8-1-2008

Distributed Object Tracking Using a Cluster-Based Kalman Filter in Wireless Camera Networks

Henry Medeiros
Marquette University, henry.medeiros@marquette.edu

Johnny Park
Purdue University

Avinash Kak
Purdue University

Distributed Object Tracking Using a Cluster-Based Kalman Filter in Wireless Camera Networks

Henry Medeiros, Johnny Park, *Member, IEEE,* and Avinash Kak

***Abstract*—Local data aggregation is an effective means to save sensor node energy and prolong the lifespan of wireless sensor networks. However, when a sensor network is used to track moving objects, the task of local data aggregation in the network presents a new set of challenges such as the necessity to estimate, usually in real-time, the constantly-changing state of the target based on information acquired by the nodes at different time instants. To address these issues, we propose a distributed object tracking system which employs a cluster-based Kalman filter in a network of wireless cameras. When a target is detected, cameras that can observe the same target interact with one another to form a cluster and elect a cluster head. Local measurements of the target acquired by members of the cluster are sent to the cluster head, which then estimates the target position via Kalman filtering and periodically transmits this information to a base station. The underlying clustering protocol allows the current state and uncertainty of the target position to be easily handed-off among clusters as the object is being tracked. This allows Kalman filter based object tracking to be carried out in a distributed manner. An extended Kalman filter is necessary since measurements acquired by the cameras are related to the actual position of the target by a non-linear transformation. In addition, in order to take into consideration the time uncertainty in the measurements acquired by the different cameras, it is necessary to introduce non-linearity in the system dynamics. Our object tracking protocol requires the transmission of significantly fewer messages than a centralized tracker that naively transmits all the local measurements to the base station. It is also more accurate than a decentralized tracker that employs linear interpolation for local data aggregation. Besides, the protocol is able to perform real-time estimation because our implementation takes into consideration the sparsity of the matrices involved in the problem. The experimental results show that our distributed object tracking protocol is able to achieve tracking accuracy comparable to the centralized tracking method, while requiring a significantly smaller number of message transmissions in the network.**

***Index Terms*—cameras, wireless camera networks, wireless sensor networks, sensor clustering, distributed tracking, Kalman filtering**

I. Introduction

It is well known that local data aggregation is an effective means to save sensor node energy and prolong the lifespan of wireless sensor networks. This has motivated many previous researchers to employ sensor clustering techniques to enable local data aggregation for environment monitoring applications [1], [2], [3], [4], [5]. However, when a sensor network is used to track moving objects, the task of local data aggregation in the network presents a new set of challenges. One challenge is that the system must be able to estimate the current state of the target based on information acquired by the nodes at different time instants while the state of the target is constantly changing. Another challenge comes from the fact that most object tracking systems demand the position of the target object to be estimated in real-time which puts heavy constraints on the time it takes to carry out local data aggregation in the network. The work presented in this paper attempts to address these issues.

In our earlier work [6], we have presented a clustering protocol to allow for dynamic formation of camera clusters as a target with specific visual features is detected in the network. In this paper we extend that work by employing the Kalman filter [7] — one of the most commonly used and time-honored techniques for reliable parameter estimation — to aggregate information collected by different nodes. We use our clustering algorithm to manage a decentralized Kalman filter to locally aggregate the data collected by the cameras. The information on a target object is acquired by the cluster members and transmitted to the cluster head. The cluster head then aggregates the data and transmits the information to a base station at a predefined rate. As the target moves in physical space, so does the corresponding cluster in the network. During cluster propagation, the state information regarding the target is handed off from cluster head to cluster head. As we will demonstrate, it is possible for a single target to result in multiple clusters — this is owing to the directional properties of the cameras. Multiple clusters will also result from multiple targets executing motions in the physical space. The results we show in this paper are limited to the case of clusters formed by the motion of a single target.

This paper is organized as follows. In section II, we present an overview of some works related to both clustering in wireless sensor networks and distributed Kalman filtering. In section III, we discuss some of the challenges involved in cluster-based object tracking using wireless camera networks and present the clustering protocol we have designed to cope with these challenges. Section IV presents the main contribution of this paper, the cluster-based Kalman filter. In section V, we present some experimental results obtained using our wireless camera network simulator and our network of wireless cameras. Finally, we conclude the paper in section VI.

II. Related Work

A. Event-driven Clustering Protocols

In environment monitoring applications, the nodes of a sensor network are usually clustered using one of the three

The authors are with the School of Electrical and Computer Engineering, Purdue University, Electrical Engineering Building, 465 Northwestern Ave., West Lafayette, IN 47907-2035. E-mail: {hmedeiro, jpark, kak}@purdue.edu.

different strategies: (1) the nodes may be clustered only once at the system initialization time; (2) periodically based on some predefined network-wide time interval; and (3) aperiodically on the basis of some internal node parameter, such as the remaining energy reserve at the nodes [1], [2], [3]. However, in object tracking applications, clustering must be triggered by the detection of an event of interest external to the network. This section presents some of the works that take external events into consideration in the cluster formation process.

Chen et al. [8] have proposed an algorithm for distributed target tracking using acoustic information. Their system is composed of sparsely placed high-capability nodes and densely spaced low-end sensors. The high-capability nodes act as cluster heads and the low-end sensors as cluster members. Cluster heads close to the detected event become active with higher probability than cluster heads that are farther from the event. Similarly, the probability that a cluster member sends data to the cluster head is proportional to its distance to the event.

Fang et al. [9] have proposed a distributed aggregate management (DAM) protocol, in which nodes that detect energy peaks become cluster heads, and a tree of cluster members is formed by their neighbors that detect lower energy levels. When many targets lie within the same cluster, they use their energy-based activity monitoring (EBAM) algorithm to count the number of targets. EBAM assumes that all the targets are equally strong emitters of energy and counts the number of targets within a cluster based on the total energy detected by the cluster. To drop this assumption, they proposed the expectation-maximization like activity monitoring (EMLAM) algorithm. This algorithm assumes that the targets are initially well separated and uses a motion prediction model along with message exchanges among cluster leaders to keep track of the total number of objects.

Zhang and Cao [10] proposed the dynamic convoy tree-based collaboration (DCTC) algorithm, in which the nodes that can detect an object create a tree rooted at a node near the detected object. As the object moves, nodes are added to and pruned from the tree, and the root moves to nodes closer to the object. This work is similar to our clustering protocol in that it also provides mechanisms for cluster propagation as the object moves. However, no mechanisms are provided for interaction among clusters since a single cluster is formed to keep track of the target. As we will see, in camera networks, it may be necessary to have multiple clusters simultaneously tracking the same target.

Blum et al. [11] proposed a middleware architecture to allow for distributed applications to communicate with groups of sensors assigned to track multiple events in the environment. Their architecture is divided into two modules: the entity management module (EMM) and the entity connection module (ECM). The EMM is responsible for creating unique groups of sensors to track each event, keeping persistent identities to these groups, and storing information about the state of the event. The ECM provides end-to-end communication among different groups of sensors.

All of these works have in common the fact that they are designed for omni-directional sensors. Therefore, they do not account for challenges specific to directional sensors such as cameras. One of these challenges is the fact that physical proximity between a sensor and the target does not imply that the sensor is able to acquire information about the target. Hence, distance based cluster formation protocols are not directly applicable to camera networks. The challenges of sensor clustering in wireless camera networks will be addressed in detail in section III.

B. Distributed Kalman Filtering

The idea of distributing the computations involved in estimation problems using Kalman filters in sensor networks has been a subject of research since the late 1970's [12]. This section presents some of the recent contributions in this area.

Olfati-Saber [13] presented a distributed Kalman filter wherein a system with an np-dimensional measurement vector is first split into n sub-systems of p-dimensional measurement vectors, then these sub-systems are individually processed by micro Kalman filters in the nodes of the network. In this system, the sensors compute an average inverse-covariance and average measurements using consensus filters. These averaged values are then used by each node to individually compute the estimated state of the system using the information form of the Kalman filter. Even though this approach is effective in an environment monitoring application where the state vector is partially known by each node in the network, it is not valid for an object tracking application where, at a given time, each node in a small number of nodes knows the entire state vector (although possibly not accurately).

Nettleton et. al. [14] proposed a tree-based architecture in which each node computes the update equations of the Kalman filter in its information form and sends the results to its immediate predecessor in the tree. The predecessor then aggregates the received data and computes a new update. Node asynchrony is handled by predicting asynchronously received information to the current time in the receiving node. This approach is scalable since the information transmitted between any pair of nodes is fixed. However, the size of the information matrix is proportional to m^2 where m is the dimension of the state vector. In a sensor network setting, this information may be too large to be transmitted between nodes, therefore, methods to effectively quantize this information may need to be devised

Regarding quantization, the work by Ribeiro et. al. [15] studied a network environment wherein each node transmits a single bit per observation, the sign of innovation (SOI), at every iteration of the filter. The system assumes an underlying sensor scheduling mechanism so that only one node transmits the information at a time. It also assumes the update information, i.e., the signs of innovations, to be available to each node of the network. They showed that the mean squared error of their SOI Kalman filter is closely related to the error of a clairvoyant Kalman filter, which has access to all the data in analog form.

There is an interesting trade-off between the works by Nettleton et. al. and Ribeiro et. al.. The former presents a high level of locality, i.e., each node only needs information

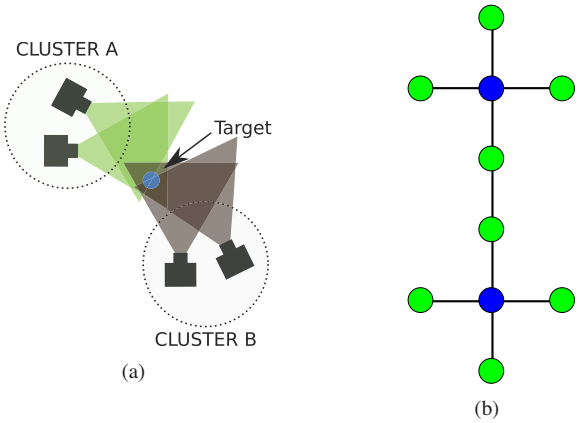


Figure 1. (a) Multiple clusters tracking the same object in a wireless camera network, dotted circles represent the communication ranges of the clusters. (b) Two single-hop clusters in a network of cameras that can communicate in multiple hops. Blue (dark) circles represent cluster heads, green (light) circles represent cluster members. The lines connecting the nodes correspond to communication links among them.

about its immediate neighbors. On the other hand, a reasonably large amount of information must be transmitted by each node. The later, by its turn, requires the transmission of a very small amount of information by each node, however, the algorithm does not present locality since the information must be propagated throughout the network. This kind of trade-off must be carefully considered when designing an algorithm for real wireless sensor network applications.

