Distributed Multi-Target Tracking In A Self-Configuring Camera Network

Cristian Soto, Bi Song, Amit K. Roy-Chowdhury [*†]
Department of Electrical Engineering
University of California, Riverside
{cwilder,bsong,amitrc}@ee.ucr.edu

Abstract

This paper deals with the problem of tracking multiple targets in a distributed network of self-configuring pan-tilt-zoom cameras. We focus on applications where events unfold over a large geographic area and need to be analyzed by multiple overlapping and non-overlapping active cameras without a central unit accumulating and analyzing all the data. The overall goal is to keep track of all targets in the region of deployment of the cameras, while selectively focusing at a high resolution on some particular target features. To acquire all the targets at the desired resolutions while keeping the entire scene in view, we use cooperative network control ideas based on multi-player learning in games. For tracking the targets as they move through the area covered by the cameras, we propose a special application of the distributed estimation algorithm known as Kalman-Consensus filter through which each camera comes to a consensus with its neighboring cameras about the actual state of the target. This leads to a camera network topology that changes with time. Combining these ideas with single-view analysis, we have a completely distributed approach for multi-target tracking and camera network self-configuration. We show performance analysis results with real-life experiments on a network of 10 cameras.

1. Introduction

Networks of video cameras are being installed in many applications, *e.g.*, video surveillance, national and homeland security, assisted living facilities, etc. It is natural to expect that these camera networks would be used to track targets at multiple resolutions, *e.g.*, multiple people, a single person, a face. For efficiency and maximum resource utilization, it is desirable to *actively* control the cameras so as to track the targets based on the requirements of the scene being analyzed. It would be prohibitively expensive to have a static setup that would cater to all possible situations. Currently, similar applications try to cover the entire area or the most important parts of it with a set of passive cameras but have difficulty in acquiring high resolution shots selectively.

It is also desirable that the tracking and control mechanism be distributed due to constraints of bandwidth, secure transmission facilities, and difficulty in analyzing a huge amount of data centrally. In such situations, the cameras would have to act as *autonomous agents* and decisions would have to be taken in a distributed manner. However, to be able to track all the targets in an area under surveillance accurately, the cameras should be working cooperatively with each other. This is because each camera's parameter settings entail certain constraints on other cameras. Also, if a target is observed by multiple cameras, there should be a consensus on the state (*e.g.*, position) of the target even if each camera is an autonomous agent.

1.1. Problem Description

The overall goal of this paper is to develop a distributed multi-target tracking and camera network control framework to observe and keep track of all targets at the desired resolutions in an active camera network. We consider a network of synchronized calibrated cameras with pan, tilt, and zoom capabilities. Each camera has an embedded processing unit for local processing of the sensed video data. Since the cameras are calibrated, they can determine through their homographies the position of a target on a ground plane.

Some of the cameras in the network may have overlapping fields of view and a target may therefore be viewed by several cameras simultaneously. Due to inaccuracies in calibration and single-view target tracking methods, the different cameras viewing the same target will not have exactly the same measurement of the target's state on the ground plane. Consequently, it is necessary for the cameras to collaborate to reach a consensus on the actual state of the target. In our framework, this consensus must be reached only through point-to-point communications between neighboring cameras without the use of any central processing unit.

[*]The authors were supported by NSF grants ECS-0622176 and CNS-0551741, and ARO grant W911NF-07-1-0485.
[†]The authors would like to thank Anesdo Fong for help with the experimental evaluations.

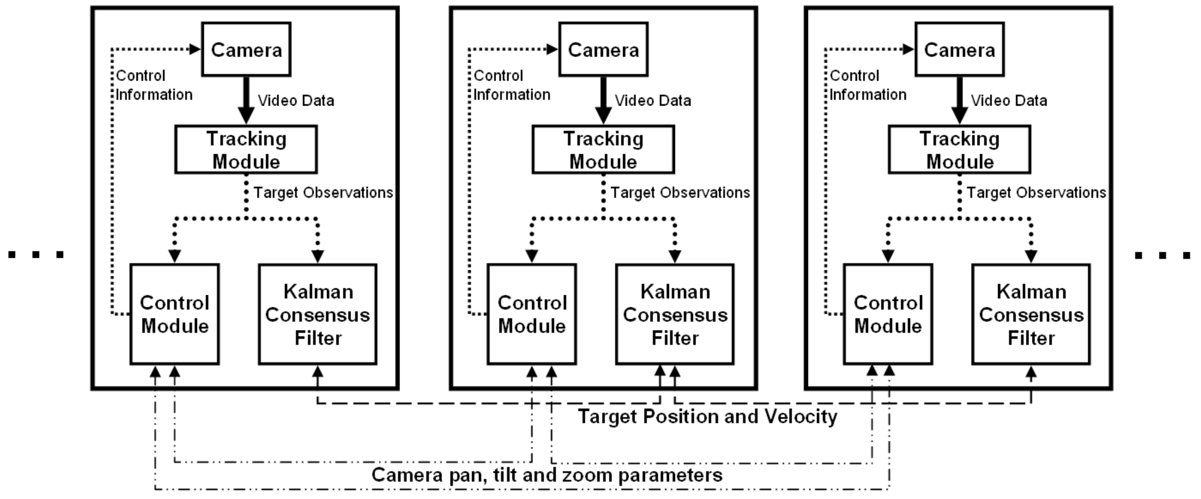


Figure 1. A diagrammatic representation of the proposed distributed tracking and control system with three neighboring smart cameras.

