera networks. *IEEE Trans. on Image Processing*, 19(10):2564–2579, Oct. 2010. 3, 8, 9, 11, 14, 36, 68

[29] R. Tron and R. Vidal. Distributed image-based 3-D localization of camera sensor networks. In *IEEE Conf. on Decision and Control*, pages 901 –908, Dec. 2009. 36

[30] R. Tron and R. Vidal. Distributed computer vision algorithms. *IEEE Signal Processing Magazine*, 28(3):32–45, May 2011. 3, 14, 34, 36

[31] R. Tron and R. Vidal. Distributed computer vision algorithms through distributed averaging. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2011. 3, 14

Appendix A

EKF: Information Form

Here we will derive the centralized Extended Kalman Filter (EKF) in information form, which is required to derive the distributed EICF algorithm. We have, measurement residual,

$$\widetilde{\boldsymbol{\mathcal{Z}}} = \boldsymbol{\mathcal{Z}} - \mathbf{h}_c(\hat{\mathbf{x}}_c^-). \tag{A.1}$$

Kalman gain,

$$\begin{aligned}
\mathbf{K}_c &= \mathbf{P_c}^- \boldsymbol{\mathcal{H}}^T \left(\boldsymbol{\mathcal{H}} \mathbf{P}_c^- \boldsymbol{\mathcal{H}}^T + \boldsymbol{\mathcal{R}}\right)^{-1} \\
&= \mathbf{P}_c^- \boldsymbol{\mathcal{H}}^T \left(\boldsymbol{\mathcal{R}}^{-1} - \boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}} \left((\mathbf{P}_c^-)^{-1} + \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}}\right)^{-1} \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\right) \\
&\quad \text{(using Matrix Inversion Lemma)} \\
&= \left(\mathbf{P}_c^- - \mathbf{P}_c^-\boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}} \left((\mathbf{P}_c^-)^{-1} + \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}}\right)^{-1}\right) \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1} \\
&= \left(\mathbf{P}_c^- \left((\mathbf{P}_c^-)^{-1} + \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}}\right) - \mathbf{P}_c^-\boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}}\right) \left((\mathbf{P}_c^-)^{-1} + \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}}\right)^{-1} \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1} \\
&= \left(\mathbf{I}_p + \mathbf{P}_c^-\boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}} - \mathbf{P}_c^-\boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}}\right) \left((\mathbf{P}_c^-)^{-1} + \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}}\right)^{-1} \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1} \\
&= \left((\mathbf{P}_c^-)^{-1} + \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}\boldsymbol{\mathcal{H}}\right)^{-1} \boldsymbol{\mathcal{H}}^T\boldsymbol{\mathcal{R}}^{-1}.
\end{aligned} \tag{A.2}$$

State estimate,

$$
\begin{aligned}
\hat{\mathbf{x}}_c^+ &= \hat{\mathbf{x}}_c^- + \mathbf{K}_c \widetilde{\boldsymbol{\mathcal{Z}}} \\
&= \hat{\mathbf{x}}_c^- + \left((\mathbf{P}_c^-)^{-1} + \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{H}}\right)^{-1} \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \left(\boldsymbol{\mathcal{Z}} - \mathbf{h}_c(\hat{\mathbf{x}}_c^-)\right) \\
&= \hat{\mathbf{x}}_c^- + \left(\mathbf{J}_c^- + \mathbf{U}_c\right)^{-1} \left(\mathbf{u}_c - \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \mathbf{h}_c(\hat{\mathbf{x}}_c^-)\right) \\
&= \left(\mathbf{J}_c^- + \mathbf{U}_c\right)^{-1} \left(\mathbf{J}_c^- \hat{\mathbf{x}}_c^- + \mathbf{U}_c \hat{\mathbf{x}}_c^- + \mathbf{u}_c - \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \mathbf{h}_c(\hat{\mathbf{x}}_c^-)\right) \\
&= \left(\mathbf{J}_c^- + \mathbf{U}_c\right)^{-1} \left(\mathbf{J}_c^- \hat{\mathbf{x}}_c^- + \mathbf{u}_c + \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \left(\boldsymbol{\mathcal{H}} \hat{\mathbf{x}}_c^- - \mathbf{h}_c\left(\hat{\mathbf{x}}_c^-\right)\right)\right) \qquad \text{(A.3)} \\
\mathbf{P}_c^+ &= \left((\mathbf{P}_c^-)^{-1} + \boldsymbol{\mathcal{H}}^T \boldsymbol{\mathcal{R}}^{-1} \boldsymbol{\mathcal{H}}\right)^{-1} \qquad \text{(A.4)} \\
\mathbf{J}_c^+ &= (\mathbf{P}_c^+)^{-1} \\
&= \mathbf{J}_c^- + \mathbf{U}_c \qquad \text{(A.5)}
\end{aligned}
$$