To the best of our knowledge, the only work that applies Kalman filtering to a cluster-based architecture for object tracking using camera networks is that proposed by Goshorn et. al. [16]. Their system assumes that the network is previously partitioned into clusters of cameras with similar fields of view. As the target moves, information within a cluster is handed-off to a neighboring cluster.

III. Cluster-Based Object Tracking with Wireless Camera Networks

Most of the current event-driven clustering protocols assume that sensors closest to an event-generating target can best acquire information about the target. In wireless camera networks, however, the distance-based criteria for sensor node clustering are not sufficient since, depending on their pointing directions, physically proximal cameras may view segments of space that are disjointed and even far from one another. What that means is that even when only a single object is being tracked, a clustering protocol must allow for the formation of multiple disjointed clusters of cameras to track the same object. An example is illustrated in Figure 1(a) where, in spite of the fact that the cameras in cluster A cannot communicate with the cameras in cluster B, both clusters of cameras can track the object. Therefore, multiple clusters must be allowed to track the same target.

Even if all the cameras that can detect a common object can communicate with one another in multiple hops, the communication overhead involved in tracking the object using a large cluster may be unacceptable as collaborative processing

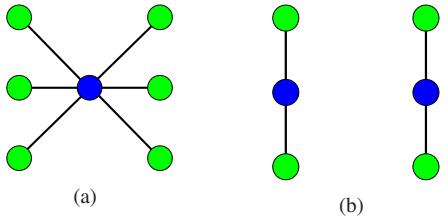


Figure 2. Fragmentation of a single cluster. As the cluster head in (a) leaves the cluster, it is fragmented into two clusters as illustrated in (b).

generally requires intensive message exchange among the cluster members. Therefore, rather than creating a single large multi-hop cluster to track an object, it is often desirable to have multiple single-hop clusters that may interact as needed. An example is illustrated in Figure 1(b) where, whereas all the cameras that can see the same object may constitute a connected graph if we allow multi-hop communications, it may be more efficient to require that two single-hop clusters be formed in this case.

Dynamic cluster formation requires all cluster members to interact to select a cluster head. There are many algorithms available [17],[18] that could be used for electing a leader from amongst *all* the cameras that are able to see the same object. Nevertheless, these algorithms would not work for us since we must allow for the formation of multiple single-hop clusters (for the reasons previously explained) and for the election of a separate leader for each cluster. Therefore, it is necessary to devise a new leader election protocol suitable for the creation of single-hop clusters in a wireless camera network setting.

After clusters are created to track specific targets, these clusters must be allowed to propagate through the network as the targets move. Cluster propagation refers to the process of accepting new members into the cluster as they identify the same object, removing members that can no longer see the object, and assigning new cluster heads as the current cluster head leaves the cluster. Since cluster propagation in wireless camera networks can be based on distinctive visual features of the target, it is possible for clusters tracking different objects to propagate independently, or even overlap if necessary. In other words, cameras that can detect multiple targets may belong simultaneously to multiple clusters. Including a new member into a cluster and removing an existing member from a cluster are rather simple operations. However, when a cluster head leaves the cluster, mechanisms must be provided to account for the possibility that the cluster be fragmented into two or more clusters, as illustrated in Figure 2.

Since multiple clusters are allowed to track the same target, if these clusters overlap they must be able to coalesce into a single cluster. Coalescence of clusters is made possible by permitting the overhearing of intra-cluster communications as different clusters come into each other's communication range. Overhearing obviously implies inter-cluster communication. It is important to note that inter-cluster communication can play a role in intra-cluster computation of a parameter of the environment even when cluster merging is not an issue. For example, a cluster composed of overhead cameras may

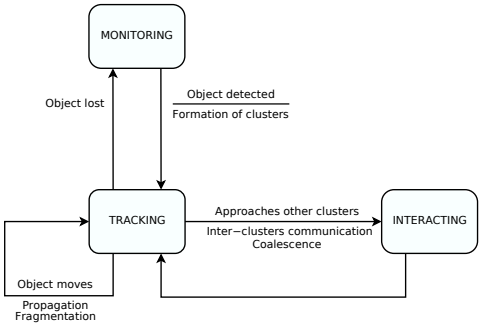


Figure 3. State transition diagram of a cluster-based object tracking system using a wireless camera network.

request information about the z coordinate of the target from a neighboring cluster composed of wall-mounted cameras. Therefore, it is necessary to provide mechanisms to allow inter-cluster interactions in wireless camera networks.

To summarize these points, Figure 3 illustrates the state transition diagram of a cluster-based object tracking system using a wireless camera network. The network initially monitors the environment. As an object is detected, one or more clusters are formed to track this object. To keep track of the object, these clusters must propagate through the network as the object moves and, if necessary, fragment themselves into smaller clusters. Finally, if two or more clusters tracking the same object meet each other, they may interact to share information or coalesce into larger clusters.

One of the primary contributions of the clustering protocol we present below is that it does allow for the formation and propagation of multiple clusters. When needed, the protocol also allows for clusters to interact or coalesce into larger clusters and for large clusters to fragment into smaller clusters. Moreover, our clustering protocol allows for distributed applications to be easily implemented in wireless camera networks since it releases the application of much of the burden of assigning roles to the cameras (i.e., leader/member) and of the collection of the data generated by the cameras.

A. Clustering Protocol

In this section we present our clustering protocol. We believe that the best way to present the protocol would be to show the state transition diagram at each node. Such a diagram would define all of the states of a node as it transitions from initial object detection to participation in a cluster, to possibly its role as a leader, and, finally, to relinquishing its membership in the cluster. Unfortunately, such a diagram would be much too complex for a clear presentation. So instead we have opted to present this diagram in three pieces. The individual pieces we will present in this section correspond to the *cluster formation and head election*, *cluster propagation*, and *inter-cluster communications*. The state transition diagram for cluster propagation includes the transitions needed for cluster coalescence and fragmentation. As the reader will note, our state transitions allow for wireless camera networks to dynamically create one or more clusters to track objects based on visual features. Note that our protocol is lightweight in the

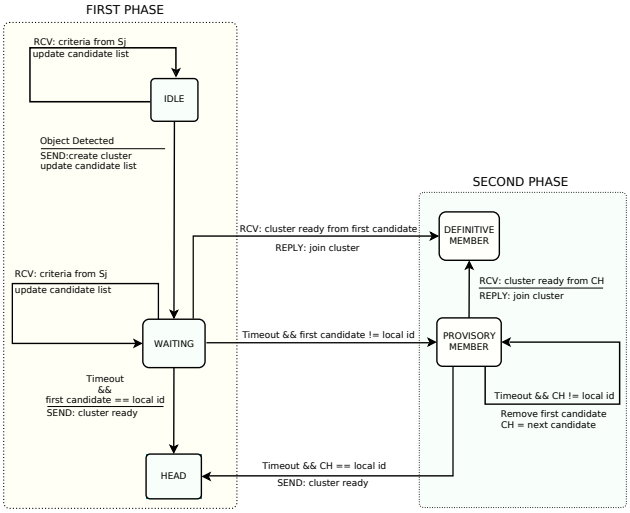


Figure 4. Cluster head election state transition diagram.

sense that it creates single-level clusters, i.e. clusters composed only of cameras that can communicate in a single hop, rather than multiple-level clusters, which incur large communication overhead and latency during collaborative processing and require complex cluster management strategies. Cameras that can communicate in multiple hops may share information as needed by inter-cluster interactions.

1) Cluster Head Election: To select cluster heads for single-hop clusters, we employ a two-phase cluster head election algorithm. In the first phase, nodes compete to find a node that minimizes (or maximizes) some criterion, such as the distance from the camera center to the object center in the image plane. By the end of this phase, at most one camera in a single-hop neighborhood elects itself leader and its neighbors join its cluster. During the second phase, cameras that were left without a leader (because their leader candidate joined another cluster) identify the next best leader candidate.

As illustrated by the state transition diagram on the left side of Figure 4, in the first phase of the cluster head election algorithm, each camera that detects an object sends a message requesting the creation of a cluster and includes itself in a list of cluster head candidates sorted by the cluster head selection criteria. The cluster creation message includes the values of the cluster head selection criteria from the sender. After a camera sends a cluster creation message, it waits for a predefined timeout period for cluster creation messages from other cameras. Whenever a camera receives a cluster creation message from another camera, it updates the list of cluster head candidates. To make sure that cameras that detect the object at later moments do not lose information about the available cluster head candidates, all the cameras that can hear the create cluster messages update their candidate lists. After the end of the timeout period, if the camera finds itself in the first position of the candidate list, it sends a message informing its neighbors that it is ready to become the cluster head. If the camera does not decide to become a cluster head, it proceeds to the second phase of the algorithm.

The first phase of the algorithm guarantees that a single

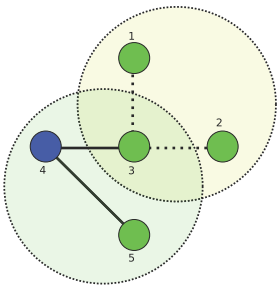


Figure 5. Orphan cameras after the first stage of the leader election algorithm.

camera chooses to become a cluster head within its communication range. However, it might be the case that cameras that can communicate to the cluster head in multiple hops are left without a leader. Figure 5 shows an example of this situation. Cameras 1 and 2 decide that camera 3 is the best cluster head candidate. However, camera 3 chooses to become a member of the cluster headed by camera 4. Hence, cameras 1 and 2 are left orphans after the first stage of the leader election and must proceed to the second phase of the algorithm to choose their cluster heads.

