Information Consensus for Distributed Multi-Target Tracking

Ahmed T. Kamal, Jay A. Farrell, Amit K. Roy-Chowdhury *
Department of Electrical Engineering
University of California, Riverside
{akamal,farrell,amitrc}@ee.ucr.edu

Abstract

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. In this paper, we propose consensus-based distributed multi-target tracking algorithms in a camera network that are designed to address this issue of naivety. The estimation errors in tracking and data association, as well as the effect of naivety, are jointly addressed leading to the development of an information-weighted consensus algorithm, which we term as 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. Experimental analysis is provided to support the theoretical results.

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 paper, we focus on the problem of *distributed*[1] *multi-target tracking in a camera network*. To motivate the core contribution of this work, we first describe the inter-relationship between distributed estimation and camera networks.

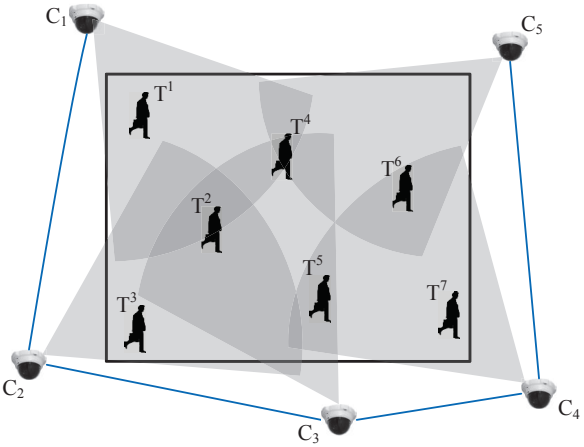


Figure 1: In this figure, there are five sensing nodes, $C_1, C_2, \ldots, C_5$ observing an area (black rectangle) consisting of seven targets $T^1, T^2, \ldots T^7$. The solid blue lines show the communication channels between different nodes. This figure also depicts the presence of "naive" nodes. 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.

Most of the work in distributed tracking has been in the multi-agent systems community [7]. 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) where each camera can view only a limited portion of the entire area. This limits the

*This work was partially supported by ONR award N00014091066 titled Distributed Dynamic Scene Analysis in a Self-Configuring Multimodal Sensor Network.

[1] 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. The 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.

1063-6919/13 $26.00 © 2013 IEEE
DOI 10.1109/CVPR.2013.311

observability of each sensor to a subset of all the targets. In this paper, our goal is to design a distributed multi-target tracking scheme which is 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* [7] 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 prior state and measurement information of 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., [11, 12, 13]).

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, 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.

A well-known consensus-based scheme for distributed state estimation is the Kalman Consensus Filter (KCF) [6]. 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. Recently, the Information-weighted Consensus Filter (ICF) [5] was proposed to address the issues with both naivety and optimality for the distributed state estimation problem.

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) [9] and the Joint Probabilistic Data Association Filter JPDAF [1] 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 paper 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. 2 the problem formulation is provided, along with a review of different consensus-based estimation methods. In Sec. 3, the JPDAF approach is reviewed and extended to a multi-sensor framework. In Sec. 4, the Multi Target Information Consensus (MTIC) tracker is proposed. Finally, in Sec. 5, the proposed method is compared against others experimentally.

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 [4], a centralized approach for tracking in a multi-camera setup was proposed where the cameras were distributed spatially over a large area. In [2], an efficiently target hand-off scheme was proposed but no multi-camera information fusion was involved. However, in this paper, 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 [3], a solution to the dis-

tributed 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 [8, 10, 11], 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 [5], which deals with both these issues.

2. Distributed Estimation and Naivety

2.1. 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

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

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

$$\mathbf{z}_i^n = \mathbf{H}_i^j \mathbf{x}_i^j + \boldsymbol{\nu}_i^j. \tag{2}$$

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. Throughout this paper, the inverse of the state covariance matrix (information/precision matrix) will be used and 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 paper.

