International Journal of Engineering and Applied Sciences (IJEAS)
ISSN: 2394-3661, Volume-5, Issue-12, December 2018

Tracking The Distributed Camera Networks Through Using Kalman

Md.Azharuddin, Kaleem Fathima

Abstract— **Camera networks are being deployed for various applications like security and surveillance, disaster response and environmental modeling. However, there is little automated processing of the data. Moreover, most methods for multicamera analysis are centralized schemes that require the data to be present at a central server. In many applications, this is prohibi-tively expensive, both technically and economically. In this paper, we investigate distributed scene analysis algorithms by leveraging upon concepts of consensus that have been studied in the context of multiagent systems, but have had little applications in video analysis. Each camera estimates certain parameters based upon its own sensed data which is then shared locally with the neighboring cameras in an iterative fashion, and a final estimate is arrived at in the network using consensus algorithms. We specifically focus on two basic problems—tracking and activity recognition. For multitarget tracking in a distributed camera network, we show how the Kalman-Consensus algorithm can be adapted to take into account the directional nature of video sensors and the network topology. For the activity recognition problem, we derive a probabilistic consensus scheme that combines the similarity scores of neighboring cameras to come up with a probability for each action at the network level. Thorough experimental results are shown on real data along with a quantitative analysis.**

Index Terms— Activity recognition, camera networks, con-sensus, distributed image processing, tracking.

I. INTRODUCTION

Networks of video cameras are being installed in many applications, e.g., surveillance and security, disaster re-sponse, environmental monitoring, etc. Currently, most of the data collected by such networks is analyzed manually, a task that is extremely tedious and reduces the potential of the in-stalled networks. Therefore, it is essential to develop tools for analyzing the data collected from these cameras and summa-rizing the results in a manner that is meaningful to the end user Tracking and activity recognition are two fundamental tasks in this regard. In this paper, we develop methods for tracking and activity recognition in a distributed network of cameras.

For many applications, for a number of reasons it is desir-able that the video analysis tasks be decentralized. For example, there may be constraints of bandwidth, secure transmission, and difficulty in analyzing a huge amount of data centrally. In such situations, the cameras would have to act as autonomous agents making decisions in a decentralized manner. At the same time, however, the decisions of the cameras need to be coordinated so that there is a consensus on the state (e.g., position, activity) of the target even if each camera is an autonomous agent. Thus, the cameras, acting as autonomous agents, analyze the raw data lo-cally, exchange only distilled information that is relevant to the collaboration, and reach a shared, global analysis of the scene.

Although there are a number of methods in video analysis that deal with multiple cameras, and even camera networks, *dis-tributed* processing in camera networks has received very little attention. In Section II, we will review the current state of the art in camera networks and will see that very few methods are capable of distributed analysis of video. On the other hand, dis-tributed processing has been extensively studied in the multi-agent systems and cooperative control literature [29]. Methods have been developed for reaching consensus on a state observed independently by multiple sensors. However, there is very little study on the applicability of these methods in camera networks.

In this paper, we show how to develop methods for tracking and activity recognition in a camera network where processing is distributed across the cameras. For this purpose, we show how consensus algorithms can be developed that are capable of con-verging to a solution, i.e, target state, based upon local deci-sion making and exchange of these decisions (not sensed data) among the cameras. We focus on two problems. For distributed tracking, we show how the Kalman consensus algorithm [28] can be adapted to camera networks taking into account issues like network topology, handoff and fault tolerance. For activity recognition, we derive a new consensus algorithm based upon the recognized activity at each camera and the transition prob-abilities between various activities. Experimental results and quantitative evaluation for both these methods are presented. Note that here we assume ideal communication between cam-eras which are connected, i.e., communication is not a bottle-neck. This proposed work is a proof-of-concept study in using distributed processing algorithms for video analysis. In the fu-ture, the practical constraints of using consensus algorithms in camera networks should be considered.

We start with a review of consensus algorithms for distributed estimation. Thereafter, in Section IV, we present a variant of the Kalman-Consensus approach for distributed tracking in the camera network and show experimental results, that are ana-lyzed quantitatively. In Section V, we study the problem of ac-tivity recognition in a consensus framework. For this purpose, we derive a completely new algorithm that shows how local de-cisions at each camera node can be combined to come up with a consensus on the state representing the activity. Again, exper-imental results are shown and analyzed.

Md.Azharuddin Dept of ECE, University College of Engineering (A) Osmania University Hyderabad- 500007, India 7893456345
Kaleem Fathima, Proffesor, SMIEEE, Dept of Electronic and communication. Osmania University, Hyderabad, India

II. PAST WORK ON SCENE ANALYSIS IN CAMERA NETWORKS

Our review of scene analysis algorithms will be limited to those directly related to the application domain of camera net-works.

There have been a few papers in the recent past that deal with networks of video sensors. Particular interest has been fo-cused on learning a network topology [21], [40], i.e., config-uring connections between cameras and entry/exit points in their view. Some of the existing methods on tracking over the net-work, include [34], [37]. Other interesting problems in camera networks, like object/behavior detection and matching across cameras, camera handoff and camera placement have been ad-dressed in [1], [10], [16], [39], and [46]. There has also been recent work on tracking people in a multicamera setup [8], [17]. However, these methods do not address the issue of distributed processing.

In [22], a distributed target tracking approach using a cluster-based Kalman filter was proposed. Here, a camera is selected as a cluster head which aggregates all the measurements of a target to estimate its position using a Kalman filter and sends that esti-mate to a central base station. Our proposed tracking system dif-fers from this method in that each camera has a consensus-based estimate of the target's state and, thus, there is no need for addi-tional computation and communication to select a cluster head. As will be described in Section IV, we apply in a special way the distributed Kalman-Consensus filter [28] which has been shown to be more effective than other distributed Kalman filter schemes. Consensus schemes have been gaining popularity in computer vision applications involving multiple cameras [41]. A related work that deals with tracking targets in a camera net-work with PTZ cameras is [33]. Here, the authors proposed a mixture between a distributed and a centralized scheme using both static and PTZ cameras in a virtual camera network envi-ronment. Our approach to tracking in the camera network, how-ever, is completely distributed using consensus algorithms. An-other problem that has received some attention in this context is the development of distributed embedded smart cameras [3]. The focus of this paper, however, is on the algorithm side, rather than building a specific smart camera architecture.

The problem of multiview activity recognition have been ad-dressed in many papers, e.g., [44] and [45], but the information of multiple views is fused centrally. Our proposed framework is decentralized: each camera determines a probabilistic measure of similarity of its own observed activities to a predefined dictio-nary and information is dispersed to compute a consensus-based estimate. A preliminary framework for distributed tracking and control in camera network was presented in [38]. However, in-stead of only considering target-based network topology [38], in this paper we also define a network topology based upon com-munication constraints which is more important in practice. Be-sides tracking through consensus, we also address another fun-damental task of distributed activity recognition, derive a prob-abilistic consensus scheme, and show experimental results on real data with a quantitative analysis.

III. CONSENSUS ALGORITHMS FOR DISTRIBUTED ESTIMATION

In the multiagent systems literature, *consensus* means that the agents reach an agreement regarding a certain quantity of in-terest that depends upon the measurements of all sensors in a network. The network may not be fully connected, so there is no central unit that has access to all the data from the sensors. Consequently, a *consensus algorithm* is an interaction rule that specifies information exchange between a sensor and its neigh-bors that guarantees that all the nodes reach a consensus. The in-teraction topology of a network of sensors is represented using

a graph $G=(V E)$ with the set of nodes $V=\{1,2,\dots,n\}$ and edges $E\subseteq V\times V$. Each sensor node $i=1\dots n$ main-tains an estimate $x_i \in\mathbb{R}^m$ of a quantity $\mathbf{x}\in\mathbb{R}^m$. Consensus is achieved when $\mathbf{x}_1=\mathbf{x}_2=\dots=x_n$, which is an n-dimensional subspace of $\mathbb{R}^{mn}$. A thorough review of consensus in networked multiagent systems can be found in [29]. Here we briefly review some of the basic approaches needed for this paper.

A. Brief Review

In a network of agents, consensus can be defined as reaching an agreement through cooperation regarding a certain quantity of interest that depends upon the information available to mea-surements from all agents. An interaction rule that specifies the information exchange between an agent and all of its neigh-bors in the network and the method by which the information is used, is called a consensus algorithm (or protocol). Cooperation means giving consent to providing one's state and following a common protocol that serves group objective.

For example, in a network of temperature sensor, the sensors' estimates of temperature could be different due to sense noise and local variation. The sensors then interchange information with their neighboring sensors, and use the information to re-fine their local estimates. Consensus is reached when all sensors agree on a single value.
Distributed computing [20] has been a challenging field in computer science for the last few decades. A lot of work has been done on consensus algorithms which formed the baseline for distributed computing. Formally the study of consensus orig-inated in management science and statistics in 1960s (see [6]). The work in [42] on asynchronous asymptotic agreement prob-lems in distributed decision making systems and parallel com-puting [2] were the initial works in systems and control theory on a distributed network. A theoretical framework for defining and solving consensus problems for networked dynamic sys-tems was introduced in [30] building on the earlier work of [11]. Consensus algorithms for reaching an agreement without com-puting any objective function appeared in the work of [15]. Fur-ther theoretical extensions of this work were presented in [35] with a focus towards treatment of directed information flow in networks. In [15], a formal analysis was provided for emergence of alignment. The setup in [30] was originally created with the vision of designing agent-based amorphous computers for col-laborative information processing in networks. Later, [30] was used in development of flocking algorithms with guaranteed convergence and the capability to deal with obstacles and ad-versarial agents [27]. Recent works

related to multi agent net-worked systems include consensus [19], collective behavior of flocks and swarms [27], sensor fusion [28], random networks [13], synchronization of coupled oscillators [32], algebraic con-nectivity of complex networks [26], asynchronous distributed algorithms [23], formation control for multi robot systems [9], dynamic graphs [24], and complexity of coordinated tasks [14]. The goals of most consensus algorithms usually include [12]:

1) **Validity**: The final answer that achieves consensus is a valid answer.
2) **Agreement**: All processes agree as to what the agreed upon answer was by the end of the process.
3) **Termination**: The consensus process eventually ends with each process contributing.
4) **Integrity**: Processes vote only once.

Many consensus algorithms contain a series of events (and re-lated messages) during a decision-making round. Typical events include Proposal and Decision. Here, proposal typically means the communication of the state of each agent and decision is the process of an agent deciding on proposals received from its neighbors after which it is not going to receive any proposal from the neighbors to come a different conclusion. In our appli-cation domain of camera networks, the agents are the cameras and the state vector we are trying to estimate are the position and velocity of a set of targets and the ID of an activity based upon a learned dictionary of activities.

B. Consensus in Distributed Camera Networks

In distributed camera networks, the cameras act as au-tonomous agents. Each camera determines its own estimate of the object's state (e.g., position, activity label). The cameras then share local estimates with their neighboring cameras in an iterative fashion, and a final estimate is arrived at in the network using consensus algorithms [29].