As the targets move from one camera's FOV into another, and as the cameras change their parameters, the distributed tracking system must also be able to keep track of the targets in a seamless way. Also, a camera may have to change its parameters in coordination with other cameras so as to keep the targets imaged at the given desired resolutions. All this requires the tracking algorithm to be robust to dynamically changing camera network topologies.

1.2. Overview of solution strategy

We propose a special application of the Kalman-Consensus Filter presented in [10] to solve the problem of finding a consensus on the state (position and velocity) of multiple targets in a dynamic camera network with possibly overlapping fields of view. In our proposed distributed tracking algorithm, the cameras will only communicate with their neighbors and their specific communication links will vary in time as the targets move through the area under surveillance. Details of the tracking approach are provided in Section 3.

The cooperative and distributed nature of this problem also leads us to explore a game theoretic solution for the camera control problem as detailed in Section 4. This is achieved by the optimization of local sensor utility functions leading to an optimal value for a global utility, i.e. keeping view of all targets at an acceptable resolution and some at high resolution.

Figure 1 shows an overview of our distributed multi-resolution tracking system in a network of self-configuring cameras. Each of the three neighboring smart cameras in this figure has its own embedded tracking module, control module and Kalman-Consensus filter. The tracking module receives the video data from the camera and performs the tracking of the targets in its FOV. Since the camera is calibrated, the tracking module can determine the (noisy) position of each target on the ground plane. The Kalman-Consensus filter then uses this target position information together with position and velocity information from other neighboring cameras to come to a consensus with these cameras about the actual state of the target.

The control module on the other hand, changes the parameters of the camera as necessary to track the different targets at the desired resolutions while also keeping the entire area in view. The parameter change is decided based on the parameters of neighboring cameras following the game-theoretic framework presented in Section 4. The parameters of the neighboring cameras are obtained through a negotiation process. The Kalman-Consensus filter and control module run *independently and asynchronously* in each smart camera.

As can be seen in Figure 1, each camera exchanges only information regarding its own PTZ parameters and states of the targets that are in its FOV. There is no video information being sent to any other camera or to a central processing station. It should also be pointed out that although the control module communicates with all of its neighbors, the Kalman-Consensus filter in our framework communicates only with a subset of the neighboring cameras. This is advantageous because the state information exchange for the consensus tracking requires a higher data rate than the control negotiation information. By reducing the number of neighboring cameras that need to communicate with each other during the consensus tracking, we reduce the overall communication bandwidth required for the system. As can be seen clearly, this entire framework leads to a completely distributed approach for multi-target tracking and camera self-configuration.

1.3. Relation to previous work

Some recent work has dealt with networks of vision sensors, namely computing the statistical dependence between cameras, computing the camera network topology, tracking over the network, and camera handoff [3, 5, 6, 12, 14]. There has also been recent work on tracking people in a multi-camera setup with overlapping fields of view [2, 4]. However, these methods are not totally distributed and do not deal with the the problem of tracking *and* control in an active camera network.

In [7], 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 estimate to a central base station. Our proposed tracking system differs from this method in that each camera in a neighborhood has a consensus-based estimate of the target's state and thus there is no need for additional computation and communication to select a cluster head. Furthermore, we consider a dynamic camera network in which, as the cameras change their parameters, the targets are being kept track of seamlessly. As will be described in Section 3, we apply in a special way the distributed Kalman-Consensus filter [10] 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 [15].

A related work that deals with tracking targets in a camera network with PTZ cameras is [11]. Here, the authors proposed a mixture between a distributed and a centralized scheme using both static and PTZ cameras in a virtual camera network environment. Our approach to camera control and tracking, however, is completely distributed using consensus algorithms and a game-theoretic framework [1]. Preliminary work on this game theoretic approach for camera control was presented in [13]. However, our goal in that paper was merely to cover an entire area, and there was no attempt to address the tracking problem, which is critical for a distributed approach to work. In this paper, we also show real-life experiments in a network of 10 cameras.

The remainder of this paper is organized as follows: Section 3 presents our distributed target tracking approach through conensus. Section 4 states the problem of distributed camera control and its solution in game theoretic terms. Our experimental results are presented in Section 5. We summarize our work in Section 6.