2.2. Average consensus

Average consensus [7] 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. 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

$$\begin{aligned} a_i(k) &= a_i(k-1) + \epsilon \sum_{i'\in\mathcal{N}_i} (a_{i'}(k-1) - a_i(k-1)) \\ &= \mathcal{A}(a_i(k-1)) \end{aligned} \tag{3}$$

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 average consensus algorithm can be used to compute the average of vectors and matrices by applying it to their individual elements separately. 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. Average consensus assumes all agents have an estimate for all elements of a and that all estimates are of equal accuracy and uncorrelated. None of these assumptions usually apply to camera network.

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 [13]. These distributed estimation frameworks have been applied in various fields including camera networks for distributed implementations of 3-D point triangulation, pose estimation [12], and action recognition [11]. 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.

2.3. Kalman Consensus Filter

The Kalman Consensus Filter (KCF) [6] is a popular distributed dynamic state estimation framework. KCF utilizes the average consensus algorithm to average the state estimates over different nodes at each time step. The KCF state estimation equations are given in the following.

$$\begin{aligned}\hat{\mathbf{x}}_i^+ &= \hat{\mathbf{x}}_i^- + \left(\mathbf{J}_i^- + \mathbf{B}_i\right)^{-1}\left(\mathbf{b}_i - \mathbf{B}_i\hat{\mathbf{x}}_i^-\right) \\ &\quad + \frac{\epsilon}{1+||(\mathbf{J}_i^-)^{-1}||}(\mathbf{J}_i^-)^{-1}\sum_{i'\in\mathcal{N}_i}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right) \quad (4)\\ \mathbf{J}_i^+ &= \mathbf{J}_i^- + \mathbf{B}_i \quad (5)\end{aligned}$$

where,

$$\mathbf{b}_i = \sum_{i'\in\mathcal{N}_i\cup i}\mathbf{H}_{i'}^T{\mathbf{R}_{i'}}^{-1}\mathbf{z}_{i'}, \quad \mathbf{B}_i = \sum_{i'\in\mathcal{N}_i\cup i}\mathbf{H}_{i'}^T{\mathbf{R}_{i'}}^{-1}\mathbf{H}_{i'} \quad (6)$$

In (4), the first term is the prior state estimate, the second term is the innovation from the measurements in the local neighborhood of the sensor and the third term is an averaging term over the priors in the local neighborhood. Note that each neighbor's prior $\hat{\mathbf{x}}_{i'}^-$ is considered equally in the estimation process. This can lead to poor estimates with naive nodes due to difference in information content.

2.4. Information Weighted Consensus

In [5], the Information-weighted Consensus Filter (ICF) algorithm was proposed, which is a distributed state estimation framework that accounts for the naivety issue and can achieve optimal performance equivalent to a centralized solution. There, the prior information $\{\hat{\mathbf{x}}_i^-, \mathbf{J}_i^-\}$ and measurement information $\{\mathbf{z}_i, \mathbf{R}_i\}$ are first fused into an information vector $\mathbf{v}_i \in \mathcal{R}^p$ and an information matrix $\mathbf{V}_i \in \mathcal{R}^{p\times p}$ at each node as follows:

$$\begin{aligned}\mathbf{v}_i &= \frac{1}{N_C}\mathbf{J}_i^-(t)\hat{\mathbf{x}}_i^-(t) + \mathbf{H}_i^T{\mathbf{R}_i}^{-1}\mathbf{z}_i \quad (7)\\ \mathbf{V}_i &= \frac{1}{N_C}\mathbf{J}_i^-(t) + \mathbf{H}_i^T{\mathbf{R}_i}^{-1}\mathbf{H}_i \quad (8)\end{aligned}$$

Next, using the average consensus algorithm, the average of these vectors and matrices are computed at each node over the network as $\bar{\mathbf{v}}$ and $\bar{\mathbf{V}}$. Finally, the optimal state estimate and its information matrix is computed as,

$$\hat{\mathbf{x}}_i^+(t) = \bar{\mathbf{V}}^{-1}\bar{\mathbf{v}}, \qquad \mathbf{J}_i^+(t) = N_C\bar{\mathbf{V}} \quad (9)$$

The reason that the performance of ICF is not affected by naivety is that the prior information state of each node is appropriately weighted by the prior information matrix $\mathbf{J}_i^-$ at that node before sending it to the neighbors. Thus a node which has less information about a target's state is given less weight in the overall estimation process. As proved in [5], the ICF reaches the maximum a posteriori estimate of the centralized solution.

