First ACM/IEEE International Conference on
Distributed Smart Cameras, ICDSC '07.
Vienna, Austria, 25-28 Sept. 2007

A LIGHT-WEIGHT EVENT-DRIVEN PROTOCOL FOR SENSOR CLUSTERING IN WIRELESS CAMERA NETWORKS

Henry Medeiros, Johnny Park and Avinash Kak

School of Electrical and Computer Engineering
Purdue University, West Lafayette, Indiana 47907-2035
{hmedeiro, jpark, kak}@purdue.edu

ABSTRACT

We propose a light-weight event-driven protocol for wireless camera networks to allow for formation and propagation of clusters of cameras for the purpose of collaborative processing during object tracking. Cluster formation is triggered by the detection of objects with specific features. Our protocol allows for simultaneous formation and propagation of multiple clusters. Cameras being directional devices, more than one cluster may track a single object since groups of cameras outside each others communication range may see the same object. Entry into a cluster and cluster membership maintenance require a sensor node to confirm the presence of features of the object being tracked. Each cluster elects its own leader among the cameras that observe the same target. When a cluster leader loses track of an object, it assigns the leadership role to another cluster member. To avoid high communication overhead among cluster members, single-hop clusters are formed, i.e., every member of a cluster is within the communication range of the cluster head. We have implemented a simple version of this protocol on a test-bed and provide an experimental evaluation.

Index Terms— sensor clustering, object tracking, collaborative processing

1. INTRODUCTION

Previous work on sensor clustering has focused primarily on extending the lifetime of a network by partitioning it into clusters to enable data aggregation at a local level [1], [2]. When sensor networks are used for event-driven applications (as opposed to environment monitoring applications), not all sensors provide useful information at the same time. The goal in event-driven clustering is to select a subset of sensors that maximize some information function that depends on the position of the event source and on the characteristics of the sensors. This function must be maximized while the cost related to exchanging information among cluster members is minimized [3].

Most of the current event-driven clustering algorithms assume that the distances between the sensors and the event-generating targets are somehow related to the information function mentioned above. 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 algorithm must allow for the formation of multiple disjointed clusters of cameras for tracking the same object. One of the primary contributions of our protocol is that it does allow for the formation and propagation of multiple clusters. When needed, the protocol also allows for clusters to coalesce into larger clusters and for large clusters to fragment into smaller clusters. Coalescence of clusters is made possible by the permitted 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 request information about the z coordinate of the target from a neighboring cluster composed of wall-mounted cameras.

Object tracking is the specific focus of the camera clustering protocol we present in this paper. Cluster formation is triggered by the detection of object features that are keyed to specific objects. Our protocol allows for simultaneous formation and propagation of multiple clusters and interaction between them. Each cluster uses simple selection rules to elect its own leader. When a cluster leader loses track of an object, it assigns the leadership role to one of its members that is in the best position to maintain a "lock" on the target object.

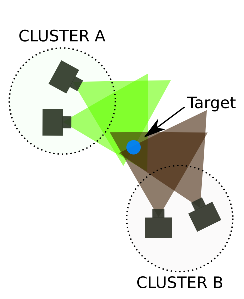
In order to test its practical feasibility, we implemented a simple version of the protocol on a testbed consisting of 12 ceiling-mounted Cyclops [4] cameras attached to micaZ motes. This camera network was used to track a simple object scurrying around on the floor.

This paper is organized as follows. The next section presents some of the related work on event-based cluster formation for collaborative processing. In section 3 we present an overview of our work on cluster-based object tracking using wireless camera networks. In section 4 we present the proposed clustering protocol. In section 5 we present our testbed implementation. Section 6 then presents the experiments carried out using the testbed. Finally, in section 7, we conclude and discuss possible future extension of our work.

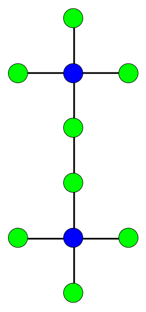
2. RELATED WORK

Among the works that take into consideration external events in the cluster formation process, Chen et al. [5] 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. [6] have proposed a distributed aggregate management (DAM) algorithm in which nodes that detect energy peaks become cluster heads, and a tree of cluster members is formed by its neigh-

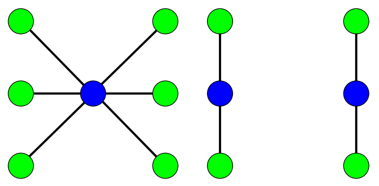


(a)



(b)

Fig. 1. (a) Multiple clusters tracking the same object in a wireless camera network. (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.



