INVITED PAPER

Smart Camera Networks in Virtual Reality

Simulated smart cameras track the movement of simulated pedestrians in a simulated train station, allowing development of improved control strategies for smart camera networks.

By Faisal Qureshi, *Member IEEE*, and Demetri Terzopoulos, *Fellow IEEE*

ABSTRACT | This paper presents our research towards smart camera networks capable of carrying out advanced surveillance tasks with little or no human supervision. A unique centerpiece of our work is the combination of computer graphics, artificial life, and computer vision simulation technologies to develop such networks and experiment with them. Specifically, we demonstrate a smart camera network comprising static and active simulated video surveillance cameras that provides extensive coverage of a large virtual public space, a train station populated by autonomously self-animating virtual pedestrians. The realistically simulated network of smart cameras performs persistent visual surveillance of individual pedestrians with minimal intervention. Our innovative camera control strategy naturally addresses camera aggregation and handoff, is robust against camera and communication failures, and requires no camera calibration, detailed world model, or central controller.

KEYWORDS | Camera networks; computer vision; persistent human observation; sensor networks; smart cameras; virtual reality; visual surveillance

I. INTRODUCTION

Future visual sensor networks will rely on *smart cameras* for sensing, computation, and communication. Smart cameras are self-contained vision systems, complete with increasingly sophisticated image sensors, power circuitry, (wireless) communication interfaces, and on-board processing and storage capabilities. They provide new opportunities to develop camera sensor networks capable of effective visual coverage of extensive areas—public spaces, disaster zones, battlefields, and even entire ecosystems. These multicamera systems lie at the intersection of Computer Vision and Sensor Networks, raising research problems in the two fields that must be addressed simultaneously.

In particular, as the size of the network grows, it becomes infeasible for human operators to monitor the multiple video streams and identify all events of possible interest, or even to control individual cameras directly in order to maintain persistent surveillance. Therefore, it is desirable to design camera sensor networks that are capable of performing advanced visual surveillance tasks autonomously, or at least with minimal human intervention.

In this paper, we demonstrate a model smart camera network comprising *uncalibrated*, static and active, simulated video surveillance cameras that, with minimal operator assistance, provide perceptive coverage of a large virtual public space—a train station populated by autonomously self-animating virtual pedestrians (Fig. 1). Once a pedestrian of interest is selected either automatically by the system or by an operator monitoring surveillance video feeds, the cameras decide among themselves how best to observe the subject. For example, a subset of the active pan/tilt/zoom (PTZ) cameras can collaboratively monitor the pedestrian as he or she weaves through the crowd. The problem of assigning cameras to persistently observe pedestrians becomes even more challenging when multiple pedestrians are involved. To deal with the myriad possibilities, the cameras must be able to *reason* about the dynamic situation. To this end, we propose a distributed camera network control strategy that is capable of

Manuscript received December 4, 2007; revised April 13, 2008. Current version published 00/00/2008. This work was made possible in part by a grant from the Defense Advanced Research Projects Agency (DARPA) of the U.S. Department of Defense.

F. Qureshi is with the Faculty of Science, University of Ontario Institute of Technology, Oshawa, ON, L1H 7K4, Canada (e-mail: faisal.qureshi@uoit.ca).

D. Terzopoulos is with the Computer Science Department, University of California, Los Angeles, CA 90095-1596 USA (e-mail: dt@cs.ucla.edu).

