Cluster-Based Distributed Face Tracking in Camera Networks

Josiah Yoder, Henry Medeiros, Johnny Park, and Avinash C. Kak

Abstract—In this paper, we present a distributed multicamera face tracking system suitable for large wired camera networks. Unlike previous multicamera face tracking systems, our system does not require a central server to coordinate the entire tracking effort. Instead, an efficient camera clustering protocol is used to dynamically form groups of cameras for in-network tracking of individual faces. The clustering protocol includes cluster propagation mechanisms that allow the computational load of face tracking to be transferred to different cameras as the target objects move. Furthermore, the dynamic election of cluster leaders provides robustness against system failures. Our experimental results show that our cluster-based distributed face tracker is capable of accurately tracking multiple faces in real-time. The overall performance of the distributed system is comparable to that of a centralized face tracker, while presenting the advantages of scalability and robustness.

Index Terms—Camera networks, distributed tracking, face tracking, object detection.

I. INTRODUCTION

AS humans, our faces play a central role in how we communicate with one another in face-to-face encounters. While the importance of face recognition in such communications is universally known, less widely acknowledged are the roles played by the orientation of a face and the movement of the head that help us understand many aspects of nonverbal communications. The orientation of a face typically indicates the visual focus of attention [1], [2] and can be an important source of information in ascertaining how a person is interacting with his/her environment. For example, in an application scenario involving a supermarket, if a computer vision system needed to figure out as to what object a customer was currently looking, the orientation of the face would be a strong indicator of that. Face/head orientation is also used in several important forms of nonverbal communications, such as when a person is nodding his/her head to express agreement, or when a person is shaking his/her head sideways to express the opposite.

The fact that the pose of a person's face holds important clues as to how he/she is interacting with the environment has mo-

tivated several researchers to work on face detection [3]–[6], pose estimation [5], [7]–[12], and on face tracking [13]–[19]. It goes without saying that the ability of a computer system to detect and track people's faces in real time will open doors to a host of new applications ranging from human-computer interaction to surveillance. The contributions we have cited previously have focused on extracting the needed information from single camera images. However, it stands to reason that simultaneously using multiple images taken from different viewpoints can only lead to more robust estimation of the pose of a face, not to speak of the enhanced ability to track the face/head of a person in motion. Indeed, in the generic context of object tracking, there has been much interest in combining information from multiple cameras [20]–[22].

In the existing work on the use of several cameras simultaneously for estimating the pose of a face, a single computer pulls together either all of the images captured by the cameras or the features extracted from all the images. These centralized approaches to pose estimation and tracking may involve extensive comparisons of the images, as in dense-stereo reconstruction [23] or as in the construction of active appearance models [12], [24]. Such approaches are not easy to implement in a distributed computing environment composed of smart cameras — a theme central to the work reported here. More appropriate for distributed implementations are those prior contributions that use lightweight object features extracted from the individual camera images [25]–[28]. Note that these contributions still require a central server to process either all the images or the features extracted from all the images. There are two major shortcomings to all methods that use a single processor for the computation of the face pose: 1) The processor creates a single point of failure and a prominent point of vulnerability in the system; and, perhaps even more importantly, 2) the number of cameras that can be connected to the processor is determined by the capabilities of the processor. For those reasons, our focus here is on face pose estimation and tracking algorithms that, from the ground up, are designed specifically for a distributed implementation.

A distributed approach to the estimation of face pose and to its tracking evidently requires coordination among the camera nodes that are focusing on solving a particular instance of the problem. As is now a common practice in wireless sensor networks, such coordination is best achieved if one of the nodes is chosen to serve as a leader. In wireless sensor networks, a collaborating set of nodes is usually referred to as a *cluster* and the leader as a *cluster head* [29]–[33]. In this paper, we will use *cluster leader* or simply *leader* to refer to the camera node that is coordinating the computations being carried out by a collection of camera nodes. This is to avoid confusion with *head* that, in the present context, is used more appropriately to refer to the
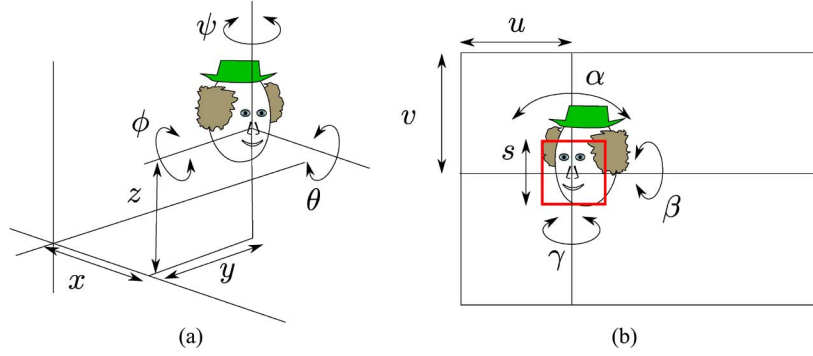
Fig. 1. 6-DOF face pose representation in the world coordinates and in the image space. (a) 6-DOF pose in the world coordinates, $\bar{\mathbf{p}}_w = [x, y, z, \theta, \phi, \psi]^T$. (b) 6-DOF pose in the image space, $\bar{\mathbf{p}}_i = [u, v, s, \alpha, \beta, \gamma]^T$.

anatomical part of the human whose face is the focus of pose estimation and tracking.

In addition to the need for a cluster leader, a distributed approach would also require only those camera nodes that are relevant to the task at hand to participate in pose and tracking computation. That is, assuming a moderate to large network of cameras, we would want only those camera nodes that can actually see a face to try to compute the pose of the face and to contribute to its tracking. As the bearer of the face moves, this collection of the camera nodes must also "move" accordingly. This dynamic formation of a camera node cluster and the subsequent propagation of the cluster requires what is known as a clustering protocol in the jargon of wireless sensor networks [34]–[40]. However, these mainstream clustering protocols are only appropriate for nondirectional sensor nodes. On account of the directional and other unique properties of wireless camera sensors [41], [42], a clustering protocol that is more suited for wireless camera networks is the one we presented in [43]. Our goal in this paper is to show how our clustering protocol can be modified so that it can be used in a network of wired cameras and serve as the underlying framework for the design of a distributed face tracking algorithm.

II. MultiCamera Face Tracking Algorithm

We start by presenting in this section a multicamera algorithm for face tracking. Our approach is based upon the fact that one can carry out a rough localization of a face, with respect to all its six degrees of freedom (6-DOF), in a single camera image. Estimates of the position and the orientation of a face in the world coordinates, as gleaned from the individual images in a camera network, constitute a set of world-based 6-DOF pose observations for the face. These observations from the individual cameras, along with the uncertainties associated with them, are then integrated in the world coordinates using a minimum variance estimator. A face is tracked in the world coordinates using this integrated estimate.

A. 6-DOF Face Pose Representations

Before we describe our approach for multicamera face tracking, let us briefly review the notation and the pose representation used in this paper. Fig. 1(a) illustrates the 6-DOF pose

of a face in the world coordinates. We use a common representation, a six-element vector $\tilde{\mathbf{p}}_w = [x, y, z, \theta, \phi, \psi]^T = [\check{\mathbf{x}}_w^T, \check{\boldsymbol{\theta}}_w^T]^T$ for the world-coordinate representation of the pose. The first three elements, denoted $\check{\mathbf{x}}_w = [x, y, z]^T$, represent the 3-D position of the object and the last three elements, denoted $\check{\boldsymbol{\theta}}_w = [\theta, \phi, \psi]^T$, the rotation of the object expressed in terms of roll, pitch, and yaw.

We also represent the 6-DOF pose in terms of the position and size of the face in the image plane and the rotation of the face with respect to a frontal face. We call this representation the image-based 6-DOF pose [Fig. 1(b)], defined as $\tilde{\mathbf{p}}_i = [u, v, s, \alpha, \beta, \gamma]^T = [\check{\mathbf{x}}_i^T, \check{\boldsymbol{\theta}}_i^T]^T$ where $\check{\mathbf{x}}_i = [u, v, s]^T$ represents the position of the face and $\check{\boldsymbol{\theta}}_i = [\alpha, \beta, \gamma]^T$ its rotation. The position $\check{\mathbf{x}}_i$ of the face is in the form of a bounding-box — (u, v) specify the center of the bounding box and s the scale. The rotation parameter vector $\check{\boldsymbol{\theta}}_i$ consists of roll (α), pitch (β), and yaw (γ) angles of the face,[1] which are measured with respect to a frontal face centered at (u, v); these are distinct from the rotation angles in both the world and camera coordinate systems [27], [44].

B. Face Pose Computation

To compute the image-based face poses, we search each frame for the occurrence of faces using standard face detection techniques (such as [6], [45]). The position and the size of the bounding box corresponding to each detected face is output by the face detector. Although our framework allows for the computation of the rotation parameter vector $\check{\boldsymbol{\theta}}_i$ using any face pose estimation technique (such as [10]), in this work we detect only frontal faces. For a survey of face pose estimation techniques, see [7].