1) Distributed Tracking: There have been recent attempts to achieve dynamic state estimation in a consensus-like manner. In contrast to a central Kalman filter where state information coming from several sensors is fused in a central station, dis-tributed Kalman filters (DKF) compute a consensus-based esti-mate on the state of interest with only point-to-point commu-nication between the sensors [28]. A distributed Kalman fil-tering (DKF) strategy that obtains consensus on state estimates was presented in [28]. The overall performance of this so-called Kalman-Consensus filter has been shown to be superior to other distributed approaches. It is on this DKF strategy that we base our distributed tracking algorithm. The mathematical details are presented in Section IV-A.

2) *Distributed Activity Recognition:* There have been methods on multiview activity recognition [44], [45], but the information of multiple views is fused centrally. In this paper, we propose a framework for distributed activity recognition. Each camera determines a probabilistic measure of similarityof its own observed activities to a predefined dictionary, and then disperses this information to compute a consensus-based estimate with only point-to-point communication between the cameras. We show mathematically how to compute this consensus based upon the similarity score computed at each camera and the transition probabilities between activities (can be uniform if no prior information is available).

IV. DISTRIBUTED TARGET TRACKING USING KALMAN CONSENSUS FILTERING

In this section, we present the first major result of this paper—how to track multiple targets in a camera network using a consensus algorithm that relies on the tracks obtained at individual cameras. For this purpose, we leverage upon the Kalman-Consensus algorithm in the distributed processing and multiagent systems literature [28], [29]. However, there are some major differences due to the nature of cameras, and we show how to handle them.

Cameras are directional sensors and, thus, geographically neighboring cameras may be viewing very different portions of the scene. On the other hand, cameras that are geographi-cally far away may be observing the same target. Therefore, we can define a target-based network topology, where the neighborhood structure is defined with respect to each target. Since targets are dynamic, this target-based topology changes over time. However, the communication constraints due to bandwidth limitation or physical network connection, which is most important in practice, naturally determine the communi-cation-based topology of network. The communication-based topology is somewhat static, since the bandwidth limitation or physical connection won't change in a short time period. The distributed tracking is achieved by considering both the communication and target-based network topologies. In the next section, we will describe this process in more detail. Also, we will show how to take into account the handoff of targets as they move between cameras.

A. Problem Formulation

Let $\mathcal{C}$ be the set of all cameras in the network. We can then define the subset of all cameras viewing target T_l as $\mathcal{C}_v^l \subset \mathcal{C}$ and the rest of the cameras as $\mathcal{C}_{\bar{v}}^l \subset \mathcal{C}$. Each camera C^i will also have its set of *neighboring cameras* $\mathcal{C}_n^i \subset \mathcal{C}$. Based upon the communication constraints due to bandwidth limitation and network connection, we define the set $\mathcal{C}_n^i$ as all the cameras with which C^i is able to communicate directly. In other words, C^i can assume that no other cameras other than its neighbors $\mathcal{C}_n^i$ exist as no information flows directly from non-neighboring cameras to C^i. Note that the set of neighbors need not be geographical neighbors. We also define the set of *overlapping cameras* of C^i as $\mathcal{C}_o^i \subset \mathcal{C}$; since all the cameras can change their PTZ parame-ters and have therefore several possible fields of view, we define the set $\mathcal{C}_o^i$ as all the cameras with which C^i can *potentially* have an overlapping field of view. By definition, it becomes clear then that for each $C^i \in \mathcal{C}_v^l$, it is true that $\mathcal{C}_v^l \subset \{\mathcal{C}_o^i \cup C^i\}$. We define $\mathcal{C}_c^i \subset \mathcal{C}$ as the connected component that C^i is in. We assume $\mathcal{C}_o^i \subset \mathcal{C}_c^i$, that is to say, C^i is able to exchange information with its overlapping cameras directly or via other cameras (**Assump-tion** $*$). An example of the camera network is shown in Fig. 1.

As mentioned earlier, we propose a special application of the Kalman-Consensus Filter presented in [28] to solve the problem of finding a consensus on the state vectors of multiple targets in a camera network. We consider the situation where targets are moving on a ground plane and a homography between each camera's image plane and the ground plane is known. We will show how the state vector estimation for each target by each camera (i.e., each camera's estimates based upon its individual measurements) can be combined together through the consensus scheme. This method is independent of the tracking scheme em-ployed in each camera, which may or may not be based upon the Kalman filter.

B. Kalman-Consensus Tracking

To model the motion of a target T on the ground plane as observed by camera C, we consider a linear dynamical system with time propagation and observation models

$$\mathbf{x}^l(k+1) = \mathbf{A}^l(k)\mathbf{x}^l(k) + \mathbf{B}^l(k)\mathbf{w}^l(k); \quad \mathbf{x}^l(0) \tag{1}$$

$$\mathbf{z}_i^l(k) = \mathbf{F}_i^l(k)\mathbf{x}^l(k) + \mathbf{v}_i^l(k) \tag{2}$$

where $\mathbf{w}^l(k)$ and $\mathbf{v}_i^l(k)$ are zero mean white Gaussian noise $(\mathbf{w}^l(k) \sim \mathcal{N}(0, \mathbf{Q}^l), \mathbf{v}_i(k) \sim \mathcal{N}(0, \mathbf{R}_i^l))$ and $\mathbf{x}^l(0) \sim \mathcal{N}(\mathbf{x}_0^l, \mathbf{P}_0)$ is the initial state of the target. We define the state of the target at time step k as $\mathbf{x}^l(k) = (x^l(k), y^l(k), \dot{x}^l(k), \dot{y}^l(k))^T$ where $(x^l(k), y^l(k))$ and $(\dot{x}^l(k), \dot{y}^l(k))$ are the position and velocity of target T_l in the x and y directions, respectively. The vector $\mathbf{x}_i^l$ is the state of target T_l by C_i based upon the measurements in C_i only. The vector $\mathbf{z}_i^l(k)$ is the noisy measurement at camera C_i. $\mathbf{z}_i^l(k)$

can be measured on either ground plane or image plane. We consider both cases and can show that it does not affect the per-formance of distributed Kalman-Consensus tracking algorithm, i.e., these two different cases of $\mathbf{z}_i^l(k)$ give equivalent tracking results.

Case 1: $\mathbf{z}_i^l(k)$ is the sensed target position $(x_i^l(k), y_i^l(k))$ on the ground plane based upon the precomputed homography be-tween image plane and ground plane. We use a subscript to represent this case, i.e.

$$(\mathbf{z}_i^l)_g(k) = (\mathbf{F})_g\mathbf{x}^l(k) + (\mathbf{v}_i^l)_g(k)$$

$$\text{and} \quad (\mathbf{F})_g = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{3}$$

Case 2: $\mathbf{z}_i^l(k)$ is the observed target position on the image plane of C. To differentiate with Case 1, we use a subscript c, i.e.

$$(\mathbf{z}_i^l)_c(k) = (\mathbf{F}_i)_c\mathbf{x}^l(k) + (\mathbf{v}_i^l)_c(k)$$

$$\text{and} \quad (\mathbf{F}_i)_c = \begin{bmatrix} (f_{11})_i & (f_{12})_i & 0 & 0 \\ (f_{21})_i & (f_{22})_i & 0 & 0 \end{bmatrix} \tag{4}$$

where $\mathcal{F}_i = \begin{bmatrix} (f_{11})_i & (f_{12})_i \\ (f_{21})_i & (f_{22})_i \end{bmatrix} \in \{\mathbf{F}: \mathbb{R}^2 \to \mathbb{R}^2\}$ denotes the mapping from ground plane to the image plane of C.[1] □

Our special implementation of the Kalman-Consensus dis-tributed tracking algorithm is presented in Algorithm 1. We de-scribe it for the general system model of (1) and (2) and is ap-plicable for the two special cases described previously. This al-gorithm is performed in a distributed fashion by each camera node C. At each time step k and for each target T, we assume we are given the prior estimated target state $\mathbf{x}_i^l$ and the error covariance matrix $\mathbf{P}_i^l$ at k using measurements up to and in-cluding time $(k-1)$. At time step $k=0$, the Kalman-Con-sensus filter is initialized with $\mathbf{P}_i^l = \mathbf{P}_0$ and $\mathbf{x}_i^l = \mathbf{x}_0^l =$ average of $(\mathbf{z}_i^l)(0)$'s of cameras viewing T.

Algorithm 1 Distributed Kalman-Consensus tracking algorithm performed by every C at discrete time step k. The state estimate of T by C is represented by $\mathbf{x}_i^l$ with error covariance matrix $\mathbf{P}_i^l$ (see Section IV-A).

```
Input: $\mathbf{x}_i^l$ and $\mathbf{P}_i^l$ valid at $k$ using measurements from time step $k-1$
for each $T$ that is being viewed by $\{C_i \cup C_i^n\}$ do
    Obtain measurement $\mathbf{z}_i^l$ with covariance $\mathbf{R}_i^l$
    Compute information vector and matrix
        $\mathbf{u}_i^l = \mathbf{F}_i^{l\,T} (\mathbf{R}_i^l)^{-1} \mathbf{z}_i^l$
        $\mathbf{U}_i^l = \mathbf{F}_i^{l\,T} (\mathbf{R}_i^l)^{-1} \mathbf{F}_i^l$
    Send messages $\mathbf{m}_i^l = (\mathbf{u}_i^l, \mathbf{U}_i^l, \mathbf{x}_i^l)$ to neighboring cameras $C_i^n$
    Receive messages $\mathbf{m}_j^l = (\mathbf{u}_j^l, \mathbf{U}_j^l, \mathbf{x}_j^l)$ from all cameras $C_j \in C_i^n$
    Fuse information matrices and vectors
        $\mathbf{y}_i^l = \sum_{j \in (C_i \cup C_i^n)} \mathbf{u}_j^l, \quad \mathbf{S}_i^l = \sum_{j \in (C_i \cup C_i^n)} \mathbf{U}_j^l.$    (5)
    Compute the Kalman-Consensus state estimate
        $\mathbf{M}_i^l = ((\mathbf{P}_i^l)^{-1} + \mathbf{S}_i^l)^{-1}$
        $\hat{\mathbf{x}}_i^l = \bar{\mathbf{x}}_i^l + \mathbf{M}_i^l(\mathbf{y}_i^l - \mathbf{S}_i^l\bar{\mathbf{x}}_i^l) + \gamma\mathbf{M}_i^l \sum_{j \in C_i^n} (\bar{\mathbf{x}}_j^l - \bar{\mathbf{x}}_i^l)$
        $\gamma = 1/(\|\mathbf{M}_i^l\| + 1), \quad \|\mathbf{X}\| = (tr(\mathbf{X}^T\mathbf{X}))^{\frac{1}{2}}.$    (6)
    Propagate the state and error covariance matrix from time step $k$ to $k+1$
        $\mathbf{P}_i^l \leftarrow \mathbf{A}^l\mathbf{M}_i^l\mathbf{A}^{l\,T} + \mathbf{B}^l\mathbf{Q}^l\mathbf{B}^{l\,T}$
        $\bar{\mathbf{x}}_i^l \leftarrow \mathbf{A}^l\hat{\mathbf{x}}_i^l.$    (7)
end for
```

[1]As homography is applied on homogeneous coordinates, the mapping from ground plane to the image plane is nonlinear, and is a linear approximation. Since $\mathcal{F}: \mathbb{R}^2 \to \mathbb{R}^2$, is invertible.