During the second phase of the cluster head election, cameras that did not receive a cluster ready message after a time interval remove the first element of the cluster head candidate list. If the camera then finds itself in the first position of the candidate list, it sends a cluster ready message and becomes a cluster head. Otherwise, the camera waits for a timeout period for a cluster ready message from the next candidate in the list. This process is illustrated in the right side of the state transition diagram of Figure 4. Eventually, the camera will either become a cluster head or join a cluster from a neighboring camera. To avoid that multiple cameras decide to become cluster heads simultaneously, it is important that the cluster head election criteria impose a strict ordering to the candidates (if it does not, ties must be broken during the first phase).

The second phase of our leader election algorithm bears some similarities with Garcia-Molina's bully election algorithm [19]. As a consequence, the algorithm is not robust to communication failures in the network. However, the consequences of communication failures are relatively mild in the sense that, as the algorithm terminates, every cluster will have exactly one cluster head, even if more than one cluster is formed where a single cluster should. This property holds because each camera eventually chooses a cluster head, even if it is itself, and after receiving a cluster ready message from a cluster head, a camera no longer accepts cluster ready messages. Therefore, we believe that the simplicity of the algorithm overcomes its relative lack of robustness.

In the final step of the algorithm, to establish a bidirectional connection among the cluster head and its members, each member sends a message to report the cluster head that it has joined the cluster. This step is not strictly necessary if the cluster head does not need to know about the cluster members. However, in general, for collaborative processing, the cluster head needs to know its cluster members so that it can assign them tasks and coordinate the distributed processing.

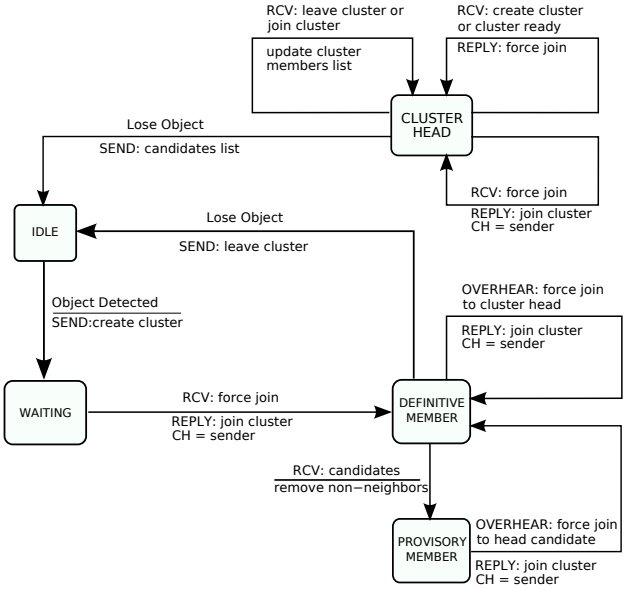


Figure 6. State transition diagram for cluster propagation.

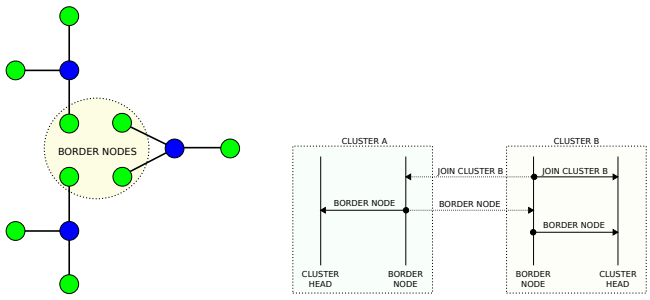
2) Cluster Propagation: Inclusion of new members into active clusters takes place as follows. When a camera detects a new target, it proceeds normally as in the cluster formation step by sending to its neighbors a create cluster message and waiting for the election process to take place. However, if there is an active cluster tracking the same object in the neighborhood of this camera, the cluster head replies with a message requesting the camera to join its cluster. The camera that initiated the formation of a new cluster then halts the election process and replies with a join cluster message.

If there are multiple cluster heads near a camera that has detected a target, the camera could, at the cost of a unit of time delay, choose the cluster head which is closest to the target and become its member. However, we believe that any extra waiting period during cluster propagation should be avoided since it could degrade the tracking performance. Hence, we simply allow a new camera (that has just seen the target) to join the cluster whose cluster head first responds to the camera.

Removal of cluster members is trivial; when the target leaves the field of view of a cluster member, all it has to do is send a message informing the cluster head that it is leaving the cluster. The cluster head then updates its list of cluster members. If the cluster member can track multiple targets, it terminates only the connection related to the lost target.

Figure 6 shows the state transition diagram for cluster propagation. The diagram shows the transitions for inclusion and removal of cluster members as well as cluster fragmentation and coalescence, which we explain below.

a) Cluster Fragmentation: When the cluster head leaves the cluster, we must make sure that, if the cluster is fragmented, each fragment will be assigned a new cluster head. Cluster head reassignment works as follows. We assume that the cluster head has access to the latest information about the position of the target with respect to each cluster member and, consequently, is able to keep an updated list of the best cluster head candidates. We also assume that cluster members



(a) (b)

Figure 7. (a) Border nodes. (b) Messages transmitted to establish inter-cluster connections.

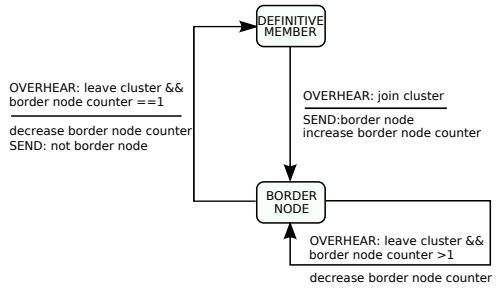


Figure 8. Inter-cluster communication state transition diagram.

know their neighbors. When the cluster head decides to leave the cluster, it sends a message to its neighbors containing a sorted list of the best cluster head candidates. Each cluster member removes from that list all the nodes that are not within its neighborhood. Leader election then takes place as in the second phase of the regular cluster head election mechanism.

b) Cluster Coalescence: When two clusters come within each other's communication range, there can be two possible scenarios: (1) we may either have a non-coalescing inter-cluster interaction, or (2) the clusters may coalesce to form a larger cluster. We will address the non-coalescing inter-cluster interactions in the next section. As far as two clusters coalescing into one is concerned, our cluster head reassignment procedure allows for seamless cluster coalescence. Consider two clusters, A and B, that are propagating toward each other. As the reader will recall, cluster propagation entails establishing a new cluster head as the previous head loses sight of the object. Now consider the situation when a camera is designated to become the new cluster head of cluster A and that this camera is in the communication range of the cluster head of B. Under this circumstance, the camera that was meant to be A's new leader is forced to join cluster B. The members of cluster A that overhear their prospective cluster head joining cluster B also join B. If there are members of cluster A that are not within the communication range of the cluster head of cluster B, they do not join cluster B. Instead, they proceed to select another cluster head for what remains of cluster A following the second phase of the regular cluster head election mechanism.

3) Non-coalescing Inter-cluster Interaction: It may be the case that members of multiple clusters come into one another's communication range but their respective cluster heads are not able to communicate in a single hop. In that case, the clusters are not able to coalesce, but they may need to interact to share information. The same situation prevails when a new cluster comes into existence in the vicinity of an existing cluster but without the head of the former being able to communicate with the head of the later. In both these cases, information can be shared among clusters through border nodes. Border nodes are the nodes that can communicate with other nodes in neighboring clusters, as illustrated in Figure 7(a).

Members of neighboring clusters become border nodes through the border node discovery procedure that takes place as follows. Consider two clusters, A and B, and suppose a new node that joins cluster B is in the communication range of a member of cluster A. As previously explained, when a node joins a cluster, either at cluster creation time or during cluster propagation, it sends a join cluster message to its corresponding cluster head. This is illustrated by the first arrow on the right side of the space-time diagram shown in Figure 7(b). Since the member of cluster A is in the communication range of the new member of cluster B, it can overhear that message (first dashed line) and be aware that it has become a border node. This new border node in cluster A then sends a border node message to its own cluster head informing it that it has become a border node, as illustrated in the cluster A part of Fig. 7(b). When the cluster B member overhears that message (second dashed line), it also becomes a border node and informs its own cluster head of that fact by sending it a border node message.

However, it is not sufficient for a border node to know that it is in the communication range of some member of another cluster. As we illustrated in Figure 7(a), border nodes may communicate with multiple border nodes. Hence, it is necessary for each border node to keep track of how many connections it has to other clusters. This can be achieved by simply incrementing a counter each time a new connection among border nodes is established and decrementing it when a connection is terminated. Figure 8 shows the state transition diagram for inter-cluster communication.

When a cluster head is informed that one of its members became a border node, it can, in effect, request information from the neighboring clusters as needed.

4) Cluster Maintenance: Additional robustness vis-a-vis communication failures is achieved by a periodic refresh of the cluster status. Since our protocol is designed for clusters to perform collaborative processing, we assume that cluster members and cluster heads exchange messages periodically. Therefore, we can use a soft-state based approach [20] to keep track of cluster membership. What that implies is that if the cluster head does not hear from a member within a certain designated time interval, that membership is considered terminated (by the same token, if a cluster member stops receiving messages from its cluster head, it assumes the cluster no longer exists and starts the creation of its own cluster). If a specific application requires unidirectional communication, i.e. communication only from head to members or only from members to head, refresh messages can be sent by the receiver

side periodically to achieve the same soft-state based updating of cluster membership.

Inter-cluster communication can also be maintained in a similar manner. If a border node does not hear from nodes outside its own cluster for a predefined timeout period, it assumes it is no longer a border node. If communication is unidirectional, border nodes can overhear the explicit refresh messages sent by the neighboring cluster's border nodes to their respective cluster heads.

IV. Cluster-based Kalman Filter Algorithm

As we have shown in the previous section, our clustering protocol provides a simple and effective means to dynamically create clusters of cameras while tracking objects with specific visual features. After a cluster is created, cameras within this cluster are able to share information about the target with very small overhead. Besides, the information shared by the cameras is automatically carried by the cluster as it propagates. This protocol provides a natural framework for the design of a decentralized Kalman filter wherein data acquired by the cameras is aggregated by the cluster head and the estimated target position is carried along with the cluster as it propagates.