We transform a 6-DOF pose observation from the image coordinates into the world coordinates as follows: The transformation is divided into two components, one for obtaining the world position vector $\check{\mathbf{x}}_w$ and the other for obtaining the world rotation vector $\check{\boldsymbol{\theta}}_w$

$$\tilde{\mathbf{p}}_w = \mathbf{f}(\tilde{\mathbf{p}}_i) = \begin{bmatrix} \check{\mathbf{x}}_w \\ \check{\boldsymbol{\theta}}_w \end{bmatrix} = \begin{bmatrix} \mathbf{f}_x(u, v, s) \\ \mathbf{f}_\theta(u, v, \alpha, \beta, \gamma) \end{bmatrix} = \begin{bmatrix} \mathbf{f}_x(\check{\mathbf{x}}_i) \\ \mathbf{f}_\theta(\check{\mathbf{x}}_i, \check{\boldsymbol{\theta}}_i) \end{bmatrix}. \quad (1)$$

[1]There are other possible representations for the rotation of an object in world or image-based coordinates. We discuss these in more detail in the conclusions (Section VI).

The world position vector is obtained by a method similar to the one reported in [28]. That is

$$\tilde{\mathbf{x}}_w = \mathbf{f}_x(\tilde{\mathbf{x}}_i) = {_w\mathbf{R}_c}\left(\frac{K_s}{s}\hat{\mathbf{d}}_c\right) + {_w\mathbf{t}_c} \qquad (2)$$

where $_w\mathbf{R}_c$ and $_w\mathbf{t}_c$ are, respectively, the rotation matrix and the translation vector from the camera coordinate system to the world coordinate system, $\hat{\mathbf{d}}_c = (u\hat{\mathbf{i}}_c + v\hat{\mathbf{j}}_c + \hat{\mathbf{k}}_c)/(\sqrt{u^2 + v^2 + 1})$ the unit vector pointing toward the face in the camera coordinate frame, and K_s a constant that relates the image-based face scale s to the distance of the face from the camera.

The rotation vector in the world space, $\tilde{\boldsymbol{\theta}}_w$, is obtained by

$$\tilde{\boldsymbol{\theta}}_w = \mathbf{f}_\theta(\tilde{\mathbf{x}}_i, \tilde{\boldsymbol{\theta}}_i) = \left[{_w\mathbf{R}_c}\, {_c\mathbf{R}_l}[\tilde{\boldsymbol{\theta}}_i]_{3\times3}\right]_{3\times1} \qquad (3)$$

where the rotation $_c\mathbf{R}_l$ compensates for perspective distortion. This rotation is based upon the unit vector $\hat{\mathbf{d}}_c$ defined previously and the unit vector pointing along the optic axis of the camera, $\hat{\mathbf{k}}_c$. It is defined as the rotation about the axis $\hat{\mathbf{k}}_c \times \hat{\mathbf{d}}_c$ by the angle $\cos^{-1}(\hat{\mathbf{k}}_c \cdot \hat{\mathbf{d}}_c)$ as in [27]. As described by the previously shown equation, we convert the three-element image-based rotation vector $\tilde{\boldsymbol{\theta}}_i$ into an equivalent 3×3 rotation matrix $[\tilde{\boldsymbol{\theta}}_i]_{3\times3}$ and multiply that by the appropriate rotation matrices to obtain a 3×3 rotation matrix in the world coordinate frame. We then transform the resulting rotation matrix back into an equivalent 3-element rotation vector $\tilde{\boldsymbol{\theta}}_w$, this operation being represented by the notation $[\cdot]_{3\times1}$.

C. Uncertainty Modeling

To take into account the uncertainty in the computation of the face pose, we represent each image-based observation as a 6-D Gaussian distribution $\mathbf{p}_i \sim \mathcal{N}(\overline{\mathbf{p}}_i, \Sigma_{\mathbf{p},i})$, where the mean $\overline{\mathbf{p}}_i$ is given by the actual image-based[2] face pose $\tilde{\mathbf{p}}_i$, and the covariance $\Sigma_{\mathbf{p},i}$ is a diagonal matrix whose elements are empirically chosen parameters.

To obtain the distribution of the observations in world coordinates $\mathbf{p}_w$, the distribution $\mathbf{p}_i$ of the image-based observation is propagated through the function $\mathbf{f}$ of (1) using the Unscented Transform [46]. This transform compensates for nonlinearities by transforming a small collection of deterministic *sigma points* through the function $\mathbf{f}$. The mean and the covariance of the transformed distribution are determined from the transformed sigma points. The transformed observation is then given by the 6-D Gaussian random variable $\mathbf{p}_w \sim \mathcal{N}(\overline{\mathbf{p}}_w, \Sigma_{\mathbf{p},w})$.

This is obviously an approximation, but a reasonable one. Sankaranarayanan and Chellapa, for example, have successfully employed a similar approach in [47]. In fact, they show that when a Gaussian random variable in $\mathbb{R}^2$ is mapped to a different plane through a homography, its distribution remains Gaussian as long as its mean is sufficiently far from the line at infinity. We believe that a similar argument can be made for the spatial component $\tilde{\mathbf{x}}_w = \mathbf{f}_x(\tilde{\mathbf{x}}_i)$ of the transformation. Even for more general transforms, such as the rotational component, $\tilde{\boldsymbol{\theta}}_w = \mathbf{f}_\theta(\tilde{\mathbf{x}}_i, \tilde{\boldsymbol{\theta}}_i)$, the Unscented Transform is generally acknowledged

[2]Recall that a subscript i denotes an image-based observation, and a subscript w a world observation.
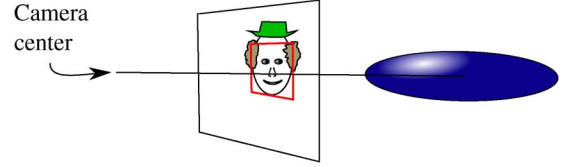

Fig. 2. We represent world observations as a 6-DOF Gaussian distributions. The spatial marginal of this distribution has a large amount of uncertainty along the backprojected ray which intersects the face center.

to accurately estimate the mean and covariance of the transformed distribution.

Note that the marginal distribution of the translational elements of the face pose $\mathbf{x}_w$ has an intuitive interpretation — its high-probability region forms an elongated ellipsoid which has the largest uncertainty along the backprojection ray (Fig. 2).

D. An Evidence Accumulation Framework for Multicamera Face Pose Estimation

As observations of a given face from multiple cameras become available, they are integrated into a single estimate of the position and the orientation of the face.

We use the Mahalanobis distance to compare the world-frame versions of the face pose as obtained from two different cameras

$$d\left(\mathbf{p}_w^j, \mathbf{p}_w^k\right) = \left(\overline{\mathbf{p}}_w^j - \overline{\mathbf{p}}_w^k\right)^T \left(\Sigma_{\mathbf{p},w}^j + \Sigma_{\mathbf{p},w}^k\right)^{-1} \left(\overline{\mathbf{p}}_w^j - \overline{\mathbf{p}}_w^k\right) \quad (4)$$

where the superscripts j and k denote the two observations being compared. To decide if two such observations are consistent (that is, they correspond to the same face in two different images), we compare the Mahalanobis distance to a matching threshold T. We declare two observations to be consistent if $d(\mathbf{p}_w^j, \mathbf{p}_w^k) < T$.

Based upon the matching distances given by $d(\mathbf{p}_w^j, \mathbf{p}_w^k)$, we determine a set $\mathcal{E}$ of estimates that correspond to a person's face. We employ two different approaches for computing these consistent sets, one for a distributed implementation of the final world pose estimation, and the other for a centralized implementation of the same. These will be discussed in Sections IV and V.

Once the observations of a person's face that are consistent in the sense described previously are collected in a set $\mathcal{E}$, we can construct from the set the following minimum variance estimate of the 6-DOF pose of the face in the world frame:

$$E[\hat{\mathbf{p}}_w] = (Cov[\hat{\mathbf{p}}_w]) \sum_{\mathbf{p}_w^k \in \mathcal{E}} \left(\Sigma_{\mathbf{p},w}^k\right)^{-1} \overline{\mathbf{p}}_w^k \qquad (5)$$

$$Cov[\hat{\mathbf{p}}_w] = \left(\sum_{\mathbf{p}_w^k \in \mathcal{E}} \left(\Sigma_{\mathbf{p},w}^k\right)^{-1}\right)^{-1}. \qquad (6)$$

Here, $\overline{\mathbf{p}}_w^k$ and $\Sigma_{\mathbf{p},w}^k$ are the mean and covariance associated with the world observation $\mathbf{p}_w^k$. One of the advantages of the uncertainty reduction formulas (5) and (6) is that they can be easily computed iteratively.

