2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

A Generalized Kalman Consensus Filter for Wide-Area Video Networks

A. T. Kamal, C. Ding, B. Song, J. A. Farrell and A. K. Roy-Chowdhury
University of California, Riverside, CA-92521

Abstract— **Distributed analysis of video captured by a large network of cameras has received significant attention lately. Tracking moving targets is one of the most fundamental tasks in this regard and the well-known Kalman Consensus Filter (KCF) has been applied to this problem. However, existing solutions do not consider the specific characteristics of video sensor networks, which are necessary for robustness across various application scenarios. Cameras are directional sensors with limited sensing range (field-of-view), and thus, targets are often not observed by many of the cameras. The network may also be spread over a wide area, preventing direct communication between all of the cameras. This limited field-of-view, combined with sparse communication and coverage topologies, motivates us to propose modifications to the traditional KCF framework. Specifically, we consider the covariance matrices of the state estimates of the neighbors and compute a weighted average consensus estimate at each node. Also, the update at each node is computed in two steps, first towards the weighted consensus estimate and then towards the final Kalman measurement update. This leads us to propose a Generalized KCF herein. Experimental results clearly show the advantage of the GKCF compared to the KCF in the considered application scenario.**

I. Introduction

As large networks of cameras become common, it is necessary to develop efficient solutions for analyzing their sensed data. Tracking targets in the captured video is one of the most basic tasks in this regard. In many applications, a distributed network architecture is necessitated whereby video is analyzed in a distributed manner over the entire network rather than at a central server. An example could be a wireless network with limited bandwidth, which is easy to install and can be mobile. In this paper, we consider such distributed camera networks and propose a consensus-based framework that is capable of tracking multiple targets throughout the network.

The problem of tracking multiple targets in a distributed sensor network has been studied previously. The Kalman Consensus Filter (KCF) [1] is a state-of-the-art distributed algorithm for fusing multiple measurements from different sensors. It is a distributed approach that combines the Distributed Kalman Filter (DKF) [2] and the average consensus algorithm [3]. At each camera (i.e., node), the KCF fuses the measurements from that node and its immediate neighbors into the state estimate to attain convergence toward the optimal state estimate and uses average consensus to ensure that the state estimates of the node and its neighbors converge toward the same values.

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

The KCF is a very appropriate framework for camera networks and has been applied in [4], [5]. 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. 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 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 the paper 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 [4], [5] performs well.

To allow for a clear discussion of the literature and our motivation, Sec II states the problem of interest and its related notation. Section III presents a technical description of our motivation, overviews the contributions of the paper,

978-1-61284-801-3/11/$26.00 ©2011 IEEE

and reviews the related literature. In Sec IV-A, we derive our Generalized Kalman Consensus Filter algorithm for a single target. Sec IV-B shows the implementation of GKCF in a multi-target scenario, where data association is necessary. Finally Sec V, shows simulation results comparing the performance of our approach with other methods.

II. Problem Formulation

We assume that sensors $\{C_i\}_{i=1}^{N_C}$ are monitoring an area containing targets $\{T_j\}_{j=1}^{N_T}$. The sensors are interconnected via a network represented by the undirected connected graph G. $\mathcal{N}_{C_i}$ is the collection of sensors which are directly connected to C_i. The state $\mathbf{x}^j$ of target T_j evolves in discrete time model according to

$$\mathbf{x}^j(k+1) = \Phi \mathbf{x}^j(k) + \gamma^j(k), \tag{1}$$

where $\mathbf{x}^j(k)$ is the state vector of T_j at time k, Φ is the state propagation matrix and $\gamma^j(k) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is the process noise. For the simulations and certain discussion herein, without loss of generality, $\mathbf{x}^j$ is a 6-dimensional vector which consists of the 3d position and velocity components of the target T_j. The state of the system is the concatenation of the target states: $\mathbf{x} = [\mathbf{x}^1; \ldots; \mathbf{x}^{N_T}]$ and $\mathbf{x} \in \Re_{(6N_T X1)}$.

