# Look there! Predicting where to look for motion in an active camera network

Ugur Murat Erdem

Computer Science Department
Boston University
Boston, MA 02215

Stan Sclaroff

Computer Science Department
Boston University
Boston, MA 02215

## Abstract

*A framework is proposed that answers the following question: if a moving object is observed by one camera in a pan-tilt-zoom (PTZ) camera network, what other camera(s) might be foveated on that object within a predefined time window, and what would be the corresponding PTZ parameter settings? No calibration is assumed, and there are no restrictions on camera placement or initial parameter settings. The framework accrues a predictive model over time. To start out, the cameras follow randomized "tours" in discretized PTZ space. If a moving object is detected in the field of view of more than one camera at a particular instant or within a predefined time window, then the model is updated to record the cameras' associations and the corresponding parameter settings. As more and more moving objects are observed, the model adapts and the most frequent associations are discovered. The formulation also allows for verification of its predictions, and reinforces its correct predictions. The system is demonstrated in observing people in an office environment with a three PTZ camera network.*

## 1. Introduction

In a typical surveillance application involving a camera network an important task that researchers face is selecting cameras and their parameter settings to accomplish a specific set of objectives. Given that a person is seen from some camera, three important questions effecting the choice of camera parameters become,

- *Which* camera(s) to foveate?
- *Where* to foveate?
- *When* to foveate?

to see the person in the near future. The overall success and complexity of the system in these cases is tightly coupled to three types of information. Intrinsic and extrinsic calibration information of the camera network (where the cameras are located and how do they relate the scene to the image plane), model of the environment (what is the geometry of the environment) and the motion dynamic of the models (how and where do people tend to move).

If we had full extrinsic and intrinsic camera network calibration information, a very detailed model of the environment, a perfect tracker and a flawless dynamical model of moving people then answering the previously posed questions would be a relatively easy task. Unfortunately this usually is not the case. The trackers do not work reliably enough. They are usually sensitive to initialization and to several sources of errors and occlusion ambiguities. The environment is not static. Furniture may be relocated, lighting conditions and natural elements change. The dynamical models of moving people are often approximate. People's movements are not completely predictable.

Furthermore even if we assume the availability of perfect extrinsic and intrinsic calibration information for the camera network, this calibration information's reliability decreases over time. This is especially true in an active camera setup, where the calibration parameters may deviate from the given ones due to mechanical and environmental issues. This in turn would require periodic re-calibration of the system, which in a large multi-camera network is a daunting task.

In this paper we propose a PTZ camera network system which is capable of computing a list of cameras to allocate for a time in the near future given the current observation(s) from some camera view. We strive to constrain the system with as little assumptions as possible, i.e. no camera calibration information, no restrictions on where the cameras are placed nor on their initial parameters, no information about the environment and the dynamic model of moving people. One of the main difficulties introduced by the inclusion of active cameras is dealt with discretizating the problem. The system is capable of encapsulating the underlying target traffic patterns in specially constructed data structures which in turn are used to create camera allocation strategies. Currently we exclude scheduling and control issues since the main target of this work is to find the actual camera allocations, not how to implement them.

## 1.1. Related Work

A considerable amount of work can be found for camera network calibration problem. Svoboda, et al. [1] introduce a full system to recover calibration parameters of a fixed camera network using a laser pointer. Baker, et al. [2] attack the same problem using a calibration widget, where as Barreto, et al. [3] use structured lighting to also recover the radial distortion with cameras having wide field of views. Note that these approaches use special artificial widgets (a calibration object, laser pointer or structured lights) with tightly synchronized cameras. In contrast, Rahimi, et al. [4] use the actual traffic observations to accomplish the extrinsic calibration of the cameras. However their method allow only a single moving object in the environment at any given time. Lee, et al. [5] also recover extrinsic calibration parameters of a fixed camera network using tracking information gathered over time. They use a planar model where it is assumed that the individual intrinsic camera calibration parameters are known. It is worth noting that none of the cited work deal with the calibration of a network of *active* cameras.