When designing our Kalman filter algorithm, we took into consideration the two major constraints of wireless sensor networks. First, the processing power of each node is very limited; hence, sophisticated algorithms cannot be implemented in real time. Even relatively simple algorithms must be carefully implemented to use the available hardware resources wisely. Second, communication is very expensive in terms of its energy overhead and in terms of the increased probability of packet collisions should there be too many messages. The second constraint implies that the data must be transmitted sparingly. To deal with the first constraint, we took advantage of the sparsity of the matrices involved in the estimation problem at hand, as explained in detail in section IV-A. To deal with the second constraint, we keep to a minimum the number of messages that need to be exchanged between the nodes for the Kalman filter to do its work. Besides, since most of the traffic in an object tracking application in a wide area network consists of object information transmitted to the base station via multi-hop messages, we limit that traffic by transmitting such information at a predefined rate. The result is an algorithm that is able to estimate the target position in a distributed manner, accurately, and in real-time while reducing overall energy consumption in the network compared to a centralized approach.

A. Kalman Filter Equations

We model our state as a 5-dimensional vector which includes the target position (x_k, y_k) at discrete time instant k, its velocity $(\dot{x}_k, \dot{y}_k)$, and the time interval δ_k between the two latest measurements. That is, the state vector is given by:

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \delta_k & \dot{x}_k & \dot{y}_k \end{bmatrix}^T$$

The dynamic equations of the system are described by the non-linear equations:

$$\mathbf{x}_{k+1} = \begin{bmatrix} x_k + \delta_k \dot{x}_k + \frac{a_x}{2}\delta_k^2 \\ y_k + \delta_k \dot{y}_k + \frac{a_y}{2}\delta_k^2 \\ \delta_k + \varepsilon \\ \dot{x}_k + a_x \delta_k \\ \dot{y}_k + a_y \delta_k \end{bmatrix}$$

In our system, following [21], the target acceleration (a_x, a_y) is modeled by white Gaussian noise. We also model the time uncertainty ε between the latest measurements as white Gaussian noise. It is necessary to consider the time uncertainty since we only want to loosely synchronize the cameras to allow for consistency among their measurements. That is, we do not want to use complex time synchronization algorithms and, therefore, there may be a small time offset in the measurements from different cameras. The dynamic equations can be represented more compactly as:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{w}_k)$$

where $\mathbf{w}_k = \begin{bmatrix} a_x & a_y & \varepsilon \end{bmatrix}^T$ is the process noise vector, assumed white Gaussian with covariance matrix Q.

The measurements are given by the pixel coordinates of the target and the time elapsed between the two most recent measurements. We assume the target moves on the xy plane of the reference frame and that the cameras were previously calibrated, i.e., the homographies between the xy plane of the reference frame and the image plane of each camera are known. Hence, the homographies relate the pixel coordinates to the elements of the state vector corresponding to the target coordinates of the object. The measurement model can now be defined as:

$$\mathbf{z}_k = h_i(\mathbf{x}_k) + \mathbf{v}_k$$

where

$$h_i(\mathbf{x}_k) = \begin{bmatrix} h_{i1}(\mathbf{x}_k) \\ h_{i2}(\mathbf{x}_k) \\ \delta_k \end{bmatrix} = \begin{bmatrix} \frac{H^i_{11}x_k + H^i_{12}y_k + H^i_{13}}{H^i_{31}x_k + H^i_{32}y_k + H^i_{33}} \\ \frac{H^i_{21}x_k + H^i_{22}y_k + H^i_{23}}{H^i_{31}x_k + H^i_{32}y_k + H^i_{33}} \\ \delta_k \end{bmatrix}$$

Here $(h_{i1}(\mathbf{x}_k), h_{i2}(\mathbf{x}_k))$ are the pixel coordinates based on the homography H^i corresponding to camera i, δ_k is the time elapsed between the two most recent measurements, and $\mathbf{v}_k$ is the measurement noise, assumed white Gaussian with covariance matrix R.

Since both the dynamic equations of the system as well as the measurement equations are given by non-linear functions, we designed an extended Kalman filter to estimate the state of the target. The time update equations of the extended Kalman filter are given by:

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, 0) \quad (1)$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + W_k Q W_k^T \quad (2)$$

where $\hat{\mathbf{x}}_{k|k-1}$ and $\hat{\mathbf{x}}_{k-1|k-1}$ are the predicted and the previously estimated state vectors, and similarly $P_{k|k-1}$ and $P_{k-1|k-1}$ are the predicted and previously estimated covariance matrices for the state vector. F_k, the Jacobian matrix of

the state transition function $f(\cdot)$ with respect to $\mathbf{x}_k$, is given by:

$$F_k = \left.\frac{\partial f}{\partial \mathbf{x}_k}\right|_{\hat{\mathbf{x}}_{k-1|k-1},0} = \begin{bmatrix} 1 & 0 & \dot{x}_k & \delta_k & 0 \\ 0 & 1 & \dot{y}_k & 0 & \delta_k \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Similarly, W_k is the Jacobian matrix of $f(\cdot)$ with respect to $\mathbf{w}_k$, given by:

$$W_k = \left.\frac{\partial f}{\partial \mathbf{w}_k}\right|_{\hat{\mathbf{x}}_{k-1|k-1},0} = \begin{bmatrix} \frac{\delta_k^2}{2} & 0 & 0 \\ 0 & \frac{\delta_k^2}{2} & 0 \\ 0 & 0 & 1 \\ \delta_k & 0 & 0 \\ 0 & \delta_k & 0 \end{bmatrix} \quad (4)$$

The measurement update equations for the filter are given by:

$$K_k = P_{k|k-1}H_k^T(H_kP_{k|k-1}H_k^T + R)^{-1} \quad (5)$$
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k(\mathbf{z}_k - h_i(\hat{\mathbf{x}}_{k|k-1}, 0)) \quad (6)$$
$$P_{k|k} = (I - K_kH_k)P_{k|k-1} \quad (7)$$

where H_k is the Jacobian matrix of the function $h_i(\cdot)$ with respect to $\mathbf{x}_k$ evaluated at $\hat{\mathbf{x}}_{k-1|k-1}$, given by:

$$H_k = \left.\frac{\partial h_i}{\partial \mathbf{x}_k}\right|_{\hat{\mathbf{x}}_{k-1|k-1},0} = \begin{bmatrix} \frac{\partial h_{i1}}{\partial x_k} & \frac{\partial h_{i1}}{\partial y_k} & 0 & 0 & 0 \\ \frac{\partial h_{i2}}{\partial x_k} & \frac{\partial h_{i2}}{\partial y_k} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (8)$$

As we can see, the Jacobian matrices in Eqs. (3), (4), and (8) are relatively sparse. This allows us to rewrite the filter equations to achieve an efficient implementation more suitable for wireless sensor nodes with limited processing power. To do so, we first divide the Jacobian matrix F_k of Eq. (3) into four sub-matrices as follows. Let $F_A = I_2$, $F_B = [\mathbf{v}\,|\delta I_2]$, $F_C = 0_{3\times 2}$, and $F_D = I_3$, where I_n is an n-dimensional identity matrix, $0_{n\times m}$ is an $n \times m$ zero matrix, and $\mathbf{v} = (\dot{x}_k, \dot{y}_k)^T$. Then, Eq. (3) becomes[1]:

$$F = \begin{bmatrix} F_{A_{(2\times 2)}} & F_{B_{(2\times 3)}} \\ F_{C_{(3\times 2)}} & F_{D_{(3\times 3)}} \end{bmatrix} = \left[\begin{array}{c|c|cc} I_2 & \mathbf{v} & \multicolumn{2}{c}{\delta I_2} \\ \hline 0_{3\times 2} & \multicolumn{3}{c}{I_3} \end{array}\right]$$

Now let the error covariance matrix $P_{k|k-1}$ be represented by:

$$P = \begin{bmatrix} P_{A_{(2\times 2)}} & P_{B_{(2\times 3)}} \\ P^T_{B_{(3\times 2)}} & P_{D_{(3\times 3)}} \end{bmatrix}$$

then the first term on the right hand side of equation (2) can be rewritten as:

$$FPF^T = U = \begin{bmatrix} U_A & U_B \\ U_B^T & U_D \end{bmatrix} =$$
$$\begin{bmatrix} P_A + F_BP_B^T + \left(F_BP_B^T\right)^T + F_BP_DF_B^T & P_B + F_BP_D \\ \left(P_B + F_BP_D\right)^T & P_D \end{bmatrix}$$

Now if we further subdivide the matrices P_B and P_D into:

$$P_B = \begin{bmatrix} P_{BA_{(2\times 1)}} & P_{BB_{(2\times 2)}} \end{bmatrix}$$

$$P_D = \begin{bmatrix} p_{DA_{(1\times 1)}} & P_{DB_{(1\times 2)}} \\ P^T_{DB_{(2\times 1)}} & P_{DD_{(2\times 2)}} \end{bmatrix}$$

then we have

$$U_A = P_A + \mathbf{v}P_{BA}^T + \delta P_{BB}^T + P_{BA}\mathbf{v}^T + \delta P_{BB} + \mathbf{v}p_{DA}\mathbf{v}^T + \delta\mathbf{v}P_{DB} + \delta P_{DB}^T\mathbf{v}^T + \delta^2P_{DD}$$

$$U_B = \begin{bmatrix} U_{BA} & U_{BB} \end{bmatrix} = \begin{bmatrix} P_{BA} + \mathbf{v}p_{DA} + \delta P_{DB}^T & P_{BB} + \mathbf{v}P_{DB} + \delta P_{DD} \end{bmatrix}$$

$$U_D = P_D$$

Similarly, let the second term on the right hand side of Eq. (2) be represented by:

$$WQW^T = V = \begin{bmatrix} V_A & V_B \\ V_B^T & V_D \end{bmatrix}$$

and let

$$Q = \begin{bmatrix} Q_{xy_{(2\times 2)}} & 0 \\ 0 & q_t \end{bmatrix}$$

then, we have

$$V_A = \frac{\delta^4}{4}Q_{xy}$$

$$V_B = \begin{bmatrix} V_{BA} & V_{BB} \end{bmatrix} = \begin{bmatrix} 0_{(2\times 1)} & \frac{\delta^3}{2}Q_{xy} \end{bmatrix}$$

$$V_D = \begin{bmatrix} v_{DA} & V_{DB} \\ V_{DB}^T & V_{DD} \end{bmatrix} = \begin{bmatrix} q_t & 0_{(1\times 2)} \\ 0_{(2\times 1)} & \delta^2Q_{xy} \end{bmatrix}$$