Let $\hat{\mathbf{x}}_i^{j+}(k)$ and $\mathbf{W}_i^{j+}(k)$ denote the state estimate and information matrix for target j at node i. Throughout this paper, we will use the information matrix notation, where the information matrix is the inverse of the state covariance matrix. The time propagation equation for each estimate is

$$\hat{\mathbf{x}}^{j-}(k+1) = \Phi \hat{\mathbf{x}}^{j+}(k), \tag{2}$$

$$\mathbf{W}^{j-}(k+1)^{-1} = \Phi \mathbf{W}^{j+}(k)^{-1} \Phi^T + \mathbf{Q}. \tag{3}$$

The state transition matrix Φ is assumed to have its eigenvalues in the closed unit disk, with at least one eigenvalue on the unit disk (i.e., neutral stability). When T^j is observed by C_i, we assume the following sensing model:

$$\mathbf{z}_i^j = \mathbf{H}_i^j \mathbf{x}^j + \eta_i^j \tag{4}$$

where $\eta_i^j \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i^j)$ is measurement noise.

When the sensor is a camera, $\mathbf{z}_i^j$ is the 2-dimensional coordinate of the projection of T^j on C_i's image plane. The matrix $\mathbf{H}_i^j \in \Re_{(2X6)}$ is the transformation (linearized camera calibration matrix [6]) from the 3d target position to the camera image plane. The matrix $\mathbf{H}_i^j$ is a time varying function of the state estimates. For simplicity of notation, we drop the time index from $\mathbf{H}_i^j$ hereafter.

Let $\mathbf{W} \in \Re_{(6N_T X 6N_T)}$ be the information matrix of the state vector $\mathbf{x}$. Under the (reasonable) assumptions that the initial information matrix $\mathbf{W}(0) = cov(\mathbf{x}(0))^{-1}$ is block diagonal, the measurement noise between targets is uncorrelated ($cov(\eta_i^j, \eta_{i'}^{j'}) = \mathbf{0}$), and the process noise between targets is uncorrelated ($cov(\gamma^j, \gamma^{j'}) = \mathbf{0}$). Then it is straightforward to show that $\mathbf{W}(t) = cov(x(t))^{-1}$ is block diagonal for all times. This is true for the centralized Kalman filter and for the consensus based filters in this article. Due to this block diagonal structure of $\mathbf{W}(t)$, the DKF and KCF algorithms can be considered for each target separately, which greatly decreases the amount of required calculations.

As explained in Sec I, specific characteristics of video networks leads to naive nodes, which we now define precisely.
Definition: Naive Node. In a realistic camera network a node might exist where neither the node C_i nor its immediate neighbors $C_{i'}, i' \in \mathcal{N}_{C_i}$ can see a specific target T^j. In this particular scenario, C_i is naive about T^j in the sense that it cannot *directly* receive any observation update about T^j. We call such a node C_i *'Naive'* relative to target T^j.

III. Motivation

We briefly review the KCF [1], [7] (see Algorithm 1) and explain the motivation for proposing the GKCF. Here

Algorithm 1 KCF for T^j at C_i

Given $\mathbf{W}_i(1)$, $\hat{\mathbf{x}}_i^-(1)$, ϵ and K
for k = 1 to K **do**

1) Get measurement $\mathbf{z}_i$.
2) Compute information vector and matrix

$$\mathbf{u}_i = (\mathbf{H}_i)^T (\mathbf{R}_i)^{-1} \mathbf{z}_i \tag{5}$$

$$\mathbf{U}_i = (\mathbf{H}_i)^T (\mathbf{R}_i)^{-1} \mathbf{H}_i \tag{6}$$

3) Broadcast message $\mathcal{M}_i$ to neighbors containing $\mathbf{u}_i$, $\mathbf{U}_i$, $\hat{\mathbf{x}}_i^-(k)$
4) Receive message $\mathcal{M}_{i'}$ from neighbors $i' \epsilon \mathcal{N}_{C_i}$
5) Fuse the information vectors and matrices and calculate weight matrices

$$\mathbf{y}_i = \sum_{i' \in \mathcal{N}_{C_i} \cup \{i\}} \mathbf{u}_{i'} \tag{7}$$

$$\mathbf{S}_i = \sum_{i' \in \mathcal{N}_{C_i} \cup \{i\}} \mathbf{U}_{i'} \tag{8}$$