There is also work on camera handoff, i.e, finding the next camera to see the target object once it leaves the field of view of the current camera. Khan, et al. [7] recover relative camera calibration by finding pairwise overlapping field of view borders using target observations over time and use this information to label targets across different cameras. Their strongest assumption is the existence of overlapping field of views. Javed, et al. [8] instead assume disjoint views and establish across-camera object labeling in a probabilistic way. They also assume the single camera tracking problem is already solved. Kettnaker, et al. [9] use a similar approach with similar assumptions. All these works assume immediate target handoff across camera field of views where as in the presented work time is a parameter for the handoff function.

In the field of robotics and control, Hutchinson, et al. [10] provide a very nice tutorial on visual servo control presenting and analyzing different strategies. Stewart, et al. [11] compare different scheduling algorithms for a sensor network. In a recent work, Costello, et al. [12] present scheduling strategies for motion tracking from a single PTZ camera. Miura, et al. [14] propose a multi-camera target assignment and planning system which allows multiple targets at any given time but requires a strong camera calibration and precise geometrical information for the environment.

In this paper we contribute to the field by addressing the following issues simultaneously which limit most of the cited work:

- No a priori information about the location and intrinsic calibration of the cameras is needed.
- There is no high-level information like tracking, data association or labeling but only low-level image foreground information allowing use of low-resolution cameras.
- The system can handle multiple targets, which is a realistic and relaxed assumption for a real-time system.
- The system uses active cameras which introduce additional camera control parameters.

The rest of the paper is organized as follows: In the next section the formal problem definition is given. Section 3 presents individual components of the system including the *delayed co-occurrence matrix* which is the core for the prediction. In Section 4 we give the results of experiments performed together with their respective implementation details. Finally a short discussion and potential extensions to the proposed system are given in Section 5.

## 2. Problem Definition

We will use ***italic boldface*** to show row vectors, *italic lowercase* to show scalars and *ITALIC UPPERCASE* to show sets. Let $CN = \{c_1, \ldots, c_n\}$ be the camera network composed of $n$ active cameras. Let $c_i$, $i = 1 \ldots n$ denote the $i^{th}$ camera in the network. Each camera has a corresponding parameter vector denoted by $\mathbf{p}_i = [\theta_i, \phi_i, \zeta_i]$ where $\theta$ is the pan, $\phi$ is the tilt and $\zeta$ is the zoom parameter. Whenever the need arises, we will show the particular time instance of a variable with superscript $t$, i.e. $\mathbf{p}_i^t$ will show the parameter set of camera $i$ at time $t$. Now define the configuration of $CN$ as, $C = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ . We want to answer the following question:

"If a moving object is observed by a camera $c_i$ with $\mathbf{p}_i^t$, in a PTZ camera network $CN$, which other camera(s) $c_k$, $k \in K \subset \{1, \ldots, n\}$ could be foveated to observe motion within a predefined time window $\Delta t$, and what would be the corresponding PTZ parameter settings $\mathbf{p}_k^{(t+\Delta t)}$ ?".

In mathematical terms we want a multi-valued function $f : \mathcal{Z}^+ \mapsto 2^{CN} \times 2^C$ such that,

$$f_{\Delta t}(i) = \{A \in 2^{CN}, B \in 2^{C^{\Delta t}}\} \tag{1}$$

where $i$ is the camera index where motion is observed, $\Delta t$ is the time delay and $2^X$ denotes the power set of $X$.

The shape of function $f_{\Delta t}$ depends on the traffic patterns exhibited by moving objects in the union of the field of views of $CN$. If the analytic solutions of the traffic pattern trajectories and full camera network calibration information were available, it would be possible to derive $f_{\Delta t}$ and perfectly predict which cameras to foveate when and where to see the object in motion assuming it is strictly obeying a given dynamic model. This scenario is, of course, a very trivial one and unfortunately it is not realistic to expect this nice behavior from a real life environment.