Digital Object Identifier: 10.1109/JPROC.2008.928932

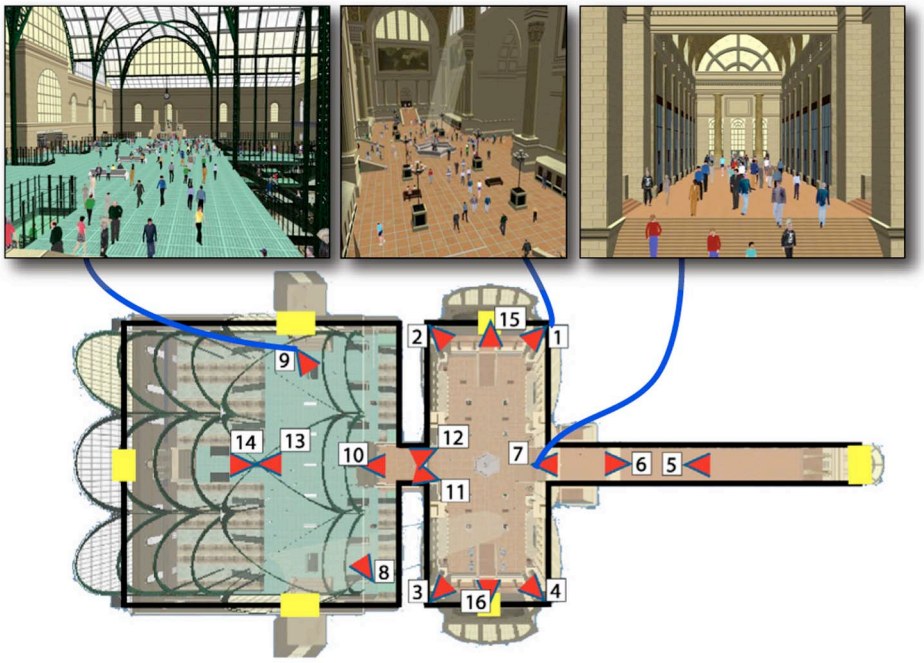


Fig. 1. *Plan view of the (roofless) virtual Penn Station environment, revealing the concourses and train tracks (left), the main waiting room (center), and the shopping arcade (right). (The yellow rectangles indicate pedestrian portals.) An example camera network is illustrated, comprising 16 simulated active (PTZ) video surveillance cameras. Synthetic images from cameras 1, 7, and 9 (from [1]).*

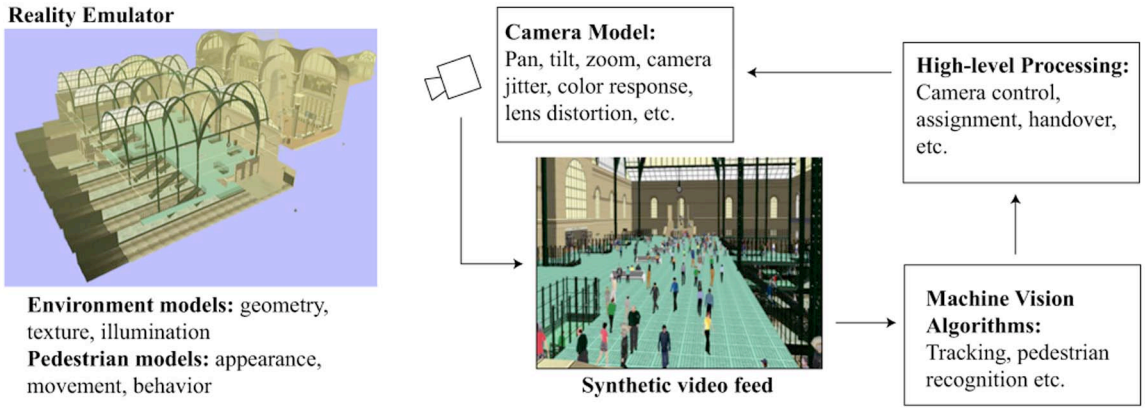


Fig. 2. *The virtual vision paradigm (image from [1]).*

dynamic, task-driven node aggregation through local decision-making and internode communication.

A. Virtual Vision

The type of research that we report here would be very difficult to carry out in the real world given the expense of deploying and experimenting with an appropriately complex smart camera network in a large public space such as an airport or a train station. Moreover, privacy laws generally restrict the monitoring of people in public spaces for experimental purposes.[1] To bypass the legal and cost impediments, we advocate *virtual vision*, a unique synthesis of computer graphics, artificial life, and computer vision technologies (Fig. 2). Virtual vision is an advanced simulation framework for working with machine vision

[1]See [2] for a discussion of privacy issues related to smart camera networks.