6) Compute Kalman Consensus estimate

$$\mathbf{M}_i = (\mathbf{W}_i + \mathbf{S}_i)^{-1} \tag{9}$$

$$\hat{\mathbf{x}}_i^+(k) = \hat{\mathbf{x}}_i^-(k) + \mathbf{M}_i \left(\mathbf{y}_i - \mathbf{S}_i \hat{\mathbf{x}}_i^-(k) \right) + \gamma \mathbf{W}_i(k)^{-1} \sum_{i' \in \mathcal{N}_{C_i}} \left(\hat{\mathbf{x}}_{i'}^-(k) - \hat{\mathbf{x}}_i^-(k) \right) \tag{10}$$

$$\gamma = \epsilon / (1 + ||\mathbf{W}_i(k)^{-1}||), \quad ||X|| = tr(X^T X)^{\frac{1}{2}} \tag{11}$$

7) Update the state of the Kalman-Consensus filter

$$\mathbf{W}_i(k+1) \leftarrow (\Phi \mathbf{M}_i \Phi^T + \mathbf{Q})^{-1} \tag{12}$$

$$\hat{\mathbf{x}}_i^-(k+1) \leftarrow \Phi \hat{\mathbf{x}}_i^+(k) \tag{13}$$

end for

K is the total number of imaging time instants and ϵ is a parameter, specified by the designer, that affects the rate of convergence toward consensus. If ϵ is set too high, oscillations or divergence might occur in the state estimation process. If it is set too low, the convergence speed will be slow. It has been shown that we must choose $0 < \epsilon < 1/\Delta$ for convergence [7] where Δ is the maximum degree of the network G. Thus, a value close to (but less than) $1/\Delta$ can be chosen for ϵ for the fastest convergence.

It should be noted that the KCF algorithm described above 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 [8], 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. In this paper, we are interested in solving the problem using a completely distributed architecture.

With the above introductory materials, we are in a position to discuss various specific conditions that require attention when the KCF is applied to sparse (e.g., camera) networks with naive nodes, and to propose solution strategies for each of them.

1) Average vs. weighted average: The basic KCF algorithm uses *average* consensus to combine state estimates from neighboring nodes (see eqn. (10)). 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. (9) 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 [1]. It has been stated in [1] 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. T^j, $\mathbf{y}_i$ and $\mathbf{S}_i$ are both zero. Therefore, $\mathbf{M}_i = {\mathbf{W}_i}^{-1}$ at eqn. (9). Consequently, from eqn. (12) it can be seen that the diagonal elements of $\mathbf{W}_i$ tend to zero at each time update as long as C_i remains naive with respect to T^j. 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. [9] and [10], 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. (10) are both functions of the prior state estimate $\hat{\mathbf{x}}_i^-(k)$. 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.

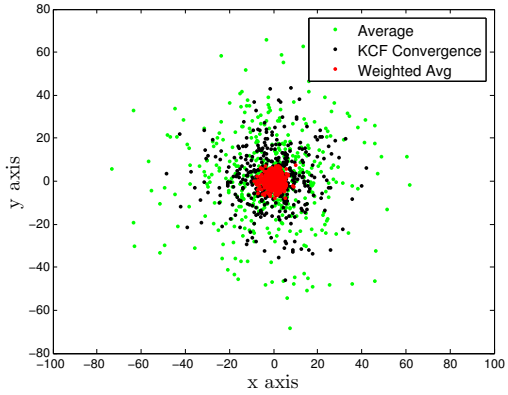

Fig. 1: This figure shows the distribution of the estimates for different algorithms (after removing the ground state offset from the estimates) in the presence of naive nodes in the network. **Green dots** represent the average state estimates of the nodes. **Black dots** represent the converged states of the KCF algorithm. **Red dots** represent the weighted average of all the nodes' estimates.

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 DKF algorithm will use measurements to update this consensus state and covariance, thus preventing the overcorrection issue mentioned above.

The motivation for using weighted average consensus instead of average consensus can be verified from Fig 1.

This figure (from our simulation explained in Sec V) shows the position of the average state estimates over all sensors (green dots), the convergence value of the KCF approach (black dots) and the weighted average of the state estimates over all sensors (red dots), relative to the ground truth state. It is evident from the plot that weighted average is a better estimate of the state.

IV. Generalized Kalman Consensus Filter

The proposed GKCF approach is presented in Algorithm 2. For the first portion of the discussion, we assume that there is only one target present in the scene. Next we generalize the concept for multiple targets. The notation in the Algorithm 2 is generalized for multiple targets. Thus, for the single target case we can just omit the target ID j and also omit Steps 2 and 6 from the algorithm, which are specifically needed for the multi-target case and will be elaborated on later.