2. A Review of Distributed Estimation and Cooperative Control in A Sensor Network

We briefly review some basic concepts related to distributed consensus and cooperative control in a sensor network that are directly relevant to our work.

2.1. Distributed state estimation

In the multi-agent systems literature, *consensus* means to reach an agreement regarding a certain quantity of interest that depends on the state of all sensors in a network. 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 neighbors so that all the nodes reach a consensus. The interaction topology of a network of sensors is represented using a graph $G = (V, E)$ with the set of nodes $V = \{1, 2, ..., n\}$ and edges $E \subseteq V \times V$. Each sensor node $i = 1, ..., n$ is associated with a state $x_i \in \mathbb{R}$. Reaching a consensus means asymptotically converging to a one-dimensional *agreement space* characterized by the equation $x_1 = x_2 = ... = x_N$.

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, Distributed Kalman Filters (DKF) compute a consensus-based estimate on the state of interest with only point-to-point communication between the sensors[10]. A distributed Kalman filtering (DKF) strategy that obtains consensus on state estimates was presented in [10].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 3.2. A thorough review of consensus in networked multi-agent systems can be found in [9].

2.2. Distributed control through cooperation

The theme of *cooperative control* has received significant attention in recent years, especially the design of autonomous vehicles with intelligent and coordinated action capabilities to achieve an overall objective. In [1], an autonomous vehicle-target assignment problem in which a group of vehicles are expected to assign themselves to a set of targets to optimize a global utility function is considered. However, rather than optimizing the global utility function directly, the emphasis was on designing vehicles that are individually capable of making coordinated decisions to optimize their own local utilities, which then indirectly translated into the optimization of a global utility function. This approach enabled the vehicles to operate in environments with limited information, communication, and computation, and still be able to optimize a global utility autonomously.

This problem was formulated as a multi-player game where each vehicle was interested in optimizing its own utility. The notion of pure Nash equilibrium can be used to represent the target assignments that are agreeable to the rational vehicles, i.e. the assignments at which there is no incentive for any vehicle to unilaterally deviate. As has been shown in [8], the combination of using the Wonderful Life Utility (WLU) as the vehicle utility and Spatial Adaptive Play (SAP) as the negotiation mechanism leads to an optimal assignment of targets within the set of pure Nash equilibria. The precise definitions of the utility functions and negotiation mechanisms for our application will be provided in Section 4.

3. Distributed Multi-Target Tracking Through Consensus

3.1. Dynamic Network Topology

As mentioned earlier, we propose a special application of the Kalman-Consensus Filter presented in [10] to solve the problem of finding a consensus on the state of multiple targets in a dynamic camera network with possibly overlapping fields of view. However, unlike the sensor nodes in [10], the nodes in our camera network are directional sensors and a new method for establishing the network topology between the camera nodes must therefore be developed. Furthermore, since the camera network is composed of cameras that change their parameters and therefore their fields of view as needed, the network topology will be dynamically changing also.

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 $\mathcal{C}_l^v \subset \mathcal{C}$ and the rest of the cameras as $\mathcal{C}_l^p \subset \mathcal{C}$. Each camera C_i will also have its set of *neighboring cameras* $\mathcal{C}_i^n \subset \mathcal{C}$. Since all the cameras can change their PTZ parameters and have therefore several possible fields of view, we define the set $\mathcal{C}_i^n$ 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 $C_i \in \mathcal{C}_l^v$, it is true that $\mathcal{C}_l^v \subset \{\mathcal{C}_i^n \cup C_i\}$, i.e. all of the cameras viewing a specific target are also neighbors. Note that the set of neighbors need not be geographical neighbors.

In our proposed distributed tracking system, only cameras that have an observation of a target T_l will communicate with their neighbors. This feature limits the amount of data exchanged for the consensus tracking. By reducing the number of neighboring cameras that need to communicate with each other during the consensus tracking, we also reduce the power requirements of each camera node and the overall communication bandwidth required for the system.

This way of defining the network connections between the camera nodes brings about a dynamic graph $G_l(k) = (V_l(k), E_l(k))$ representing the network topology of the system with respect to target T_l at time instant k. When there are multiple targets in the area under surveillance, there will be a distinct network topology for each target. Obviously, as T_l moves through the area under surveillance, $G_l(k)$ will also change since other cameras will be viewing T_l and only these cameras will communicate with their neighbors thus creating a dynamic network topology. The network topologies of all the targets will also change when one or more cameras change their parameters. As will be explained in the following section, this definition of network topology for distributed tracking does not affect in any way the performance of the Kalman-Consensus filter presented next.