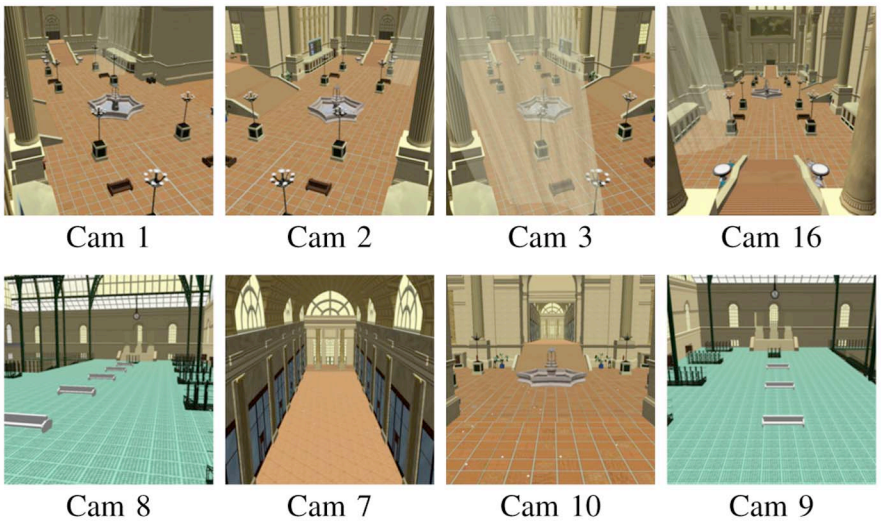


Fig. 3. *Synthetic video feeds from multiple virtual surveillance cameras situated in the (empty) Penn Station environment. Camera locations are shown in Fig. 1.*

systems, including smart camera networks, that also offers wonderful rapid prototyping opportunities. Exploiting visually and behaviorally realistic environments, called *reality emulators*, virtual vision offers significantly greater flexibility and repeatability during the camera network design and evaluation cycle, thus expediting the scientific method and system engineering process.

In our work, we employ a virtual train station populated by autonomous, lifelike virtual pedestrians, wherein we deploy virtual cameras that generate synthetic video feeds emulating those acquired by real surveillance cameras monitoring public spaces (Fig. 3). Despite its sophistication, our simulator runs on high-end commodity PCs, thereby obviating the need to grapple with special-purpose hardware and software. Unlike the real world, 1) the multiple virtual cameras are very easily reconfigurable in the virtual space, 2) we can readily determine the effect of algorithm and parameter modifications because experiments are perfectly repeatable in the virtual world, and 3) the virtual world provides readily accessible ground-truth data for the purposes of camera network algorithm validation. It is important to realize that our simulated camera networks always run online in real time within the virtual world, with the virtual cameras actively controlled by the vision algorithms. By suitably prolonging virtual-world time relative to real-world time, we can evaluate the competence of computationally expensive algorithms, thereby gauging the potential payoff of efforts to accelerate them through efficient software and/or dedicated hardware implementations.

An important issue in camera network research is the comparison of camera control algorithms. Simple video capture suffices for gathering benchmark data from time-shared physical networks of passive, fixed cameras, but gathering benchmark data for networks that include any smart, active PTZ cameras requires scene reenactment for every experimental run, which is almost always infeasible when many human subjects are involved. Costello *et al.* [3], who compared various schemes for scheduling an active camera to observe pedestrians, ran into this hurdle and resorted to Monte Carlo simulation to evaluate camera scheduling approaches. They concluded that evaluating scheduling policies on a physical testbed comprising even a single active camera is extremely problematic. By offering convenient and limitless repeatability, our virtual vision approach provides a vital alternative to physical active camera networks for experimental purposes.