A. Generalized Kalman Consensus Filter for a Single Target

To derive our approach in Algorithm 2, we first introduce the weighted average consensus. Next, we show how to incorporate this consensus scheme into our framework. We then implement the Distributed Kalman Filter (DKF) with the results from the weighted average consensus and show how to propagate our covariance and state estimates.

1) Weighted Average Consensus: Let the initial state estimate of all N_C agents be $\mathbf{x}_i(0)$ with information matrix $\mathbf{W}_i(0)$. 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. Also let $\mathbf{W}(0) = \sum_{i=1}^{N_C} \mathbf{W}_i(0)$. So, the global weighted average of the initial states is

$$\mathbf{x}^* = \mathbf{W}(0)^{-1} \sum_{i=1:N_C} \mathbf{W}_i(0)\mathbf{x}_i(0). \tag{14}$$

Define the weighted initial state of each agent as

$$\tilde{\mathbf{x}}_i(0) = \mathbf{W}_i(0)\mathbf{x}_i(0). \tag{15}$$

Weighted average consensus [7] states that if the iterative update in eqns. (21) and (22) is performed for all $i = 1, \ldots, N_c$, then each of the terms $\mathbf{W}_i(\kappa)^{-1}\tilde{\mathbf{x}}_i(\kappa)$ tends to the global weighted average $\mathbf{x}^*$ as $\kappa \to \infty$. As a by-product, the weights also converge to the average of the initial weights. Both these properties of the weighted average consensus will be utilized in our approach.

We assume that the initial information matrix $\mathbf{W}_i(0)$, is provided at the initial time step by the target detection mechanism. It would ideally be zero for nodes that are not detecting the target. For nodes that are detecting the target, the initial value would be $\mathbf{W}_i(k-1) = \mathbf{H}_i{}^\top \mathbf{R}^{-1}\mathbf{H}_i$.

At the k^{th} iteration, the agents communicate with each other with the $\mathbf{W}_i(k-1)$ and $\tilde{\mathbf{x}}_i(k-1)$ information. Then, using the previously discussed average consensus scheme, they get an updated prior state estimate $\hat{\mathbf{x}}_i^-(k)$ and weight estimate $\mathbf{W}_i(k)$ (see eqns. (21), (22) and (23)). This prior estimate tends towards the global normalized weighted average as stated before.

Algorithm 2 Multi-target GKCF on sensor C_i

Given $\mathbf{W}_i^j(0)$, $\hat{\mathbf{x}}_i^{j+}(0)$, ϵ and K. Also let,

$$\tilde{\mathbf{x}}_i^j(0) = \mathbf{W}_i^j(0)\hat{\mathbf{x}}_i^{j+}(0) \tag{16}$$

for k = 1 to K **do**

1) Get measurements $\{{}^i\mathbf{z}_l\}_{l=1}^L$
2) Associate observations to targets using Hungarian Algorithm. Let the observation associated with T_j in C_i be $\mathbf{z}_i^j$. If no observation in associated, set $\mathbf{z}_i^j = \mathbf{0}$ and $(\mathbf{R}_i^j)^{-1} = \mathbf{0}$
3) Compute information vector and matrix

$$\mathbf{u}_i^j = (\mathbf{H}_i^j)^T(\mathbf{R}_i^j)^{-1}\mathbf{z}_i^j \tag{17}$$

$$\mathbf{U}_i^j = (\mathbf{H}_i^j)^T(\mathbf{R}_i^j)^{-1}\mathbf{H}_i^j \tag{18}$$

4) Broadcast message $\mathcal{M}_i$ to neighbors containing $\mathbf{u}_i^j, \mathbf{U}_i^j, \tilde{\mathbf{x}}_i^j(k-1), \mathbf{w}_i^j(k-1) \quad \forall j$
5) Receive message $\mathcal{M}_{i'}$ from neighbors $i' \in \mathcal{N}_{C_i}$
6) Compute cross camera data association (CCDA) matchings using the method described in IV-B.2 and sort all data accordingly.
7) Fuse the information matrices and vectors

$$\mathbf{y}_i^j = \sum_{i' \in \mathcal{N}_{C_i} \cup \{i\}} \mathbf{u}_{i'}^j \tag{19}$$

$$\mathbf{S}_i^j = \sum_{i' \in \mathcal{N}_{C_i} \cup \{i\}} \mathbf{U}_{i'}^j \tag{20}$$