Once all of the observations are integrated into a single estimate, a Kalman filter can be used to track each person's face pose from frame to frame. This incorporates a motion model into
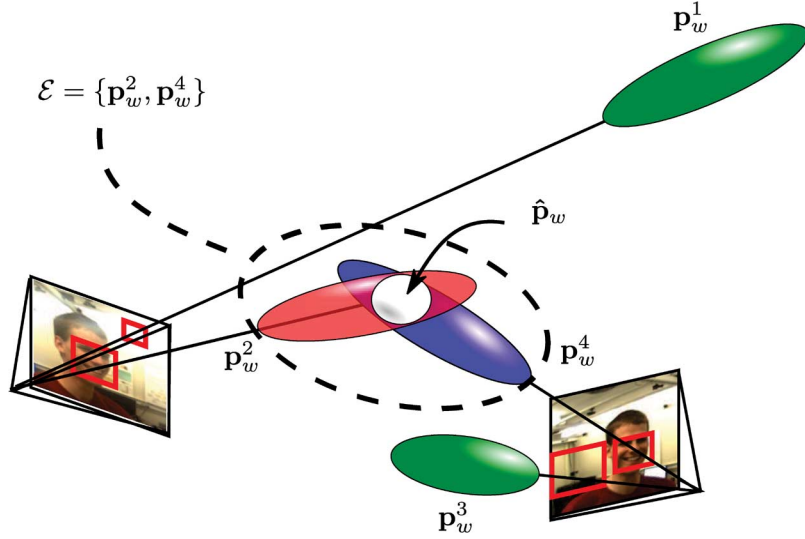
Fig. 3. Evidence accumulation framework. Consistent world observations are found using the Mahalanobis distance. Corresponding observations are integrated using a minimum variance estimator. The framework is naturally robust to incorrect detections.

the estimation process and provides a prediction of the face pose in one frame based upon the estimates in the previous frames.

Fig. 3 is a pictorial depiction of how the 3-D pose evidence gleaned from the individual cameras is integrated into a minimum variance estimate of the pose of the face in the world frame. The example shown involves two cameras that provide the observations $\mathbf{p}_w^2$ and $\mathbf{p}_w^4$ for the 6-DOF pose. These two observations are assumed to be consistent in the sense of satisfying the Mahalanobis distance requirement mentioned earlier. Both these observations would therefore be placed in the set $\mathcal{E}$ for evidence integration to yield the minimum variance estimate $\hat{\mathbf{p}}_w$. It is important to mention that this approach is naturally robust to false face detections, as illustrated by the observations $\mathbf{p}_w^1$ and $\mathbf{p}_w^3$ depicted in the figure. For each such presumably anomalous observation in one camera, the likelihood of there existing consistent observations in the other cameras would be low (except, perhaps, for accidental matchings for the case of images of crowded scenes when each image could include several faces).

III. EVENT-DRIVEN CLUSTERING IN WIRED CAMERA NETWORKS

In this section, we describe a wired camera network protocol that allows a group of cameras to collaboratively calculate the 6-DOF pose of a human face by dynamically forming a cluster of cameras and electing a leader of the cluster to serve as the coordinator for the collaborative effort. The clustering protocol was devised with the purpose of facilitating distributed tracking of objects, and it includes efficient propagation mechanisms that allow the computational load to be transferred to different camera nodes as the target object moves. In addition, the protocol is robust to the presence of errors in the visual features extracted from the images of the objects being tracked.

A. Cluster-Based Object Tracking With Wired Camera Networks

There are two commonly used graphs for representing a camera network: 1) A *communication graph* in which an edge between two camera nodes exists if they can directly communicate with each other; and 2) a *vision graph* in which an edge between two camera nodes exists if they have overlapping fields of view. In a wired camera network, since each camera can communicate with all the other cameras in the network, the communication graph is fully connected. In a wireless camera network, on the other hand, each camera can only communicate with cameras within its radio range; therefore, the communication graph for this case contains edges only between physically proximal camera nodes. Therefore, the idea of employing a clustering protocol designed for wireless camera networks in wired networks may seem rather counter-intuitive. Nonetheless, as we will show in this section, the communication graph of wireless camera networks and the vision graph of wired camera networks share many similarities. As a consequence, the clustering protocol for wireless cameras that we proposed in [43] can be applied to wired camera applications with only small changes.

In practice, for a camera network to effectively use its vision graph to carry out, say, object tracking, a set of cameras in some neighborhood of the network must all see the object so that each camera in the set can extract visual features from its image of the object and, perhaps, form hypotheses about the identity/pose of the object. All the cameras that can see an object at any given instant of time will constitute a graph which would be a subgraph of the vision graph. But that raises the question as to what we mean by a set of cameras seeing an object simultaneously. This question is answered by considering the set of cameras on a pairwise basis. *For any pair of cameras to see an object simultaneously means that the visual features extracted in the respective images match. That is, a similarity criterion used to compare such visual features passes a decision threshold.* A set of cameras that sees an object at the same time will be referred to as a *tracking graph*. Obviously, a tracking graph is a dynamic concept, in the sense that this graph will change from moment to moment, depending upon which cameras are best able to detect the object, extract its features, and then pass the feature comparison similarity tests. Fig. 4(a) illustrates one example of a
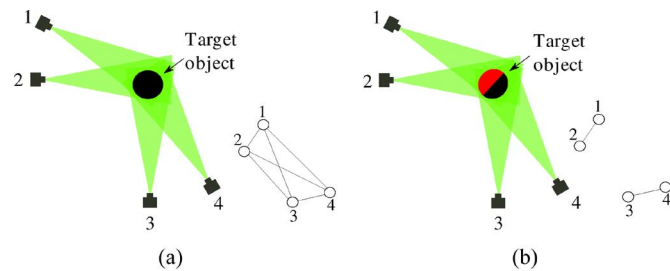
Fig. 4. (a) Multiple cameras detecting the same object and the corresponding tracking graph. (b) Multiple cameras detecting an object with distinct features and the corresponding tracking graph.

tracking graph. In the example, all the cameras can identify a target and, thus, may belong to a tracking graph. We place an edge between two nodes of a tracking graph if there exist sufficient similarities between the features extracted by the two cameras. (Of course, a camera may participate in multiple tracking graphs if it detects multiple objects.) The tracking graph shown in 4(a) is complete because we assume that all the cameras in this graph can extract similar features. Now consider the case when the object features measured in two cameras are distinct, as would be the case when the surface of an object is colored partly red and partly black. In this case, as illustrated in Fig. 4(b), although all the cameras can observe the same target, it is not possible to establish a correspondence between the observations at cameras 1 and 2, on the one hand, and at cameras 3 and 4, on the other. Consequently the tracking graph in this case will not be fully connected.

As we previously mentioned, the purpose of our protocol is to form clusters of cameras that observe targets with similar visual features. In the example shown in Fig. 4(a), the tracking graph is complete, meaning that all the cameras are able to identify the target with essentially the same set of features. In this case, all the cameras in the tracking graph form a cluster for the clustering protocol. Obviously, the same cannot be done in the example shown in Fig. 4(b). In this case, each clique in the tracking graph forms its own cluster. For the example shown, cameras 1 and 2 would form one cluster and cameras 3 and 4 another. In wireless camera networks, multiple clusters must be allowed to track the same target because of communication constraints. In the case of wired cameras, although the communication graph is complete, the tracking graph is not. Therefore, multiple clusters must also be allowed to track the same target, at least initially until more information about the target can be extracted at which time these multiple clusters can be merged into a single cluster.

After the clusters are created to track specific targets, they must be allowed to propagate through the network as the targets move. Cluster propagation refers to the process of 1) accepting new camera nodes into a cluster as they identify and recognize the same object, 2) removing the camera nodes that can no longer see the object, and 3) electing a new cluster leader as the current leader leaves the cluster. Adding a new camera node to a cluster and removing an existing camera node from a cluster are simple operations. However, when the cluster leader leaves a cluster, proper mechanisms must be provided to elect a new leader. In addition, since multiple clusters are allowed to track the same target, as these clusters collect further infor-

mation about the target, they may eventually be able to conclude that they are in fact tracking a common object. In wireless camera networks, as the clusters propagate, new cameras that join them may introduce previously nonexisting communication links between the clusters, thereby allowing them to coalesce [43]. In wired camera networks, the acquisition of additional information about the target may produce a similar result in the tracking graphs. In this case, what were previously two separate partitions of a tracking graph coalesce into a single clique, allowing what were two different clusters of camera nodes to operate as a single cluster.

As the reader would expect, dynamic cluster formation requires comparing the features extracted by the different camera nodes. Feature comparisons between different images of the same object can always be expected to be erroneous, not the least because of the differences in the images recorded from different viewpoints. We can expect difficulties even for the simplest of the objects — unless they look the same from all viewpoints. In addition to the difficulties that are inherent to feature comparisons, another source of difficulty in our case is that even when two of the camera nodes believe that they are looking at different objects (because of the differences in the feature values recorded), the observations made by a third camera may bear sufficient similarity to those in the first two cameras and this third camera may believe that all three cameras are looking at the same object. To illustrate this point, assume our object is very simple and consists of a multicolored ball, as shown in Fig. 5(a), and that the main feature extracted from the images is the average color value in the image. Obviously, whereas cameras 2 and 3 will not be able to establish a correspondence between their observations, camera 1 has enough information to know that the object it sees corresponds simultaneously to the observations of cameras 2 and 3. Similarly, cameras 1 and 4 will not be able to recognize that their observations correspond to the same object, but cameras 2 and 3 can create an effective visual connection between cameras 1 and 4 by partially matching the target's features. Under these conditions, ideally it should be possible to create a single cluster that consists of all the cameras that can detect the same target (cameras 1, 2, 3, and 4 in our example). However, this would also increase the chances of grouping together camera nodes that are actually seeing different objects, as illustrated in Fig. 5(b) and (c). Evidently, in the two examples presented in Fig. 5(b) and (c), one could use camera calibration information to figure out that the objects involved are different. This is not always possible, however, because of spatial uncertainties involved in computing the position of an object, especially along the camera axis.