We will attack the problem by approximating the function $f_{\triangle t}$. We will keep records of the observations obtained over time and establish camera relationships exploiting the pairwise observation frequencies for different time windows assuming no knowledge about the camera extrinsic [1] and intrinsic parameters. We will also do this in a feed-back fashion such that the system will be able to update the approximation to $f_{\triangle t}$ using the recent history hence adapting to changes in the environment without external intervention.

## 3. Approach

The main idea is to make camera assignment predictions using an approximation of $f_{\triangle t}$ evolving over time. This approximation is computed using observations gathered over a sliding time window, establishing pairwise temporal associations among cameras. The approximation of $f_{\triangle t}$ is actually done by using *delayed co-occurrence matrices*. Before going further into the definition of the *delayed co-occurrence matrices*, we need to define the sampling of the PTZ parameter space for each camera.

### 3.1. Sampling of PTZ parameter space

The first step of the algorithm is to sample over the parameter space of each camera and use these sample points as *stations* for the rest of the algorithm. Note that this step is performed only once.

A *station* $\mathbf{s}_{ij},\ i = 1,\ldots,n,\ j = 1,\ldots,s_i$ is a predefined parameter point $\mathbf{p}_i$ for camera $i$ where $s_i$ is the number of *stations*. A lexicographically ordered set of stations constitutes a *tour* $T_i = \{[i, \mathbf{s}_{ij}]\},\ j = 1,\ldots,s_i$. Since no other information about the camera location is given, in order to increase the union of field of views (from now on denoted by $\mathrm{FOV}_i$) of $T_i$, we favor a uniform sampling over $\theta, \phi$ and $\zeta$ while keeping at a minimum level the pairwise field of view overlaps for the same camera. Note that depending on the actual placement of individual cameras, some of the *stations* may correspond to orientations with little or no information at all, like facing an obstruction. Nevertheless there is no way of knowing this before running the system and analyzing the collected data. Now we can define the *delayed co-occurrence matrix* which is the core of the algorithm.

### 3.2. Delayed Co-occurrence Matrix

Let $H_{\triangle t}$ denote a square matrix where $\Delta t$ denotes the time delay. The row and column indices of the $H_{\triangle t}$ are defined as the indices to the set $Z = \{T_1|T_2|\ldots|T_n\}$ where $|$ is the concatenation operator. Using this construction, the entries of $H_{\triangle t}$ will reflect the delayed temporal relations among the cameras. For instance, the entry $H_{\triangle t}(k,l)$ will contain the total number of events seen so far at $Z(k)$, $\Delta t$ time units after seing an event at $Z(l)$. Note that due to the way $H_{\triangle t}$ is constructed, sparse matrix methods can be used. We call $H_{\triangle t}$ the *delayed co-occurrence matrix*. How the $H_{\triangle t}$ is populated and updated is the topic of the following subsections.

[1]Except its pan, tilt, zoom parameters, provided by the PTZ control system.

```
Function ConsumeRequestQueue(J)
  input: Request queue.
  begin
    while $J \neq \emptyset$ do
      $request \leftarrow$ pop ($J$);
      $requestStation \leftarrow request(1)$;
      $requestTime \leftarrow request(2)$;
      $stationsToGo \leftarrow$
      $|requestStation - currentStation|$;
      $timeToGo \leftarrow$
      $stationsToGo \times travelTimePerStation$;
      $arrivalTime \leftarrow currentTime + timeToGo$;
      if $requestTime \geq arrivalTime$ then
        sleepUntil ($requestTime$);
        checkForMotion ();
  end
```

### 3.3. Verification of the Current Prediction

The scope of this step is to validate predictions made for the current point in time and use this information to keep a record of the number of successful predictions for a specific $Z(l)$. Each camera has an associated *request priority queue* $J$ containing *requests* defined as the predicted station $\mathbf{s}_{ij}$ and associated predicted time $t_{now} + \Delta t$. Whenever a prediction for a camera $c_i$ is made, the corresponding *request* is pushed into the queue $J_i$ which is sorted in increasing order of the *request* times.