8) Compute weighted average consensus estimate

$$\tilde{\mathbf{x}}_i^j(k) = \tilde{\mathbf{x}}_i^j(k-1) + \epsilon \sum_{i' \in \mathcal{N}_{C_i}} \left(\tilde{\mathbf{x}}_{i'}^j(k-1) - \tilde{\mathbf{x}}_i^j(k-1)\right) \tag{21}$$

$$\mathbf{W}_i^j(k) = \mathbf{W}_i^j(k-1) + \epsilon \sum_{i' \in \mathcal{N}_{C_i}} \left(\mathbf{W}_{i'}^j(k-1) - \mathbf{W}_i^j(k-1)\right) \tag{22}$$

$$\hat{\mathbf{x}}_i^{j-}(k) = \mathbf{W}_i^j(k)^{-1}\tilde{\mathbf{x}}_i^j(k) \tag{23}$$

9) Compute Kalman consensus estimate

$$\mathbf{W}_i^j(k) = \mathbf{W}_i^j(k) + \mathbf{S}_i^j \tag{24}$$

$$\hat{\mathbf{x}}_i^{j+}(k) = \hat{\mathbf{x}}_i^{j-}(k) + \mathbf{W}_i^j(k)^{-1}\left(\mathbf{y}_i^j - \mathbf{S}_i^j\hat{\mathbf{x}}_i^{j-}(k)\right) \tag{25}$$

10) Propagate weight and weighted state estimate

$$\mathbf{W}_i^j(k) \leftarrow \left(\Phi\mathbf{W}_i^j(k)^{-1}\Phi^T + \mathbf{Q}\right)^{-1} \tag{26}$$

$$\tilde{\mathbf{x}}_i^j(k) \leftarrow \mathbf{W}_i^j(k)\Phi\hat{\mathbf{x}}_i^{j+}(k) \tag{27}$$

end for

2) Covariance/Information Matrix Propagation: After communicating with its neighbors and prior to using measurement information, the optimal 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 $\mathbf{W}_i(k)$ in eqn. (22) 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. (26).

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 3 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. (22) and (23)) appropriately in a standard Distributed Kalman Filter, we get the final state and weight estimate at time k. Thus using DKF in eqns. (24) and (25) 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 in a more general algorithm, at the expense of additional communications, the weighted consensus of Step 8 could be performed multiple times between measurements. Then the state estimates would converge even closer to the global weighted average (by virtue of the weighted average consensus steps). Also note that the GKCF achieves its improved performance at the expense of additional communication, as it requires communication of the information matrix for each observed target whereas the KCF does not.

B. Generalized Kalman Consensus Filter for Multiple Targets

This section discusses tracking of multiple targets in a distributed sensor network. The main difference between single and multi-target tracking is the requirement of *data association*. Several methods have been proposed for data association in multi-target multi-sensor scenarios. The authors in [11] used *Joint Probabilistic Data Association* to perform data association and embedded it into the KCF. We will show how the proposed GKCF can be used with data association.

We assume that all the sensors know the total number of targets and their initial states (this can be done in many ways like observing the entrances and exits). The sensor which detects a target gets an initial covariance/weight prior of the target state from the detection mechanism. The initial weights for that target in other sensors have to be set to zero (because they are not initially observing the target). To track multiple targets, two extra steps (Steps 2 and 6 in Algorithm 2) are needed relative to the single target GKCF algorithm. We describe these two additional steps in detail in this section.

Since a camera does not know the association between its observations and the targets, data association becomes necessary. When two sensors communicate information about multiple targets, they should also be able to associate the different targets more reliably than without communication. Data association is performed in two different stages. The *self data association* step is where a cameras associates its observations with the targets. The *cross camera data association* step is where two cameras associate the different targets among themselves.

1) Self Data Association: At the beginning of an iteration, assume that C_i has L_i observations: $\{{}^i\mathbf{z}_l\}_{l=1}^{L_i}$. The weighted state estimates of all the targets are available from the previous iteration. In order to compare an observation with a target, the weighting factors need to be removed from the weighted state estimates and transformed to the observation space. The unweighted prior state estimate at iteration k is

$$\hat{\mathcal{X}}_i^{j-}(k) = \mathbf{W}_i^j(k-1)^{-1}\tilde{\mathbf{x}}_i^j(k-1). \tag{28}$$