3. Multi-target data association

The KCF and the ICF algorithms 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) [1] 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 $\tilde{\mathbf{z}}^{jn}$ 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,

$$\begin{aligned}\tilde{\mathbf{z}}^{jn} &= \mathbf{z}^n - \mathbf{H}^j\hat{\mathbf{x}}^{j-} \quad (10)\\ \mathbf{S}^j &= \mathbf{H}^j\mathbf{P}^{j-}\mathbf{H}^{jT} + \mathbf{R}^j \quad (11)\end{aligned}$$

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 [1] for details about computing these probabilities. The Kalman gain $\mathbf{K}^j$, mean measurement $\mathbf{y}^j$ and mean measurement innovation $\tilde{\mathbf{y}}^j$ for target T^j are defined as

$$\begin{aligned}\mathbf{K}^j &= \mathbf{P}^{j-}\mathbf{H}^{jT}(\mathbf{S}^j)^{-1}, \quad (12)\\ \mathbf{y}^j &= \sum_{n=1}^{l}\beta^{jn}\mathbf{z}^n, \quad (13)\\ \tilde{\mathbf{y}}^j &= \sum_{n=1}^{l}\beta^{jn}\tilde{\mathbf{z}}^{jn} = \mathbf{y}^j - (1-\beta^{j0})\mathbf{H}^j\hat{\mathbf{x}}^{j-}. \quad (14)\end{aligned}$$

The state and its covariance estimate for JPDAF is given as

$$\begin{aligned}\hat{\mathbf{x}}^{j+}(t) &= \hat{\mathbf{x}}^{j-}(t) + \mathbf{K}^j\tilde{\mathbf{y}}^j \quad (15)\\ \mathbf{P}^{j+}(t) &= \mathbf{P}^{j-}(t) - (1-\beta^{j0})\mathbf{K}^j\mathbf{S}^j\left(\mathbf{K}^j\right)^T \\ &\quad + \mathbf{K}^j\tilde{\mathbf{P}}^j\left(\mathbf{K}^j\right)^T \quad (16)\end{aligned}$$

where,

$$\tilde{\mathbf{P}}^j = \left(\sum_{n=1}^{l}\beta^{jn}\tilde{\mathbf{z}}^{jn}\left(\tilde{\mathbf{z}}^{jn}\right)^T\right) - \tilde{\mathbf{y}}^j\left(\tilde{\mathbf{y}}^j\right)^T. \quad (17)$$

3.1. 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 equations (15-16) can be written in the information form as the following (see supplementary

materials):

$$\hat{\mathbf{x}}^{j+} = \left(\mathbf{J}^{j-} + \mathbf{U}^{j}\right)^{-1}\left(\mathbf{J}^{j-}\hat{\mathbf{x}}^{j-} + \mathbf{u}^{j} + \beta^{j0}\mathbf{U}^{j}\hat{\mathbf{x}}^{j-}\right) \quad (18)$$

$$\mathbf{J}^{j+} = \mathbf{J}^{j-} + \mathbf{G}^{j} \quad (19)$$

where,

$$\mathbf{G}^{j} = \mathbf{J}^{j-}\mathbf{K}^{j}\left(\left(\mathbf{C}^{j}\right)^{-1} - \mathbf{K}^{jT}\mathbf{J}^{j-}\mathbf{K}^{j}\right)^{-1}\mathbf{K}^{jT}\mathbf{J}^{j-} \quad (20)$$

$$\mathbf{C}^{j} = (1-\beta^{j0})\mathbf{S}^{j} - \tilde{\mathbf{P}}^{j} \quad (21)$$

$$\mathbf{u}^{j} = \mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{y}^{j} \quad \text{and} \quad \mathbf{U}^{j} = \mathbf{H}^{jT}\mathbf{R}^{j-1}\mathbf{H}^{j}. \quad (22)$$