Algorithm 1 Distributed Kalman-Consensus tracking algorithm performed by every C_i at discrete time step k. The state of T_l is represented by $\mathbf{x}_i^l$ with error covariance matrix $\mathbf{P}_i^l$ (see Sec. 3.2).

```
Input: $\bar{\mathbf{x}}_i^l$ and $\mathbf{P}_i^l$ from time step $k-1$
for each $T_l$ that is being viewed by $\{\mathcal{C}_i^n \cup C_i\}$ do
  if $C_i \in \mathcal{C}_l^v$ (i.e. $C_i$ is viewing $T_l$) then
    Obtain ground plane measurement $\mathbf{z}_i^l$ with covariance $\mathbf{R}_i^l$
    Compute information vector and matrix
      $\mathbf{u}_i^l = {\mathbf{H}_i^l}^T {\mathbf{R}_i^l}^{-1} \mathbf{z}_i^l$
      $\mathbf{U}_i^l = {\mathbf{H}_i^l}^T {\mathbf{R}_i^l}^{-1} \mathbf{H}_i^l$
    Send message $\mathbf{m}_i^l = (\mathbf{u}_i^l, \mathbf{U}_i^l, \bar{\mathbf{x}}_i^l)$ to neighboring cameras $\mathcal{C}_i^n$
  end if
  Receive messages $\mathbf{m}_j = (\mathbf{u}_j^l, \mathbf{U}_j^l, \bar{\mathbf{x}}_j^l)$ from all cameras $C_j \in \mathcal{C}_l^v$
  Fuse information matrices and vectors
    $\mathbf{y}_i^l = \sum_{j \in \mathcal{C}_l^v} \mathbf{u}_j^l, \ \mathbf{S}_i^l = \sum_{j \in \mathcal{C}_l^v} \mathbf{U}_j^l$
  Compute the Kalman-Consensus state estimate
    $\mathbf{M}_i^l = ((\mathbf{P}_i^l)^{-1} + \mathbf{S}_i^l)^{-1}$
    $\hat{\mathbf{x}}_i^l = \bar{\mathbf{x}}_i^l + \mathbf{M}_i^l(\mathbf{y}_i^l - \mathbf{S}_i^l \bar{\mathbf{x}}_i^l) + \gamma \mathbf{M}_i^l \sum_{j \in \mathcal{C}_l^v} (\bar{\mathbf{x}}_j^l - \bar{\mathbf{x}}_i^l)$
    $\gamma = 1/(||\mathbf{M}_i^l|| + 1), ||\mathbf{X}|| = (tr(\mathbf{X}^T\mathbf{X}))^{\frac{1}{2}}$
  Update the state and error covariance matrix for time step $k$
    $\mathbf{P}_i^l \leftarrow \mathbf{A}^l \mathbf{M}_i^l {\mathbf{A}^l}^T + \mathbf{B}^l \mathbf{Q}^l {\mathbf{B}^l}^T$
    $\bar{\mathbf{x}}_i^l \leftarrow \mathbf{A}^l \hat{\mathbf{x}}_i^l$
end for
```

3.2. Kalman-Consensus tracking

To model the motion of a target T_l on the ground plane as observed by camera C_i, we consider a linear dynamical system with process and sensing models

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

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

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 $\mathbf{x}^l(0)$ 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. $\mathbf{x}_i^l$ is the state of T_l based on the observations in C_i only. The noisy measurement $\mathbf{z}_i^l(k)$ at camera C_i is the sensed target position $(x_i^l(k), y_i^l(k))$ on the ground plane based on the pre-computed homography.

Our special implementation of the Kalman-Consensus distributed tracking algorithm is presented in Algorithm 1. This algorithm is performed distributedly in each camera node C_i. At each time step k and for each target T_l, we assume to be given the estimated target state $\bar{\mathbf{x}}_i^l$ and the error covariance matrix $\mathbf{P}_i^l$ from time step $(k-1)$. Each C_i also knows its set of neighbors $\mathcal{C}_i^n$ from the information used by the camera's control module. At time step $k = 0$, the Kalman-Consensus filter is initialized with $\mathbf{P}_i^l = \mathbf{P}_0$ and $\bar{\mathbf{x}}_i^l = \mathbf{x}_i^l(0)$ = average of $\mathbf{z}^l(k)$'s of neighbors viewing T_l.

The following gives a verbal description of Algorithm 1 performed at each C_i distributedly for each T_l that is viewed by $\{\mathcal{C}_i^n \cup C_i\}$. If C_i is viewing a target T_l, it determines T_l's ground plane position $\mathbf{z}_i^l$ and computes the corresponding information vector and matrix with the given measurement covariance matrix $\mathbf{R}_i^l$ and output matrix $\mathbf{H}_i^l$. C_i then sends a message $\mathbf{m}_i^l$ to its neighbors which includes the computed information vector and matrix, and its estimated target state $\bar{\mathbf{x}}_i^l$ at previous time step $(k-1)$. C_i then receives similar messages $\mathbf{m}_j$ only from the cameras in its neighborhood that are also viewing T_l. The information matrices and vectors received from these messages, and its own information matrix and vector if C_i is viewing T_l, are then fused and the Kalman-Consensus state estimate is computed in a way similar to the method proposed in [10]. Finally, the ground plane state $\bar{\mathbf{x}}_i^l$ and error covariance matrix $\mathbf{P}_i^l$ are updated according to the assumed linear dynamical system.