To ensure that disparate objects seen by the different cameras in a vision graph do not result in the same tracking graph, we could also raise the bar on the decision thresholds used for cross-camera feature similarity comparisons. (Obviously, that would still not prevent the difficulties created by the case when different objects do look the same from different viewpoints.) However, if cross-camera feature similarity thresholds are set too high, that could impede the formation of a tracking graph with more than one camera node. We have therefore adopted a middle approach that consists of establishing a distinction between a tracking graph and a cluster. Only those camera nodes
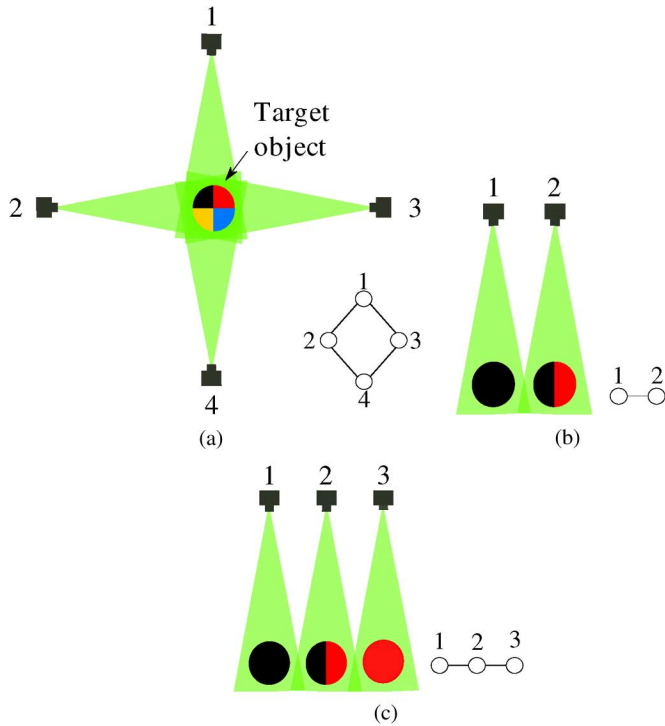
Fig. 5. (a) Multiple cameras detecting a common object by partial feature matching and the corresponding tracking graph. (b) Incorrectly matched objects and the corresponding tracking graph. (c) Sequence of incorrectly matched objects and the corresponding tracking graph.

Fig. 6. State transition diagram of a cluster-based object tracking system using a camera network.

that are one another's immediate neighbors in a tracking graph can form a cluster. That is, two camera nodes are allowed to join the same cluster if they both pass the cross-camera feature comparison similarity tests and they are each other's immediate neighbors in a tracking graph (and, by implication in the vision graph). Therefore, instead of creating a large cluster to track the same target, we prefer to tolerate the formation of multiple small clusters and to allow them to coalesce later as more information about the target is collected. Here, we can draw a parallel with the complex strategies required to construct and update large multihop clusters in wireless sensor networks [35]. Although it is possible to create such large clusters, the overhead involved in the process makes them undesirable for real-time, lightweight distributed applications. Similarly, although it may be possible to construct large clusters of wired cameras by employing more sophisticated multicamera object tracking algorithms, the required message and time complexity could be prohibitively large.

It is true that even the constraint that only immediate neighbors in a tracking graph be allowed to form a cluster may result in a cluster that tracks multiple objects thinking that they are the same. This is best exemplified by the simple situation illustrated in Fig. 5(c). Fortunately, unless all of the objects are moving together along the same trajectory (in which case one could argue that they be treated as a single extended object), such a cluster is likely to break into multiple clusters as a cluster leader departs because it can no longer see the object. For example, suppose camera 2 in Fig. 5(c) is elected a cluster leader, and cameras 1 and 3 incorrectly decide that they are detecting the same object as detected by camera 2. In that case, a single cluster would
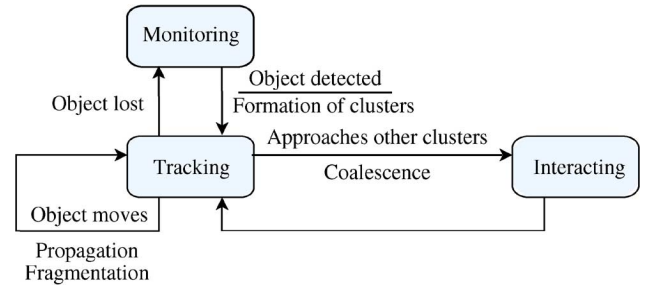
be created to track all three targets as if they corresponded to a single target. But if the object detected by camera 2 leaves the camera's field of view, the camera node will leave the cluster, and two new independent clusters will be formed by cameras 1 and 3 to detect the remaining (and now clearly distinct) objects. It is interesting to note that a similar situation arises in wireless camera networks [43]. In such networks, all the members of a single-hop cluster must be able to communicate with the cluster leader; however, the sensor nodes may not necessarily be able to communicate with one another directly. Therefore, when the cluster leader leaves the cluster, it may be necessary to create multiple new clusters.

Fig. 6 shows the state transition diagram of our cluster-based object tracking system using a camera network. Upon initialization, the network monitors the environment for any objects of interest. As objects are detected, for each object one or more clusters are formed to track it. These clusters propagate through the network to keep track of the objects in motion. Finally, if two or more clusters conclude that they are tracking the same object, they may coalesce into a larger cluster. Moreover, since the network is able to keep track of multiple objects simultaneously, each camera may belong to more than one cluster at the same time. In practice, that requires that each camera maintain a different state for each of the objects that it recognizes and that are currently being tracked by the network.

B. Clustering Protocol

In this section, we briefly describe our clustering protocol for camera networks. A more detailed description of the protocol in the context of wireless camera networks can be found in [43], [48]. The messages exchanged by the cameras for cluster formation and propagation activities include data packets that consist of visual features extracted and their corresponding values for each object detected in the scene. Obviously, when one camera node receives such a message from another camera node, the receiving node accepts the message only if it has itself detected an object with similar attributes and values. As the reader would expect, at network initialization time, the calibration information at each camera node is used to construct a vision graph that is stored at every node of the network in the form of an adjacency list. Each node uses the vision graph to filter out the messages from those camera nodes that are not directly connected to it in the vision graph.

1) Cluster Leader Election: We employ a two-phase cluster leader election algorithm. In the first phase, the nodes in the

same tracking graph compete to become the leader of the cluster. (One possible criterion for this competition is given in Section IV-B.) After the first phase, at most one camera node elects itself as the cluster leader among its direct neighbors in a tracking graph, and the rest join the cluster. During the second phase, the cameras that were left without a cluster leader (because their cluster leader candidate joined another cluster) elect the next best candidate as the cluster leader.

2) Cluster Propagation: Inclusion of new members into active clusters takes place as follows: When a camera detects a new object, 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, the cluster leader replies with a message requesting that the camera 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.

Removal of cluster members is trivial. When an object leaves the field of view of a cluster member, all the member has to do is send a message informing the cluster leader that it is leaving the cluster. The cluster leader then updates its list of cluster members. If the cluster member is tracking multiple objects at the moment, it terminates only the connection related to the object that left its field of view.

Cluster propagation also involves leader reselection and cluster coalescence, which we explain in the following.

   a) *Cluster Leader Reselection*: Assuming that the cluster leader has access to the latest information about the position of the target with respect to each cluster member, it is able to keep an updated list of the best cluster leader candidates. When the cluster leader decides to leave the cluster, it sends a message to the remaining camera nodes containing a sorted list of the best cluster leader candidates. (This message also includes any additional state information for the cluster head, such as the state of the Kalman filter we will discuss in Section IV-C). A new cluster leader is then selected following the second phase of the regular cluster leader election mechanism. This approach not only allows for cluster fragmentation, but also for seamless cluster coalescence.

   b) *Cluster Coalescence*: Consider two clusters, A and B, that are propagating toward each other. As explained previously, cluster propagation entails establishing a new cluster leader as the previous leader loses sight of the object. Now consider the situation when a camera is designated to become the new leader of cluster A and is tracking the same object as cluster B. Under this circumstance, the camera node that was meant to be A's new leader is forced to join cluster B. As the members of cluster A overhear their prospective cluster leader joining cluster B, they also join B.

   c) *Cluster Maintenance*: Additional robustness to failures is achieved by a periodic refresh of the cluster status. Since the protocol is designed to enable clusters to carry out collaborative processing, it is reasonable to assume that cluster members and cluster leaders exchange messages periodically. Therefore, we can use a soft-state based approach [49] to keep track of cluster membership.
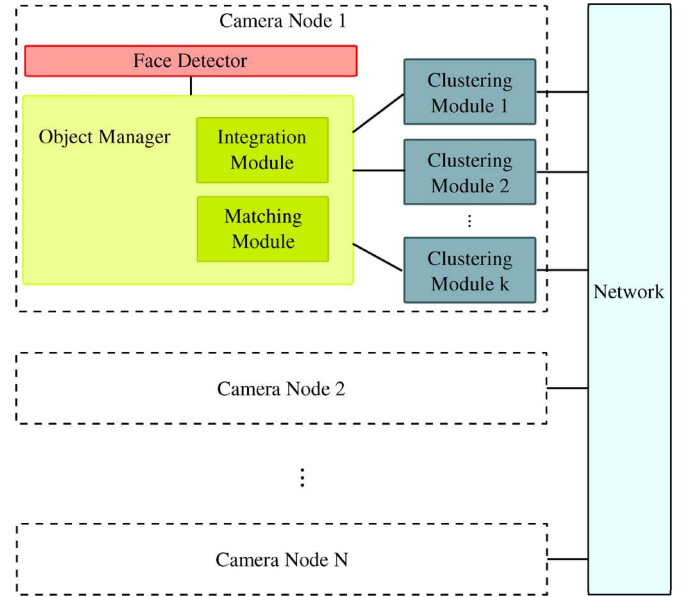

Fig. 7. Block diagram of our distributed face tracker.