In Eqn. (18), $\mathbf{J}^{j-}\hat{\mathbf{x}}^{j-}$ 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:

$$\hat{\mathbf{x}}^{j+} = \left(\mathbf{J}^{j-} + \sum_{i=1}^{N_C}\mathbf{U}_i^{j}\right)^{-1} \left(\mathbf{J}^{j-}\hat{\mathbf{x}}^{j-} + \sum_{i=1}^{N_C}\left(\mathbf{u}_i^{j} + \beta_i^{j0}\mathbf{U}_i^{j}\hat{\mathbf{x}}^{j-}\right)\right), \quad (23)$$

$$\mathbf{J}^{j+} = \mathbf{J}^{j-} + \sum_{i=1}^{N_C}\mathbf{G}_i^{j}. \quad (24)$$

4. Distributed Multi-Target Tracking In a Camera Network

We are now ready to present the main result of the paper. 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.

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 previous time step, the prior information at each node will be equal, i.e., $\mathbf{J}_i^{j-} = \mathbf{J}^{j-}$ and $\hat{\mathbf{x}}_i^{j-} = \hat{\mathbf{x}}^{j-}\ \forall i, j$. From this, we can rewrite (23) and (24) as follows:

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

$$= \left(\sum_{i=1}^{N_C}\mathbf{V}_i^{j}\right)^{-1}\sum_{i=1}^{N_C}\mathbf{v}_i^{j} = \left(\frac{\sum_{i=1}^{N_C}\mathbf{V}_i^{j}}{N_C}\right)^{-1}\frac{\sum_{i=1}^{N_C}\mathbf{v}_i^{j}}{N_C} \quad (25)$$

$$\mathbf{J}_i^{j+} = \sum_{i=1}^{N_C}\left(\frac{\mathbf{J}^{j-}}{N_C} + \mathbf{G}_i^{j}\right) = \sum_{i=1}^{N_C}\mathbf{W}_i^{j} = N_C\frac{\sum_{i=1}^{N_C}\mathbf{W}_i^{j}}{N_C} \quad (26)$$

where,

$$\mathbf{V}_i^{j} = \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{U}_i^{j}, \qquad \mathbf{W}_i^{j} = \frac{\mathbf{J}_i^{j-}}{N_C} + \mathbf{G}_i^{j}$$

$$\text{and} \quad \mathbf{v}_i^{j} = \mathbf{u}_i^{j} + \left(\frac{\mathbf{J}_i^{j-}}{N_C} + \beta_i^{j0}\mathbf{U}_i^{j}\right)\hat{\mathbf{x}}_i^{j-} \quad (27)$$

The three averaging terms in (25) and (26) can be computed in a distributed manner using the average consensus algorithm [7]. The algorithm is summarized in Algorithm 1. 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).

4.1. Comparison of KCF, ICF and MTIC

We now compare the state estimation equations of KCF (38-39), ICF (40-41) and MTIC (42-43) for one particular target and a single consensus iteration step. The derivation of these particular forms of KCF, ICF and MTIC are shown in the supplementary material.

KCF:

$$\hat{\mathbf{x}}_i^{+} = \hat{\mathbf{x}}_i^{-} + \left(\mathbf{J}_i^{-} + \mathbf{B}_i\right)^{-1}\left(\mathbf{b}_i - \mathbf{B}_i\hat{\mathbf{x}}_i^{-}\right) + \frac{\epsilon}{1 + ||(\mathbf{J}_i^{-})^{-1}||}(\mathbf{J}_i^{-})^{-1}\sum_{i'\in\mathcal{N}_i}\left(\hat{\mathbf{x}}_{i'}^{-} - \hat{\mathbf{x}}_i^{-}\right) \quad (38)$$

$$\mathbf{J}_i^{+} = \mathbf{J}_i^{-} + \mathbf{B}_i \quad (39)$$

ICF:

$$\hat{\mathbf{x}}_i^{+} = \hat{\mathbf{x}}_i^{-} + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^{-}}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^{-} + \epsilon\sum_{i'\in\mathcal{N}_i}\frac{\mathbf{J}_{i'}^{-}}{N_C}\left(\hat{\mathbf{x}}_{i'}^{-} - \hat{\mathbf{x}}_i^{-}\right)\right) \quad (40)$$

$$\mathbf{J}_i^{+} = N_C\left(\mathcal{A}\left(\frac{\mathbf{J}_i^{-}}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right) \quad (41)$$