3.3. Handoff, Network Reconfiguration 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 its neighboring cameras, 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 through the game theoretic control framework presented in Section 4, the target will always be seen by at least one camera. Also, by our definition of neighboring cameras, a target T_l will always move from one camera C_i's FOV to the FOV of a neighboring camera $C_j \in \mathcal{C}_i^n$. 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. Furthermore, even as a camera changes its parameters, it can also take over the tracking of the targets in its new FOV immediately since it also knew the position of the targets in its neighborhood beforehand through the consensus-tracking algorithm.

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_i, the targets that were viewed by C_i are not suddenly lost by the camera network. The neighboring cameras can adjust their parameters to cover the area that was left uncovered by C_i's failure and continue with the consensus-tracking algorithm without any interruption.

We have also considered the fact that a camera may take a short amount of time to change its parameters to a new position. 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 just follows the assumed linear state equations. This does not translate to a significant decrease in tracking performance as seen in our experiments. In the next section, we explain how the camera parameters are changed in order to keep the different targets imaged at the specified resolution.

4. Distributed Self-Configuration of Camera Network Using Game Theory

Our goal is to develop a distributed strategy for coordinated self-configuration of the camera network that relies on local decision-making at the camera nodes, while being aligned with the suitable global criterion of observing multiple targets at multiple resolutions. For this purpose, we use *game theoretic* ideas that rely on *multi-player learning and negotiation mechanisms*.

4.1. Game Theoretic Motivation

Let us consider N_t targets in the entire area of deployment and N_c sensors that need to be assigned to these targets. The targets $\{\mathcal{T}\}$ are the opponents. A target loses and is eliminated from the game if it is captured in an image at the desired resolution. Our team of cameras scores each time it obtains a desired resolution image of a target. A camera C_i will select its own assignment profile, a_i, (i.e. the set of targets to track) by optimizing its own utility function $U_{C_i}(a_i)$. Our problem is to design these utility functions and appropriate negotiation procedures that lead to a mutually agreeable assignment of targets resulting in meeting the global criterion.

4.2. Choice of utility functions

Target Utility: Let V_l be the value of observing target T_l and p_{il} the probability that target T_l is acquired at the desired resolution by camera C_i engaging T_l. Then, we define the utility of observing T_l using a particular assignment profile $\mathcal{A} = \bigcup_i a_i$ to be

$$U_{T_l}(\mathcal{A}) = V_l \left[1 - \prod_i (1 - p_{il}) \right]. \quad (3)$$

p_{il} is defined as

$$p_{il} = \left\{ \begin{array}{cc} 1 - e^{-\lambda x_{il}} & \text{if } x_{il} > r_0/r_{max} \\ 0 & \text{otherwise} \end{array} \right., \quad (4)$$

where r_0 is the minimum acceptable resolution in terms of target pixel height at which the targets should be viewed at and r_{max} is the height in pixels of C_i's image plane. If r_{il} is the resolution at which T_l is being viewed at by C_i, then $x_{il} = r_{il}/r_{max}$. The term λ can be changed according to how well the single-view tracking algorithm performs as the height of the target on the image plane increases or decreases.

Global Utility: From the target utility function, we can now define the global utility function as the sum of the utilities generated by observing all the targets, i.e.,

$$U_g(\mathcal{A}) = \sum_{T_l} U_{T_l}(\mathcal{A}). \quad (5)$$

Camera Utility: In our application, we use what is known as Wonderful Life Utility (WLU). In WLU, the utility of a sensor observing a particular target is the marginal contribution to the global utility as a result of this action, i.e., the sensor utility is the change in the global utility as a result of that sensor observing that particular target as opposed to not observing it. Since each camera C_i in our distributed camera network can influence only the target assignments of its neighbors $\mathcal{C}_i^n$, we will define the camera utility depending only on the assignments of its neighboring cameras, i.e.

$$\begin{aligned} U_{C_i}(\mathcal{A}) &= U_g(\mathcal{A}) - U_g(a_{-i}) \\ &= \sum_{T_l \in \mathcal{C}_i^n} U_{T_l}(\mathcal{A}) - U_{T_l}(a_{-i}), \end{aligned} \quad (6)$$

where $a_{-i} = \mathcal{A} - a_i$ is the assignment profile of all the cameras except C_i.