IV. CLUSTER-BASED DISTRIBUTED FACE TRACKING

In this section, we present the architecture of our cluster-based distributed face tracking system. Fig. 7 shows a block diagram of the system. At each camera node, as a new image frame becomes available, a face detector module detects all the faces present in the frame and computes their corresponding world poses. This information is then delivered to the object manager, which is responsible for checking whether the faces detected in this frame correspond to any of the existing faces currently being tracked or to a new detection. This is done by the matching module that compares the identities of the faces detected in this frame to the identities of all the existing faces currently being tracked. (We will describe the features used to establish identity in the next subsection.) If a face detected in this frame corresponds to a new detection, the object manager instantiates a new clustering module with the responsibility to keep track of this new face. In the example shown in Fig. 7, k faces are being tracked by camera node 1. When a clustering module is instantiated, it starts a new cluster leader election process. If, on the other hand, a face detected in this frame is identified as one of the existing faces currently being tracked, all that the object manager has to do is to transfer the face pose observation to the corresponding clustering module.

When a clustering module starts a cluster leader election, it broadcasts this intent to its vision neighbors. When another clustering module receives this message, it checks whether the face pose contained in the message corresponds to the face for which it is responsible. Again this is done by the matching module within the object manager. After the cluster leader election finishes, the clustering module reports to the object manager its current cluster status, i.e., whether it became a cluster leader or a cluster member. The object manager can then use this information to decide how to process new observations of the specific face as follows.

For each face detected in the current frame, if the face is associated with an existing cluster and the camera node is currently

a member of that cluster, the object manager requests that the clustering module send this observation to its cluster leader.[3] If a face detected in the current frame is associated with an existing cluster and the camera node is the leader of that cluster, the object manager simply updates the face pose estimate using this new observation. The same happens if the clustering module receives an observation from one of its cluster members. In both cases, the face pose is updated by the integration module using (5) and (6).

If the object manager at a given camera node does not detect a particular individual's face for several frames, it terminates the corresponding clustering module. If the camera node is currently the leader of that cluster, its termination triggers cluster propagation so that a new cluster leader can be assigned to keep track of the face. On the other hand, if the camera node is only a cluster member, its termination simply triggers a report to the cluster leader that it is leaving the cluster.

A. Use of Position as a Face Identification Feature

The clustering protocol as described in Section III distinguishes between the *features* used to identify targets and the *estimates* of the target position. However, since robust face recognition methods that work in real time under realistic conditions are not yet available [50], at this time we use the pose of a face as the feature that defines its identity. In [51], the authors also use a spatial feature (object motion) for track identification. One of the challenges in using the pose for face identification is that, as the face moves, the face identifier also changes. Nonetheless, as long as the people being tracked do not move too abruptly and as long as the pose estimates are kept up-to-date by the object manager on a frame by frame basis whenever their faces are detected in the camera images, we can expect our approach to face identification to work without difficulties.

B. Cluster Leader Election Criterion

Cluster leader election requires that we define a criterion that must be satisfied by a camera node if it is to become a cluster leader. The criterion that we use currently is based upon the distance between the location of the face as given by the projection of x_w into the camera image plane and the camera center of the camera in question. The main advantages of this criterion are that it generally creates relatively long-lived cluster leaders and the distance can be computed easily in each camera.

C. System Synchronization

It is well known that synchronous distributed systems are in general less complex and, for some network types, more efficient than asynchronous systems [52]. Since synchronizing the time clocks of camera nodes interconnected by a local area network is relatively simple, we chose to employ a synchronous approach for our distributed camera network system. In our system, camera clocks are synchronized using the network time protocol (NTP). Moreover, we assume that all the cameras in the network have a consistent frame capturing rate. However, even when that is true, it is not simple to synchronize the image capture times without using special hardware. That is, it is difficult

---

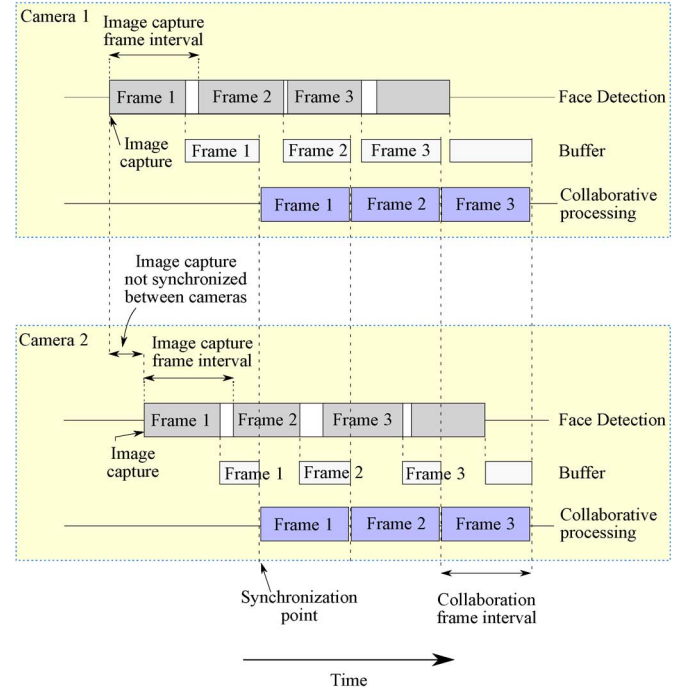[3]All internode communication takes place through the clustering modules at each node.

Fig. 8. Buffer mechanism for camera synchronization. Camera capture and face detection are executed asynchronously. The results are stored in a buffer which is processed synchronously during the collaboration frame intervals.
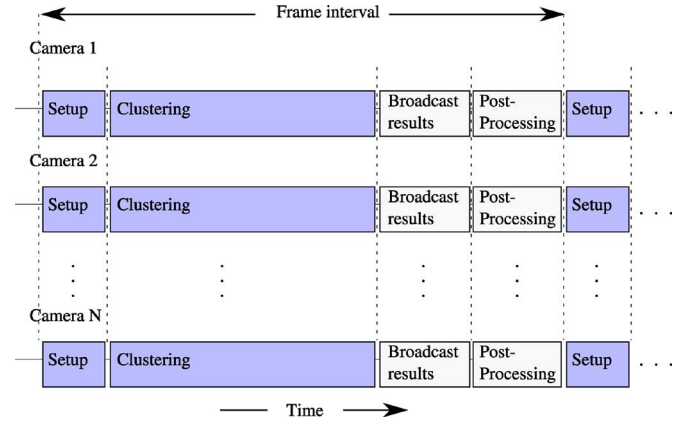

Fig. 9. Collaboration interval steps. After a short setup period, all the cameras share information during the clustering step. Afterwards, cluster leaders broadcast the results to the cluster members, and postprocessing algorithms may take place.

to guarantee that the time instant at which the different cameras capture their frames will be exactly the same for all cameras. To overcome this problem, we designed a buffering mechanism that allows the cameras to store the current detections until they reach a synchronization point. The buffering mechanism is illustrated in Fig. 8. As the cameras acquire the initial frame, they store the information in the buffer until they reach a synchronization point. At this point, collaborative processing is initiated and the cameras are allowed to share information. In subsequent frames, new synchronized collaboration frame intervals occur at a predefined rate that is common to all cameras. This works as long as face detection can be carried out in less than one frame interval. However, since face detection time is variable, it may occasionally take longer than one frame interval. In that case,
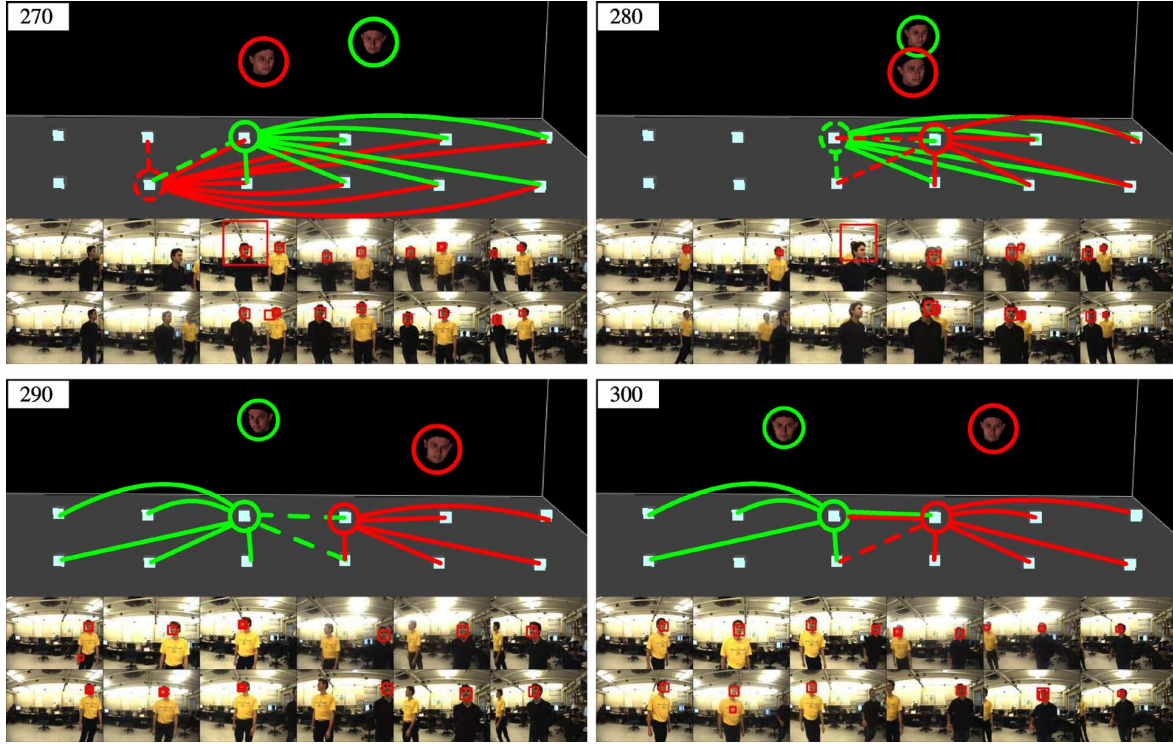
Fig. 10. Snapshot of our tracking results. Top figures: graphical representation of the camera array and estimated poses. Bottom figures: images captured by each camera and the detected faces. The two frames at the top illustrate the propagation of a cluster leader.