Appendix B

JPDAF: Information Form

The JPDAF estimation and covariance update equation are given in the following which are then converted to the equivalent information form below. The time index t has been dropped for simplicity. Now, Kalman gain,

$$
\begin{aligned}
\mathbf{K}^j &= \mathbf{P}^{j-}\mathbf{H}^{jT}\mathbf{S}^{j-1} \\
&= \mathbf{P}^{j-}\mathbf{H}^{jT}\left(\mathbf{H}^j\mathbf{P}^{j-}\mathbf{H}^{jT}+\mathbf{R}^j\right)^{-1} \\
&= \mathbf{P}^{j-}\mathbf{H}^{jT}\left(\mathbf{R}^{j-1}-\mathbf{R}^{j-1}\mathbf{H}^j\left((\mathbf{P}^{j-})^{-1}+\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j\right)^{-1}\mathbf{H}^{jT}\mathbf{R}^{j-1}\right) \\
&\qquad \text{(using Matrix Inversion Lemma)} \\
&= \left(\mathbf{P}^{j-}-\mathbf{P}^{j-}\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j\left((\mathbf{P}^{j-})^{-1}+\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j\right)^{-1}\right)\mathbf{H}^{jT}\mathbf{R}^{j-1} \\
&= \left(\mathbf{P}^{j-}\left((\mathbf{P}^{j-})^{-1}+\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j\right)-\mathbf{P}^{j-}\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j\right) \\
&\quad \left((\mathbf{P}^{j-})^{-1}+\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j\right)^{-1}\mathbf{H}^{jT}\mathbf{R}^{j-1} \\
&= \left(\mathbf{I}_p+\mathbf{P}^{j-}\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j-\mathbf{P}^{j-}\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j\right) \\
&\quad \left((\mathbf{P}^{j-})^{-1}+\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j\right)^{-1}\mathbf{H}^{jT}\mathbf{R}^{j-1} \\
&= \left((\mathbf{P}^{j-})^{-1}+\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j\right)^{-1}\mathbf{H}^{jT}\mathbf{R}^{j-1} \qquad \text{(B.1)} \\
\tilde{\mathbf{y}}^j &= \mathbf{y}^j-(1-\beta^{j0})\mathbf{H}^j\hat{\mathbf{x}}^{j-} \qquad \text{(B.2)}
\end{aligned}
$$

$$
\begin{aligned}
\hat{\mathbf{x}}^{j+} &= \hat{\mathbf{x}}^{j-} + \mathbf{K}^j \tilde{\mathbf{y}}^j \\
&= \hat{\mathbf{x}}^{j-} + \left((\mathbf{P}^{j-})^{-1} + \mathbf{H}^{j^T} \mathbf{R}^{j^{-1}} \mathbf{H}^j \right)^{-1} \mathbf{H}^{j^T} \mathbf{R}^{j^{-1}} \left(\mathbf{y}^j - (1 - \beta^{j0}) \mathbf{H}^j \hat{\mathbf{x}}^{j-} \right) \\
&= \hat{\mathbf{x}}^{j-} + \left(\mathbf{J}^{j-} + \mathbf{U}^j \right)^{-1} \left(\mathbf{u}^j - (1 - \beta^{j0}) \mathbf{U}^j \hat{\mathbf{x}}^{j-} \right) \\
&= \left(\mathbf{J}^{j-} + \mathbf{U}^j \right)^{-1} \left(\mathbf{J}^{j-} \hat{\mathbf{x}}^{j-} + \mathbf{U}^j \hat{\mathbf{x}}^{j-} + \mathbf{u}^j - (1 - \beta^{j0}) \mathbf{U}^j \hat{\mathbf{x}}^{j-} \right) \\
&= \left(\mathbf{J}^{j-} + \mathbf{U}^j \right)^{-1} \left(\mathbf{J}^{j-} \hat{\mathbf{x}}^{j-} + \mathbf{u}^j + \beta^{j0} \mathbf{U}^j \hat{\mathbf{x}}^{j-} \right) \qquad \text{(B.3)} \\
\mathbf{P}^{j+} &= \mathbf{P}^{j-} - (1 - \beta^{j0}) \mathbf{K}^j \mathbf{S}^j \mathbf{K}^{j^T} + \mathbf{K}^j \tilde{\mathbf{P}}^j \mathbf{K}^{j^T} \\
&= \mathbf{P}^{j-} - \mathbf{K}^j \left((1 - \beta^{j0}) \mathbf{S}^j - \tilde{\mathbf{P}}^j \right) \mathbf{K}^{j^T} \qquad \text{(B.4)}
\end{aligned}
$$