If $J$ is non-empty then the top *request* is popped and checked for feasibility which answers the following question:“Can the camera reach the predicted *station* given its current *station*, the prediction time delay $\Delta t$ and its servo speed ?”. If the answer is “yes” then the camera is sent to the predicted *station* and becomes committed until the *request*’s time arrives. A pseudo-code is supplied in ConsumeRequestQueue.

Even if this approach is a trivial way of handling *requests*, it is still sufficient enough to produce empirical performance values for the proposed prediction algorithm.

## 3.4. Observation

If the camera is not committed then it is set to go through a random sequence of its respective *tour*, $T_i$. Intuitively this corresponds to sampling randomly through $\text{FOV}_i$. The idea here is to collect motion information through $T_i$ with minimum bias. At each *station* during the random *tour*, each camera grabs a sequence of image frames and checks for motion. If there exists significant motion then this information is propagated to the next step in the algorithm. Motion estimation and measure of significant motion are application specific. One example is described in Section 4.

## 3.5. Update

In order to explain how the $H_{\Delta t}$ is updated, we need to introduce *temporal sliding windows*. Let $Q_k$ be a queue associated with the entry $Z(k)$ such that $q \in Q_k \wedge q \in \mathcal{Z}^+$. Suppose the observation step reports motion from camera $i$ at station $T_i(j)$. Let $Z(k) = T_i(j)$. Then the current time-stamp $t_{now}$ is pushed into the queue $Q_k$. Since the time-stamp is non-decreasing, $Q_k$ is also non-decreasing by definition. In order to make the $Q_k$ be a sliding window with a window size equal to $\Delta t_{max}$, which is the farthest prediction interval requested, we prune all entries $q \in Q_k : (t_{now} - q) > \Delta t_{max}$ at each iteration. From now on we will refer to $Q_k$ as *temporal sliding window*. The pseudo-code for the update function is given in UpdateDelayedCoOccurrenceMatrix. Example $H_{\Delta t}$s are

```
Function UpdateDelayedCoOccurrenceMatrix(k)
  input: Index to $Z$ where motion is observed.
  begin
      for $l \leftarrow 1$ to $|Z|$ do
          while $(t_{now} - q) > \Delta t_{max}$ do
              $q \leftarrow$ pop ($Q_l$);
          forall $q \in Q_l$ do
              $\Delta t \leftarrow t_{now} - q$;
              $H_{\Delta t}(k, l) \leftarrow H_{\Delta t}(k, l) + 1$;
  end
```

given in Fig. 3. Note that separate $H_{\Delta t}$s are necessary to enable the corresponding $\Delta t$ delay prediction. We can now compute the conditional probability $P_{\Delta t}(z_l|z_k)$ for a given $\Delta t$,

$$P_{\Delta t}(z_l|z_k) = \frac{H_{\Delta t}(k,l)}{\sum_{l=1}^{l=|Z|} H_{\Delta t}(k,l)} \tag{2}$$

where $z_k$ is a binary random variable which is equal to 1 if motion is detected at $Z(k)$ and vice-versa. Now define the *allocation probability matrix* $\Pi_{\Delta t}$ for $\Delta t$ as,

$$\Pi_{\Delta t}(k,l) = P_{\Delta t}(z_l|z_k) \tag{3}$$

The *allocation probability matrix* $\Pi_{\Delta t}$ provides a nice probabilistic basis and understanding for the rest of the algorithm.

## 3.6. Prediction

If at time $t_{now}$ some significant motion is detected at $Z(k)$, then the prediction is made by using the *Golden Rule of Sampling* [15] or *CDF inversion sampling* method over the $k^{th}$ row of $\Pi_{\Delta t}$, since the rows of $\Pi_{\Delta t}$ represent approximate discrete probability distribution of detecting motion in $\Delta t$ time over $Z$. The sampling is performed by first computing the CDF, $F$, of $\Pi_{\Delta t}(k,\cdot)$ and dividing the [0,1] interval into $|Z|$ segments with lengths equal to $\Pi_{\Delta t}(k,l),\ l = 1,\cdots,|Z|$. A uniformly distributed random number $r \in [0,1]$ is then generated. The $Z(l)$ corresponding to,

$$F(l) < r < F(l+1)$$

is then selected as the predicted camera and its respective *station*.