Algorithm 1 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 $\mathbf{S}_i^j, \mathbf{y}_i^j, \beta_i^{j0}, \mathbf{K}_i^j$ and $\mathbf{C}_i^j$

3) Compute information vector and matrices:

$$\mathbf{u}_i^j \leftarrow \mathbf{H}_i^{jT}\mathbf{R}_i^{j\,-1}\mathbf{y}_i^j \tag{28}$$

$$\mathbf{U}_i^j \leftarrow \mathbf{H}_i^{jT}\mathbf{R}_i^{j\,-1}\mathbf{H}_i^j \tag{29}$$

$$\mathbf{G}_i^j \leftarrow \mathbf{J}_i^{j-}\mathbf{K}_i^j\left(\mathbf{C}_i^{j\,-1} - \mathbf{K}_i^{jT}\mathbf{J}_i^{j-}\mathbf{K}_i^j\right)^{-1}\mathbf{K}_i^{jT}\mathbf{J}_i^{j-} \tag{30}$$

4) Initialize consensus data

$$\mathbf{v}_i^j(0) \leftarrow \mathbf{u}_i^j + \left(\frac{\mathbf{J}_i^{j-}}{N} + \beta_i^{j0}\mathbf{U}_i^j\right)\hat{\mathbf{x}}_i^{j-} \tag{31}$$

$$\mathbf{V}_i^j(0) \leftarrow \frac{\mathbf{J}_i^{j-}}{N} + \mathbf{U}_i^j \tag{32}$$

$$\mathbf{W}_i^j(0) \leftarrow \frac{\mathbf{J}_i^{j-}}{N} + \mathbf{G}_i^j \tag{33}$$

5) Perform average consensus (Sec. 2.2) on $\mathbf{v}_i^j(0)$, $\mathbf{V}_i^j(0)$ and $\mathbf{W}_i^j(0)$ independently for K iterations.

6) Estimate:

$$\hat{\mathbf{x}}_i^{j+} \leftarrow \left(\mathbf{V}_i^j(K)\right)^{-1}\mathbf{v}_i^j(K) \tag{34}$$

$$\mathbf{J}_i^{j+} \leftarrow N_C\mathbf{W}_i^j(K) \tag{35}$$

7) Predict:

$$\hat{\mathbf{x}}_i^{j-}(t+1) \leftarrow \mathbf{\Phi}\hat{\mathbf{x}}_i^{j+}(t) \tag{36}$$

$$\mathbf{P}_i^{j-}(t+1) \leftarrow \mathbf{\Phi}\mathbf{P}_i^{j+}(t)\mathbf{\Phi}^T + \mathbf{Q} \tag{37}$$

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

MTIC:

$$\hat{\mathbf{x}}_i^+ = \hat{\mathbf{x}}_i^- + \left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{U}_i)\right)^{-1} \left(\mathcal{A}(\mathbf{u}_i) - \mathcal{A}(\mathbf{U}_i)\hat{\mathbf{x}}_i^- + \epsilon\sum_{i'\in\mathcal{N}_i}\frac{\mathbf{J}_{i'}^-}{N_C}\left(\hat{\mathbf{x}}_{i'}^- - \hat{\mathbf{x}}_i^-\right) + \mathcal{A}(\beta_{i0}\mathbf{U}_i\hat{\mathbf{x}}_i^-)\right) \tag{42}$$

$$\mathbf{J}_i^+ = N_C\left(\mathcal{A}\left(\frac{\mathbf{J}_i^-}{N_C}\right) + \mathcal{A}(\mathbf{G}_i)\right) \tag{43}$$

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 and MTIC. This handles the issue with naivety as the innovation from a naive neighbor's prior state will be given less weight. However, in KCF, the innovation from each neighbor's prior is given equal weight which may deteriorate the performance of KCF in the presence of naive nodes.