where,

$$
\tilde{\mathbf{P}}^j = \left(\sum_{n=1}^{l} \beta^{jn} \tilde{\mathbf{z}}^{jn} \left(\tilde{\mathbf{z}}^{jn} \right)^T \right) - \tilde{\mathbf{y}}^j \left(\tilde{\mathbf{y}}^j \right)^T \qquad \text{(B.5)}
$$

Let,

$$
\mathbf{C}^j = (1 - \beta^{j0}) \mathbf{S}^j - \tilde{\mathbf{P}}^j \qquad \text{(B.6)}
$$

Thus, using matrix inversion lemma and by definition of $\mathbf{J}^{j+} = (\mathbf{P}^{j+})^{-1}$ and $\mathbf{J}^{j-} = (\mathbf{P}^{j-})^{-1}$ we get,

$$
\begin{aligned}
\mathbf{J}^{j+} &= \mathbf{J}^{j-} + \mathbf{J}^{j-} \mathbf{K}^j \left(\left(\mathbf{C}^j \right)^{-1} - \mathbf{K}^{j^T} \mathbf{J}^{j-} \mathbf{K}^j \right)^{-1} \mathbf{K}^{j^T} \mathbf{J}^{j-} \\
&= \mathbf{J}^{j-} + \mathbf{G}^j \qquad \text{(B.7)}
\end{aligned}
$$

where

$$
\mathbf{G}^j = \mathbf{J}^{j-} \mathbf{K}^j \left(\left(\mathbf{C}^j \right)^{-1} - \mathbf{K}^{j^T} \mathbf{J}^{j-} \mathbf{K}^j \right)^{-1} \mathbf{K}^{j^T} \mathbf{J}^{j-}. \qquad \text{(B.8)}
$$

Thus, Equ. (B.3) and (B.7) are the JPDAF estimation equations in the information form.

Appendix C

EMTIC: Derivation

For a single sensor, the JPDAF equations can be rewritten for the non-linear case as the following,

$$\mathbf{K}^j = \left((\mathbf{P}^{j-})^{-1} + \mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j\right)^{-1}\mathbf{H}^{jT}\mathbf{R}^{j-1} \tag{C.1}$$

$$\tilde{\mathbf{y}}^j = \mathbf{y}^j - (1-\beta^{j0})\mathbf{h}(\hat{\mathbf{x}}^{j-}) \tag{C.2}$$

The state estimate,

$$\begin{aligned}
\hat{\mathbf{x}}^{j+} &= \hat{\mathbf{x}}^{j-} + \mathbf{K}^j\tilde{\mathbf{y}}^j \\
&= \hat{\mathbf{x}}^{j-} + \left((\mathbf{P}^{j-})^{-1} + \mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^j\right)^{-1}\mathbf{H}^{jT}\mathbf{R}^{j-1}\left(\mathbf{y}^j - (1-\beta^{j0})\mathbf{h}(\hat{\mathbf{x}}^{j-})\right) \\
&= \hat{\mathbf{x}}^{j-} + \left(\mathbf{J}^{j-} + \mathbf{U}^j\right)^{-1}\left(\mathbf{u}^j - (1-\beta^{j0})\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{h}(\hat{\mathbf{x}}^{j-})\right) \\
&= \left(\mathbf{J}^{j-} + \mathbf{U}^j\right)^{-1}\left(\mathbf{J}^{j-}\hat{\mathbf{x}}^{j-} + \mathbf{U}^j\hat{\mathbf{x}}^{j-} + \mathbf{u}^j - (1-\beta^{j0})\mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{h}(\hat{\mathbf{x}}^{j-})\right) \\
&= \left(\mathbf{J}^{j-} + \mathbf{U}^j\right)^{-1}\left(\mathbf{J}^{j-}\hat{\mathbf{x}}^{j-} + \mathbf{u}^j + \mathbf{H}^{jT}\mathbf{R}^{j-1}(\mathbf{H}^j\hat{\mathbf{x}}^{j-} - (1-\beta^{j0})\mathbf{h}(\hat{\mathbf{x}}^{j-}))\right)
\end{aligned} \tag{C.3}$$