Comparing with the Kalman filter with centralized fusion (i.e., all the cameras send their measurements to a central pro-cessor, and tracking is preformed centrally, see Appendix A), we can see the fundamentals of Kalman-Consensus tracking al-gorithm described in Algorithm 1. If C is viewing a target T, it obtains T's measurement $\mathbf{z}_i^l$ and computes the corresponding

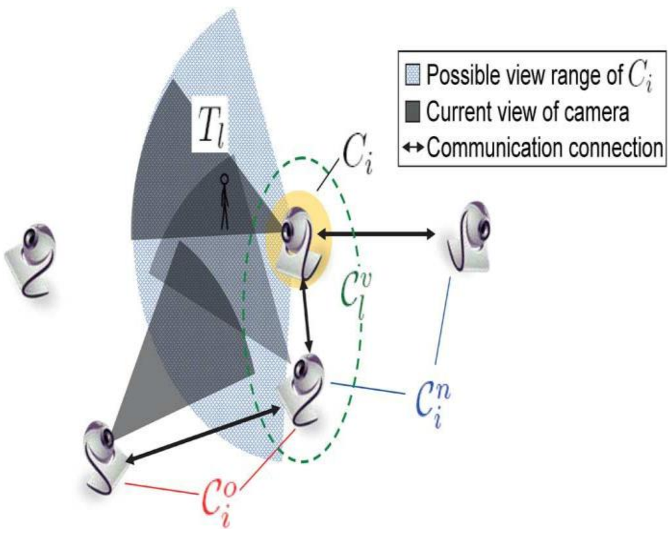


Fig. 1. Conceptual illustration of camera network topologies. $\mathcal{C}_l^v$ is the subset of all cameras viewing target l and the rest of the cameras are $\mathcal{C}_{v-l}$. $\mathcal{C}_i^n$ is the set of *neighboring cameras* of C_i and defined as all the cameras with which C_i is able to communicate. $\mathcal{C}_i^o$ is the set of *overlapping cameras* of C_i, and is defined as all the cameras with which C_i can *potentially* have an overlapping field of view.

information vector $\mathbf{u}_i^l$ and matrix $\mathbf{U}_i^l$. Similar to [31], we define the information matrix and vector of $C_i \in \mathcal{C}_{v-l}$ as $\mathbf{U}_i^l = 0$ and $\mathbf{u}_i^l = 0$ by assuming that their output matrices are zero, i.e., $\mathbf{F}_i^l = 0$ for all $C_i \in \mathcal{C}_{v-l}$ to avoid any ambiguity arising from the lack of measurements in these cameras. If $C_i \in \mathcal{C}_v^l$ and the communication graph for $\mathcal{C}_v^l$ is fully connected, such that C_i can receive information from all the other cameras viewing the same target, by fusing information vectors and matrixes, the local state estimation at C_i is the same as central estimation. However, in the more typical situation, the neighbors of each cameras are different; therefore, at each time instant the infor-mation each camera receives to fuse may also be different. There is no guarantee that the state estimates at different cameras re-main cohesive. Thus, a consensus step is implemented right as part of the estimation step. By comparing the fusion step (5) and Kalman-consensus state estimation step (6) in Algorithm 1 with the centralized state estimation (26) in Appendix A, it can be seen that our Kalman-consensus filter is essentially a distributed implementation of the centralized case with the consideration of communication constraint by adding a consensus term in (6). It is proved in [28] that all estimators asymptotically reach an un-

biased consensus, i.e., $\mathbf{x}_1 = \ldots = \mathbf{x}_N = \mathbf{x}$.

As shown in Algorithm 1, the information vector $\mathbf{u}_i$ and $\mathbf{U}_i$ exchanged between camera nodes are computed with measure-ment $\mathbf{z}_i^l$, covariance matrix $\mathbf{R}_i^l$ and output matrix $\mathbf{F}_i^l$. Consider the two cases of measurement $(\mathbf{z}_i^l)_g$ and $(\mathbf{z}_i^l)_c$ as in (3) and (4). We denote their corresponding information vector and matrix as $(\mathbf{u}_i)_g (\mathbf{U}_i)_g$ and $(\mathbf{u}_i)_c (\mathbf{U}_i)_c$, respectively. The following shows that $(\mathbf{u}_i)_c = (\mathbf{u}_i)_g$ and $(\mathbf{U}_i)_c = (\mathbf{U}_i)_g$.

Recall that $(\mathbf{z}_i^l)_g$ and $(\mathbf{z}_i^l)_c$ are the measurements on ground plane and on the image plane of C_i, respectively and $\bar{F}_i$ is the mapping from ground plane to the image plane. It is obvious that

$$
\begin{aligned}
&(\mathbf{z}_i^l)_c = \bar{F}_i (\mathbf{z}_i^l)_g \\
&\Rightarrow (\mathbf{F}_i)_c \mathbf{x}^l + (\mathbf{v}_i^l)_c = \bar{F}_i (\mathbf{F})_g \mathbf{x}^l + \bar{F}_i (\mathbf{v}_i^l)_g \\
&\qquad - \text{From (3) and (4).}
\end{aligned}
\tag{8}
$$

Then

$$
\begin{aligned}
&(\mathbf{F}_i)_c = \bar{F}_i (\mathbf{F})_g \\
&(\mathbf{v}_i^l)_c = \bar{F}_i (\mathbf{v}_i^l)_g \Rightarrow (\mathbf{R}_i^l)_c = \bar{F}_i (\mathbf{R}_i^l)_g \bar{F}_i^T \\
&\qquad - \text{from (8) and definition of} \\
&\qquad \text{covariance matrix}
\end{aligned}
\tag{9}
$$

So the information vector and matrix are

$$
\begin{aligned}
(\mathbf{u}_i)_c &= (\mathbf{F}_i^l)_c^T (\mathbf{R}_i^l)_c^{-1} (\mathbf{z}_i^l)_c \\
&= \left(\bar{F}_i (\mathbf{F}_i^l)_g\right)^T \left(\bar{F}_i (\mathbf{R}_i^l)_g \bar{F}_i^T\right)^{-1} \bar{F}_i (\mathbf{z}_i^l)_g \\
&\quad - \text{substituting (8) and (9)} \\
&= (\mathbf{F}_i^l)_g^T \bar{F}_i^T \left(\bar{F}_i^T\right)^{-1} (\mathbf{R}_i^l)_g^{-1} \bar{F}_i^{-1} \bar{F}_i (\mathbf{z}_i^l)_g \\
&= (\mathbf{F}_i^l)_g^T (\mathbf{R}_i^l)_g^{-1} (\mathbf{z}_i^l)_g \\
&= (\mathbf{u}_i)_g
\end{aligned}
\tag{10}
$$

and

$$
\begin{aligned}
(U_i)_c &= (\mathbf{F}_i^l)_c^T (\mathbf{R}_i^l)_c^{-1} (\mathbf{F}_i^l)_c \\
&= \left(\bar{F}_i (\mathbf{F}_i^l)_g\right)^T \left(\bar{F}_i (\mathbf{R}_i^l)_g \bar{F}_i^T\right)^{-1} \bar{F}_i (\mathbf{F}_i^l)_g \\
&\quad - \text{substituting (8) and (9)} \\
&= (\mathbf{F}_i^l)_g^T \bar{F}_i^T \left(\bar{F}_i^T\right)^{-1} (\mathbf{R}_i^l)_g^{-1} \bar{F}_i^{-1} \bar{F}_i (\mathbf{F}_i^l)_g \\
&= (\mathbf{F}_i^l)_g^T (\mathbf{R}_i^l)_g^{-1} (\mathbf{F}_i^l)_g \\
&= (U_i)_g .
\end{aligned}
\tag{11}
$$

Since the information message exchanged between cameras are the same for both cases of $\mathbf{z}_i^l$, whether the measurement $\mathbf{z}_i^l$ is measured on ground plane or image plane does not affect the tracking algorithm; these two cases give the same result.

C. *Handoff and Fault Tolerance*

Through this algorithm, each C_i has a consensus-based ground plane state estimate of each target that is being viewed by the cameras with which C_i can exchange information directly or indirectly, even if C_i has never seen some of the targets. Since we are assuming that the network of cameras as a whole is always covering the entire area under surveillance, each target will always be seen by at least one camera. Also, by our definition of overlapping cameras, a target T_l will always move from one camera C_i's FOV to the FOV of an overlapping camera $C_j \in \mathcal{C}_i^o$. Moreover, by **Assumption** $*$, C_i can exchange information with its overlapping cameras, $\mathcal{C}_i^o$, directly or via other cameras. Therefore, C_j can take over the tracking of T_l and find the target correspondence in a seamless way since it had knowledge of T_l's ground plane position through the consensus-tracking before it even entered its FOV. Additional target features could be used to find the target correspondences in a cluttered scene.

Another advantage of the fact that cameras have knowledge of all the targets in their neighborhood is that in the event of a sudden failure of camera node C, the targets that were viewed by C are not suddenly lost by the camera network.

We have also considered the fact that a camera may take a short amount of time to change its parameters to a new position in a nonstatic camera network. If no camera is viewing the target for the short amount of time it takes for the cameras to come to a new set of parameters to cover the entire area, the target state estimate and covariance continue to propagate by (7). This does not translate to a significant decrease in tracking performance as seen in our experiments.

D. *Experimental Results*

We tested our approach for tracking in a real camera network composed of 10 PTZ cameras looking over an outdoor area of approximately 10000 sq. feet. In the area under surveillance, there were eight targets in total that were to be tracked using our distributed Kalman-Consensus filtering approach. In our experiment, the measurements (i.e., the observed positions of targets) are obtained using histogram of gradient (HOG) human detector [5]. The association of measurements to targets is achieved based upon appearance (color) and motion infor-mation. Fig. 2 shows the tracking results as viewed by each camera at four time instants.

The results are shown on a nonstatic camera network. The cameras are controlled to always cover the entire area under surveillance through a game theoretic control framework we proposed in [38]. As explained previously, the change of camera settings does not affect the procedure of the Kalman-consensus filter. Fig. 2(a) shows the initial settings of the camera network that covers the entire area. As the targets are observed in this area, the single-view tracking module in each camera deter-mines the ground plane position of each target in its FOV and sends that information to the Kalman-Consensus filter which processes it together with the information received from the Kalman-Consensus filters of neighboring cameras as described in Section IV.

Fig. 2(b) shows the instant when a camera C_6 is focused on a target T_h. Fig. 2(b) and (c) shows the dynamics of the targets in the camera network. All targets are tracked using the Kalman-consensus scheme, although we show the marked track for only one target. The handoff of T_h is clearly shown in Fig. 2(d) from C_6 to C_3. It is to be noted that every time a target goes from one camera's FOV into another one, or when a camera changes its parameters, the network topologies for the targets, i.e., $\mathcal{C}_v$ and $\mathcal{C}_{v-}$, also change.

Fig. 3(a) shows the distributed Kalman-Consensus tracks for the eight targets. The measurements of the different cameras are shown in a light gray color. As can be seen, the Kalman-Consensus filter in each camera comes to a smooth estimate of the actual state for each target.

Fig. 3(b) shows the distributed tracking results on the ground plane for one of the targets, T. The dots correspond to the ground plane measurements from different cameras viewing the target while the solid line is the consensus-based estimate. As can be expected, the individual positions are different for each camera due to calibration and single-view tracking inaccura-cies. As can be seen clearly, even though $\mathcal{C}_v$ is time varying, the Kalman-Consensus filter estimates the target's position seam-lessly at all times.