As shown in [8], the WLU utility leads to a potential game with the global utility function as the potential function, and hence they are aligned with the global utility. This ensures that the resulting set of targets that are chosen will be included within the set of pure Nash equilibria.

4.3. Negotiation mechanisms

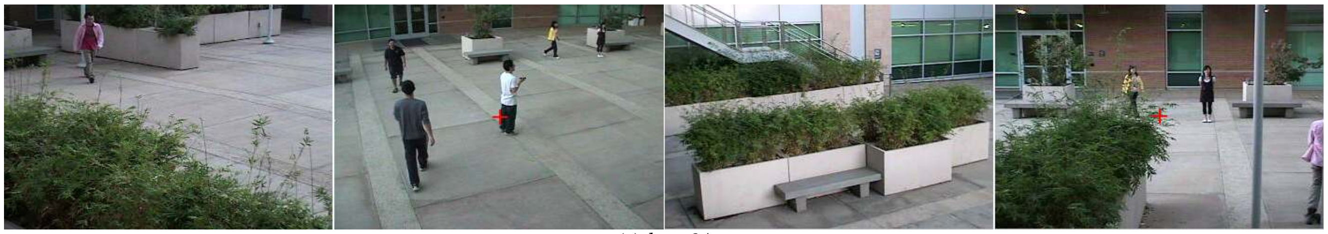
Persistently observing the objects of interest in a dynamic setting requires negotiation mechanisms between the different sensors, allowing them to come up with the strategic decisions described above. Each sensor negotiates with other sensors to accurately predict their team-mates' parameters, and decide its own action. A particularly appealing strategy for this problem is Spatial Adaptive Play (SAP) [8]. This is because it can be implemented with a low computational burden on each camera and leads to an optimal assignment of targets with arbitrarily high probabilities for the WLU described above.

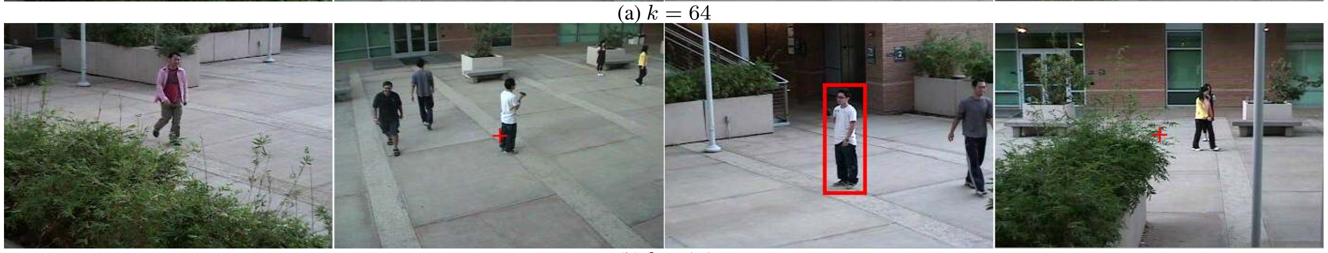
In a particular step of the SAP negotiation strategy, a camera C_i is randomly chosen from the pool of cameras in the network according to a uniform distribution and only this camera is given the chance to update its proposed parameter settings. Let us now consider the case where a specific application requires the tracking of a target T_l^h at a high resolution r_h. When a camera C_i is chosen to update its parameters at any negotiation step, it will first determine if it can adjust its parameters to view the target at the desired high resolution. If it can not do so, C_i will simply continue with the negotiation cycle maximizing its own utility and sending its new parameters to its neighbors $\mathcal{C}_i^n$ as presented above. If C_i is however able to view T_l^h, it will set its parameters accordingly, take over the task of viewing the target and transmit its parameters to its neighbors $\mathcal{C}_i^n$. When C_i is not able to change its parameters to view T_l^h, it will send a handoff flag to its neighbors $\mathcal{C}_i^n$ indicating that another camera needs to take over the task of viewing the target. C_i then returns to the negotiation mechanism maximizing its own utility U_{C_i}. It is to note that the time between negotiation steps does not need to be the same as the discrete time step size in the Kalman-Consensus tracking described in Section 3 since the two processes run parallelly and asynchronously to each other in each smart-camera.