the detections corresponding to that frame are discarded and a new frame is processed without detriment to the collaborative processing synchronization.

Fig. 9 shows in detail the sequence of events that takes place during the collaboration intervals. During the brief setup step, all new clustering modules are initialized and prepared to receive messages from other cameras. This step provides tolerance for small errors in the synchronization of the cameras. Although not strictly necessary, this step greatly improves system efficiency by not having to buffer messages received by uninitialized clustering modules. At the beginning of the clustering step, the clustering protocol is executed and clusters are created, dissolved, or propagated as necessary. After the states of all the clusters are updated, the cluster members send their observations to the cluster leaders, which then integrate them into face pose estimates. After the clustering step, the cluster leaders update the estimate and prediction of the location of each face using a Kalman filter. The cluster leaders also broadcast the predicted face poses to the cluster members for association with the face detections in the next frame. To avoid the cluster switching from one person to another, if the cluster leader detects that the new face pose estimate is far from the Kalman filter's prediction, it terminates the cluster by sending a message to its members and deactivating itself. Finally, after the required processing is concluded, our system provides a time slot for the application of postprocessing algorithms. In our current system, we use this slot to log the results of our experiments.

V. EXPERIMENTS

We implemented our algorithm on a network of twelve firewire cameras connected to three quad-core desktop com-

puters. The cameras are arranged side-by-side in the form of a 2×6 array all facing approximately the same direction. Each camera has a separate process assigned to it. In each process, we manage the face position estimates and camera clustering assigned to that camera, and detect faces with a boosted classifier cascade [6], [45] trained using the FERET database [53]. Fig. 10 shows four snapshots of the graphical user interface of our system. The top half of each snapshot shows a computer graphics representation of the 3-D positions of the camera array, the clusters tracking the faces, and the estimated poses of the detected faces. Each camera is represented by a small cube and the face poses are indicated by a 3-D face model [54]. The circles represent cluster leaders and the lines represent the members. A dashed circle or line indicates that the cluster leader or member did not contribute an observation for that particular frame. The bottom part of each snapshot shows the images captured by the cameras and the corresponding face detections as computed by the individual cameras.

Fig. 11 shows qualitative results of one run of our experiments for one of the two people in the same sequence shown in Fig. 10. The left column shows the 3-D positions of the face, and the right column shows its orientation. In this figure, ground truth is represented by solid lines and the markers with different colors represent the estimated face poses. The reason for using different colors on the trajectories is to illustrate the moments when the system loses track of a person's face — each time the track is lost, a different color is used. Notice that when a cluster loses track, another cluster almost immediately is created and starts to track the face again.

In the experiments we present here, we require that all the cameras in a cluster detect a frontal face. A face can be accu-
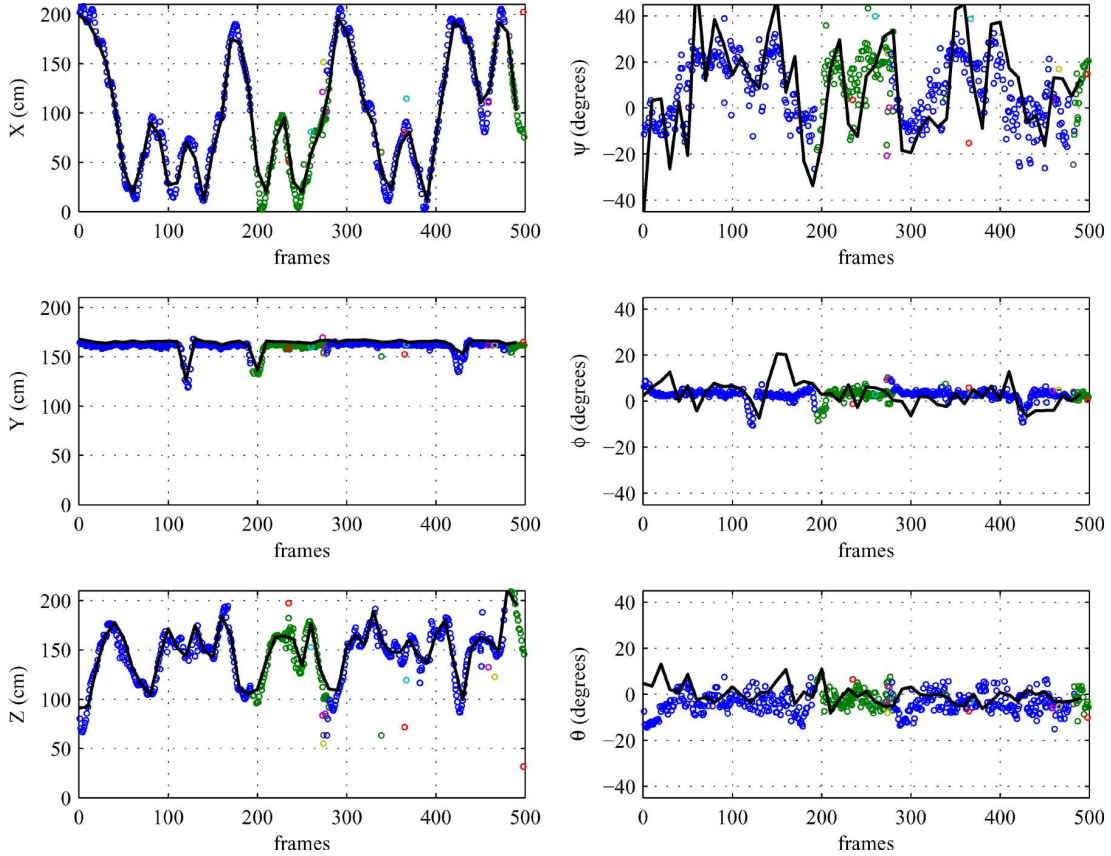
Fig. 11.   Qualitative results of one experimental run of the distributed face tracker on one of two faces (the second face is omitted for clarity).

rately tracked if it is detected by at least two cameras. When the individual cameras have low detection rates, additional camera views can increase the likelihood of detecting the face. If non-frontal face poses can be localized in the individual images, fewer cameras will be needed by our protocol. Even when detecting only frontal faces, many camera configurations can be used with our current protocol. Because the clustering protocol allows for the propagation of the cluster, new cameras in different orientations can pick up the tracking of the face as the face rotates away from the old cameras.

A. Comparison With a Centralized Method

To provide a quantitative evaluation of our distributed approach, we compare it to a centralized method which operates in a similar framework but does not include the clustering protocol.

The centralized version has many of the features of the distributed approach. As in the distributed approach, we perform face detection locally in each camera. Thus, face detection — which takes most of the processing time in our experiments — is still distributed in the centralized approach. The timing of both approaches is also similar. In both systems, we synchronize the processing of images as illustrated in Fig. 8. Despite these similarities, there are fundamental differences between the centralized and distributed versions. In the centralized version, no leader is elected. Instead, collaborative processing takes place in two steps. In the first step, every camera sends the observations to a single node for central processing. In the second step, the

central node processes all the observations it received for that frame in batch.

Because all of the observations are available in batch, we partition them into sets $\mathcal{E}_l$, $l = 1, \ldots, L$, where L is the number of people in the environment, using an approximate clique clustering algorithm [55]. The sets are chosen to approximately minimize the sum of intraset costs

$$\arg\min_{L, \mathcal{E}_l} \sum_{l=1}^{L} \left(\sum_{\mathbf{p}_w^j \in \mathcal{E}_l, \mathbf{p}_w^k \in \mathcal{E}_l, \mathbf{p}_w^j \neq \mathbf{p}_w^k} cost\left(\mathbf{p}_w^j, \mathbf{p}_w^k\right) \right). \quad (7)$$

This algorithm requires both positive and negative costs to produce nontrivial clusterings, so we use the cost function $cost(\mathbf{p}_w^j, \mathbf{p}_w^k) = d(\mathbf{p}_w^j, \mathbf{p}_w^k) - T_{clique}$, where T_{clique} determines the zero-cost distance, and $d(\cdot)$ is given in (4).