The theoretical justification behind the choice of this sampling approach is that it will favor selection of a camera and its corresponding *station* with a chance proportional to its current allocation probability. Nevertheless the choice of the approach is application dependent. For instance, for a surveillance application *CDF inversion sampling* will decrease a potential intruder's chances to evade detection by simply observing the system for some time with the introduction of randomness into the camera prediction step. On the other hand, for an application aimed solely to face recognition the preferred way may be to pick the camera with the highest allocation probability to maximize the chances of observing a person.

# 4. Experiments

The experimental setup consists of a PTZ IP camera network with three cameras attached to ceiling tiles and placed in a university lab area, e.g. Fig. 1. The network is connected to a single PC. The algorithm is implemented using C++. Frames grabbed are 176x120 RGB images. Framerate is 15 fps. The system is fully real-time. The blob detection

Table 1: Prediction accuracy for the experiment. *Success* is defined as an allocation request finalizing with motion detection.

| CAMERAS | $\Sigma$ REQUEST | SUCCESS RATE |
|---|---|---|
| **1** | 137 | %12 |
| **2** | 375 | %36 |
| **3** | 373 | %41 |

is performed by using a single Gaussian background model

[16]. The number of blobs, their areas and moments are found using morphology and connected components analysis. Only blobs with areas above a given threshold proportional to the image size (in this case 2 percent of the total number of pixels in the image), are considered as significant motion. No other high level analysis, e.g. tracking, labeling, is performed. $\Delta t_{max}$ is set to be six seconds. Each camera has five *stations* with pan parameters covering uniformly $[0, 180°]$ interval with $45°$ increments as shown in Fig. 1. The tilt parameter is set to $-20°$. The zoom factor is set to $\times 1$. Servo travel time in between adjacent *stations* is 1 second.

The experiment is performed from 16:00 pm to 18:00 pm with several ($\sim 8$) students working in the area and several others going to-from offices and the lounge area creating traffic. The complete set of *delayed co-occurrence matrices* are given in Fig. 3. An intuitive explanation for the peaks on the diagonals of *delayed co-occurrence matrices* is the existence of students discussing with each other, creating significant motion around the same place over time. Most of the peaks seen in Fig. 3 agree with the high activity locations of the environment as expected. Furthermore a close observation of *delayed co-occurrence matrix* corresponding to $\Delta t = 0$ show that the peaks seen highly correlate with the overlapping FOVs of the cameras. We believe this information can be used for estimating the topology of the camera network. The performance results are given in Table 1 where *success* is achieved when a prediction *request* is finalized by a motion detection, e.g. Fig. 2. The relatively low success rates can be attributed to motion bursts generated by stationary people (students working in the area) which trigger the system to allocate cameras. Such *requests* in turn fail to detect motion since their triggering motion is stationary, not an actual traffic.

## 5. Conclusion

In this work we presented a real-time PTZ camera network system capable of generating camera allocations and their respective parameters for detecting motion in a near future given that motion is detected from some camera. We assume no camera calibration information, no information about the environment and no knowledge about the dynamical model of motion. The system gains the prediction power by constructing and updating *delayed co-occurrence matrices* over time using motion observations.

The method is tested in a difficult real-time environment and preliminary results are also presented. It is important to note that even though the method was demonstrated on predicting associations between cameras seeing motion in their FOV within some time interval, this does not guarantee that the cameras actually see the same moving object. In our preliminary study, though, we found that the predictions tend to be for the same moving object. Nonetheless, it is possible to add additional constraints in building and verifying the *delayed co-occurrence matrix* that would further increase the likelihood of observing the same object that was observed in another FOV $\Delta t$ time units ago. For instance, it is possible to use the color histograms of detected motion blobs to constraint the temporal association among the cameras.