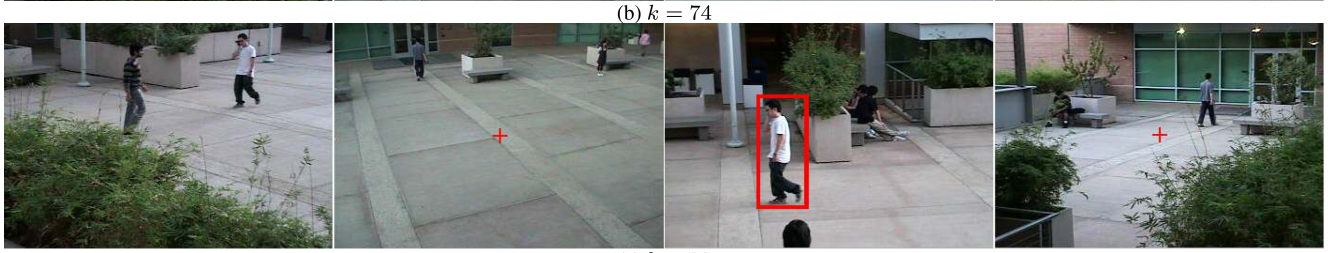
5. Experimental Results

We tested our approach for tracking and camera self-configuration in a real camera network composed of 10 PTZ cameras looking over an outdoor area of approximately 10000 sq. feet. We divided the area into contiguous blocks and each of the centers of those blocks was considered a virtual target of height 1.70m. Therefore, if the camera network as a whole is covering all of the virtual targets in each block at an acceptable resolution $r_0 = 40$ pixels in the vertical direction, all of the actual targets in the area under surveillance are also being viewed at an acceptable resolution.

In the area under surveillance, there were 8 targets in total that were to be tracked using our distributed Kalman-Consensus filtering approach. Figure 2 shows the tracking and control results as viewed by each camera at 4 time instants. Due to space constraints, we show only 4 ($C_1, ..., C_4$) of the 10 cameras in the camera network.


(a) $k = 64$


(b) $k = 74$


(c) $k = 93$

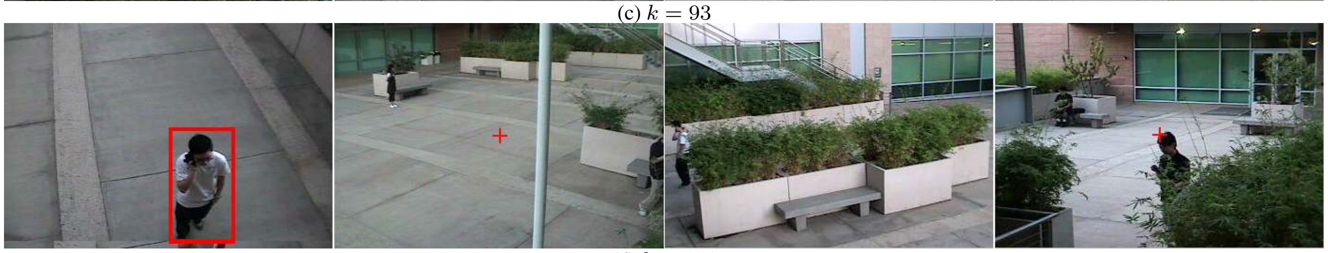

(d) $k = 151$

Figure 2. Each column shows one of 4 of the 10 cameras at four time instants denoted by k. A target marked with a box is always tracked at a high resolution. Note that the camera parameters are changing to achieve this while covering the entire area at an acceptable resolution. The other targets are tracked using the Kalman-Consensus filtering approach, but are not marked for clarity. The video is available as supplementary material and at http://www.ee.ucr.edu/ amitrc/CameraNetworks.htm .

At an initial state, all of the cameras have random PTZ parameters and are not covering the entire area under surveillance. After the cameras start running their control modules, they converge to the final configuration seen partly in Figure 2(a) covering the entire area. As the targets are observed in this area, the single-view tracking module in each camera determines 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 3.

Figure 2(b) shows the instant when a camera C_3 zoomed into a target T_l^h that was marked for being tracked at a high resolution. Since C_3 changed its parameters and left some area it was covering uncovered, the other cameras in the network adjust their own parameters to cover that uncovered area following the negotiation mechanisms described above. These changes in parameters can be seen in Figure 2(b) and (c) for cameras C_3 and C_4. As shown in Figure 2(d), when C_3 is not able to change its parameters to keep tracking T_l^h, C_1 takes over the tracking and C_3 changes its parameters to cover some area that was left uncovered by C_1. It is to note 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 change also. [1]

Figure 3 shows the distributed Kalman-Consensus tracking trajectories for the 8 targets. The observations of the different cameras are shown in a light gray color. As can be seen, even though the homography-based observations are noisy and the network topology is changing constantly, the Kalman-Consensus filter in each camera comes to a smooth consensus about the actual state of the target.

[1] The code for a simulation of the camera network control and Kalman-Consensus filter is available at http://www.ee.ucr.edu/ amitrc/CameraNetworks.htm .