The term $\mathbf{u}_i$, in (40) and (42) 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 $\mathbf{y}_i^j$.

Another difference between (40) and (42) is the term, $\mathcal{A}(\beta_{i0}\mathbf{U}_i\hat{\mathbf{x}}_i^-)$, which is present in MTIC due to the reason that there is chance with probability β_{i0} that none of the measurements belong to the target, i.e., the probability that the estimate is biased by wrong measurements is β_{i0}. This bias is accounted for by incorporating information $\mathcal{A}(\beta_{i0}\mathbf{U}_i\hat{\mathbf{x}}_i^-)$ from the prior estimate. The information matrix update equations, i.e., (41) and (43), 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.

5. Experiments

In this section, we evaluate the performance of the proposed MTIC algorithm in a simulated environment and compare it with other methods: JPDA-KCF, ICF with nearest-neighbor data association (ICF-NN), ICF with ground truth data association (ICF-GT) and centralized Kalman Filter with ground truth data association (CKF-GT). In ICF-NN, the nearest observation $\mathbf{z}_i^n$ is associated with a target T^j only if the target is predicted to be in C_i's FOV. For ICF-NN, the data association is estimated using the Hungarian algorithm. 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, JPDA-KCF and ICF-NN 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 (1). The state transition matrix (used both in track generation and estimation) $\mathbf{\Phi}$ and process covariance $\mathbf{Q}$ were chosen as

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

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 (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 (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 [1] 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. 2. 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 while ICF-NN is highly sensitive to it (note that the x-axis of the plot is in semilog scale). Due to this reason, to be able to observe the performance of ICF-NN while other parameters were varied, the amount of clutter was kept low at $\lambda = \frac{1}{32}$ for the other experiments.

Fig. 3a 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. 3a, it can be seen that as the overlap increases, the performance of different approaches deteriorated. However, MTIC performs better than JPDA-KCF and ICF-NN. 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.