Finally, the covariance update equation (2) becomes:

$$P_{k|k-1} = U_{k-1|k-1} + V_{k-1|k-1} = \begin{bmatrix} U_A + V_A & U_{BA} & U_{BB} + V_{BB} \\ U_{BA}^T & p_{DA} + v_{DA} & P_{DB} \\ U_{BB}^T + V_{BB}^T & P_{DB}^T & P_{DD} + V_{DD} \end{bmatrix}_{k-1|k-1}$$

Let us now turn our attention to the measurement update equations. Following a similar derivation, the Jacobian matrix H_k of Eq. (8) can be divided into:

$$H = \begin{bmatrix} H_{A_{(2\times 2)}} & H_{B_{(2\times 3)}} \\ H_{C_{(1\times 2)}} & H_{D_{(1\times 3)}} \end{bmatrix} = \left[\begin{array}{cc|ccc} \multicolumn{2}{c|}{H_{A_{(2\times 2)}}} & \multicolumn{3}{c}{0_{(2\times 3)}} \\ \hline \multicolumn{2}{c|}{0_{(1\times 2)}} & 1 & 0 & 0 \end{array}\right]$$

Then, since $H_B = 0_{(2\times 3)}$ and $H_C = 0_{(1\times 2)}$, the first element inside the parenthesis on the right hand side of Eq. (5) becomes:

$$HPH^T = \begin{bmatrix} H_AP_AH_A^T & H_AP_{BA} \\ (H_AP_{BA})^T & p_{DA} \end{bmatrix}$$

Let the covariance matrix of the measurement noise R be represented by

$$R = \begin{bmatrix} R_{A_{(2\times 2)}} & 0 \\ 0 & r_B \end{bmatrix}$$

Then, it follows from the Schur complement that

$$(HPH^T + R)^{-1} = M = \begin{bmatrix} M_A & M_B \\ M_B^T & M_D \end{bmatrix} = \begin{bmatrix} S & -\frac{SH_AP_{BA}}{d} \\ -\left(\frac{SH_AP_{BA}}{d}\right)^T & \frac{P_{BA}^TH_A^TSH_AP_{BA}+d}{d^2} \end{bmatrix}$$

[1]Note that we have dropped the discrete time subscript to simplify the notation.

where $d = r_B + p_{DD}$ and

$$S_{(2\times 2)} = \left(H_A P_A H_A^T + R_A - \frac{H_A P_{BA} P_{BA}^T H_A^T}{d} \right)^{-1}$$

Finally, the equation for the Kalman gain, Eq. (5), then becomes:

$$K = PH^T M = \begin{bmatrix} P_A H_A^T & P_{BA} \\ P_{BA}^T H_A^T & p_{DA} \\ P_{BB}^T H_A^T & P_{DB}^T \end{bmatrix} \begin{bmatrix} M_A & M_B \\ M_B^T & M_D \end{bmatrix}$$

It is important to note that in the above equations all operations are carried out on small matrices. Besides, many elements used in each step of the computations are reused in later steps. Therefore, it is possible to temporarily store them and reuse them later, saving computation time.

B. State Estimation

Algorithm 1 summarizes the state estimation algorithm that runs at each node of the wireless camera network. As we already mentioned, we use our clustering protocol as the underlying framework for the implementation of the Kalman filter. Therefore, the algorithm is initialized when a camera joins a cluster (either as a cluster member or as a cluster head) and is terminated when the camera leaves the cluster. Therefore, the algorithm only runs while a camera is actively tracking a target. After the target leaves the field of view of the camera, it may switch to an energy saving mode that periodically observes the environment to detect the presence of new targets. In that sense, the initialization step presented in Algorithm 1 is a local intra-cluster process that prepares the cameras to track a specific target, as opposed to the network wide system initialization procedure described in section IV-C.

The initialization in the state estimation algorithm takes place after cluster formation is concluded, and its main goals are to initialize the Kalman filter and to synchronize the cluster members so that they can estimate the state of the target consistently. To synchronize the cluster members, the newly elected cluster head broadcasts a message to its cluster members informing its current time. The cluster members then synchronize their internal clocks to the time received from the cluster head and time-stamp any future measurements based on that time. It is important to note that neither clock drifts between the cluster head and its cluster members nor potential message delays (caused by the underlying algorithm for medium access control or by message queuing) are taken into consideration in this approach. Therefore, this time synchronization is inherently not very accurate. This is the reason why we modeled the uncertainty in the measurement times in the state estimation equations, as explained in section *IV-A*.

It is interesting to note that our synchronization approach is similar to the post-facto synchronization using undisciplined clocks [22], except for the fact that the source of the synchronization message is the cluster head rather than a third-party node. Therefore, this method provides only a local consistent time suitable for the Kalman filter to operate. To achieve global time synchronization for the nodes, the time stamp can be propagated along with the clusters using a synchronization protocol such as TPSN [23]. Evidently, this method does not provide a time stamp consistent with any external reference (such as the time in the base station). Therefore, it is only possible to know the position of the target at a given time with respect to the instant it is initially detected. To obtain a global reference consistent with an external clock, a more complex time synchronization algorithm is required. Some of the existing options are surveyed in [24].

Algorithm 1 Cluster-Based Kalman Filter

```
Initialization
    if cluster_head = local_ID
        initialize target state
        broadcast time stamp

Maintenance
    new local measurement available:
        if cluster_head = local_ID
            estimate target state
        else
            send measurement to
            cluster head
    measurement received:
        estimate target position
    sample period elapsed:
        if cluster_head = local_ID
            send current estimated
            state to the user

Termination
    if cluster_head = local_ID
        send current estimated state
        to new cluster head
```

In our approach, the cluster head is responsible for estimating the state of the target and reporting it to the base station. Therefore, whenever a cluster member acquires a new measurement, it sends it to the cluster head which updates the estimated target state based on that measurement. Similarly, when a cluster head acquires a new local measurement, it also updates the state of the target based on this measurement. However, since the base station may be multiple hops away from the cluster, transmitting every new estimate to the user could result in high energy consumption. Therefore, the cluster head sends information to the base station at a predefined rate.

Since the cluster head is responsible for keeping track of the state of the target, as the cluster propagates and a new cluster head is chosen, it is necessary to hand-off the state information to the new cluster head. As explained in section III-A2, our clustering protocol allows information about the state of the target to be carried by the cluster as new cluster heads are assigned. Therefore, all our Kalman filter algorithm has to do is piggy-back a message containing the target state to the cluster head reassignment message. After the new cluster head is assigned, it continues state estimation.

It is important to note that the simplicity of Algorithm 1 is only possible due to the underlying clustering protocol that handles all the aspects of distributed data collection and information hand-off among different cameras.

C. System Initialization

As we observe in the Kalman filter equations presented in section IV-A, the cluster head needs to know the homographies between the cluster members and the xy plane in order to estimate the position of the target based on the measurements acquired by the cluster members. Therefore, to avoid transmitting large amounts of data while tracking the object, when the system is initialized each node stores the homographies of its one-hop neighbors, i.e., its potential cluster members.

The system initialization works as follows. When a new camera joins the network, it broadcasts its own homography to its one-hop neighbors. When a camera receives a homography from a new neighboring camera, it stores this homography and replies to this camera by sending its own homography. Even though this procedure can take $O(n^2)$ steps, where n is the number of nodes in the network, it is a local algorithm, meaning that no information needs to be broadcast beyond a single hop neighborhood. Therefore, if we assume that m is the average number of nodes in a single hop region, the algorithm terminates in expected $O(m^2)$ iterations. Since we can assume that $m \ll n$ for a wide area camera network, the algorithm is feasible in practice.

V. Experiments

To evaluate the performance of our algorithm, we carried out a number of experiments in a simulated multi-hop network. Also, in order to demonstrate the feasibility of the system in practice, we implemented the object tracking system in a real network of wireless cameras.

A. Simulations

We implemented a camera network simulator that provides the pixel coordinates of a moving target based on each camera's calibration matrix. Figures 9(a) and (b) show two views of the graphical user interface of the simulator. The simulator creates a target that moves randomly or that follows a predefined trajectory in the xy plane in the world frame. The simulated cameras operate independently and the data generated by each camera is output via TCP connection to an individual node in a sensor network simulation using the Avrora simulator [25]. Avrora is a simulator for the Atmel AVR family of microcontrollers used in the Mica motes [26]. Avrora is capable of performing simulations with high accuracy on code natively generated for the AVR microcontrollers. The simulator also provides a free space radio model that allows for simulation of wireless communications in sensor networks. Our Kalman filter code, along with the clustering protocol, both implemented in the nesC language [27] and running under the TinyOS operating system [28], were executed directly in Avrora.