So in the observation space, the state estimate and its covariance would have the form $\mathbf{H}_i^j\hat{\mathcal{X}}_i^{j-}(k)$ and $\mathbf{H}_i^j\mathbf{W}_i^j(k-1)^{-1}\mathbf{H}_i^{jT}$ respectively. Let us also define the covariance of the l^{th} observation in C_i as $\mathbf{R}_i^l$. We then compute the self data association using the Mahalanobis distance:

$$d(T_i^j, {}^i\mathbf{z}_l)(k) = \left(\mathbf{H}_i^j\hat{\mathcal{X}}_i^{j-}(k) - {}^i\mathbf{z}_l\right)^T \left(\mathbf{H}_i^j\mathbf{W}_i^j(k-1)^{-1}\mathbf{H}_i^{jT} + \mathbf{R}_i^l\right)^{-1} \left(\mathbf{H}_i^j\hat{\mathcal{X}}_i^{j-}(k) - {}^i\mathbf{z}_l\right). \tag{29}$$

We can now perform a matching between the observation set and the target set. In our experiments, we use the *Hungarian algorithm* [12], which is a *bipartite matching algorithm* that finds the association across two sets by minimizing the sum of the distances over all associated pairs. We do not consider matching pairs that have a distance over a certain threshold.

After this association is done, we denote the observation associated with T_j in C_i as $\mathbf{z}_i^j$. If T_j is not associated to any of the observations in C_i, we set $\mathbf{z}_i^j = \mathbf{0}$ and $(\mathbf{R}_i^j)^{-1} = \mathbf{0}$.

2) Cross Camera Data Association: After the cameras communicate their weighted state estimates, weights and measurement information, we need to associate the targets across the cameras. If the cameras maintain a consistent ordering scheme for the targets, then this step is not necessary. If that ordering information is not available, the cameras can estimate this association using a cross-camera data association scheme. Just as in the self data association issue, we use the bipartite matching between the targets across two cameras using Mahalanobis distance as

$$d(T_i^j, T_{i'}^{j'})(k) = \left(\hat{\mathcal{X}}_i^{j-}(k) - \hat{\mathcal{X}}_{i'}^{j'-}(k)\right)^T \left(\mathbf{W}_i^j(k-1)^{-1} + \mathbf{W}_{i'}^{j'}(k-1)^{-1}\right)^{-1} \left(\hat{\mathcal{X}}_i^{j-}(k) - \hat{\mathcal{X}}_{i'}^{j'-}(k)\right). \tag{30}$$

For simplicity of notation, let us assume that after the cross camera data association is performed, the information from

each neighboring camera is sorted such that all data with the index j in camera i represents T_j's information.

V. Experimental Evaluation

To validate our approach, we prepared a simulation framework. In our simulation, we considered a 500×500 grid. In that area, we considered four moving targets. Each target's state vector is 4×1, i.e. 2d position and 2d velocity components. The initial positions were uniformly picked from the grid. The starting velocity was uniformly picked from 2-20 and with a random direction. The targets were propagated through time using the dynamics similar to (1). The values of the parameters we used for this dynamical model are as follows:

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{Q} = \begin{bmatrix} 50 & 5 & 5 & 5 \\ 5 & 50 & 5 & 5 \\ 5 & 5 & 50 & 5 \\ 5 & 5 & 5 & 50 \end{bmatrix}.$$

The targets were also programmed to randomly change direction and to be deflected back when they reached the boundary of the grid. The tracks were propagated for 40 iterations.

There were 5 cameras monitoring the area. The partially overlapping FOVs are depicted by the smaller black rectangles in Fig 3. The communication adjacency matrix $\mathbf{A}$ for the cameras is given below. The observations were generated using the same model as (4). The parameters used in this sensing and network model are as follows:

$$\mathbf{H}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \mathbf{R}=\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \mathbf{A}=\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

The initial state of all the targets were provided to each of the cameras. The initial weights were selected depending on whether a camera was viewing the target or not as

$$\mathbf{W}_{viewing}(0) = (0.05)\mathbf{I}_4, \qquad \mathbf{W}_{notviewing}(0) = (1.0e^{-6})\mathbf{I}_4.$$

The targets were tracked using both KCF and GKCF methods separately. The tracking results using both KCF and GKCF methods are shown in Fig 3 (ground truth tracks are given in Fig 2). In this figure, the left column shows the state estimates in different cameras using the KCF algorithm and right column shows the state estimates in different cameras using the GKCF algorithm. Here the circle at one end of the tracks symbolizes the final point on the track. From the left column, we can see that the cameras' tracks lagged from the ground truths (e.g., blue target) and in some cases got associated with another target's track (red and green target). In the right column, we can clearly see that the tracking was successful using the GKCF approach.