(a) (b)

Fig. 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).

bors that detect lower energy levels. When many targets lie within the same cluster, Fang et al. use their energy-based activity monitoring (EBAM) algorithm to count the number of targets. By assuming a motion prediction model, they present a target-counting algorithm in which, as targets approach each other, their corresponding cluster heads exchange information and the clusters merge into a single cluster.

In a previous contribution that is closely related to ours, Zhang and Cao propose the dynamic convoy tree-based collaboration (DCTC) [3] in which 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.

Blum et al. [7] have 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, to keep persistent identities to these groups, and to store information about the state of the event. The ECM provides end-to-end communication among different groups of sensors.

3. OBJECT TRACKING WITH WIRELESS CAMERA NETWORKS

Wireless camera networks allow for tracking of multiple objects based on their unique visual features. To be able to track the targets robustly and precisely, resource-constrained wireless cameras may need to collaborate to process information acquired from the targets.

Clustering is a common technique for data aggregation and collaborative processing in wireless sensor networks. In object tracking applications, clusters are usually created to keep track of a specific target. Once a cluster is created to track an object, connections among cluster members can be established to allow for collaborative processing.

Clustering in wireless camera networks gives rise to issues not present in networks of omnidirectional sensors. In a camera network, different sensors tracking the same object are not necessarily close to one another, thus clusters may be created in different regions of the network 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 requires, in general, intensive message exchange among the cluster members. Therefore, rather than requiring 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.

Dynamic cluster formation requires all cluster members to interact to select a cluster head. There are many algorithms available [8],[9] that could be used for electing a leader from amongst *all* the cameras that are able to see the same object. But these algorithms will not work for us since we must allow for the formation of multiple clusters (for reasons previously explained) and for the election of a separate leader for each cluster. As illustrated in figure 1 (b), whereas all the cameras that can see the same object may constitute a connected graph if you allow for multiple-hop communications, our protocol would require that two single-hop clusters be formed in this case.

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 is based on object features, it is possible for the 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 by 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. In addition, as these clusters approach each other, they may interact to exchange information about the state of the target to improve their estimates about the target position. 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

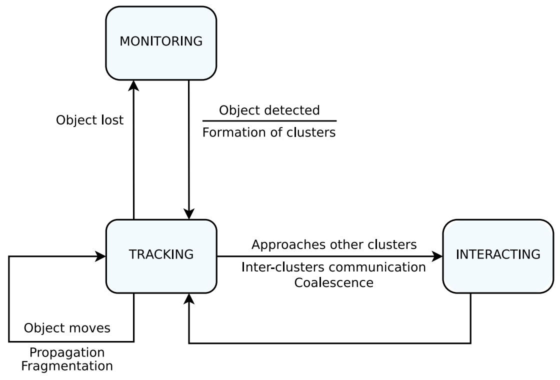


Fig. 3. State transition diagram of an object tracking system based on our protocol

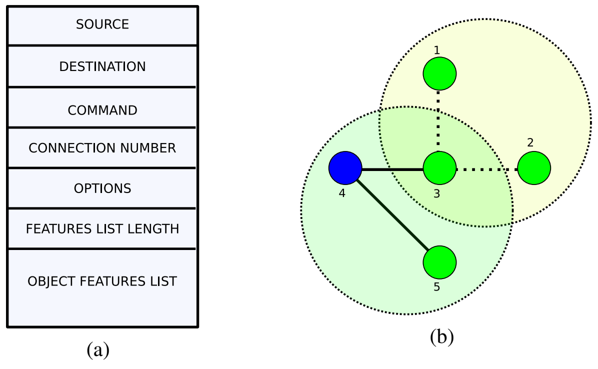


Fig. 4. (a) Protocol message format. (b) Orphan cameras after the first stage of the leader election algorithm.

diagram of an 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.

4. 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 large for the presentation here. 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 light-weight in the 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.