In Fig. 3(a) and (b), the cameras that are viewing the same target can communicate with each other directly, i.e., $\mathcal{C}_v$ is a fully connected graph. As shown in Section IV-B, the results are exactly the same as a centralized case similar to each cluster of [22]. We denote the results of this fully connected case as KCF1. In order to show the effect of the network communication topology on the Kalman-consensus tracking, we consider an example of a partially connected network, which is shown on the right-top of Fig. 3(c). Compared to the fully connected one, direct communication does not exist between camera 1 and camera 3, neither between camera 4 and camera 8. Fig. 3(c) shows the KCF tracking results at Camera 1 for this case, which is denoted as KCF2. It is slightly different with KCF1, due to the difference of fused information. The consensus method is guaranteed to have the same result as centralized case if there are no limitations on the communication capabilities. In the case of partial connection between cameras, KCF will converge to the same estimate centralized result as the number of consensus iterations goes to infinity [28]. However, the limited communication will result in differences from the centralized result for finite steps [as shown in Fig. 3(c)]. However, even in this case, the consensus result is better than that obtained at each individual camera, as shown in Fig. 3(d) and explained in the following.

In order to measure tracking performance, we compare the tracking results with the groundtruth trajectory, which is shown in Fig. 3(c). In the table at the bottom, we show the minimum, maximum and average distances to the groundtruth of KCF1, KCF2, and individual camera tracks. It can be seen that KCF1 performs best and KCF2 is better than individual camera tracks. We also look at the output error covariance matrix $\mathbf{P}$ of the Kalman filter. The higher the trace of $\mathbf{P}$ is, the lower the tracking accuracy is. Fig. 3(d) shows the traces of the covariance matrix of the tracking error for the same target as in Fig. 3(b) and (c). The colored lines with symbols correspond to tracking results from different cameras using their own measurements only (as each camera runs an independent Kalman filter), while the solid black line is the result of consensus-based estimate for the fully connected case (which will be the same for the centralized case) and dashed purple line is for the partially connected one. As can be seen clearly, the Kalman-Consensus filter with full connec-tion performs the best, and partially connected one does better than individual Kalman filters without consensus.

V. Distributed Activity Recognition Through Consensus

In this section, we consider the problem of activity recog-nition in a camera network where processing power is distributed across the network and there is no central pro-cessor accumulating and analyzing all the data. Each camera computes a similarity measure of the observed activities in its views against a dictionary of predefined activities. Also, the transition probability between activities is known. This is a common assumption used in many activity recognition approaches and can be learned *a priori* from training data [4], [7], [18], [25], [36]. If no such information is available, the transition matrix can be assumed to be uniform. Based

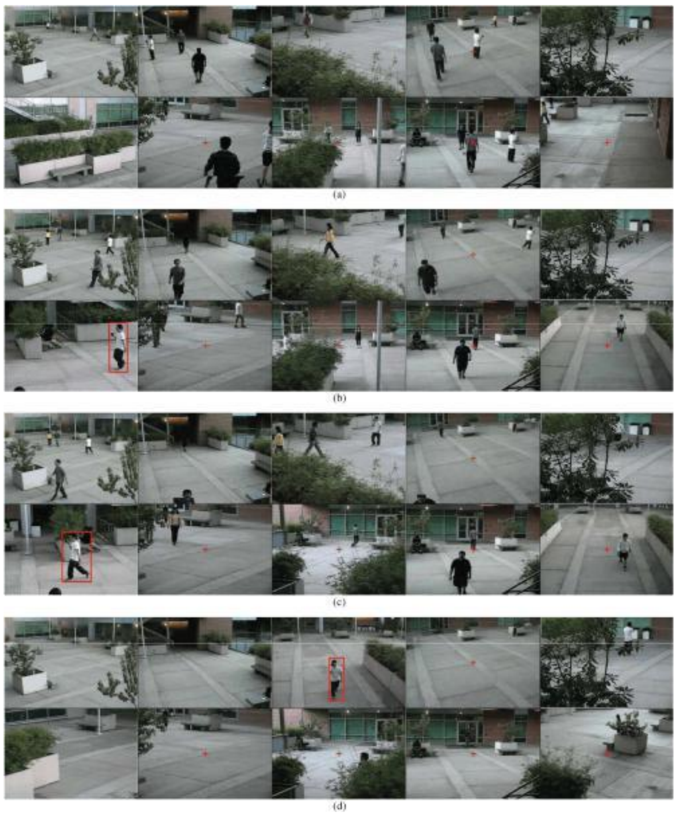


Fig. 2. Each subfigure shows ten cameras at one of four time instants denoted by . The track of one target, marked with a box, is shown. All targets are tracked using the Kalman-Consensus filtering approach, but are not marked for clarity. (a) ; (b) ; (c) ; (d)

upon the computed similarities at each camera node and the learned transition matrices, we show how to compute the consensus estimate in a probabilistic framework. Essentially, the consensus is a probability of similarity of the observed activity against the dictionary taking into account the deci-sions of the individual cameras.

A. Problem Formulation and Main Result

Let us assume that there are N_c cameras viewing a person per-forming some actions. The observation of camera C_i in the kth
time interval is denoted as $O_i(k)$, $i = 1 \ldots N_c$. Let $\mathbf{O}(k)$ be the collection of observations from all the cameras, i.e., $\mathbf{O}(k) = \{O_1(k), \ldots, O_{N_c}(k)\}$. Its history is $\mathcal{O}^k = \{\mathbf{O}(1) \ldots \mathbf{O}(k)\}$, ;
The problem of activity recognition can be formulated so as to estimate the conditional probability, $P(y(k)|\mathcal{O}^k)$, where $y(k) \in \{1, \ldots, Y\}$ is the label of the class of activity in a dictionary of Y activities with history $\mathcal{Y}^k = \{y(1) \ldots y(k)\}$. ,

It is a somewhat general assumption that the state transitions of activity class y are governed by the transition matrix for a first-order Markov chain [36]

$$\begin{aligned} P\left(y(k) = a | y(k-1) = a', \mathcal{Y}^{k-2}\right) &= P\left(y(k) = a | y(k-1) = a'\right) \\ &= m(a', a). \end{aligned} \quad (12)$$

$m(a', a)$ can be learned *a priori* from training data; if no such information is available, the transition matrix can be assumed to be uniform. Given $y(k)$, observation $\mathbf{O}(k)$ is assumed to be independent of other observations and states, i.e.

$$P\left(\mathbf{O}(k)|\mathcal{Y}^k, \mathcal{O}^{k-1}\right) = P\left(\mathbf{O}(k)|y(k)\right). \quad (13)$$

Based upon Bayes' rule and the previously shown Markov chain assumption, we can show that the following relationship holds (see Appendix B for proof):

Result 1:

$$\begin{aligned} P\left(y(k)|\mathcal{O}^k\right) = &\frac{1}{P\left(\mathbf{O}(k)|\mathcal{O}^{k-1}\right)} \\ &\cdot \prod_{j=1}^{N_c} P\left(O_j(k)|y(k)\right) \\ &\cdot \left(\sum_{y(k-1)} P\left(y(k)|y(k-1)\right) \right. \\ &\left. \times P\left(y(k-1)|\mathcal{O}^{k-1}\right) \right) \end{aligned} \quad (14)$$

,

where $\sum_{y(k)}$ mean summing over all values of $y(k) = 1, \ldots, Y$. $\square$

Analysis of Result 1: By observing the righthand side of (14), we notice that $P(O_j(k)|y(k))$, $j = 1 \ldots N_c$ is the likelihood of camera C_j 's observation. The first term of the righthand side is a constant with respect to $y(k)$, so that it can be treated as a normalization factor and denoted by $\gamma(k)$, i.e.

$$\gamma(k) \triangleq \frac{1}{P\left(\mathbf{O}(k)|\mathcal{O}^{k-1}\right)}.$$

So we rewrite (14) as

$$\begin{aligned} P\left(y(k)|\mathcal{O}^k\right) = &\gamma(k) \prod_{j=1}^{N_c} P\left(O_j(k)|y(k)\right) \\ &\cdot \left(\sum_{y(k-1)} P\left(y(k)|y(k-1)\right) P\left(y(k-1)|\mathcal{O}^{k-1}\right) \right). \end{aligned} \quad (15)$$

We define the state of the activity at camera C_i as $\mathbf{w}_i = [w_i^1, w_i^2, \cdots, w_i^Y]^T$, where

$$w_i^a \triangleq P\left(y(k) = a|\mathcal{O}^k\right), \quad a = 1, \ldots, Y.$$

The likelihood of camera C_i 's observation is denoted by $\mathbf{v}_i = [v_i^1, v_i^2, \ldots, v_i^Y]^T$, where

$$v_i^a \triangleq P\left(O_i(k)|y(k) = a\right), \quad a = 1, \ldots, Y.$$

Thus

$$\begin{aligned} w_i^a(k) = &\gamma(k) \prod_{j=1}^{N_c} P\left(O_j(k)|y(k) = a\right) \\ &\cdot \left(\sum_{y(k-1)} P\left(y(k) = a|y(k-1) = a'\right) \right. \\ &\left. \times P\left(y(k-1) = a'|\mathcal{O}^{k-1}\right) \right) \\ = &\gamma(k) \prod_{j=1}^{N_c} v_j^a(k) \left(\sum_{a'=1}^{Y} m(a', a) w_i^{a'}(k-1) \right). \end{aligned} \quad (16)$$

Algorithm 2 Distributed Consensus based activity recognition algorithm performed by every C at step k.

```
Input: $\mathbf{w}_i(k\_1)$
for each person that is being viewed by $\{C_c \cup C\}$ do
    Obtain observations $O_i(k)$
    Compute local likelihood
        $\mathbf{v}_i(k) = \begin{bmatrix} v_i^1(k) \\ \vdots \\ v_i^Y(k) \end{bmatrix} = \begin{bmatrix} P(O_i(k)|y(k)=1) \\ \vdots \\ P(O_i(k)|y(k)=Y) \end{bmatrix}$
    Send $\mathbf{v}_i(k)$ to neighboring cameras $C_{n_i}$
    Receive $\mathbf{v}_j(k)$ from all cameras $C$  $\in C_i^n$
    Fuse information to estimate activity state see equation
    at the bottom of next page, where $\mathbf{M}$ is a $Y \times Y$ matrix
    with $(ij)$ th element to be $m(ij)$ ,
        $\Lambda(\mathbf{v}_j(k)) = \begin{bmatrix} v_j^1(k) & & \\ & \ddots & \\ & & v_j^Y(k) \end{bmatrix}$
    $Y$  $\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$
    Repeat
      Send $\mathbf{w}_i(k)$ to neighboring cameras $C_{n_i}$
      Receive $\mathbf{w}_j(k)$ from all cameras $C$  $\in C_i^n$
      Compute the Consensus state estimate
        $\bar{\mathbf{w}}_i(k) = \mathbf{w}_i(k) + \epsilon \sum_{j \in C_i^n} (\mathbf{w}_j(k) - \mathbf{w}_i(k))$
    until either a predefined iteration number is reached
    or $\sum_{j \in C_i^n}(\mathbf{w}_j(k)\_\mathbf{w}_i(k))$ is smaller than a predefined
    small value
end for
```

Based upon the previously mentioned argument, we have the activity recognition algorithm described in Algorithm 2 for each camera in the network.