Moreover the predictions generated by the proposed system can also be used as input to clever scheduling of the cameras [11, 12]. A scheduling system could use the prediction information together with additional info (i.e. servo timing, delays, access control etc.) and optimize some cost function, e.g. maximize resource utilization, minimize time conflict. Such extensions of the framework are under current investigation.

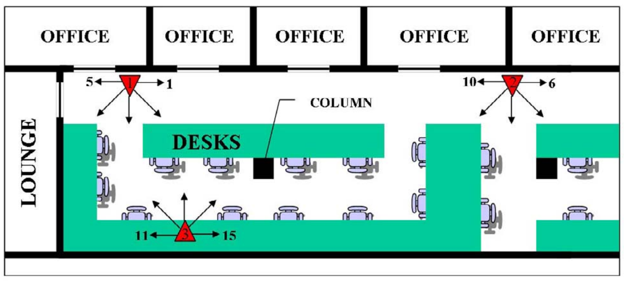

Figure 1: Illustration of the experiment area. The arrows point towards station pan orientations with corresponding indices.

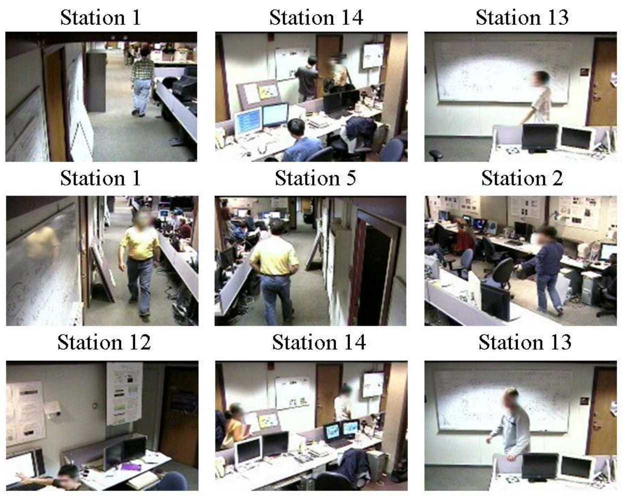

Figure 2: A couple of sample frames from successful requests. Faces are blurred for anonymity.