4.1. Message Format

Figure 4 (a) shows the format of the messages used in the clustering protocol. Source and destination fields have obvious meanings. The destination field also allows a broadcast address so that messages may be transmitted to all the neighbors in the communication range of a node. The command field corresponds to the commands used in the protocol. Connection number is a unique number defined by the cluster head to identify a connection to exchange information about an object. After clusters are formed, cluster members can use the pair $(cluster\,head\,identifier, connection\,number)$ to exchange information with the cluster head about a specific object. The options field contains command-specific information, such as the cluster leader election criteria. The features list length field specifies the length of the object features list, which may vary depending on the application. Finally, the object features list field contains the list of visual object features used during clustering to uniquely identify an object.

4.2. 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 5, 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 selection criteria. The cluster creation message includes, in the options field, the value of the cluster 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 candidates lists. After the end of the timeout period, if the camera finds itself in the first position of the candidates 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 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 4 (b) shows an example of this situation. Cameras 1 and 2 decide

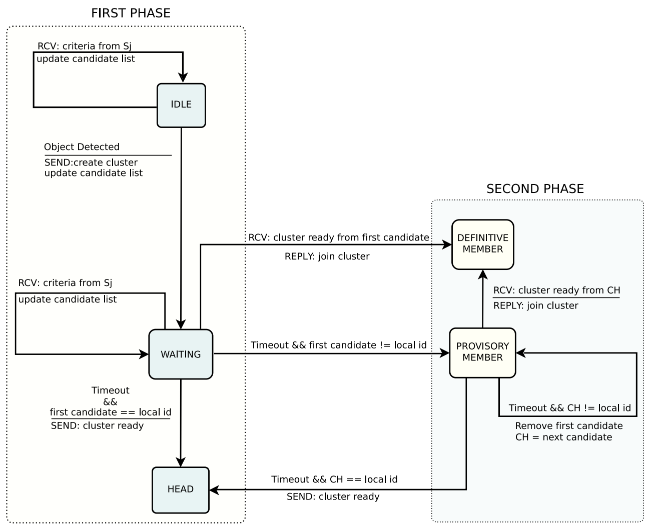


Fig. 5. Cluster head election state transition diagram.

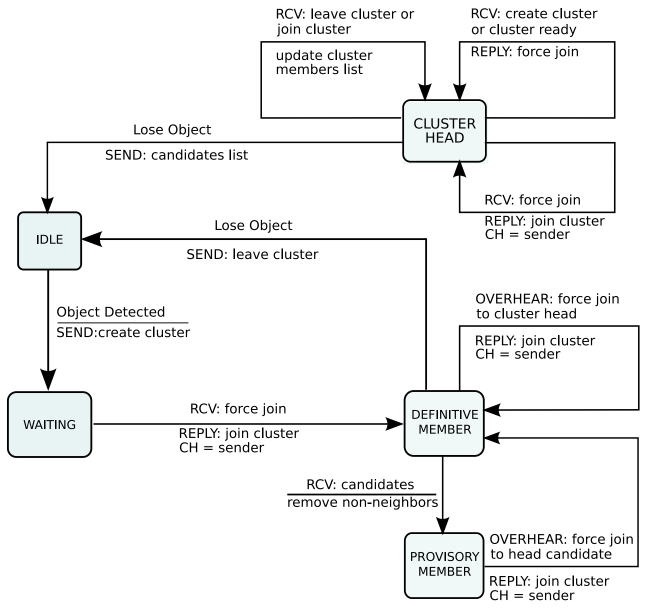


Fig. 6. State transition diagram for cluster propagation.

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 candidates list. If the camera then finds itself in the first position of the candidates 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 5. 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 [10]. 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 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 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.

4.3. 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 during cluster propagation an extra waiting period would degrade the tracking performance. Hence, we allow a new camera (that has just seen the target) to simply 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.

4.3.1. 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

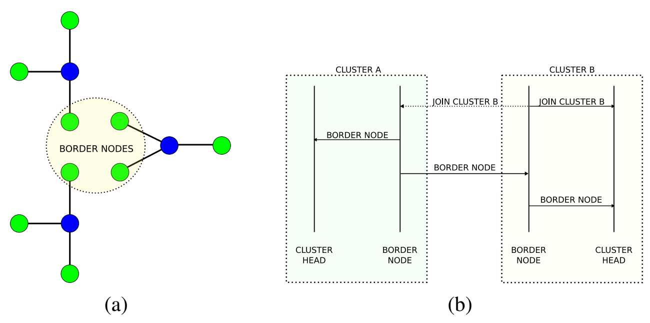


Fig. 7. (a) Border nodes. (b) Messages transmitted to establish inter-cluster connections.