Regarding the normalization factor $\gamma(k)$, we have the fol-lowing result (see Appendix C for details).

Result 2:

$$\gamma(k) = \left[\sum_{y(k)} \prod_{j=1}^{N_c} P(O_j(k)|y(k)) \cdot \left(\sum_{y(k-1)} P(y(k)|y(k-1))\, P\left(y(k-1)|O^{k-1}\right) \right) \right]^{-1}.$$

□

Result 3: The local activity recognition procedure for node i based upon fusion of the recognition results in all the cameras is

$$v_i^a(k) = P(O_i(k)|y(k)=a), \quad a = 1, \ldots, Y$$
$$w_i^a(k) = \gamma(k) \prod_{j=1}^{N_c} v_j^a(k) \left(\sum_{a'=1}^{Y} m(a',a) w_i^{a'}(k-1) \right)$$
$$a = 1, \ldots, Y$$
$$\gamma(k) = \left(\sum_{a=1}^{Y} \prod_{j=1}^{N_c} v_j^a(k) \times \left(\sum_{a'=1}^{Y} m(a',a) w_i^{a'}(k-1) \right) \right)^{-1}. \quad (17)$$

□

The proof of Result 3 follows directly from Results 1 and 2.

Based upon the network topology defined in Section IV-A, each camera can only communicate with its neighbors. Ac-cording to this local activity recognition algorithm, there is no guarantee that the estimates remain cohesive among nodes. We use an ad hoc approach by implementing a consensus step right after the estimation step to reduce the disagreement regarding the estimates obtained in Result 3, from which Algorithm 2 can be inferred. This consensus approach is similar to the one proposed in [28] for the Kalman-Consensus filtering. However, a number of iterations are done in each time segment so as to converge to a consensus estimate.

The cameras that exchange information in the consensus stage are defined based upon the communication constraints; therefore, it is possible that a camera involved in the consensus does not view the activity. In this case, such a camera transmits a value of $\mathbf{v}_i = (1/Y)\mathbf{1}_Y$, i.e., by assuming equal likelihood for all possible action classes.

B. Experimental Evaluation

To validate our proposed consensus approach for activity recognition, we carried out an experimental evaluation. We did activity recognition using multiple cameras and came to a consensus about the actions taking place using the theory of Section V-A.

For this, we used the IXMAS dataset [45]. In the dataset, there are sequences of images of different people doing sev-eral actions. The extracted silhouettes of the people in those ac-tions are also given in the dataset. Five cameras were used to capture the whole activity which were placed at pan and tilt angles of $(120^\circ, 10^\circ)$, $(90^\circ, 10^\circ)$, $(30^\circ, 30^\circ)$, $(0^\circ, 10^\circ)$ and ; $(30^\circ, 90^\circ)$, where 0° pan angle means looking at a person from the front and 90° means to look at him from the left. A 3-D motion-model of each person doing the actions is also given which has approximately 3500 voxels on a person.

We used the 3-D motion-model as our training data and the silhouettes extracted from each camera as our test data. To build our training database, we took the orthographic projection

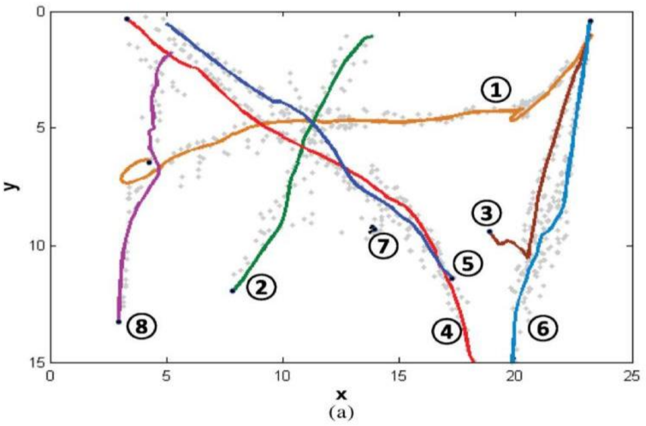
$$\mathbf{w}_i(k) = \begin{bmatrix} w_i^1(k) \\ \vdots \\ w_i^Y(k) \end{bmatrix}$$

$$= \begin{bmatrix} \gamma(k)\prod_{j\in(C_i\cup C_i^n)} v_j^1(k)\left(\sum_{a'=1}^{Y} m(a',1)\bar{w}_i^{a'}(k-1)\right) \\ \vdots \\ \gamma(k)\prod_{j\in(C_i\cup C_i^n)} v_j^Y(k)\left(\sum_{a'=1}^{Y} m(a',Y)\bar{w}_i^{a'}(k-1)\right) \end{bmatrix}$$

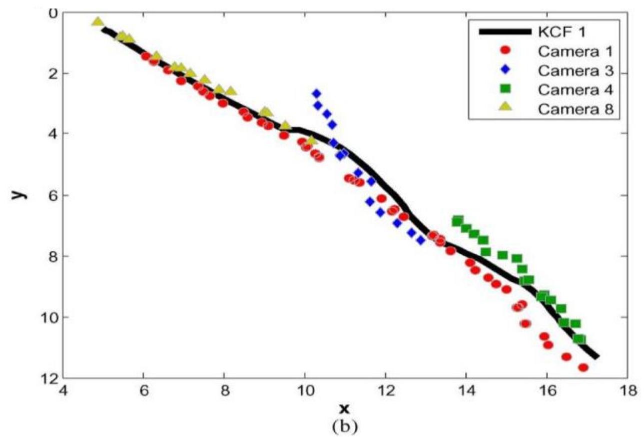
$$= \gamma(k) \prod_{j\in(C_i\cup C_i^n)} \Lambda(\mathbf{v}_j(k))\,\mathbf{M}^T\,\bar{\mathbf{w}}_i(k-1)$$

$$\gamma(k) = \left(\sum_{a=1}^{Y} \prod_{j\in(C_i\cup C_i^n)} v_j^a(k)\left(\sum_{a'=1}^{Y} m(a',a)\bar{w}_i^{a'}(k-1)\right)\right)^{-1}$$

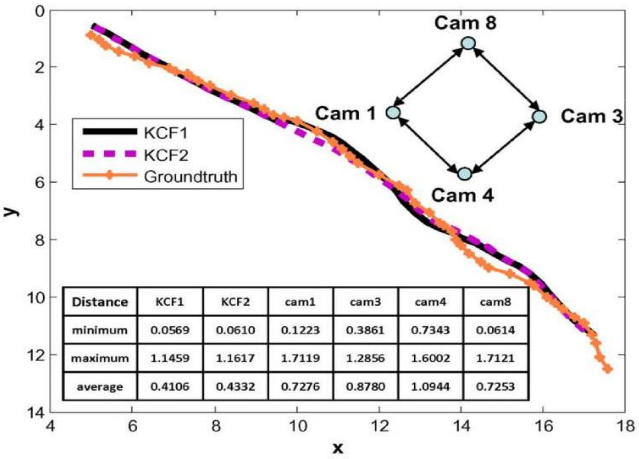
$$= \left(\mathbf{1}_Y^T \cdot \prod_{j\in(C_i\cup C_i^n)} \Lambda(\mathbf{v}_j(k))\,\mathbf{M}^T\,\bar{\mathbf{w}}_i(k-1)\right)^{-1}$$



(a)

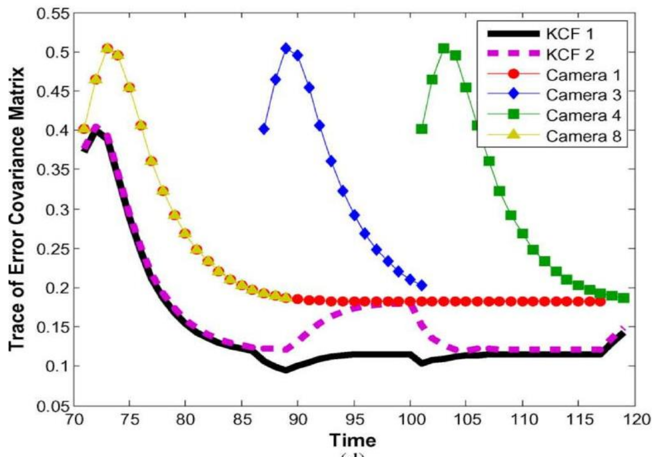


(b)



Distance	KCF1	KCF2	cam1	cam3	cam4	cam8
minimum	0.0569	0.0610	0.1223	0.3861	0.7343	0.0614
maximum	1.1459	1.1617	1.7119	1.2856	1.6002	1.7121
average	0.4106	0.4332	0.7276	0.8780	1.0944	0.7253

(c)



(d)

Fig. 3. Tracking results. (a): Distributed Kalman-Consensus tracking trajectories for 8 targets. Measurements from all cameras are shown in a light gray color.
(b): Tracking results on the ground plane for one of the targets 5. In (a) and (b), the cameras that are viewing the same target can communicate with each other
directly, i.e., v is a fully connected graph. The results are exactly same as centralized case. We denote the results of this full connection as KCF1. (c): KCF tracking results at Camera 1 given an example of a partially connected camera network, which is shown on the top-right. This case is denoted as KCF2. We can see that Cam (1,3) and Cam (4,8) cannot communicate. The groundtruth trajectory is also marked. The comparison of tracking performances (minimum, maximum and average distances to the groundtruth) of KCF1, KCF2 and individual camera tracks are shown in the table at the bottom. (d): Trace of the error covariance of the tracking results for the same target shown in (b) and (c)

of the 3-D voxels on an image plane by rotating our virtual camera around the model with pan angles from 0o to 330o in increments of 30o and for each pan angle we used tilt angles of 10o and 30o . The actions we used in our experiments from the dataset are: looking at watch, scratching head, sit, wave hand, punch, kick and pointing a gun. These are later referred to as Actions 1 through 7. For each action and each camera viewpoint, we extracted the shape silhouette using 40 landmarks, i.e. 40 uniformly distributed contour points per shape in each frame. In a similar fashion we extracted the shape sequences of the test data, i.e. the silhouettes from different camera views.

For matching two shape sequences, we used a shape-based activity recognition algorithm based upon work in [43]. The dis-tance between two shape sequences is measured by comparing the mean shapes of the two sequence. Then we took the recip-rocal of the distance measure to get a similarity measure be-tween two shape sequences and normalized the similarity mea-sures to convert them to probabilities. A block diagram of the overall activity recognition process is given in Fig. 4.

The activity recognition is performed individually in each of the cameras depending upon its own current observation. In our experiment, we have five cameras, i.e. cam0, cam1, cam2, cam3, and cam4. We consider a network topology where the net-work is not a full mesh, rather each camera is connected to two other cameras only. So, after the activity recognition stage, each camera shares the detection result with its immediate neighbor. Each camera fuses the detection results of itself and its neigh-
bors, final detection result from the previous time step k–1, and the transition probabilities between different actions, and gets a new probability distribution of the actions. After this

stage, the cameras initiate the consensus algorithm and try to converge to the same detection results.