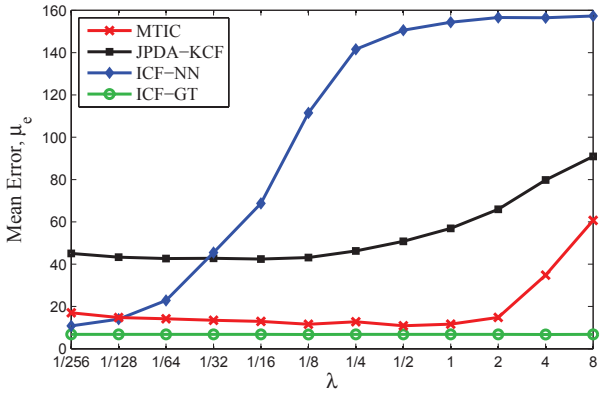


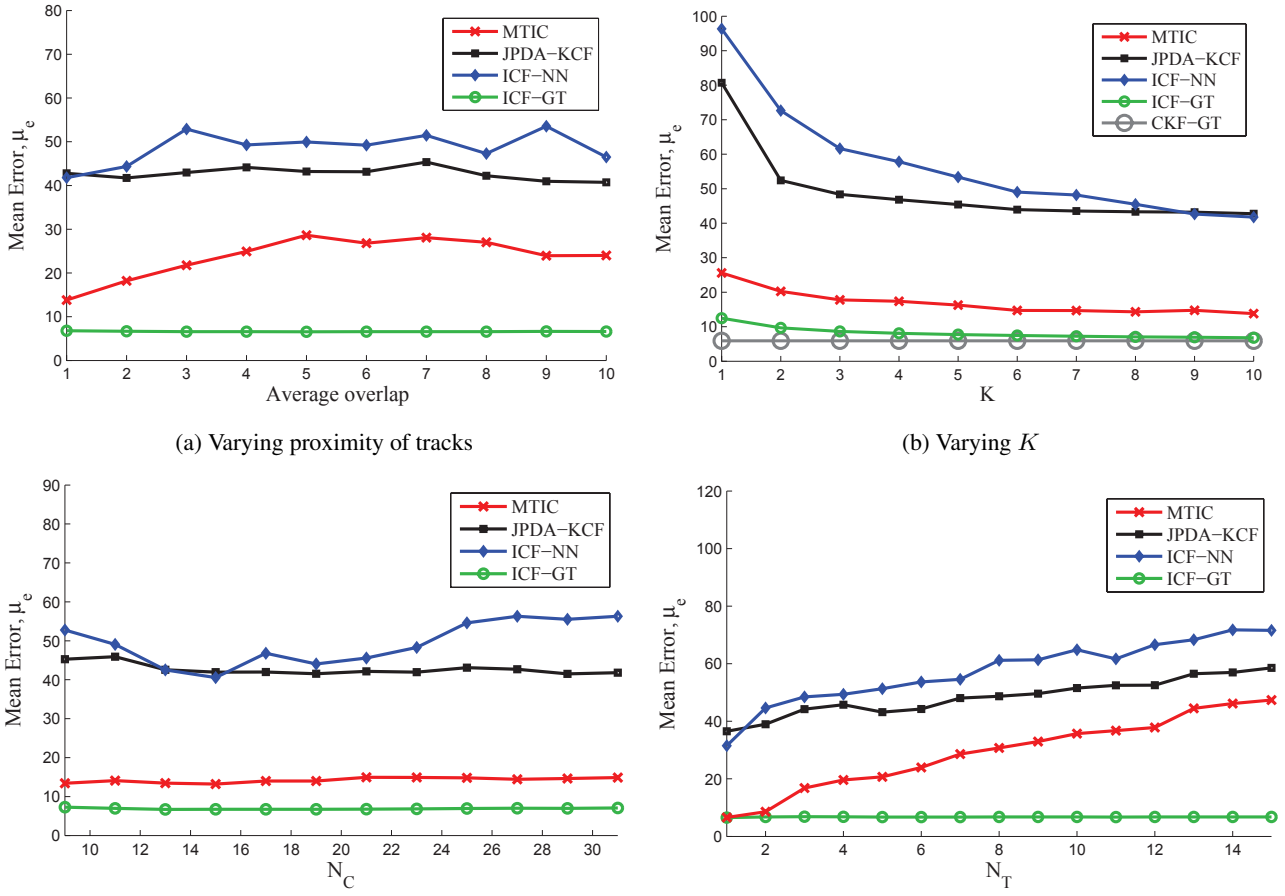
Figure 2: Performance comparison by varying amount of clutter.

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. 3b 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 and ICF-NN 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. 3c and 3d. 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. 3d.

6. Conclusion

In this paper, 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.



(a) Varying proximity of tracks

(b) Varying K

(c) Varying N_C

(d) Varying N_T

Figure 3: Performance comparison by varying different parameters.

References

[1] Y. Bar-Shalom, F. Daum, and J. Huang. The probabilistic data association filter. *IEEE Control Systems*, 29(6):82 –100, Dec. 2009. 2, 4, 7

[2] 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. 2

[3] 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. 2

[4] R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE*, 89(10):1456–1477, 2001. 2

[5] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury. Information weighted consensus. In *IEEE Conf. on Decision and Control*, 2012. 2, 3, 4

[6] R. Olfati-Saber. Kalman-consensus filter: Optimality, stability, and performance. In *IEEE Conf. on Decision and Control*, 2009. 2, 4

[7] R. Olfati-Saber, J. Fax, and R. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, Jan. 2007. 1, 2, 3, 5

[8] T. Onel, C. Ersoy, and H. Delic. On collaboration in a distributed multi-target tracking framework. In *IEEE Intl. Conf. on Communications*, June 2007. 3

[9] D. Reid. An algorithm for tracking multiple targets. *IEEE Trans. Automatic Control*, 24(6):843–854, 1979. 2

[10] 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. 3

[11] 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. 2, 3

[12] R. Tron and R. Vidal. Distributed computer vision algorithms. *IEEE Signal Processing Magazine*, 28(3):32–45, May 2011. 2, 3

[13] R. Tron and R. Vidal. Distributed computer vision algorithms through distributed averaging. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2011. 2, 3