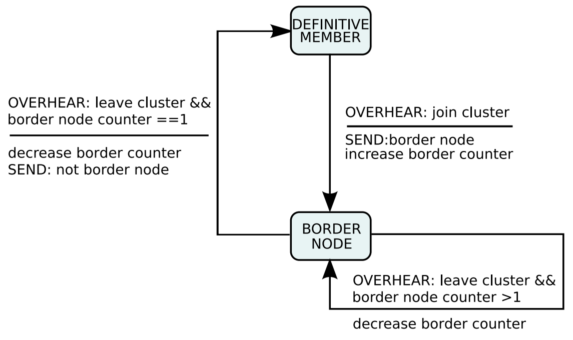


Fig. 8. Inter-cluster communication state transition diagram.

candidates. We also assume that cluster members 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 leader election mechanism.

4.3.2. 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 another. 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 leader election mechanism.

4.4. Non-coalescing Inter-cluster Interaction

There are two possible cases in which clusters may need to interact without coalescing. In the first case, two clusters propagate towards each other until their communication ranges overlap. The second case corresponds to the creation of a new cluster within the communication range of an active cluster (see figure 1 (b) for an example). In any case, information can be shared among clusters through border nodes. Border nodes correspond to nodes that can communicate to other nodes in two or more clusters, as illustrated in figure 7 (a).

As we explained in previous sections, clusters propagate as new cameras that detect an object being tracked by an active nearby cluster are forced to join that cluster. When two clusters approach each other, these messages can be overheard by members of the neighboring cluster. As illustrated by the state-space diagram in figure 7 (b), when a member of an active cluster overhears a message (dashed line) of a camera which is tracking the same object joining a different cluster, it sends a message to its cluster head informing that it became a border node. It also informs the camera whose message was overheard that it should become a border node. This camera, by its turn, also informs its cluster head that it became a border node.

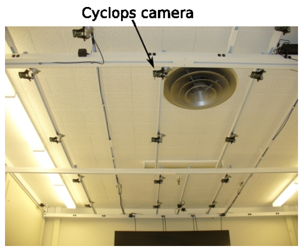
However, it is not sufficient for a border node to know that it is in the communication range of some member of another cluster. As we illustrated in figure 7 (a), border nodes may communicate with multiple border nodes. Therefore, it is necessary for each border node to keep track of how many connections 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.

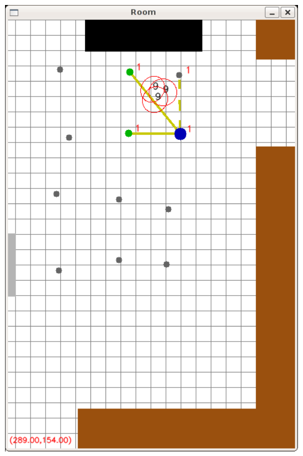
4.5. Cluster Maintenance

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 [11] 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.



(a)



(b)

Fig. 9. (a) Ceiling mounted wireless cameras for the testbed. (b) Graphical user interface implemented to display the clusters and their attributes.

5. TESTBED IMPLEMENTATION

The protocol was tested on a wireless network of 12 Cyclops cameras attached to micaZ motes mounted on the ceiling of our laboratory. The cameras are spaced about 40 inches from each other so that the field of view of each camera partially overlaps with those of its neighbors. The field of view of all the cameras covers a region of about 16 by 12 feet. Figure 9 (a) shows a picture of the testbed. The cameras were calibrated by the calculation of planar homographies between the floor of the laboratory and the camera planes. As the object to be tracked moves on the floor, each camera that sees the target is able to compute the coordinates of the centroid of its image with respect to the world coordinate frame.