In Fig. 5, we show the similarity matrices, i.e. the proba-bility of match for each test activity (the row of a matrix). The more white the cell block is, the test data it refers to is detected

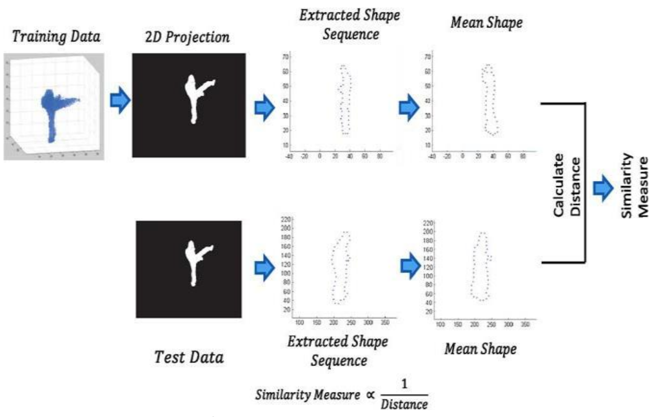


Fig. 4. Block diagram of the activity recognition process. For training using the 3-D action models, orthographic projections were taken for different viewing angles. From the projections, shape sequences were extracted from which the mean shape was calculated for each viewing angle. For testing, in similar way the mean shape were extracted. The Euclidean distance between the mean shapes were computed by comparing the test mean shape to all the training mean shapes. The ones with the lowest distance was selected for each action in the dictionary. Taking the reciprocal of the distance measure and normalizing it so that the sum of all the similarities is 1 gave the similarity measure of the actions.

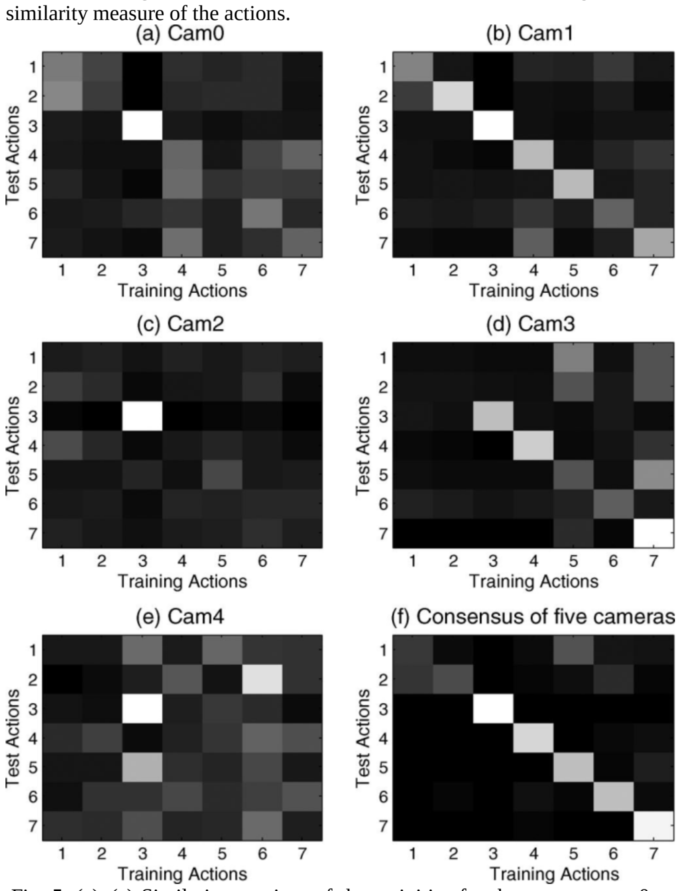


Fig. 5. (a)–(e) Similarity matrices of the activities for the cameras cam0, cam1, cam2, cam3 and cam4; (f) Similarity matrix of the activities for the consensus of all these cameras. Actions 1 through 7 are looking at watch, scratching head, sit, wave hand, punch, kick and pointing a gun, respectively, all from the IXMAS dataset.

with more probability as that action. Five of the images rep-resent the similarity matrix for the test data captured by each camera and the sixth image shows the similarity matrix of the consensus for all of these cameras. The similarity scores of cor-rect matching are the diagonal values of the similarity matrix. Comparing with other values in the matrix, the higher the di-agonal values (brighter in the image) are, the less confusing the recognition result is. By comparing the similarity matrix of con-sensus with the test data captured by each camera [compare (f) with (a)–(e)], it is clear that the recognition result after con-sensus has less confusion than others.

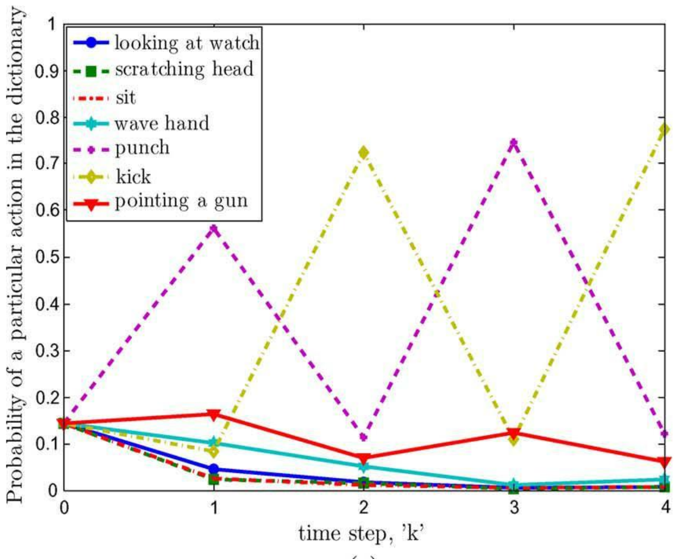
Next, in Fig. 6(a), we show a graphical representation of the final detection result for a sequence of punch-kick-punch-kick, by plotting the result of the consensus stage in each time step. The vertices in each line at each time step, represent the prob-ability of a particular action in the dictionary. It was assumed that in time step $k=0$, all the activities were equally likely. As an example, we use a nonuniform transition matrix where there is high transition probability between punch and kick, and there is also some moderately high transition probability be-tween looking at watch, scratch, sit, wave hand and point. The transition matrix between different actions is shown in Fig. 6(b). In practice, if some prior knowledge is available, the transition matrices can be learned/manually set. As the transition proba-bility between punch and kick is high, it can be seen that the recognition result (after consensus) keeps on improving.

Finally, we generate a statistics to observe the performance of the probability of correct match for individual cameras versus their consensus. We use every possible subset of the five cam-eras by considering five, four, three and two cameras to deter-mine their consensus and show that the consensus result is better than an individual camera, on average. This result shows the fault tolerance aspect of the consensus process. The result is shown in Fig. 7.

C. Discussion

- *Experimental Setup*: We did the experiments by running the algorithms on independent threads, one for each camera, and communication between the threads using existing protocols. We assume here that communication is not a bottleneck. This proposed work is a proof-of-concept study in using distributed processing algorithms for video analysis. Future work should consider the practical con-straints of using consensus algorithms in camera networks.
- *Temporal Segmentation for Activity Recognition*: In our distributed activity recognition procedure, the video se-quence is divided into segments, where each segment is treated as a observed variable (image features) and associated with a hidden variable (activity state). In our experiments, in order to provide a clear comparison of our results with ground truth, the observed sequence from each camera is temporally segmented based upon the ground truth. In practice, such a precise segmentation is not required; the observed sequence can be uniformly divided into short segments (e.g., 4 s each). Since our activity recognition results are represented in the form of probabilities, the nondominant recognition results on the segments where activity transitions happen won't affect the recognition on their subsequent segments.

Synchronization: The cameras in the network have been presynchronized, however, the frame synchronization may



(a)

action	looking at watch	scratching head	sit	wave hand	punch	kick	point
looking at watch	0.1804	0.1854	0.1790	0.1825	0.0871	0.1033	0.0823
scratching head	0.1742	0.1758	0.1896	0.1736	0.0991	0.0908	0.0969
sit	0.1914	0.1889	0.1118	0.1949	0.0992	0.1066	0.1071
wave hand	0.2172	0.1055	0.1151	0.2050	0.1093	0.1212	0.1266
punch	0.0807	0.0943	0.0801	0.0984	0.2500	0.2396	0.1569
kick	0.0810	0.0842	0.0818	0.0865	0.2485	0.2481	0.1699
point	0.1480	0.1483	0.1387	0.1364	0.1500	0.1317	0.1469

(b)

Fig. 6. (a): Graphical representation of the final detection result, i.e. the result of the consensus stage in each time step, for the sequence punch-kick-punch-kick. The vertices in each line at each time step represent the probability of a particular action. It was assumed that in time step , all the activities were equally likely. We use a nonuniform transition matrix, as shown in (b), where there are high transition probability between the punch and kick, and there is also some moderately high transition probability between looking at watch, scratch, sit, wave hand and point.

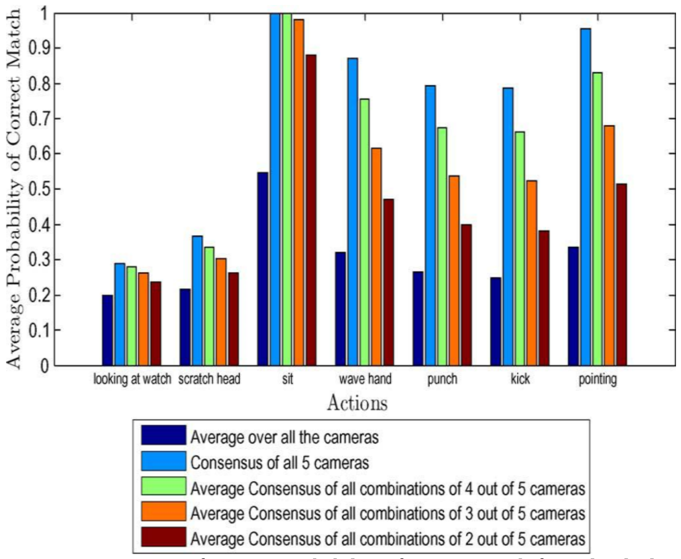


Fig. 7. Comparison of average probability of correct match for individual camera and their consensus for all the activities. Their are seven sets of bars for seven different actions and in each set, there are five bars where the leftmost one (blue) is the average probability of correct match for individual cameras and the next four bars the average probability of correct match of the consensus over all the combinations of cameras taking, respectively five, four, three and two out of five cameras.

not be perfect due to slight frame rate difference between cameras. So the transmitted information between cameras includes a time stamp. In the distributed tracking frame-work, when a camera fuses the information (e.g., state es-timations) from its neighboring cameras, it will do interpo-lation of the information vector $\mathbf{u}$ (in Algorithm 1) as nec-essary. This will ensure that the information being fused is synchronized. While the activity recognition is done on each segment, unlike the frame based Kalman-consensus tracking, a precise synchronization of the cameras is not needed; precision of presynchronization is enough.

- *Selection of Parameters*: We can see that the consensus step in Algorithm 2 is a gradient descent algorithm that minimizes the cost function $g(\mathbf{w}_i) = (1/2)\sum_{j\in C_i^n}(\mathbf{w}_i - \mathbf{w}_j)^2$. The step-size $\epsilon > 0$ should be a small number. The choice of ϵ is based upon reasoning similar to what is used for gradient descent. The simplest way is to set ϵ a fixed small number, while some suggest using an adaptive step-size. In our experiments, the step-size ϵ is fixed at 0.01.
- *Integration of tracking and activity recognition*: Since the distributed tracking and activity recognition can be achieved through analogous frameworks (though the de-tailed fundamentals are different) by estimating locally and fusing through consensus, it is possible to integrate these two by designing integrated local estimation and fusion schemes. We address the integration as a future work.