$$\begin{aligned}
\mathbf{P}^{j+} &= \mathbf{P}^{j-} - (1-\beta^{j0})\mathbf{K}^j\mathbf{S}^j\mathbf{K}^{jT} + \mathbf{K}^j\tilde{\mathbf{P}}^j\mathbf{K}^{jT} \\
&= \mathbf{P}^{j-} - \mathbf{K}^j\left((1-\beta^{j0})\mathbf{S}^j - \tilde{\mathbf{P}}^j\right)\mathbf{K}^{jT}
\end{aligned} \tag{C.4}$$

where,

$$\tilde{\mathbf{P}}^j = \left(\sum_{n=1}^{l} \beta^{jn} \tilde{\mathbf{z}}^{jn} \left(\tilde{\mathbf{z}}^{jn}\right)^T\right) - \tilde{\mathbf{y}}^j \left(\tilde{\mathbf{y}}^j\right)^T \tag{C.5}$$

Let,

$$\mathbf{C}^j = (1 - \beta^{j0})\mathbf{S}^j - \tilde{\mathbf{P}}^j \tag{C.6}$$

Thus, using matrix inversion lemma and by definition of $\mathbf{J}^{j+} = (\mathbf{P}^{j+})^{-1}$ and $\mathbf{J}^{j-} = (\mathbf{P}^{j-})^{-1}$ we get,

$$\begin{aligned} \mathbf{J}^{j+} &= \mathbf{J}^{j-} + \mathbf{J}^{j-}\mathbf{K}^j \left(\left(\mathbf{C}^j\right)^{-1} - \mathbf{K}^{jT}\mathbf{J}^{j-}\mathbf{K}^j\right)^{-1} \mathbf{K}^{jT}\mathbf{J}^{j-} \\ &= \mathbf{J}^{j-} + \mathbf{G}^j \end{aligned} \tag{C.7}$$

where

$$\mathbf{G}^j = \mathbf{J}^{j-}\mathbf{K}^j \left(\left(\mathbf{C}^j\right)^{-1} - \mathbf{K}^{jT}\mathbf{J}^{j-}\mathbf{K}^j\right)^{-1} \mathbf{K}^{jT}\mathbf{J}^{j-}. \tag{C.8}$$

As, this is in the information form, for multiple sensor, under the assumption that the additional measurement information are uncorrelated, they have to be added as the following,

$$\begin{aligned} \hat{\mathbf{x}}_c^{j+} &= \left(\mathbf{J}_c^{j-} + \sum_{i=1}^{N_C} \mathbf{U}_i^j\right)^{-1} \\ &\quad \left(\mathbf{J}_i^{j-}\hat{\mathbf{x}}_i^{j-} + \sum_{i=1}^{N_C} \left(\mathbf{u}_i^j + \mathbf{H}_i^{jT}\mathbf{R}_i^{j-1}(\mathbf{H}_i^j\hat{\mathbf{x}}_i^{j-} - (1 - \beta_i^{j0})\mathbf{h}_i(\hat{\mathbf{x}}_i^{j-}))\right)\right) \end{aligned} \tag{C.9}$$

$$\mathbf{J}_c^{j+} = \mathbf{J}_i^{j-} + \sum_{i=1}^{N_C} \mathbf{G}_i^j \tag{C.10}$$

Appendix D

Single step consensus comparison:

Proposition 1 *The state estimate of ICF using a single step of consensus iteration can be expressed as*