Since the focus of this work is on clustering protocols, we use only simple objects in our tracking experiments. For such objects, detection is carried out by thresholding the color histogram. Therefore, our list of object features consists simply of flags to indicate whether an object matches a given histogram (more robust algorithms such as [12] could be used to achieve similar tracking performance while allowing cameras to dynamically assign identifiers to the objects being tracked). The histogram based segmentation algorithm yields a binary image of the target which is processed with a standard recursive labeling algorithm to compute the coordinates of the centroid of the target with respect to the image frame. The mote then receives the pixel coordinates from the attached Cyclops camera via the serial interface and, based on the calibration parameters for the camera, computes the coordinates as well as the covariance matrix of the target location in the world reference plane. The mote also executes the clustering protocol and handles the associated communications.

During collaborative processing, cluster members share information about the state of the target. As the clusters propagate, this information is carried by the clusters so that it may be used by new cameras to improve the estimated state of the target. To implement this behavior, the cameras within a cluster share an object identifier that is defined simply by the numerical ID of the first camera that detects the target. This information is carried along by the clusters as they propagate during object tracking. Whenever this information is lost, for instance if cluster propagation fails and a new cluster is created to track the object, the network loses previous information about the target and a new object identifier is created by the next camera that detects the object. Note that our approach to maintaining cluster state can be extended to include additional parameters regarding the state of the object and its motion.

To visualize the dynamic behavior of the network, we implemented a graphical user interface that displays the clusters during all their phases. Figure 9 (b) shows the display panel of this GUI. The blue circle represents a cluster head and green circles connected to the cluster head by solid lines represent cluster members. Gray circles represent cameras that do not belong to any cluster. Yellow solid lines represent the connections among cluster members and their respective cluster head. The yellow dashed line represents a connection that should have been established but was not due to a communication failure. The red ellipses represent the 95% uncertainty region of the target position with respect to each camera that can detect the target. The expected value of the target position is displayed at the left bottom of the screen. The numbers inside the ellipses correspond to the object identifiers. The large rectangles, brown ones on the right and the bottom, black at the top, and light gray on the left, correspond to pieces of furniture present in the room that are represented in the GUI to facilitate in the visualization of the movement of the target.

6. EXPERIMENTS

We used our testbed to evaluate the performance of the proposed clustering protocol. Our initial experiments were carried out using a single target object and focus on the correctness of cluster creation and propagation in a real application.

To simulate an unsychronized network, we introduced at each camera a random delay period before starting monitoring the environment. This delay follows a uniform distribution between zero and the camera sampling time which, in our current implementation, is approximately one second.

6.1. Head Election Efficiency

To estimate the efficiency of the cluster head election algorithm, we position the target at a specific location and trigger cluster formation using a base-station. After a cluster is formed, the cluster head sends a message to the base-station informing it of that fact. Based on the position of the target and the homographies of the cameras that participate in the election, we compute the distance of the object center from the camera center in the image plane of each camera and use that information to rank order the cameras with regard to their suitability as cluster leaders. Note that rank-ordering of the cameras in each cluster is based on our knowledge of the camera positions vis-a-vis the position of the target. By head election efficiency, we mean the frequency with which the head election algorithm produces a result that agrees with the manually-generated topmost ranked camera. With the target position information, we are also able to know exactly which cameras should join the cluster. In our testbed, since the cameras are mounted in a grid layout facing the floor with partially overlapping fields of view, at most four cameras can be part of any cluster. We performed 50 runs of the experiment positioning the target in locations where clusters of 2, 3, and 4 members (including the cluster head) should be formed. Figures 10 (a) to (c) show the cluster head efficiency as a function of the election algorithm timeout period. In each case, the topmost curve in figures 10 (a) to (c) shows

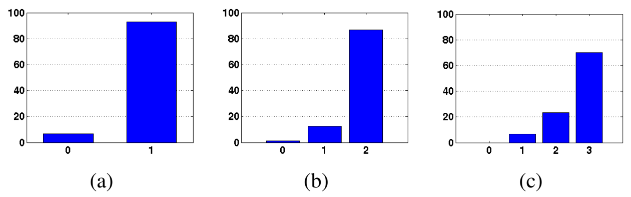


Fig. 11. Average number of members that join a cluster of (a) 2, (b) 3, and (c) 4 elements.

the average percentage of the time the camera elected to be head was also the topmost ranked camera. The curve below the first in each figure shows the percentage of the time the camera elected to be the head was actually the second-ranked camera in the manual ranking process. Similarly, when more than two cameras are present in the cluster, the percentage of the time the third and fourth-ranked cameras were elected cluster heads are represented by the bottommost curves.