VI. Conclusion and Future Work

We investigated in this paper distributed scene analysis al-gorithms by leveraging upon concepts of consensus. We ad-dressed two fundamental tasks—tracking and activity recogni-tion in a distributed camera network. We proposed a robust ap-proach to distributed multitarget tracking in a network of cam-eras. A distributed Kalman-Consensus filtering approach was used together with a dynamic network topology for persistently tracking multiple targets across several camera views. A proba-bilistic consensus scheme for activity recognition was provided, which combines the similarity scores of neighboring cameras to come up with a probability for each action at the network level. In the future, we will look at the integration of tracking and ac-tivity recognition into a single framework and more complex activities that span a larger area.

Appendix A
Kalman Filter With Centralized Information Fusion

Consider a Kalman filter with centralized information fusion, i.e., each camera sends its observation to a central processor, and tracking (i.e. state estimation) is performed centrally. As in (2), the sensing model at camera C_i of target T is $\mathbf{z}_i^t = \mathbf{F}_i^t \mathbf{x}^t + \mathbf{v}_i^t$. Thus, the central measurement, observation noise and observa-tion matrix are defined as

$$\mathbf{z}^t = \begin{bmatrix} \mathbf{z}_1^t \\ \mathbf{z}_2^t \\ \vdots \\ \mathbf{z}_{N_c}^t \end{bmatrix}, \quad \mathbf{v}^t = \begin{bmatrix} \mathbf{v}_1^t \\ \mathbf{v}_2^t \\ \vdots \\ \mathbf{v}_{N_c}^t \end{bmatrix}, \quad \mathbf{F}^t = \begin{bmatrix} \mathbf{F}_1^t \\ \mathbf{F}_2^t \\ \vdots \\ \mathbf{F}_{N_c}^t \end{bmatrix} \qquad (18)$$

where N_c is the total number of cameras. Then we get

$$\mathbf{z}^l = \mathbf{F}^l\mathbf{x}^l + \mathbf{v}^l \tag{19}$$

where $\mathbf{x}^l = (x^l, y^l, \dot{x}^l, \dot{y}^l)^T$ is the same as in Section IV-B.

By assuming ^l_i 's are uncorrelated, the covariance matrix of $\mathbf{v}^l$ is

$$\mathbf{R}^l = diag\left(\mathbf{R}^l_1, \mathbf{R}^l_2, \cdots, \mathbf{R}^l_{N_c}\right). \tag{20}$$

Thus, the Kalman filter iterations in the information form are

$$\begin{aligned}\mathbf{M}^l(k) &= \left[\left(\mathbf{P}^l(k)\right)^{-1} + \mathbf{F}^l(k)^T\mathbf{R}^l(k)^{-1}\mathbf{F}^l(k)\right]^{-1} \\ &= \Bigg[\left(\mathbf{P}^l(k)\right)^{-1} \\ &\quad + \sum_{i=1}^{N_c}\mathbf{F}^l_i(k)^T\mathbf{R}^l_i(k)^{-1}\mathbf{F}^l_i(k)\Bigg]^{-1}\end{aligned} \tag{21}$$

$$\mathbf{K}^l(k) = \mathbf{M}^l(k)\mathbf{F}^l(k)^T\mathbf{R}^l(k)^{-1} \tag{22}$$

$$\begin{aligned}\hat{\mathbf{x}}^l(k) &= \bar{\mathbf{x}}^l(k) + \mathbf{K}^l(k)\left(\mathbf{z}^l(k) - \mathbf{F}^l(k)\bar{\mathbf{x}}^l(k)\right) \\ &= \bar{\mathbf{x}}^l(k) + \mathbf{M}^l(k) \\ &\quad \times \left[\mathbf{F}^l(k)^T\mathbf{R}^l(k)^{-1}\mathbf{z}^l(k)\right. \\ &\quad\quad \left. - \mathbf{F}^l(k)^T\mathbf{R}^l(k)^{-1}\mathbf{F}^l(k)\bar{\mathbf{x}}^l(k)\right] \\ &= \bar{\mathbf{x}}^l(k) + \mathbf{M}^l(k) \\ &\quad \times \Bigg[\sum_{i=1}^{N_c}\mathbf{F}^l_i(k)^T\mathbf{R}^l_i(k)^{-1}\mathbf{z}^l_i(k) \\ &\quad\quad - \left(\sum_{i=1}^{N_c}\mathbf{F}^l_i(k)^T\mathbf{R}^l_i(k)^{-1}\mathbf{F}^l_i(k)\right) \\ &\quad\quad \times \bar{\mathbf{x}}^l(k)\Bigg]\end{aligned} \tag{23}$$

$$\mathbf{P}^l(k+1) = \mathbf{A}^l\mathbf{M}^l(k)\mathbf{A}^{lT} + \mathbf{B}^l\mathbf{Q}^l\mathbf{B}^{lT} \tag{24}$$

$$\bar{\mathbf{x}}^l(k+1) = \mathbf{A}^l\hat{\mathbf{x}}^l(k). \tag{25}$$

Denoting the information vector and matrix at camera C_i as $\mathbf{u}^l_i(k) = \mathbf{F}^l_i(k)^T\mathbf{R}^l_i(k)^{-1}\mathbf{z}^l_i(k)$ and $\mathbf{U}^l_i(k) = \mathbf{F}^l_i(k)^T\mathbf{R}^l_i(k)^{-1}\mathbf{F}^l_i(k)$, (23) can be rewritten as

$$\hat{\mathbf{x}}^l(k) = \bar{\mathbf{x}}^l(k) + \mathbf{M}^l(k)\left[\sum_{i=1}^{N_c}\mathbf{u}^l_i(k) - \left(\sum_{i=1}^{N_c}\mathbf{U}^l_i(k)\right)\bar{\mathbf{x}}^l(k)\right]. \tag{26}$$

By comparing (26) with Algorithm 1 where each camera fuses its information vector and matrix and those from its neighbors, it is clearly shown that Kalman-consensus filter is a distributed im-
plementation. If $l/\mathcal{C}_v$ is a fully connected graph, i.e., all cam-eras that are viewing the same target can communicate with each other directly, the Kalman-consensus filter will provide exactly the same result as the Kalman filter with centralized fusion.

Appendix B
Proof of Result 1

Assuming there are N cameras viewing a person per-forming some actions, the observations of camera C_i in kth time interval are denoted as $O_i(k)$, $i=1\ldots N$. Let ; $\mathbf{O}(k)$ be the collection of observations from all the cam-eras, i.e., $\mathbf{O}(k) = \{O_1(k), \ldots, O_{N_c}(k)\}$ and its history is $\mathcal{O}^k = \{\mathbf{O}(1), \ldots, \mathbf{O}(k)\}$. Then the statement

$$P\left(y(k)|\mathcal{O}^k\right) = \frac{1}{P\left(\mathbf{O}(k)|\mathcal{O}^{k-1}\right)}\prod_{j=1}^{N_c}P\left(O_j(k)|y(k)\right) \cdot \left(\sum_{y(k-1)}P\left(y(k)|y(k-1)\right)P\left(y(k-1)|\mathcal{O}^{k-1}\right)\right)$$

holds $\forall N_c \geq 1$.

Proof:

$$P\left(y(k)|\mathcal{O}^k\right) = \frac{P\left(y(k), \mathcal{O}^k\right)}{P(\mathcal{O}^k)} \quad - \text{ from Bayes' rule} \tag{27}$$

We notice that $P(y(k), \mathcal{O}^k)$ is the forward variable in hidden Markov model. According to the recursion of the Forward Al-gorithm, i.e.

$$P\left(y(k), \mathcal{O}^k\right) = \left[\sum_{y(k-1)}P\left(y(k-1), \mathcal{O}^{k-1}\right)P\left(y(k)|y(k-1)\right)\right] \cdot P\left(O(k)|y(k)\right) \tag{28}$$

(27) becomes

$$\begin{aligned}&P\left(y(k), \mathcal{O}^k\right) \\ &\quad= \frac{P\left(\mathbf{O}(k)|y(k)\right)}{P(\mathcal{O}^k)} \\ &\quad\quad \cdot \left[\sum_{y(k-1)}P\left(y(k-1), \mathcal{O}^{k-1}\right)P\left(y(k)|y(k-1)\right)\right] \\ &\quad\quad - \text{substituting (28)} \\ &= \frac{P\left(\mathbf{O}(k)|y(k)\right)}{P(\mathcal{O}^k)} \\ &\quad \cdot \left[\sum_{y(k-1)}P\left(y(k-1)|\mathcal{O}^{k-1}\right)P(\mathcal{O}^{k-1}) \right. \\ &\quad\quad \left. \times P\left(y(k)|y(k-1)\right)\right] \\ &\quad - \text{from Bayes' rule} \\ &= \frac{P(\mathcal{O}^{k-1})P\left(\mathbf{O}(k)|y(k)\right)}{P(\mathcal{O}^k)} \\ &\quad \cdot \left[\sum_{y(k-1)}P\left(y(k-1)|\mathcal{O}^{k-1}\right)P\left(y(k)|y(k-1)\right)\right] \\ &= \frac{P\left(O(k)|y(k)\right)}{P(\mathcal{O}^k)/P(\mathcal{O}^{k-1})} \\ &\quad \cdot \left[\sum_{y(k-1)}P\left(y(k-1)|\mathcal{O}^{k-1}\right)P\left(y(k)|y(k-1)\right)\right] \\ &= \frac{P\left(O(k)|y(k)\right)}{P\left(\mathbf{O}(k), \mathcal{O}^{k-1}\right)/P(\mathcal{O}^{k-1})} \\ &\quad \cdot \left[\sum_{y(k-1)}P\left(y(k-1)|\mathcal{O}^{k-1}\right)P\left(y(k)|y(k-1)\right)\right] \\ &\quad - \text{expanding } P(\mathcal{O}_k)\end{aligned} \tag{29}$$

$$= \frac{P(O(k)|y(k))}{P(\mathbf{O}(k)|\mathcal{O}^{k-1})\, P(\mathcal{O}^{k-1})/P(\mathcal{O}^{k-1})} \cdot \left[\sum_{y(k-1)} P\left(y(k-1)|\mathcal{O}^{k-1}\right) P\left(y(k)|y(k-1)\right)\right]$$

– from Bayes' rule

$$= \frac{1}{P(\mathbf{O}(k)|\mathcal{O}^{k-1})} P(\mathbf{O}(k)|y(k)) \cdot \left[\sum_{y(k-1)} P\left(y(k)|y(k-1)\right) P\left(y(k-1)|\mathcal{O}^{k-1}\right)\right]. \quad (30)$$

The observation of Camera $\mathcal{C}$, $O_j(k)$, is determined by the activity being performed and the view point of $\mathcal{C}$. If the activity is known, i.e., $y(k)$ is given, $O_j(k)$ only depends upon the view point of $\mathcal{C}$ and is independent of observations of other cameras, i.e.,

$$P(\mathbf{O}(k)|y(k)) = P(O_1(k), \ldots, O_{N_c}(k)|y(k)) = \prod_{j=1}^{N_c} P(O_j(k)|y(k)). \quad (31)$$

So we get

$$P\left(y(k)|\mathcal{O}^k\right) = \frac{1}{P(\mathbf{O}(k)|\mathcal{O}^{k-1})} \prod_{j=1}^{N_c} P(O_j(k)|y(k)) \cdot \left(\sum_{y(k-1)} P(y(k)|y(k-1))\, P\left(y(k-1)|\mathcal{O}^{k-1}\right)\right)$$

which is the statement of Result 1.