Fig. 12 shows a comparison between the centralized tracker and our distributed approach. The tracks shown in the figure correspond to the x coordinates of the two individuals' faces for the same sequence shown in Fig. 10. As in Fig. 11, different colors illustrate the moments when the system loses track of a face. Since neither approach represents an ideal tracker, lost tracks occur in both. We record two kinds of tracking errors for our system. If a track is lost, and a new one is created to track a person, we call this an "extra track." If a tracker switches from one person to another, we call this a "track switch." Track switches are worse than extra tracks because they indicate that a single tracker tracks two people. They are also much more rare in both the centralized and distributed systems. In ten runs of the
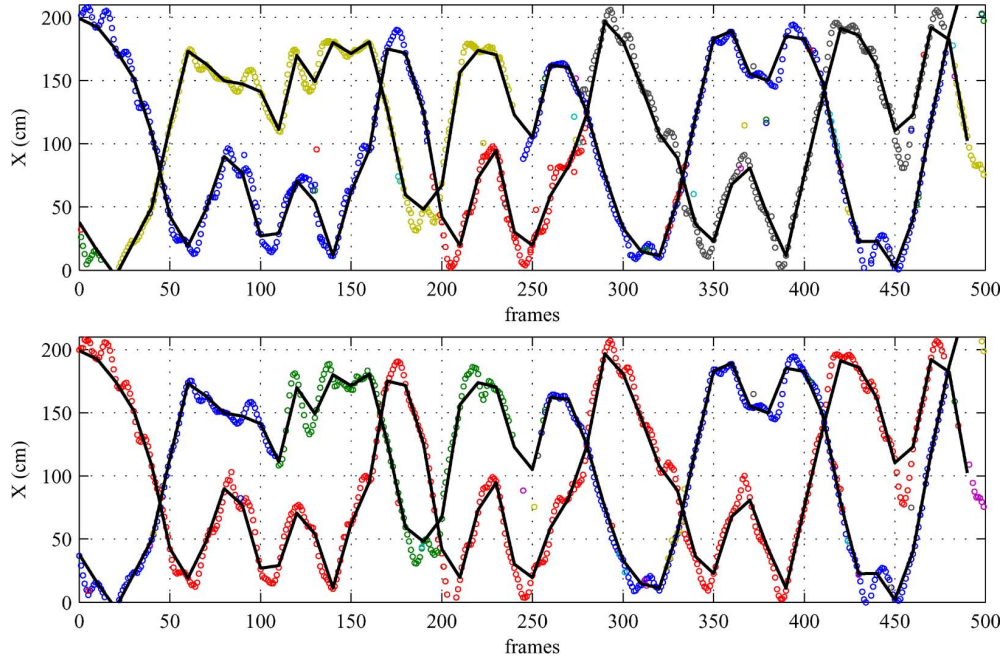
Fig. 12. Comparison of tracking in the distributed (top) and centralized (bottom) versions. Marker color indicates the ID assigned to each track. Both the centralized and the distributed approaches are approximate and track ID changes are visible in both approaches. The ground truth for each track is indicated by a black line which is sampled every 10th frame.

TABLE I
TRACKING PERFORMANCE ON THE TWO-PERSON SEQUENCE. TESTING IS
DONE ON 50 FRAMES, SPACED TEN FRAMES APART. THESE RESULTS ARE
AVERAGED OVER FIVE RUNS FOR EACH SYSTEM, EACH USING
THE SAME MULTICAMERA VIDEO SEQUENCE

	TP^a	FP^a	$rmse_{\mathbf{T}}(cm)$	$rmse_{\mathbf{R}}(°)$
Centralized	95 (95%)	12 (12%)	5.8	20.8
Distributed	94 (94%)	4 (4%)	6.1	18.7

^{a}percentages are per frame, per person

system, the centralized system required an average of 3.6 ± 0.2 extra tracks per person and 0 track-switches, and the distributed system required 2.2 ± 0.7 extra tracks and 0.1 track switches.

We also compare the centralized and distributed approaches based upon the frame-by-frame tracking performance, as shown in Table I. In each frame, we associate each ground-truth to a single integrated face estimate, if there is an estimate within a specific matching distance. True-positives (TP) represent ground-truth and integrated estimate pairs, while false-positives (FP) represent estimates which do not correspond to any ground-truth. Pairs are assigned starting with the closest ground-truth and estimates, so that it is possible for an estimate to be within the matching distance and still be considered an FP. The $RMSE$ estimates are based upon the TP pairs. These results show that the distributed approach achieves comparable performance to the centralized version.

VI. CONCLUSION

We have presented a completely distributed face tracking algorithm that estimates the 6-DOF poses of multiple faces in real time. Each camera individually computes the world pose of the faces based upon their visual features. The observations of multiple cameras are integrated using a minimum variance estimator and tracked using a Kalman filter. A clustering protocol is responsible for dynamically creating groups of cameras that track a given face and for coordinating the distributed processing.

As our experimental results show, our algorithm performs as well as a centralized approach while presenting the well-known advantages of distributed systems: scalability and robustness. Since the computational load is dynamically transferred among processors as the people move in the field of view of the camera network, our algorithm can potentially handle an arbitrary number of faces and can be scaled to much larger networks. Also, since cluster leaders are dynamically elected and the clustering protocol is robust to system failures, the algorithm does not rely on a single server to process the information, therefore avoiding a single point of failure.

One limitation of our current approach is the representation of rotations using yaw, pitch, and roll angles, which, as any representation of rotations in $\mathbb{R}^3$, have discontinuities that must be handled as special cases. We currently restrict ourselves to representations in $\mathbb{R}^3$ to allow the use of standard techniques for transforming distributions between Euclidean spaces. In the future, we would like to extend our method to use quaternions, but this requires more sophisticated techniques to transform rotation estimates from a 3-D Euclidean space to a 3-D manifold in a 4-D space.

Another aspect that requires consideration is that the minimum variance estimator — (5) and (6) — is based upon the assumption that the camera observations are independent. Although this is generally a good approximation, when we transform the observations from the image space to the world space, we incorporate prior knowledge about the size of a person's face. This introduces a bias in the world space which manifests

itself as observations which are consistently closer to the camera which detects them, or farther away, depending upon whether the person's face is larger or smaller than expected. Although we believe that most of this bias is removed in the integration of the observations from multiple cameras — and our experimental results support that claim — in the future we would like to carefully investigate the effects of this bias in the estimation of the positions of the faces.

One advantage of our approach is that while it does not require image-based tracking methods such as particle filters or mean-shift [56]–[58], it does provides a good framework for incorporating such trackers which operate independently on different camera images. These trackers would provide additional observations which could be used to keep track of the faces when the interval between face detections is large. When new face detections become available, they could be used to reduce the chance that the trackers drift off of the true face location.

ACKNOWLEDGMENT

REFERENCES

[1] R. Stiefelhagen, M. Finke, J. Yang, and A. Waibel, "From gaze to focus of attention," in *Proc. Int. Conf. Visual Information and Information Systems*, 1999, pp. 761–768.

[2] K. Smith, S. Ba, D. Gatica-Perez, and J.-M. Odobez, "Tracking the visual focus of attention for a varying number of wandering people," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1212–1229, Jul. 2008.

[3] H. Rowley, S. Baluja, and T. Kanade, "Neural network based face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 23–38, Jan. 1998.

[4] H. Schneiderman and T. Kanade, "A statistical method for 3-D object detection applied to faces and cars," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000, vol. I, pp. 746–751.

[5] C. Huang, H. Ai, Y. Li, and S. Lao, "High-performance rotation invariant multiview face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 671–686, Apr. 2007.

[6] P. Viola and M. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[7] E. Murphy-Chutorian and M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 4, pp. 607–626, Apr. 2009.

[8] L. Brown, Y. Tian, I. Center, and N. Hawthorne, "Comparative study of coarse head pose estimation," in *Proc. Workshop on Motion and Video Computing*, 2002, pp. 125–130.

[9] L. Zhao, G. Pingali, and I. Carlbom, "Real-time head orientation estimation using neural networks," in *Proc. Int. Conf. Image Processing*, 2002, pp. I-297–I-300.

[10] E. Murphy-Chutorian, A. Doshi, and M. Trivedi, "Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation," in *Proc. IEEE Intelligent Transportation Systems Conf.*, Sep. 2007, pp. 709–714.

[11] B. Raytchev, I. Yoda, and K. Sakaue, "Head pose estimation by nonlinear manifold learning," in *Proc. Int. Conf. Pattern Recognition*, 2004, pp. 462–466.

[12] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, p. 681, Jun. 2001.

[13] T. Maurer and C. von der Malsburg, "Tracking and learning graphs and pose on image sequences of faces," in *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, 1996, pp. 176–181.

[14] T. S. Jebara and A. Pentland, "Parametrized structure from motion for 3-D adaptive feedback tracking of faces," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997, pp. 144–150.