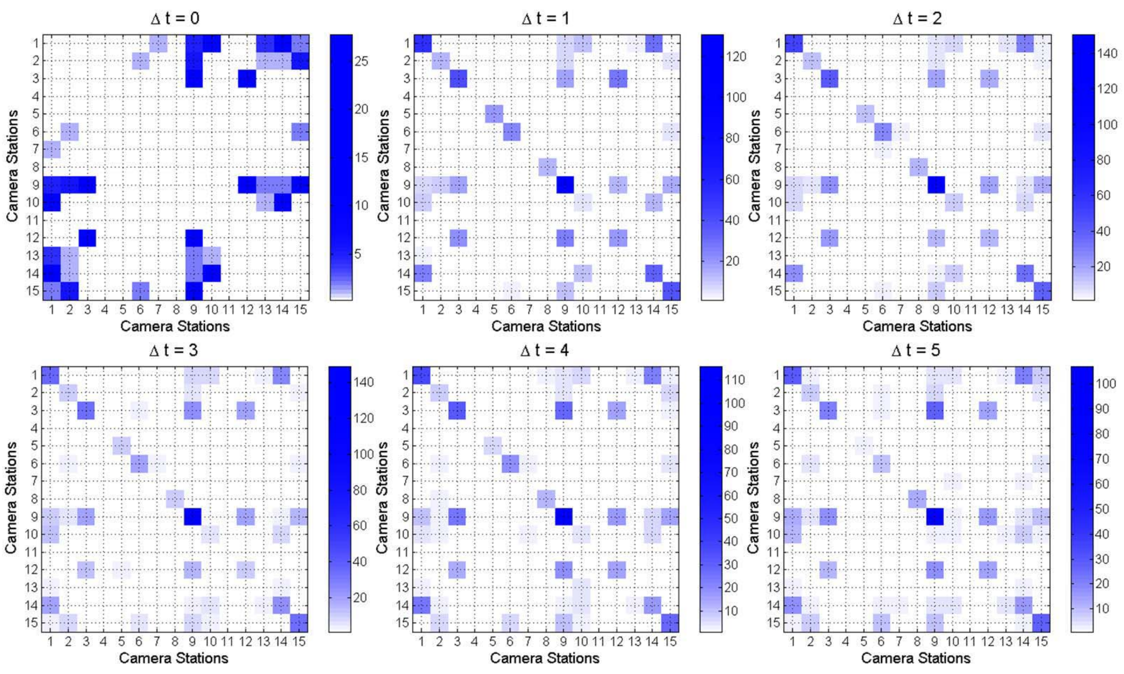

Figure 3: *Delayed co-occurrence matrices* for different $\Delta t$s. Color scaling differs for each graph.

## References

[1] T. Svoboda, D. Martinec, T. Pajdla, "A Convenient Multi-Camera Self-Calibration for Virtual Environments," *PRESENCE: Teleoperators and Virtual Environments,* The MIT Press, Vol. 14, No. 4, August 2005.

[2] P. T. Baker, Y. Aloimonos, "Calibration of a Multicamera Network," *Proc. OMNIVIS,* June, 2003.

[3] J. P. Barreto, K. Daniidilis, "Wide Area Multiple Camera Calibration and Estimation of Radial Distortion," *Proc. OMNIVIS,* May, 2004.

[4] A. Rahimi, B. Dunagan and T. Darrell, "Simultaneous Calibration and Tracking with a Network of Non-overlapping Sensors," *Proc. CVPR,* June, 2004.

[5] L. Lee, R. Romano, G. Stein, "Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame," *Trans. PAMI,* Vol. 22, No. 8, pp. 758–767, August, 2000.

[6] C. Stauffer, W. E. L. Grimson, "Learning Patterns of Activity Using Real-Time Tracking," *Trans. PAMI,* Vol. 22, No. 8, pp. 747–757, August, 2000.

[7] S. Khan, M. Shah, "Consistent Labeling of Tracked Objects in Multiple Cameras with Overlapping Fields of View," *Trans. PAMI,* Vol. 25, pp. 1355–1361, October, 2003.

[8] O. Javed, Z. Rasheed, K. Shafique, M. Shah, "Tracking Across Multiple Cameras with Disjoint Views," *Proc. ICCV,* October, 2003.

[9] V. Kettnaker, R. Zabih, "Bayesian Multi-camera Surveillance," *Proc. CVPR,* June, 1999.

[10] S. A. Hutchinson, G. D. Hager, P. I. Corke, "A Tutorial on Visual Servo Control," *Trans. Robotics and Automation,* Vol. 12, No. 5, pp. 651–670, 1996.

[11] D. Stewart, P. Khosla, "Real-Time Scheduling of Sensor Based Control Systems," *Proc. IFAC/IFIP,* 1991.

[12] C. J. Costello, C. P. Diehl, A. Banerjee, H. Fisher, "Scheduling an Active Camera to Observe People," *Proc. Video Surveillance and Sensor Networks,* 2004.

[13] T. Matsuyama, "Cooperative Distributed Vision: Dynamic Integration of Visual Perception, Action, and Communication," *Lecture Notes in Computer Science,* Springer-Verlag, Vol. 1701, pp. 75–88, 1999.

[14] J. Miura, Y. Shirai, "Parallel Scheduling of Planning and Action for Realizing an Efficient and Reactive Robotic System," *Proc. Int. Conf. on Control, Automation, Robotics and Vision,* 2002.

[15] R. Eckhardt, S. Ulam, J. Von Neumann, "The Monte Carlo Method," Los Alamos Science, p. 135, 1987.

[16] C. Wren, A. Azarbayejani, T. Darrell, A. Pentland, "Pfinder: Real-Time Tracking of the Human Body," *Trans. PAMI,* Vol. 19, No. 7, pp. 780–785, July, 1997.