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \\ &\quad \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \epsilon \sum_{i' \in \mathcal{N}_i} \frac{\mathbf{J}_{i'}^-}{N_C}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \qquad \text{(D.1)} \\ \mathbf{J}_i^+ &= N_C\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right) \qquad \text{(D.2)}\end{aligned}$$

Proof. Using the shorthand notation $\mathcal{A}()$ for a single step of consensus in (3.34), for a single step we can write,

$$\hat{\mathbf{x}}_i^+ = \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1}\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^-\right) + \mathcal{A}(\mathbf{u}_i)\right) \qquad \text{(D.3)}$$

By adding and subtracting $\hat{\mathbf{x}}_i^-$ in RHS of (D.3), we get,

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}\left(\mathbf{U}_i\right)\right)^{-1} \\ &\quad \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^-\right) + \mathcal{A}(\mathbf{u}_i) - \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)\hat{\mathbf{x}}_i^-\right) \\ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \\ &\quad \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^-\right) - \mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right)\hat{\mathbf{x}}_i^-\right) \qquad \text{(D.4)}\end{aligned}$$

$$
\begin{aligned}
\text{Now, } & \mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^-\right) - \mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right)\hat{\mathbf{x}}_i^- \\
&= \frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^- + \epsilon \sum_{i'\in\mathcal{N}_i}\left(\frac{\mathbf{J}_{i'}^-}{N_C}\hat{\mathbf{x}}_{i'}^- - \frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^-\right) - \frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^- - \epsilon\sum_{i'\in\mathcal{N}_i}\left(\frac{\mathbf{J}_{i'}^-}{N_C}\hat{\mathbf{x}}_i^- - \frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^-\right) \\
&= \epsilon\sum_{i'\in\mathcal{N}_i}\frac{\mathbf{J}_{i'}^-}{N_C}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right) \qquad \text{(D.5)}
\end{aligned}
$$

Using this result in (D.4), we get,

$$
\hat{\mathbf{x}}_i^+ = \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1}\left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \epsilon\sum_{i'\in\mathcal{N}_i}\frac{\mathbf{J}_{i'}^-}{N_C}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \qquad \text{(D.6)}
$$

Similarly, $\mathbf{J}_i^+$ can be written as

$$
\mathbf{J}_i^+ = N_C\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right) \qquad \text{(D.7)}
$$

■

Proposition 2 *The state estimate of MTIC using a single step of consensus iteration can be expressed as*

$$
\begin{aligned}
\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}(\frac{\mathbf{J}_i^-}{N_C}) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \\
&\quad \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i^- - (1-\beta_{i0})\mathbf{U}_i\hat{\mathbf{x}}_i^-) + \epsilon\sum_{i'\in\mathcal{N}_i}\frac{\mathbf{J}_{i'}^-}{N_C}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \qquad \text{(D.8)} \\
\mathbf{J}_i^+ &= N_C\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{G}_i)\right) \qquad \text{(D.9)}
\end{aligned}
$$

Proof. MTIC estimate equations (4.24) for a single consensus step, can be written with the $\mathcal{A}()$ notation as (dropping the target superscript j),

$$
\hat{\mathbf{x}}_i^+ = \left(\mathcal{A}(\frac{\mathbf{J}_i^-}{N_C}) + \mathcal{A}(\mathbf{U}_i)\right)^{-1}\left(\mathcal{A}(\frac{\mathbf{J}_i}{N_C}\hat{\mathbf{x}}_i^-) + \mathcal{A}(\beta_0\mathbf{U}_i\hat{\mathbf{x}}_i^-) + \mathcal{A}(\mathbf{u}_i)\right) \qquad \text{(D.10)}
$$

By adding and subtracting $\hat{\mathbf{x}}_i^-$ in RHS we get,

$$
\begin{aligned}
\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}(\frac{\mathbf{J}_i^-}{N_C}) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \\
&\quad \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \mathcal{A}(\frac{\mathbf{J}_i}{N_C}\hat{\mathbf{x}}_i^-) - \mathcal{A}(\frac{\mathbf{J}_i}{N_C})\hat{\mathbf{x}}_i^- + \mathcal{A}(\beta_0\mathbf{U}_i\hat{\mathbf{x}}_i^-)\right) \qquad \text{(D.11)}
\end{aligned}
$$