There are two main reasons that contribute to the election of an incorrect leader. The first and most obvious is communication failure. If the cluster ready message sent by the correct cluster head is lost, a camera may join a cluster headed by a less suitable leader. The effects of communication failures are mitigated, however, by the cluster coalescence process that forces such cameras to join the cluster headed by the best cluster head (as explained in subsection 4.3.2). The second reason for the election of an incorrect leader is due to the asynchronous nature of the network. If what would have been the correct cluster head did not acquire an image of the target by the time a cluster is formed, it has no option but to join a previously formed cluster headed by the next best camera. The protocol itself does not offer any self-correcting measures for fixing this problem. This is corroborated by the fact that fewer incorrect cluster heads are elected when we increase the cluster formation timeout period. In our implementation, for a timeout of approximately 60% of the sampling period of the cameras, the correct cluster head was selected about 90% of the time. This problem is eliminated when the timeout period is longer than the sampling period of the cameras. Of course, the price to pay for that is the reduction in the overall speed with which clusters would be able to follow a target (implying that there would be a limitation on the speed of the target if tracking is to be successful). We believe that the performance of the algorithm can be significantly improved (without incurring the speed penalty) if we impose loose synchronization among cameras that can communicate in a single hop.

6.2. Cluster formation quality

Often, due to communication failures, not all cameras that should join a cluster actually do so. To quantify partially formed clusters, we used the same experimental setup used to evaluate the election process as described in the previous section. In each message reporting the formation of a cluster, the cluster head also includes a list of its current members. Figures 11 (a) to (c) show the average over 250 runs of the experiment of the number of members (not including the cluster head) that joined the clusters for clusters of 2, 3, and 4 elements, respectively.

As in the previous experiment, the reasons for incomplete clusters are communication failures and the asynchronous nature of the network. It is important to note that the results displayed in figure 11 correspond to the status of the cluster immediately after the cluster creation process has concluded. Subsequent cluster modifications due to cluster coalescence are not considered.

6.3. Tracking Efficiency

To evaluate the performance of the system while tracking an object, we move the object randomly and simultaneously compute the target coordinates using the wireless camera network and a firewire camera at 30 frames per second. The data gathered by the firewire camera is used as ground truth. Figure 12 shows the trajectory of the object for three different runs of the experiment. The ground truth is represented by the solid black line, 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 target positions computed by the wireless cameras. We used different markers to illustrate the moments when the wireless network loses track of the object and a new object identifier is created, i.e., when cluster propagation fails and a new cluster is created to track the object.

7. CONCLUSION

We presented a light-weight event-driven clustering protocol for wireless cameras. As is well recognized, clustering is critical to energy-efficient collaborative processing in sensor networks. Any clustering protocol must address issues of cluster formation, propagation, coalescence, fragmentation, extinction, and interaction among multiple clusters. Our protocol addresses all of these. We believe that because cameras are directional devices, multiple cluster formation and coalescence are important for wireless camera networks. Our protocol addresses all the phases in a single coherent framework.

Our future goals include a more formal analysis of the correctness and performance of the protocol under different conditions, especially when the network is called upon to track multiple objects simultaneously. We also intend to evaluate, using simulations, the performance of the system in larger and denser networks. Besides, our protocol assumes that all cameras that can see the target join a cluster. Nonetheless, it is possible to extend the protocol so that, after a cluster is formed, the cluster head may choose which cameras it wishes to collaborate with using certain camera selection criteria based on how well a camera sees a target [13], [14].

8. ACKNOWLEDGMENTS

This work was supported by Olympus Corporation.

9. REFERENCES

[1] S. Bandyopadhyay and E.J. 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, pp. 660–670, Oct. 2002.

[3] W. Zhang and G. Cao, “DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks,” *IEEE Transactions on Wireless Communications*, vol. 3, Sept. 2004.