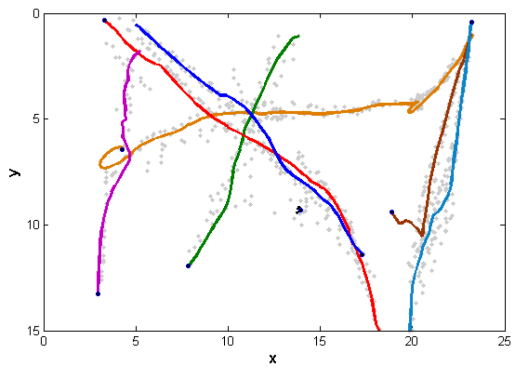


Figure 3. Distributed Kalman-Consensus tracking trajectories for 8 targets. Observations from all cameras are shown in a light gray color.

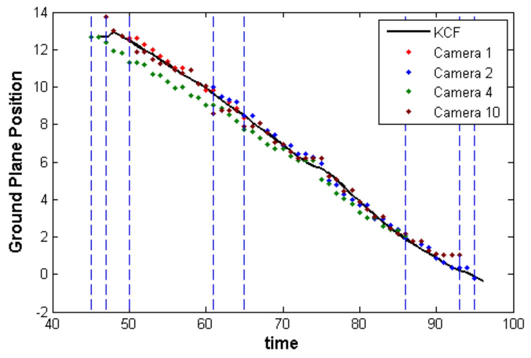


Figure 4. Tracking results on the ground plane for one of the targets.

Figure 4 shows the distributed tracking results in the y ground plane direction for one of the targets. The dots correspond to the observations from the different cameras viewing the target while the solid line is the consensus-based estimate. As can be expected, the observations are different for each camera due to calibration and single-view tracking inaccuracies. The vertical dashed lines indicate the time instants of change in the dynamic network topology with respect to the target, i.e. when the target goes into or out of a camera's FOV. As can be seen clearly, even though different combinations of cameras view the target at different time instants, the Kalman-Consensus filter finds an estimate of the target's position seamlessly at all times.

6. Conclusion

We presented in this paper a robust approach to distributed multi-target tracking in a network of self-configuring cameras. A distributed Kalman-Consensus filtering approach was used together with a dynamic network topology for tracking persistently multiple targets across several camera views in an area under surveillance viewed by a dynamic camera network. A camera control framework based on game theoretical ideas allowed for viewing some targets at a high resolution while keeping the entire area under surveillance covered at an acceptable resolution.

References

[1] G. Arslan, J. Marden, and J. Shamma. Autonomous Vehicle-Target Assignment: A Game-Theoretical Formulation. *ASME Journal of Dynamic Systems, Measurement and Control*, 129(5), September 2007.

[2] W. Du and J. Piater. Multi-camera People Tracking by Collaborative Particle Filters and Principal Axis-Based Integration. In *Asian Conf. on Computer Vision*, 2007.

[3] S. Khan, O. Javed, Z. Rasheed, and M. Shah. Camera Handoff: Tracking in Multiple Uncalibrated Stationary Cameras. In *IEEE Workshop on Human Motion*, 2000.

[4] S. Khan and M. Shah. A Multiview Approach to Tracking People in Crowded Scenes Using a Planar Homography Constraint. In *Euro. Conference on Computer Vision*, 2006.

[5] Y. Li and B. Bhanu. Utility-based dynamic camera assignment and hand-off in a video network. *IEEE/ACM Intl. Conf. on Distributed Smart Cameras*, 2008.

[6] D. Markis, T. Ellis, and J. Black. Bridging the Gap Between Cameras. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2004.

[7] H. Medeiros, J. Park, and A. Kak. Distributed object tracking using a cluster-based kalman filter in wireless camera networks. *IEEE Journal of Selected Topics in Signal Processing*, 2(4):448–463, Aug. 2008.

[8] D. Monderer and L. Shapley. Potential games. In *Games and Economic Behavior*, 1996.

[9] R. Olfati-Saber, J. Fax, and R. Murray. Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE*, 95(1):215–233, Jan. 2007.

[10] R. Olfati-Saber and N. F. Sandell. Distributed tracking in sensor networks with limited sensing range. *Proceedings of the American Control Conference*, June 2008.

[11] F. Qureshi and D. Terzopoulos. Surveillance in Virtual Reality: System Design and Multi-Camera Control. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.

[12] B. Song and A. Roy-Chowdhury. Stochastic Adaptive Tracking in a Camera Network. In *IEEE Intl. Conf. on Computer Vision*, 2007.

[13] B. Song, C. Soto, A. K. Roy-Chowdhury, and J. A. Farrell. Decentralized camera network control using game theory. *Workshop on Smart Camera and Visual Sensor Networks at ICDSC*, 2008.

[14] K. Tieu, G. Dalley, and W. Grimson. Inference of Non-Overlapping Camera Network Topology by Measuring Statistical Dependence. In *IEEE Intl. Conf. on Computer Vision*, 2005.

[15] R. Tron, R. Vidal, and A. Terzis. Distributed pose averaging in camera networks via consensus on SE(3). *IEEE/ACM Intl. Conf. on Distributed Smart Cameras*, Sept. 2008.