Using (D.5) and writing $\beta_{i0} = 1 - (1 - \beta_{i0})$ we have,

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}(\frac{\mathbf{J}_i^-}{N_C}) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \\ &\left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i^- - (1-\beta_{i0})\mathbf{U}_i\hat{\mathbf{x}}_i^-) + \epsilon \sum_{i' \in \mathcal{N}_i} \frac{\mathbf{J}_{i'}^-}{N_C}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \quad \text{(D.12)}\end{aligned}$$

Similarly, $\mathbf{J}_i^+$ can be written as

$$\mathbf{J}_i^+ = N_C\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{G}_i)\right) \quad \text{(D.13)}$$

■

Proposition 3 *The state estimate of EICF using a single step of consensus iteration can be expressed as*

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \\ &\left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i^- - \mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{h}_i(\hat{\mathbf{x}}_i^-)) + \epsilon \sum_{i' \in \mathcal{N}_i} \frac{\mathbf{J}_{i'}^-}{N_C}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \quad \text{(D.14)} \\ \mathbf{J}_i^+ &= N_C\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right) \quad \text{(D.15)}\end{aligned}$$

Proof. Using the shorthand notation $\mathcal{A}()$ for a single step of consensus in (5.17), for a single step we can write,

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \\ &\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^-\right) + \mathcal{A}(\mathbf{u}_i) + \mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i) - \mathcal{A}(\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{h}_i(\hat{\mathbf{x}}_i))\right) \quad \text{(D.16)}\end{aligned}$$

By adding and subtracting $\hat{\mathbf{x}}_i^-$ in RHS we get,

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1}\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^-\right) - \mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right)\hat{\mathbf{x}}_i^-\right. \\ &\left. + \mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i^-) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{h}_i(\hat{\mathbf{x}}_i))\right) \quad \text{(D.17)}\end{aligned}$$

Using (D.5), we have,

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1}\left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^-\right. \\ &\left. + \mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i^- - \mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{h}_i(\hat{\mathbf{x}}_i^-)) + \epsilon \sum_{i' \in \mathcal{N}_i} \frac{\mathbf{J}_{i'}^-}{N_C}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right)\right) \quad \text{(D.18)}\end{aligned}$$

Similarly, $\mathbf{J}_i^+$ can be written as

$$\mathbf{J}_i^+ = N_C \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C} \right) + \mathcal{A}(\mathbf{U}_i) \right) \tag{D.19}$$

■

Proposition 4 *The state estimate of EMTIC using a single step of consensus iteration can be expressed as*

$$\begin{aligned} \hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C} \right) + \mathcal{A}(\mathbf{U}_i) \right)^{-1} \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- \right. \\ &\left. + \mathcal{A}\left(\mathbf{U}_i\hat{\mathbf{x}}_i^- - (1-\beta_i^0)\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{h}_i(\hat{\mathbf{x}}_i^-) \right) + \epsilon \sum_{i' \in \mathcal{N}_i} \frac{\mathbf{J}_{i'}^-}{N_C} \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^- \right) \right) \end{aligned} \tag{D.20}$$

$$\mathbf{J}_i^+ = N_C \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C} \right) + \mathcal{A}(\mathbf{G}_i) \right) \tag{D.21}$$

Proof. Using the shorthand notation $\mathcal{A}()$ for a single step of consensus in (5.41), for a single step we can write,

$$\begin{aligned} \hat{\mathbf{x}}_i^+ &= \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C} \right) + \mathcal{A}(\mathbf{U}_i) \right)^{-1} \\ &\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^- \right) + \mathcal{A}(\mathbf{u}_i) + \mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i) - \mathcal{A}((1-\beta_i^0)\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{h}_i(\hat{\mathbf{x}}_i)) \right) \end{aligned} \tag{D.22}$$

By adding and subtracting $\hat{\mathbf{x}}_i^-$ in RHS we get,

$$\begin{aligned} \hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C} \right) + \mathcal{A}(\mathbf{U}_i) \right)^{-1} \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\hat{\mathbf{x}}_i^- \right) - \mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C} \right)\hat{\mathbf{x}}_i^- \right. \\ &\left. + \mathcal{A}(\mathbf{U}_i\hat{\mathbf{x}}_i^-) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \mathcal{A}(\mathbf{u}_i) - \mathcal{A}((1-\beta_i^0)\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{h}_i(\hat{\mathbf{x}}_i)) \right) \end{aligned} \tag{D.23}$$