Appendix C
Proof of Result 2

Given that

$$P\left(y(k)|\mathcal{O}^k\right) = \frac{1}{P(\mathbf{O}(k)|\mathcal{O}^{k-1})} \prod_{j=1}^{N_c} P(O_j(k)|y(k)) \cdot \left(\sum_{y(k-1)} P(y(k)|y(k-1))\, P\left(y(k-1)|\mathcal{O}^{k-1}\right)\right) \quad (*)$$

where $y(k) \in \{1, \ldots, Y\}$, then

$$\gamma(k) \triangleq \frac{1}{P(\mathbf{O}(k)|\mathcal{O}^{k-1})} = \left[\sum_{y(k)} \prod_{j=1}^{N_c} P(O_j(k)|y(k)) \cdot \left(\sum_{y(k-1)} P(y(k)|y(k-1)) P\left(y(k-1)|\mathcal{O}^{k-1}\right)\right)\right]^{-1}.$$

Proof: Since $y(k) \in \{1, \ldots, Y\}$, it can be inferred that

$$\sum_{a=1}^{Y} P\left(y(k)|\mathcal{O}^k\right) = 1.$$

By substituting $(*)$, we have

$$1 = \sum_{y(k)} \left[\frac{1}{P(\mathbf{O}(k)|\mathcal{O}^{k-1})} \prod_{j=1}^{N_c} P(O_j(k)|y(k)) \cdot \left(\sum_{y(k-1)} P(y(k)|y(k-1)) \times P\left(y(k-1)|\mathcal{O}^{k-1}\right)\right)\right]$$

$$\Rightarrow 1 = \frac{1}{P(\mathbf{O}(k)|\mathcal{O}^{k-1})} \cdot \left[\sum_{y(k)} \prod_{j=1}^{N_c} P(O_j(k)|y(k)) \times \left(\sum_{y(k-1)} P(y(k)|y(k-1)) \times P\left(y(k-1)|\mathcal{O}^{k-1}\right)\right)\right]$$

$$\Rightarrow 1 = \gamma(k) \left[\sum_{y(k)} \prod_{j=1}^{N_c} P(O_j(k)|y(k)) \cdot \left(\sum_{y(k-1)} P(y(k)|y(k-1)) \times P\left(y(k-1)|\mathcal{O}^{k-1}\right)\right)\right]$$

$$\Rightarrow \gamma(k) = \left[\sum_{y(k)} \prod_{j=1}^{N_c} P(O_j(k)|y(k)) \cdot \left(\sum_{y(k-1)} P(y(k)|y(k-1)) \times P\left(y(k-1)|\mathcal{O}^{k-1}\right)\right)\right]^{-1}.$$

REFERENCES

[1] Alahi, D. Marimon, M. Bierlaire, and M. Kunt, “A master-slave approach for object detection and matching with fixed and mobile cameras,” in *Proc. Int. Conf. Image Processing*, 2008, pp. 1712–1715

[2] D. P. Bertsekas and J. Tsitsiklis, *Parallell and Distributed Computa- tion*. Upper Saddle River, NJ: Prentice-Hall, 1989

[3] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach“Distributed embedded smart cameras for surveillance applications,” *Computer*, vol. 39, no. 2, pp. 68–75, 2006.

[4] T.-J. Cham and J. M. Rehg, “A multiple hypothesis approach to figure tracking,” in *Proc. IEEE Conf. Computer Vision and Pattern Recogni tion*, 1999, vol. 2, pp. 239–245.

[5] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005, vol. I, pp. 886–893

[6] M. H. DeGroot, “Reaching a consensus,” *J. Amer. Statist. Assoc.*, vol. 69, no. 345, pp. 118–121, Mar. 1974

[7] A. Doucet, N. d. Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. New York: Springer, 2001

[8] W. Du and J. Piater, "Multi-camera people tracking by collaborative particle filters and principal axis-based integration," in *Proc. Asian Conf. Computer Vision*, 2007, pp. 365–374.

[9] M. Egerstedt and X. Hu, "Formation control with virtual leaders and reduced communications," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6,
947–951, 2001.

[10] E. Ermis, V. Saligrama, P. Jodoin, and J. Konrad, "Abnormal behavior detection and behavior matching for networked cameras," in *Proc. IEEE/ACM Int. Conf. Distributed Smart Cameras*, 2008, pp. 1–10.

[11] J. A. Fax, "Optimal and Cooperative Control of Vehicle Formations," Ph.D. dissertation, Control Dynamical Syst., California Inst. Technol., Pasadena, CA, 2001.

[12] R. Guerraoui, M. Hurfin, A. Mostefaoui, R. Oliveira, M. Raynal, and A. Schiper, "Consensus in asynchronous distributed systems: A concise guided tour," *LNCS*, vol. 1752, pp. 33–47, 1999, Springer-Verlag.

[13] Y. Hatano and M. Mesbahi, "Agreement over random networks," *IEEE Trans. Autom. Control*, vol. 50, no. 11, pp. 1867–1872, Nov. 2005.

[14] D. Hristu and K. Morgansen, "Limited communication control," *Syst. Control Lett.*, vol. 37, pp. 193–205, Jul. 1999.

[15] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, Jun. 2003.

[16] S. Khan, O. Javed, Z. Rasheed, and M. Shah, "Camera Handoff: Tracking in multiple uncalibrated stationary cameras," in *Proc. IEEE Workshop on Human Motion*, 2000, pp. 113–118.

[17] S. Khan and M. Shah, "A multiview approach to tracking people in crowded scenes using a planar homography constraint," in *Proc. Eur. Conf. Computer Vision*, 2006, pp. 133–146.

[18] L. Liao, D. Fox, and H. Kautz, "Location-based activity recognition using relational markov networks," in *Proc. Int. Joint Conf. Artificial Intelligence*, 2005, pp. 773–778.

[19] Z. Lin, M. Brouke, and B. Francis, "Local control strategies for groups of mobile autonomous agents," *IEEE Trans. Autom. Control*, vol. 49, no. 4, pp. 622–629, Apr. 2004.

[20] N. Lynch, *Distributed Algorithms*. San Mateo, CA: Morgan Kauf-mann, 1996.

[21] D. Markis, T. Ellis, and J. Black, "Bridging the gap between cameras," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004, vol. II, pp. 205–210.

[22] H. Medeiros, J. Park, and A. Kak, "Distributed object tracking using a cluster-based Kalman filter in wireless camera networks," *IEEE J. Sel. Topics Signal Process.*, vol. 2, no. 4, pp. 448–463, Aug. 2008.

[23] M. Mehyar, D. Spanos, J. Pongsjapan, S. Low, and R. M. Murray, "Distributed averaging on asynchronous communication networks," in *Proc. IEEE Conf. Decision and Control and European Control Conf.*, Dec. 2005, pp. 7446–7451.

[24] M. Mesbahi, "On state-dependent dynamic graphs and their control-lability properties," *IEEE Trans. Autom. Control*, vol. 50, no. 3, pp. 387–392, Mar. 2005.

[25] B. North, A. Blake, M. Isard, and J. Rittscher, "Learning and classifi-cation of complex dynamics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 9, pp. 1016–1034, Sep. 2000.

[26] R. Olfati-Saber, "Ultrafast consensus in small-world networks," in *Proc. Amer. Control Conf.*, 2005, pp. 1842–1849.

[27] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algo-rithms and theory," *IEEE Trans. Autom. Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[28] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proc. IEEE Conf. Decision and Control*, 2007, pp. 5492–5498.

[29] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[30] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[31] R. Olfati-Saber and N. F. Sandell, "Distributed tracking in sensor net-works with limited sensing range," in *Proc. American Control Conf.*, Jun. 2008, pp. 3157–3162.

[32] V. M. Preciado and G. C. Verghese, "Synchronization in gener-alized Erd os-Rénye networks of nonlinear oscillators," in *Proc. IEEE Conf. Decision and Control and Eur. Control Conf.*, 2005,
pp. 4628–4633.

[33] F. Qureshi and D. Terzopoulos, "Surveillance in virtual reality: System design and multi-camera control," in *Proc. IEEE Conf. Computer Vi-sion and Pattern Recognition*, 2007, pp. 1–8.

[34] A. Rahimi and T. Darrell, "Simultaneous calibration and tracking with

[35] network of non-overlapping sensors," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004, vol. I, pp. 187–194.

A. Rahimi and T. Darrell, "Simultaneous calibration and tracking with a network of non-overlapping sensors," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004, vol. I, pp. 187–194.

[35] W. Ren and R. W. Beard, "Consensus seeking in multi-agent systems under dynamically changing interaction topologies," *IEEE Trans. Autom. Control*, vol. 50, no. 5, pp. 655–661, May 2005.

[36] J. Rittscher and A. Black, "Classification of human body motion," in *Proc. Int. Conf. Computer Vision*, 1999, vol. 1, pp. 634–639.

[37] B. Song and A. Roy-Chowdhury, "Stochastic adaptive tracking in a camera network," in *Proc. Int. Conf. Computer Vision*, 2007, pp. 1–8.

[38] C. Soto, B. Song, and A. Roy-Chowdhury, "Distributed multi-target tracking in a self-configuring camera network," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009, pp. 1486–1493.

[39] B. Stancil, C. Zhang, and T. Chen, "Active multicamera networks: From rendering to surveillance," *IEEE J. Sel. Topics Signal Process.*, vol. 2, no. 4, pp. 597–605, Aug. 2008.

[40] K. Tieu, G. Dalley, and W. Grimson, "Inference of non-overlapping camera network topology by measuring statistical dependence," in *Proc. Int. Conf. Computer Vision*, 2005, pp. 1842–1849.

[41] R. Tron, R. Vidal, and A. Terzis, "Distributed pose averaging in camera networks via consensus on SE(3)," in *Proc. IEEE/ACM Int. Conf. Dis-tributed Smart Cameras*, Sep. 2008, pp. 1–10.

[42] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, Sep. 1986.

[43] A. Veeraraghavan, A. Roy-Chowdhury, and R. Chellappa, "Matching shape sequences in video with applications in human motion analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1896–1909, Dec. 2005.

[44] Y. Wang, K. Huang, and T. Tan, "Multi-view gymnastic activity recog-nition recognith with fused hmm," in *Proc. Asian Conf. Computer Vi-sion*, 2007, pp. 667–677.

[45] D. Weinland, E. Boyer, and R. Ronfard, "Action recognition from arbi-trary views using 3d exemplars," in *Proc. Int. Conf. Computer Vision*, 2007, pp. 1–7.

[46] J. Zhao, S. C. Cheung, and T. Nguyen, "Optimal camera network con-figurations for visual tagging," *IEEE J. Sel. Topics Signal Process.*, vol. 2, no. 4, pp. 464–479, Aug. 2008.

MD.Azharuddin.Osmania University Ph.D Research Scholar, Department of Electronics Communiaction Engineering .Hyderabad.