[4] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, “Cyclops: in situ image sensing and

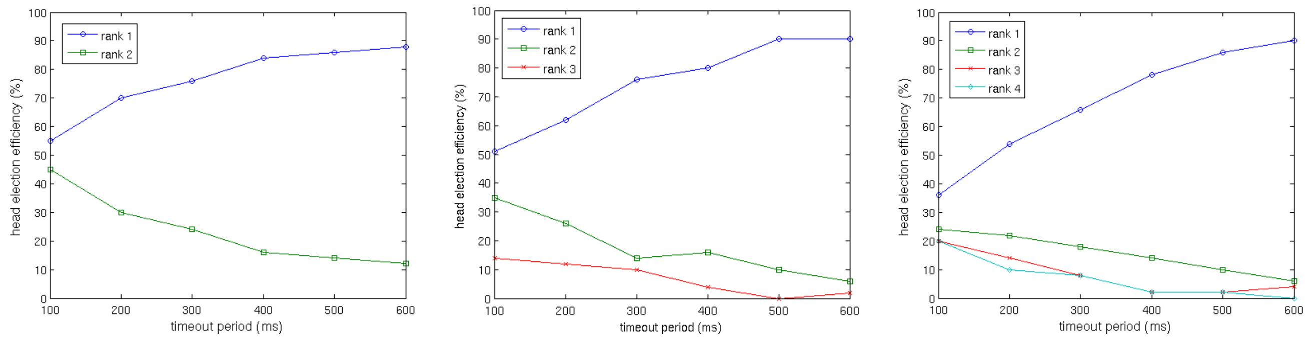


Fig. 10. Head election efficiency as a function of the timeout period for clusters of (a) 2, (b) 3, and (c) 4 members.

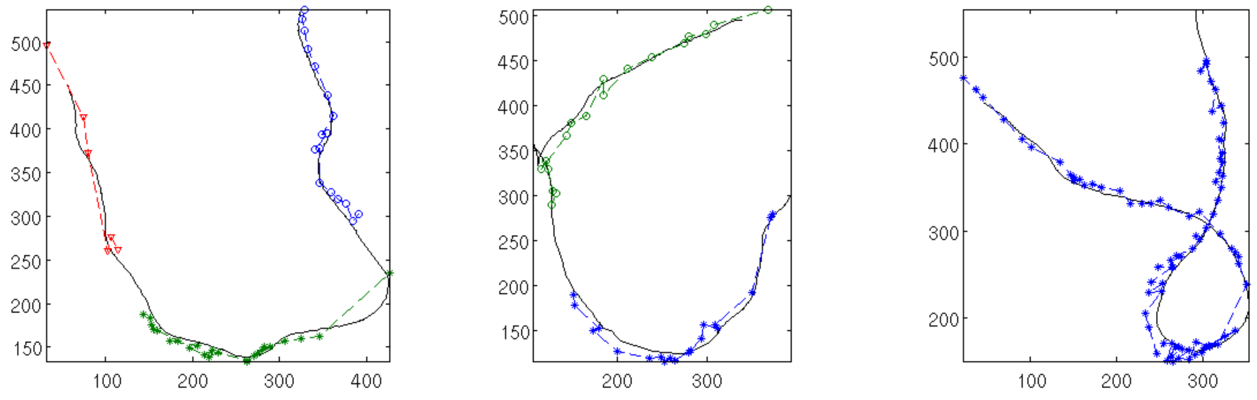


Fig. 12. Tracking performance for three different runs of the tracking experiment.

interpretation in wireless sensor networks," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005.

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

[6] 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.

[7] 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.

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

[9] Gerard Tel, *Introduction to Distributed Algorithms*. Cambridge, 1994.

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

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

[12] F. Lau, E. Oto, and H. Aghajan, "Color-Based Multiple Agent Tracking for Wireless Image Sensor Networks," in *Advanced Concepts for Intelligent Vision Systems (ACIVS)*, Sept. 2006.

[13] J. Park, P. C. Bhat, and A. C. Kak, "A Look-up Table Based Approach for Solving the Camera Selection Problem in Large Camera Networks," in *Workshop on Distributed Smart Cameras, in conjunction with ACM SenSys'06*, 2006.

[14] A. O. Ercan, D. B. Yang, and A. El Gamal, "Optimal Placement and Selection of Camera Network Nodes for Target Localization," in *Proceedings of the International Conference on Distributed Computing in Sensor Systems*, 2006.