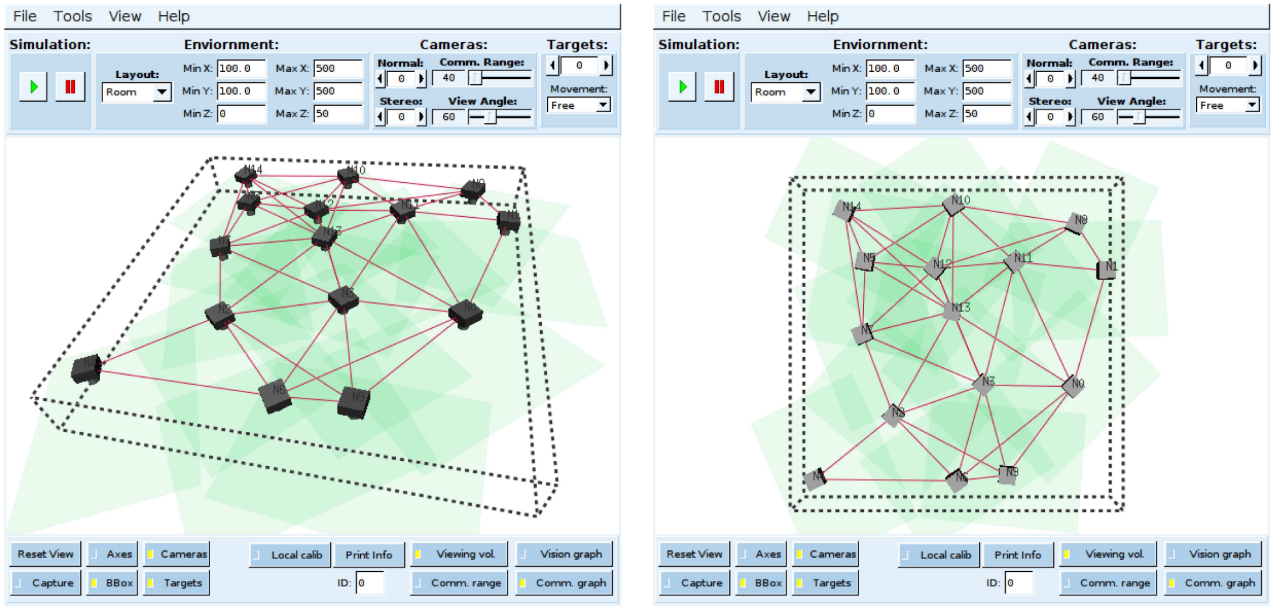
We used our testbed described above to evaluate the performance of the proposed cluster-based Kalman filter for object tracking. Figure 9 shows the configuration of 15 wireless cameras used in the experiment. All the cameras were randomly placed on the top plane of a cuboid volume with the dimension of $50 \times 50 \times 5$ meters. Each camera node is assumed to have a communication range of $18m$ in all directions, and a view angle of 120 degrees. Each red line between a pair of cameras shown in Figure 9 indicates that the two cameras are able to communicate directly (i.e., one-hop neighbors), and each green pyramid represents the viewing volume of the camera. It is assumed that all cameras have been fully calibrated (both intrinsic and extrinsic parameters of the cameras are known). Additionally, we assume that all objects move on the floor (i.e., the bottom plane of the working environment), which allows each camera to compute the 2D world coordinates of the object given its image coordinates using the homography relating the camera plane and the floor. Based on the results of the experiments in [6] we set the clustering timeout to $300ms$. This timeout value provides an adequate balance between the effectiveness of cluster formation and the speed with which a cluster is able to follow a target.

1) Estimation Accuracy: To evaluate the accuracy of our algorithm, we introduced Gaussian random noise into the camera measurements and computed the root mean squared error of the estimates obtained by our system. We then compared the performance of our cluster-based Kalman filter with the performance of a centralized tracking method. In the centralized approach, we transmitted all the data collected by the network to the base station, which then applied a centralized Kalman filter to the data. Figure 10(a) shows plots of the root mean squared error of the unfiltered data, of our distributed Kalman filter, and that of the centralized Kalman filter as a function of the standard deviation of the measurement noise. As we can see, our algorithm is able to substantially reduce the error in the unfiltered data. Although the centralized Kalman filter is able to provide more accurate results, it requires the transmission of all the data to the base station.

It is important to note that this centralized tracker is an idealized concept since *every* message generated by the motes is processed. It does not consider the message drops that would occur in a real centralized tracker as the messages are routed to the base station. These message drops are difficult to quantify, however, since they depend on the network topology and the distance between the nodes that detect the event and the base station.

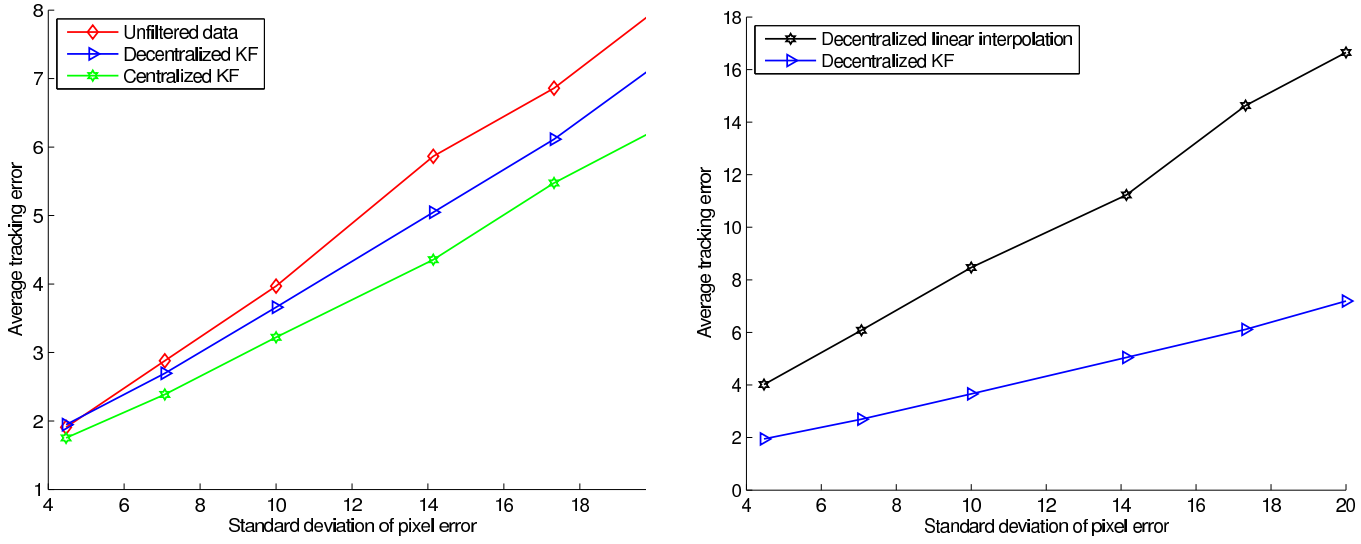
We also compared the accuracy of our cluster-based Kalman filter to that of an alternative decentralized tracking method. In the alternative method, we used linear interpolation for local data aggregation. That is, the target position was periodically estimated by linearly interpolating the two most recent measurements available to the cluster head. The results of the experiment are shown in Figure 10(b). Due to the noisy nature of the data, the performance of linear interpolation degrades significantly as the standard deviation of the pixel error increases. As the experiment shows, even for small pixel error, our algorithm is able to significantly reduce the total error when compared to local data aggregation using linear interpolation. This performance gain becomes larger as the pixel error increases.

2) Average Number of Messages Transmitted: To evaluate the potential energy savings obtained by restricting the number of multi-hop messages transmitted by the system, we measured the average number of messages transmitted per node per



(a) (b)

Figure 9. GUI showing the configuration of 15 wireless cameras. The red lines connecting pairs of cameras indicate communication links between camera nodes, and the green tetrahedral volumes represent the field of views of cameras.



(a) (b)

Figure 10. (a) Root mean squared error of the target position as a function of the standard deviation of the pixel noise. (b) Performance of the cluster-based algorithm compared to a decentralized tracker using linear interpolation.

minute in our simulator using our Kalman filter algorithm. We also measured the average number of messages needed to transmit all the information to the base station, which is required by the centralized Kalman filter approach. Figure 11 shows the number of messages transmitted as a function of the average distance to the base station. In order to estimate the number of messages required to reach the base station, we multiplied the number of messages by the average distance to the base station.

The results of the experiment show that, for networks of small diameter where the average distance to the base station is small, the number of messages transmitted by our algorithm is higher than transmitting all the data to the base station due to the overhead introduced by clustering. However, as the average distance to the base station grows, the number of messages transmitted to the base station in the centralized system increases while the clustering overhead remains constant. Eventually, a threshold is reached where sending all the messages becomes more expensive than creating the clusters. This threshold depends on the sampling period of the cluster-based Kalman filter. For example, for the case of a sampling period of 750ms, the cluster-based approach performs better than the centralized approach for an average distance to the base station larger than 2 hops.

3) Model Error: As explained in section IV-A, we model the movement of our target as a constant velocity movement

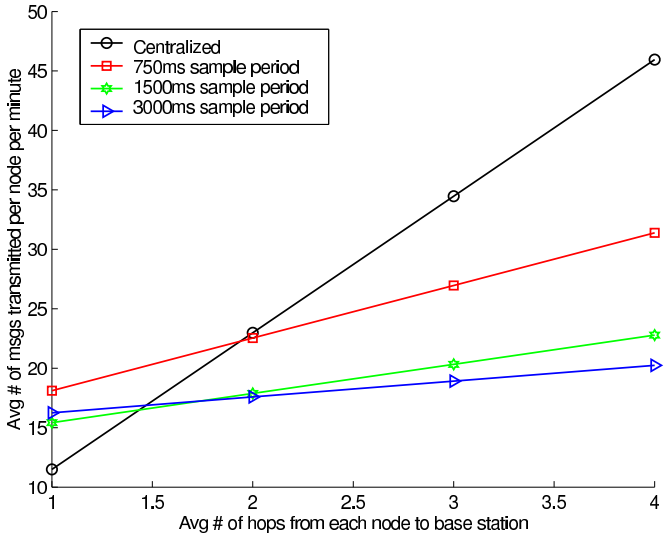


Figure 11. Number of messages transmitted by the system as a function of the average distance to the base station.