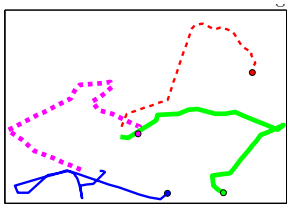

Fig. 2: This figure shows the ground truth tracks for the state estimates in Fig 3.

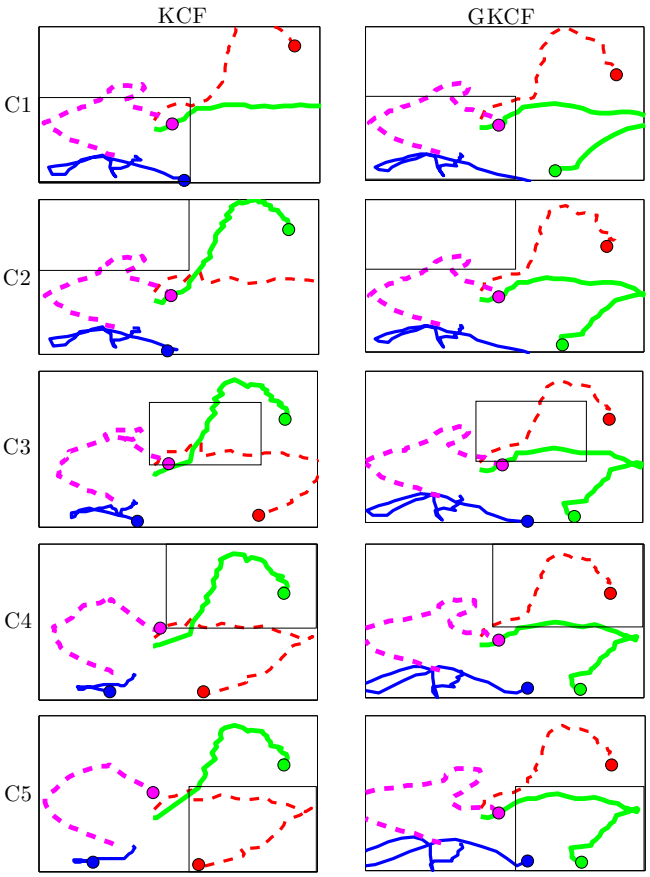

Fig. 3: This figure shows multi-target tracking (i.e., consensus estimates of the state vectors) results for four targets (shown in red, green, blue and magenta colors) using the KCF (left column) and the GKCF (right column) approaches. The cameras' FOVs are represented by black rectangles. The ground truth states for this simulation are shown in Fig 2.

Next, we compute the error statistics to compare the GKCF and KCF against each other and against the centralized Kalman filter. Fig 4 shows the mean square errors (MSE) relative to the ground truth states for the different methods. To construct this graph, the simulation was run 50 times with different randomly generated tracks. For a particular target, the mean square error of estimation is defined as,

$$MSE_j = E\big[(\hat{\mathbf{x}}_i^j - \mathbf{x}^j)^T(\hat{\mathbf{x}}_i^j - \mathbf{x}^j)\big]. \tag{31}$$

Here, $\mathbf{x}^j$ is the ground truth state. The expectation is over all iterations and all cameras. From the figure, it is evident that the performance of the GKCF was very close to the centralized Kalman filter. Please note that in this simulation only one target was used to remove the effect from data association errors. In the legend of this figure, the average

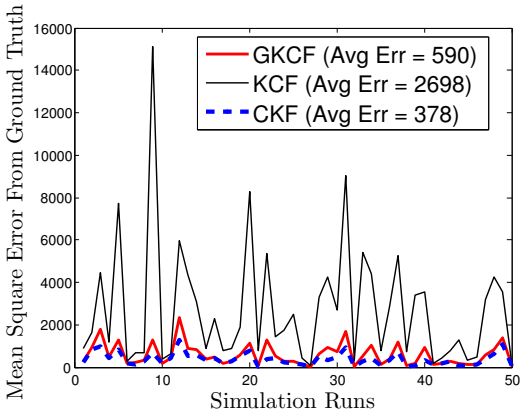

Fig. 4: This figure shows the mean square errors (MSE) relative to the ground truth states for the different methods. The x-axis is the different runs of the simulation with random tracks. The y-axis is average of the squared errors over all the iterations and cameras for each run.