Using (D.5), we have,

$$\begin{aligned} \hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C} \right) + \mathcal{A}(\mathbf{U}_i) \right)^{-1} \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- \right. \\ &\left. + \mathcal{A}\left(\mathbf{U}_i\hat{\mathbf{x}}_i^- - (1-\beta_i^0)\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{h}_i(\hat{\mathbf{x}}_i^-) \right) + \epsilon \sum_{i' \in \mathcal{N}_i} \frac{\mathbf{J}_{i'}^-}{N_C} \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^- \right) \right) \end{aligned} \tag{D.24}$$

Similarly, $\mathbf{J}_i^+$ can be written as

$$\mathbf{J}_i^+ = N_C \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C} \right) + \mathcal{A}(\mathbf{G}_i) \right) \tag{D.25}$$

■

Proposition 5 *The state estimate of GKCF using a single step of consensus iteration can be expressed as*

$$\hat{\mathbf{x}}_i^+ = \hat{\mathbf{x}}_i^- + \left(\mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i\right)^{-1} \left(\mathbf{b}_i - \mathbf{B}_i \hat{\mathbf{x}}_i^- + \epsilon \sum_{i' \in \mathcal{N}_i} \mathbf{J}_{i'}^- \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^- \right) \right) \quad \text{(D.26)}$$

$$\mathbf{J}_i^+ = \mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i \quad \text{(D.27)}$$

Proof. In the GKCF algorithm, after incorporating neighbors priors with a single step consensus we get,

$$\bar{\mathbf{x}}_i^- = \left(\mathcal{A}(\mathbf{J}_i^-)\right)^{-1} \mathcal{A}(\mathbf{J}_i^- \hat{\mathbf{x}}_i^-) \quad \text{(D.28)}$$

After incorporating measurement information and adding and subtracting $\hat{\mathbf{x}}_i^-$ ($\mathbf{b}_i$ and $\mathbf{B}_i$ are defined in (2.6-2.7)):

$$\begin{aligned}
\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- - \hat{\mathbf{x}}_i^- + \bar{\mathbf{x}}_i^- + \left(\mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i\right)^{-1} \left(\mathbf{b}_i - \mathbf{B}_i \bar{\mathbf{x}}_i^-\right) && \text{(D.29)} \\
&= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i\right)^{-1} \left(\mathbf{b}_i + \mathcal{A}(\mathbf{J}_i^-) \bar{\mathbf{x}}_i^- - \mathcal{A}(\mathbf{J}_i^-) \hat{\mathbf{x}}_i^- - \mathbf{B}_i \hat{\mathbf{x}}_i^-\right) \\
&= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i\right)^{-1} \left(\mathbf{b}_i - \mathbf{B}_i \hat{\mathbf{x}}_i^- + \mathcal{A}(\mathbf{J}_i^- \hat{\mathbf{x}}_i^-) - \mathcal{A}(\mathbf{J}_i^-) \hat{\mathbf{x}}_i^-\right) && \text{(D.30)} \\
&= \hat{\mathbf{x}}_i^- + \left(\mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i\right)^{-1} \left(\mathbf{b}_i - \mathbf{B}_i \hat{\mathbf{x}}_i^- + \epsilon \sum_{i' \in \mathcal{N}_i} \mathbf{J}_{i'}^- \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^- \right) \right) && \text{(D.31)}
\end{aligned}$$

To get (D.30), the relation in (D.28) was used. To get (D.31), the relation

$$\mathcal{A}(\mathbf{J}_i^- \hat{\mathbf{x}}_i^-) - \mathcal{A}(\mathbf{J}_i^-) \hat{\mathbf{x}}_i^- = \epsilon \sum_{i' \in \mathcal{N}_i} \mathbf{J}_{i'}^- \left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^- \right). \quad \text{(D.32)}$$

was used which can be derived in a similar way (D.5) was derived. Similarly, $\mathbf{J}_i^+$ can be written as

$$\mathbf{J}_i^+ = \mathcal{A}(\mathbf{J}_i^-) + \mathbf{B}_i. \quad \text{(D.33)}$$

■