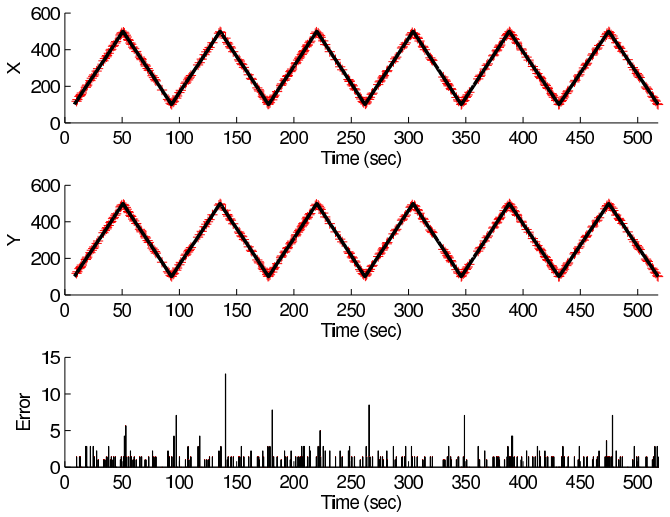


Figure 12. Estimation error in the target position for varying movement. Solid curves represent the ground truth, and the superimposed red markers the filtered data.

with random acceleration. The more the target movement resembles this model, the better the estimates obtained. This is illustrated in Figure 12; the two topmost plots in the figure show the x and y positions of the target as a function of time. The plot on the bottom shows the corresponding error in the target position after applying the Kalman filter. As the figure shows, at the time instants when the target undergoes abrupt changes in direction, the error is larger since the constant velocity model is not valid. On the other hand, the algorithm is able to accurately track the target as long as its movement is smooth. This is illustrated in Figure 13 where we plot the x and y positions of a target moving with constant velocity in the y direction and with varying accelerations in the x direction. The bottom plots show the corresponding root mean squared error with respect to the ground truth. As we can see, although the acceleration in each plot is different, the average error remains approximately constant.

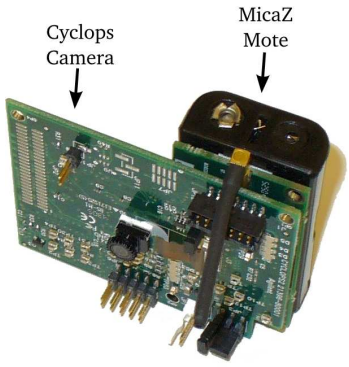


Figure 15. Cyclops camera attached to a MicaZ mote.

Overall, the total error obtained with the decentralized Kalman filter is comparable to the error obtained using the centralized Kalman filter, as qualitatively illustrated in Figure 14. Figure 14(a) shows the x and y coordinates of the target when tracked by the ideal centralized Kalman filter. Figure 14(b) shows the x and y coordinates of the target when tracked by the decentralized Kalman filter.

However, as we see in the figure, due to the delays introduced by the clustering protocol, our decentralized algorithm occasionally loses track of the target. Tracking is lost when the sensor network is engaged in cluster formation. In the example presented in Figure 14, if we consider any two measurements more than 5 seconds apart as defining a region that is not covered by the tracker, while the ideal centralized tracker is able follow the target approximately 95% of the total distance traveled by the target, the cluster-based version is able to keep track of the target approximately 77% of the total distance.

4) Computation Time: Since the Avrora simulator provides instruction level accuracy at the actual clock frequency of the microcontroller, it was possible to compute precisely the time it takes to perform each step of our algorithm. In our current implementation, the time required for the Kalman filtering update is 15ms whereas the prediction takes 4.5ms. The variance in these measurements is very small since it is only due to the time required to attend to the interrupt service routines; those times are of the order of microseconds.

B. Experiments on Real Wireless Cameras

To demonstrate the feasibility of our algorithm, we tested it on a network that consists of 12 Cyclops cameras [29] attached to MicaZ motes (Figure 15) mounted on the ceiling of our laboratory. The cameras are spaced roughly $1m$ (39 inches) apart so that there is partial overlap between the fields of view of the neighboring cameras. The field of view of all the cameras covers a region of about 5 by 3.5 meters (16.4 by 11.5 feet). Figure 16 shows a picture of our wireless camera network.

To evaluate the performance of the system while tracking an object, we moved the object randomly and, at the same time, computed the target coordinates using the wireless camera network and a single wired camera at 30 frames per second. The data gathered by the wired camera was used as the ground truth. Figure 17 shows the trajectory of the object for

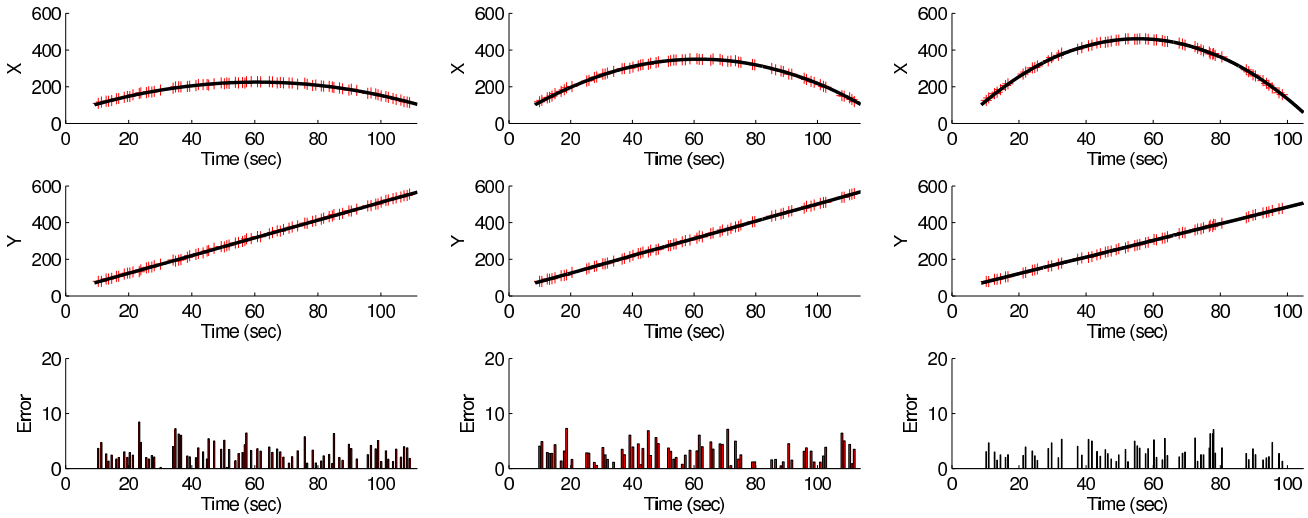


Figure 13. Position of the target as a function of time for different values of acceleration. Solid curves represent the ground truth, and the superimposed red markers the filtered data.

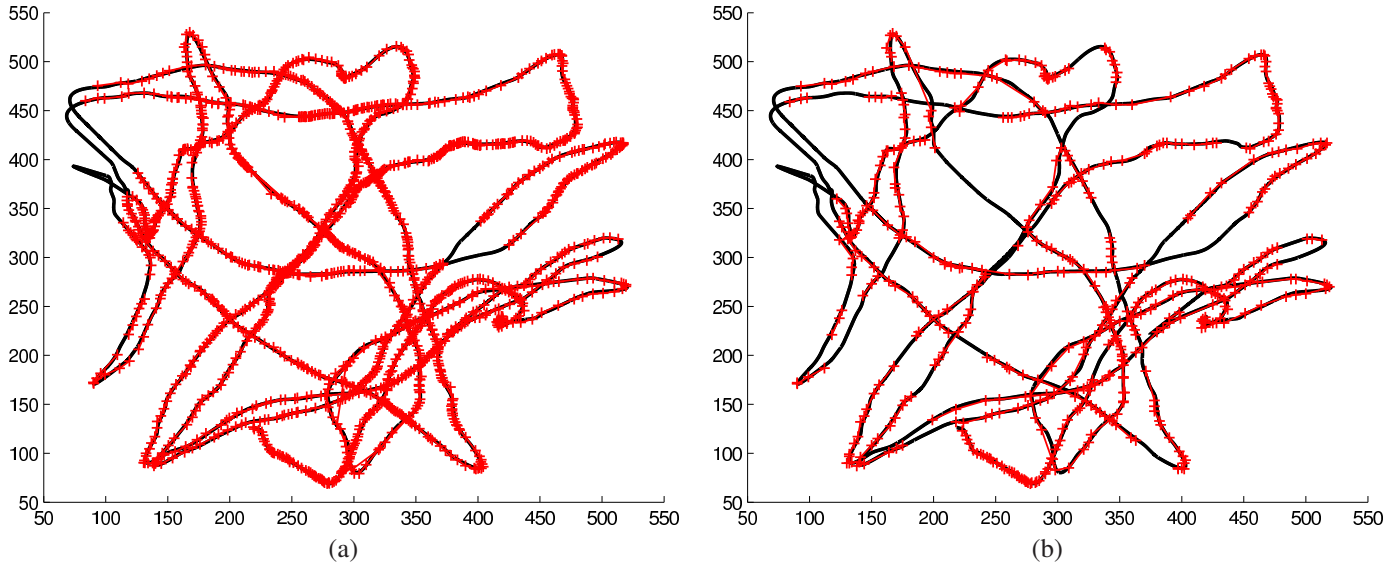


Figure 14. Tracking results for (a) ideal centralized Kalman filter and (b) decentralized Kalman filter. The solid black curves represent the ground truth of the target trajectory. The superimposed red curves with markers represents the estimated positions of the target.

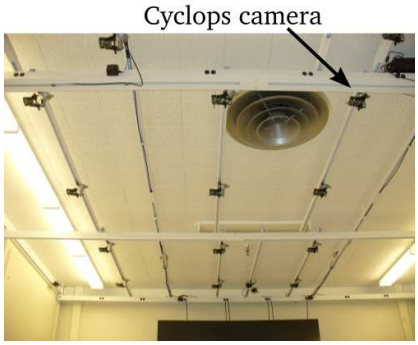


Figure 16. Ceiling mounted wireless cameras.

three different runs of the experiment. The ground truth is represented by the solid black lines, the dashed lines show the trajectory of the target as computed by the wireless cameras. The markers placed on the dashed tracks correspond to the actual target positions computed by the wireless cameras. The reason for showing both the trajectory with the dashed line and the markers on this line is to give the reader a sense of when the system loses track of the target. Each time the track is lost, the system creates a new object identifier for the moving target. This is illustrated by the different markers in Figure 17.