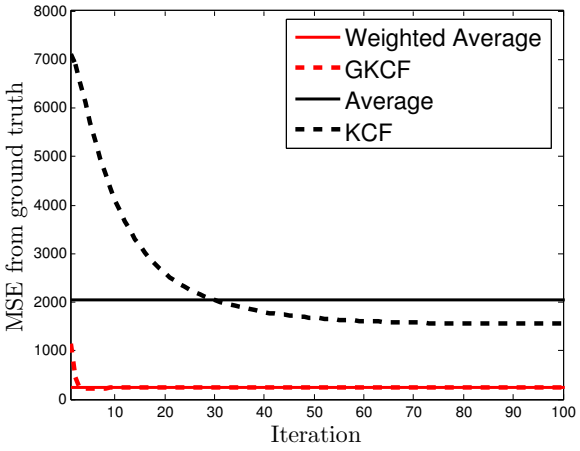

Fig. 5: In this figure we show experimentally how the KCF and GKCF schemes converge if multiple iterations are allowed before getting new measurements or propagating in time. The y-axis of this graph is the mean squared distance from the ground truth state and the x-axis is the number of iterations.

MSE for each method is shown.

In Fig 5, we show the convergence analysis for the KCF and GKCF algorithms. At a random iteration of the simulation, we programmed the algorithms to keep iterating without getting any new measurement or propagating in time. By doing this, we can see to which values these consensus schemes converge. The x-axis of the graph is the iteration number and the y-axis is the mean squared distance from the ground truth state. We have already shown in Fig 1 that statistically, the weighted average estimate is closer to the ground truth than the average estimate or the converging value of the KCF approach. This effect can also be observed in this graph, i.e. the GKCF algorithm is converging to a value much closer to the ground truth and that value is the weighted average indeed.

The states in the GKCF algorithm converged much faster and closer to the ground truth state. The average number of iterations needed to converge for the GKCF algorithm is 2.707 whereas it is 36.278 for the KCF algorithm. If within the i^{th} to $(i+10)^{th}$ iterations, the state did not change more than 0.1% of the value at the i^th iteration, it was considered to have converged at the i^{th} iteration. This statistic was generated from 1000 simulation runs. It shows that the convergence of the GKCF method is much faster than the KCF method.

VI. Conclusion

In this paper, we introduced a novel method for distributed state estimation based on the Kalman Consensus Filter (KCF). 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 algorithm with other approaches.

References

[1] R. Olfati-Saber, "Kalman-consensus filter: Optimality, stability, and performance," in *IEEE Conference on Decision and Control*, 2009.

[2] R. Olfati-Saber, "Distributed Kalman filter with embedded consensus filters," in *IEEE Conference on Decision and Control*, 2005.

[3] R. Olfati-Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *American Control Conference*, 2003.

[4] B. Song, A. T. Kamal, C. Soto, C. Ding, A. K. Roy-Chowdhury, and J. A. Farrell, "Tracking and Activity Recognition Through Consensus in Distributed Camera Networks," in *IEEE Trans. on Image Processing*, 2010.

[5] B. Song, C. Ding, A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury, "Distributed Wide Area Scene Analysis in Reconfigurable Camera Networks," in *IEEE Signal Processing Magazine - Special Issue on Distributed Image Processing and Communications*, May 2011.

[6] A. Morye, C. Ding, B. Song, A. R. Chowdhury, and J. A. Farrell, "Optimized Imaging and Target Tracking within a Distributed Camera Network," in *American Control Conference*, 2011.

[7] R. Olfati-saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," in *Proceedings of the IEEE*, vol. 95, Jan 2007.

[8] R. Olfati-Saber and N. F. Sandell, "Distributed tracking in sensor networks with limited sensing range," in *American Control Conference*, 2008.

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

[10] M. Alighanbari and J. P. How, "An unbiased Kalman consensus algorithm," in *American Control Conference*, 2006.

[11] N. Sandell and R. Olfati-Saber, "Distributed data association for multi-target tracking in sensor networks," in *IEEE Conference on Decision and Control*, 2008.

[12] M. Junger, T. M. Liebling, D. Naddef, G. L. Nemhausera, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, *50 Years of Integer Programming 1958-2008*. Springer-Verlag Berlin Heidelber, 2010.