[15] A. Schödl, A. Haro, and I. Essa, "Head tracking using a textured polygonal model," in *Proc. Workshop on Perceptual User Interfaces*, 1998, pp. 43–48.

[16] J. Heinzmann and A. Zelinsky, "3-D facial pose and gaze point estimation using a robust real-time tracking paradigm," in *Proc. Int. Conf. Automatic Face and Gesture Recognition*, 1998, pp. 142–147.

[17] D. Decarlo and D. Metaxas, "Optical flow constraints on deformable models with applications to face tracking," *Int. J. Comput. Vis.*, vol. 38, no. 2, pp. 99–127, 2000.

[18] M. La Cascia, S. Sclaroff, and V. Athitsos, "Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3-D models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 4, pp. 322–336, Apr. 2000.

[19] J. Tu, T. Huang, and H. Tao, "Accurate head pose tracking in low resolution video," in *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, 2006, pp. 573–578.

[20] Z. Zhang, A. Scanlon, W. Yin, L. Yu, and P. L. Venetianer, "Video surveillance using a multi-camera tracking," in *Multi-Camera Networks: Principles and Applications*, H. Aghajan and A. Cavallaro, Eds. New York: Elsevier, 2009, ch. 18, pp. 435–456.

[21] S. Calderara, R. Cucchiara, R. Vezzani, and A. Prati, "Statistical pattern recognition for multi-camera detection, tracking, and trajectory analysis," in *Multi-Camera Networks: Principles and Applications*, H. Aghajan and A. Cavallaro, Eds. New York: Elsevier, 2009, ch. 16, pp. 389–414.

[22] J. Wu and M. Trivedi, "A two-stage head pose estimation framework and evaluation," *Pattern Recognit.*, vol. 41, no. 3, pp. 1138–1158, 2008.

[23] L. Morency, A. Rahimi, N. Checka, and T. Darrell, "Fast stereo-based head tracking for interactive environments," in *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, 2002, pp. 390–395.

[24] S. Koterba, S. Baker, I. Matthews, C. Hu, J. Xiao, J. Cohn, and T. Kanade, "Multi-view AAM fitting and camera calibration," in *Proc. IEEE Int. Conf. Computer Vision*, 2005, vol. I, pp. 511–518.

[25] M. Yasumoto, H. Hongo, H. Watanabe, and K. Yamamoto, "Face direction estimation using multiple cameras for human computer interaction," in *Proc. Int. Conf. Advances in Multimodal Interfaces—ICMI*, 2000, pp. 222–229.

[26] C. Chang, C. Wu, and H. Aghajan, "Pose and gaze estimation in multicamera networks for non-restrictive HCI," in *Proc. IEEE Int. Workshop on HCI*, 2007, vol. 4796, p. 128.

[27] E. Murphy-Chutorian and M. Trivedi, "3-D tracking and dynamic analysis of human head movements and attentional targets," in *Proc. Int. Conf. Distributed Smart Cameras*, 2008, pp. 1–8.

[28] H. Iwaki, G. Srivastava, A. Kosaka, J. Park, and A. Kak, "A novel evidence accumulation framework for robust multi-camera person detection," in *Proc. ACM/IEEE Int. Conf. Distributed Smart Cameras*, Sep. 2008, pp. 1–10.

[29] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[30] S. Bandyopadhyay and E. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *Proc. IEEE INFOCOM*, 2003, vol. 3, pp. 1713–1723.

[31] O. Younis and S. Fahmy, "HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Trans. Mobile Comput.*, vol. 3, no. 4, pp. 366–379, Oct.–Dec. 2004.

[32] S. Soro and W. Heinzelman, "Prolonging the lifetime of wireless sensor networks via unequal clustering," in *Proc. IEEE Int. Parallel and Distributed Processing Symposium*, Apr. 2005.

[33] C. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao, "Distributed group management for track initiation and maintenance in target localization applications," in *Proc. IEEE Int. Workshop on Information Processing in Sensor Networks*, Apr. 2003.

[34] W.-P. Chen, J. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 3, no. 3, pp. 258–271, Jul. 2004.

[35] W. Zhang and G. Cao, "DCTC: Dynamic convoy tree-based collaboration for target tracking in sensor networks," *IEEE Trans. Wireless Commun.*, vol. 3, no. 5, pp. 1689–1701, Sep. 2004.

[36] Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang, "Location tracking in a wireless sensor network by mobile agents and its data fusion strategies," *Comput. J.*, vol. 47, no. 4, pp. 448–460, Apr. 2004.

[37] X.-H. Kuang, R. Feng, and H.-H. Shao, "A lightweight target-tracking scheme using wireless sensor network," *Measurement Sci. Technol.*, vol. 19, no. 2, p. 025104, 2008.

[38] Q. Fang, F. Zhao, and L. Guibas, "Lightweight sensing and communication protocols for target enumeration and aggregation," in *Proc. ACM Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003, pp. 165–176.

[39] F. Bouhafs, M. Merabti, and H. Mokhtar, "Mobile event monitoring protocol for wireless sensor networks," in *Proc. Int. Conf. Advanced Information Networking and Applications Workshops*, 2007, vol. 1, pp. 864–869.

[40] W. Yang, Z. Fu, J.-H. Kim, and M.-S. Park, "An adaptive dynamic cluster-based protocol for target tracking in wireless sensor networks," in *Proc. Joint 9th Asia-Pacific Web Conf. Advances in Data and Web Management, and 8th International Conf. Web-Age Information Managemen*, 2007, pp. 157–167.

[41] S. Soro and W. Heinzelman, "On the coverage problem in video-based wireless sensor networks," in *Proc. Int. Conf. Broadband Networks*, Oct. 2005, vol. 2, pp. 932–939.

[42] S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Adv. Multimedia*, vol. 2009, pp. 1–21, 2009.

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

[44] Y. Tian, L. Brown, J. Connell, S. Pankanti, A. Hampapur, A. Senior, and R. Bolle, "Absolute head pose estimation from overhead wide-angle cameras," in *Proc. IEEE Int. Workshop on Analysis and Modeling of Faces and Gestures*, 2003, pp. 92–99.

[45] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proc. IEEE Int. Conf. Image Processing*, Sep. 2002, vol. 1, no. 1, pp. 900–903.

[46] S. Julier and J. Uhlmann, "A new extension of the Kalman filter to non-linear systems," in *Proc. Int. Symp. Aerospace/Defense Sensing, Simulation and Controls*, 1997, vol. 3068, pp. 182–193.

[47] A. Sankaranarayanan and R. Chellappa, "Optimal multi-view fusion of object locations," in *Proc. IEEE Workshop on Motion and Video Computing*, Jan. 2008, pp. 1–8.

[48] H. Medeiros, J. Park, and A. Kak, "A light-weight event-driven protocol for sensor clustering in wireless camera networks," in *Proc. ACM/IEEE Int. Conf. Distributed Smart Cameras*, Sep. 2007, vol. 2, no. 4, pp. 203–210.

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

[50] W. Zhao, R. Chellappa, P. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 399–458, 2003.

[51] Y. Sheikh, O. Javed, and M. Shah, "Object association across multiple cameras," in *Multi-Camera Networks: Principles and Applications*, H. Aghajan and A. Cavallaro, Eds. New York: Elsevier, 2009, ch. 17, pp. 415–434.

[52] N. Lynch, *Distributed Algorithms*. San Mateo, CA: Morgan Kaufmann, 1997.

[53] P. Phillips, H. Wechsler, J. Huang, and P. Rauss, "The FERET database and evaluation procedure for face-recognition algorithms," *Image Vis. Comput.*, vol. 16, no. 5, pp. 295–306, 1998.

[54] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3-D faces," in *Proc. ACM SIGGRAPH Conf. Computer Graphics*, 1999, pp. 187–194.

[55] V. Ferrari, T. Tuytelaars, and L. Van Gool, "Real-time affine region tracking and coplanar grouping," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001, vol. 2, pp. 226–233.

[56] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.

[57] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Proc. Eur. Conf. Computer Vision*, 2002, vol. 1, pp. 661–675.

[58] K. Nummiaro, E. Koller-Meier, and L. V. Gool, "An adaptive color-based particle filter," *Image Vis. Comput.*, vol. 21, no. 1, pp. 99–110, Jan. 2003.

Josiah Yoder received the B.S. CompE degree from Rose-Hulman Institute of Technology, Terre Haute, IN, in 2005 and is currently pursuing the Ph.D. degree in the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN.

His research interests include computer vision, machine learning, human-computer interaction, and distributed computing.

Henry Medeiros received the B.E. and M.S. degrees in electrical engineering from the Federal University of Technology, Parana, Brazil, in 2003 and 2005, respectively, and is currently pursuing the Ph.D. degree in the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN.

His current research interests include sensor networks, computer vision, and embedded systems.

Johnny Park received the B.S., M.S., and Ph.D. degrees from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, in 1998, 2000, and 2004, respectively.

He was a Principal Research Scientist at Purdue University where he led a large research project on distributed camera networks. He is currently a Research Assistant Professor in the School of Electrical and Computer Engineering at Purdue University. His research interests span various topics in distributed sensor networks, computer graphics, computer vision, and robotics.

Avinash C. Kak is a Professor of electrical and computer engineering at Purdue University, West Lafayette, IN. 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* (Wiley, 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 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 2010, will be titled *Designing with Objects*.