VI. Conclusion

We have presented an object tracking algorithm suitable for a network of wireless cameras. The algorithm has a low message overhead and can be executed in real time even in resource constrained wireless sensors such as the MicaZ motes. It represents an effective approach for local data aggregation in the context of object tracking by wireless camera networks. The algorithm uses a clustering protocol to establish connections among cameras that detect the target and

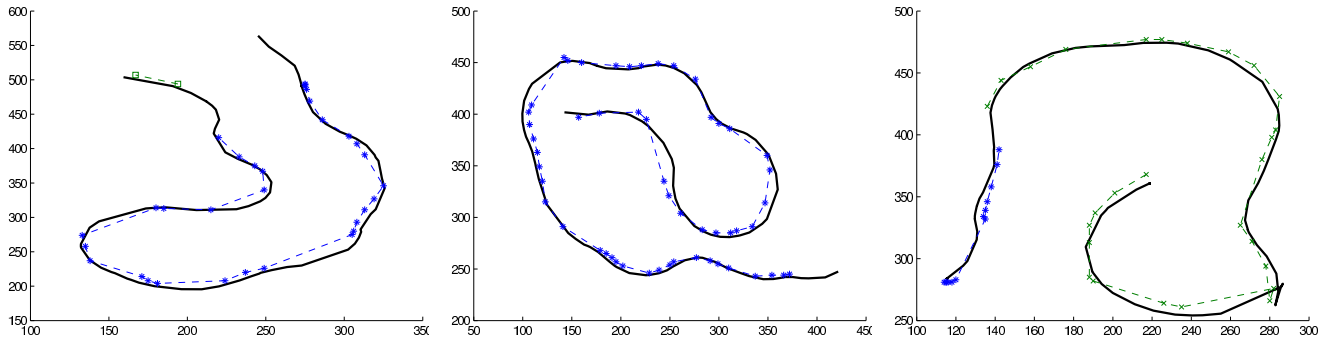


Figure 17. Tracking performance for three different runs of the tracking experiment.

to enable the propagation of the state of the target as it moves. Data aggregation is carried out using a decentralized Kalman filter.

Regarding the accuracy of our filter, our experiments have shown that the algorithm is indeed capable of tracking a target with accuracy comparable to that achieved by a centralized approach wherein every measurement is transmitted to the base station. Although Wan and van der Merwe [30] reported that the extended Kalman filter can introduce large errors in the estimated parameters and that the unscented Kalman filter (UKF) [31] usually provides better results, we found that our approach is able to substantially decrease the noise in the target position and that filter instability was not an issue in our application. Furthermore, the implementation of the UKF requires the computation of the square-root of the covariance matrix of the augmented state vector which includes the state variables and the noise variables. In our case, such matrix would be of dimension 11×11 and, even if we employ efficient square-root implementations as suggested in [32], the computation of such matrix would be too complex for the resource-limited processors used by wireless cameras. Even if we consider the fact that the measurement noise is additive and apply the technique proposed in [31] and [32] to reduce the dimension of the augmented state vector, we would still need to compute the square-root of an 8×8 matrix. Nevertheless, it may be possible to obtain a more effective implementation of UKF if we take into account the sparsity of the matrices involved in the problem as we did in this paper for the EKF. Therefore, devising an efficient implementation of a cluster-based UKF for object tracking using wireless camera networks and comparing it with the performance of our cluster-based EKF is a subject of further investigation.

In this paper we have focused on the problem of a single cluster tracking a single object. The issues of multiple clusters tracking the same object and the inter-cluster interactions involved in that process as well as tracking multiple objects simultaneously are subjects of future studies. Besides, since the focus of this work is on the cluster-based Kalman filter, further analysis of the clustering protocol itself is necessary. Although some preliminary experimental results regarding the clustering protocol were presented in [6], further investigation of our protocol is needed with respect to the density of cameras with common viewing areas as well as the density of single hop neighbors since these parameters greatly influence the overhead involved in the clustering protocol and the performance of local data aggregation.

VII. Acknowledgments

This work was supported by Olympus Corporation. We greatly acknowledge Dr. Akio Kosaka for the helpful discussions about this work.

References

[1] S. Bandyopadhyay and E. Coyle, "An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks," in *Proc. IEEE INFOCOM*, 2003.

[2] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[3] O. Younis and S. Fahmy, "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[4] I. Gupta, D. Riordan, and S. Sampalli, "Cluster-head Election Using Fuzzy Logic for Wireless Sensor Networks," in *Proceedings of the 3rd Annual Communication Networks and Services Research Conference*, 2005, pp. 255–260.

[5] V. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N. Shroff, "A Minimum Cost Heterogeneous Sensor Network with a Lifetime Constraint," *IEEE Transactions on Mobile Computing*, vol. 4, no. 1, pp. 4–15, 2005.

[6] H. Medeiros, J. Park, and A. Kak, "A Light-Weight Event-Driven Protocol for Sensor Clustering in Wireless Camera Networks," in *First IEEE/ACM International Conference on Distributed Smart Cameras*, Sep. 2007.

[7] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," University of North Carolina at Chapel Hill, Tech. Rep., 1995.

[8] W.-P. Chen, J. Hou, and L. Sha, "Dynamic Clustering for Acoustic Target Tracking in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 3, 2004.

[9] Q. Fang, F. Zhao, and L. Guibas, "Lightweight Sensing and Communication Protocols for Target Enumeration and Aggregation," in *ACM Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[10] W. Zhang and G. Cao, "DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks," *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, Sep. 2004.

[11] B. Blum, P. Nagaraddi, A. Wood, T. Abdelzaher, S. Son, and J. Stankovic, "An Entity Maintenance and Connection Service for Sensor Networks," *Proceedings of the 1st international Conference on Mobile Systems, Applications and Services (MobiSys)*, May 2003.

[12] J. L. Speyer, "Computation and Transmission Requirements for a Decentralized Linear-Quadratic-Gaussian Control Problem," *IEEE Transactions on Automatic Control*, Apr. 1979.

[13] R. Olfati-Saber, "Distributed Kalman Filter with Embedded Consensus Filter," in *44th IEEE Conference on Decision and Control, and the European Control Conference*, Dec. 2005.

[14] E. Nettleton, H. Durrant-Whyte, and S. Sukkarieh, "A Robust Architecture for Decentralised Data Fusion," in *International Conference on Advanced Robotics*, 2003.

[15] A. Ribeiro, G. Giannakis, and S. Roumeliotis, "SOI-KF: Distributed Kalman Filtering With Low-Cost Communications Using the Sign of Innovations," *IEEE Transactions on Signal Processing*, vol. 54, no. 12, pp. 4782–4795, Dec. 2006.

[16] R. Goshorn, J. Goshorn, D. Goshorn, and H. Aghajan, "Architecture for Cluster-based Automated Surveillance Network for Detecting and Tracking Multiple Persons," in *First IEEE/ACM International Conference on Distributed Smart Cameras*, Sep. 2007.

[17] N. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1997.

[18] G. Tel, *Introduction to Distributed Algorithms*. Cambridge, 1994.

[19] H. Garcia-Molina, "Elections in a Distributed Computing System," *IEEE Transactions on Computers*, vol. c-31, Jan. 1982.

[20] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, 3rd ed. Addison Wesley, 2005.

[21] Y. Bar-Shalom and Xiao-Rong Li, *Estimation and tracking: principles, techniques, and software*. Artech House, 1993.

[22] J. Elson and D. Estrin, "Time synchronization for wireless sensor networks," in *Proceedings 15th International Parallel and Distributed Processing Symposium*, Apr. 2001, pp. 1965–1970.

[23] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2003, pp. 138–149.

[24] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Network*, vol. 18, no. 4, pp. 45–50, 2004.

[25] B. L. Titzer, D. K. Lee, and J. Palsberg, "Avrora: scalable sensor network simulation with precise timing," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, 2005.

[26] J. L. Hill and D. E. Culler, "Mica: a wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, Dec. 2002.

[27] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," in *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, 2003, pp. 1–11.

[28] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, *TinyOS: An Operating System for Sensor Networks*. Springer, 12 Dec. 2005, pp. 115–148.

[29] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: in situ image sensing and interpretation in wireless sensor networks," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005.

[30] E. A. Wan and R. van der Merwe, "The unscented Kalman filter for nonlinear estimation," in *The IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium, AS-SPCC*, 2000, pp. 153–158.

[31] S. J. Julier and J. K. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear Systems," in *Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls*, 1997.

[32] E. A. Wan and R. van der Merwe, "The Unscented Kalman Filter," in *Kalman Filtering and Neural Networks*, S. Haykin, Ed. Wiley Publishing, 2001, ch. 7.

Henry Medeiros received the BE and MS degrees in Electrical Engineering from the Federal Center of Technological Education, Parana, Brazil, in 2003 and 2005 respectively. Since 2005 he has been pursuing his PhD degree in the School of Electrical and Computer Engineering at Purdue University. His current research interests include sensor networks, computer vision, and embedded systems.

Johnny Park received the BS, MS, and PhD degrees from the School of Electrical and Computer Engineering, Purdue University, in 1998, 2000, and 2004, respectively. Since 2004, he has been a principal research scientist at Purdue University. His current research interests are distributed and collaborative information processing in sensor networks and various topics in computer vision and robotics, including human posture estimation, real-time 3D reconstruction, visual servoing, and 3D object recognition. He is a member of the IEEE.

Avinash C. Kak is a professor of electrical and computer engineering at Purdue University. His research and teaching include sensor networks, computer vision, robotics, and high-level computer languages. He is a coauthor of Principles of Computerized Tomographic Imaging, which was republished as a classic in applied mathematics by SIAM, and of Digital Picture Processing, which is also considered by many to be a classic in computer vision and image processing. His recent book Programming with Objects (John Wiley & Sons, 2003) is used by a number of leading universities as a text on object oriented programming. His latest book Scripting with Objects, also published by John Wiley, focuses on object-oriented scripting. These are two of the three books for an "Objects Trilogy" that he is creating. The last, expected to be finished sometime in 2008, will be titled Designing with Objects.