Nevertheless, skeptics may argue that virtual vision relies on simulated data, which can lead to inaccurate results. Fretting that virtual video lacks all the subtleties of real video, some may cling to the dogma that it is impossible to develop a working machine vision system using simulated video. However, our high-level camera control routines do not directly process any raw video. Instead, these routines are realistically driven by data supplied by low-level recognition and tracking routines that mimic the performance of a state-of-the-art pedestrian localization and tracking system, including its limitations and failure modes. This enables us to develop and evaluate camera network control algorithms under realistic simulated conditions consistent with physical camera networks. We believe that the fidelity of our virtual vision emulator is such that algorithms developed through its use will readily port to the real world.

B. Smart Camera Network

Many of the challenges associated with sensor networks are relevant to our work. A fundamental issue is the selection of sensor nodes that should participate in a

particular sensing task [4]. The selection process must take into account the informational contribution of each node against its resource consumption or potential utility in other tasks. Distributed approaches for node selection are preferable to centralized approaches and offer what are perhaps the greatest advantages of networked sensing—robustness and scalability. Also, in a typical sensor network, each node has local autonomy and can communicate with a small number of neighboring nodes, where the neighborhood of a node can be defined automatically as the set of nodes that are, e.g., within nominal radio communications distance of that node [5]. Message delay and message loss are common occurrences in sensor networks due to bandwidth limitations, interference, etc. One must also contend with nonstationary network topology due to node failures, node additions, etc.

Mindful of these issues, we propose a novel camera network control strategy that does not require camera calibration, or a detailed world model, or a central controller. The overall behavior of the network is the consequence of the local processing at each node and internode communication. The network is robust to node and communication failures. Moreover, it is scalable because of the lack of a central controller. Visual surveillance tasks are performed by groups of one or more camera nodes. These groups, which are created on the fly, define the information sharing parameters and the extent of collaboration between nodes. A group evolves—i.e., old nodes leave the group and new nodes join it—during the lifetime of the surveillance task. One node in each group acts as the group supervisor and is responsible for group-level decision making. We also present a novel constraint satisfaction problem formulation for resolving interactions between groups.

We assume the following communication model: 1) nodes can communicate with their neighbors, 2) messages from one node can be delivered to another node if there is a path between the two nodes, and 3) messages can be sent from one node to all the other nodes. Furthermore, we assume the following network model: 1) messages can be delayed, 2) messages can be lost, and 3) nodes can fail. These assumptions ensure that our virtual camera network faithfully mimics the operational characteristic of a real sensor network.

C. Contributions and Overview

The contribution of this paper is twofold. We introduce a novel camera sensor network framework suitable for next-generation visual surveillance applications. We also demonstrate the advantages of developing and evaluating camera sensor networks within our sophisticated virtual reality simulation environment. A preliminary version of this work appeared in [6] and it extends that reported in an earlier paper [7]. Among other extensions, we introduce a novel Constraint Satisfaction Problem (CSP) formulation for resolving group–group interactions.

The remainder of the paper is organized as follows: Section II reviews relevant prior work. We explain the low-level vision emulation and behavior models for camera nodes in Section III. Section IV presents the sensor network communication model. Section V discusses the application of the model in the context of persistent visual surveillance and presents our results. Section VI concludes the paper and discusses future research directions.

II. RELATED WORK

In 1997, Terzopoulos and Rabie introduced a purely software-based approach to designing active vision systems, called *animat vision* [8]. Their approach prescribes the use of artificial animals (or animats) situated in physics-based virtual worlds to study and develop active vision systems, rather than struggling with hardware—the cameras and wheeled mobile robots typically used by computer vision researchers. They demonstrated the animat vision approach by implementing biomimetic active vision systems for virtual animals and humans [9]. The algorithms developed were later adapted for use in a vehicle tracking and traffic control system [10], which affirmed the usefulness of the animat vision approach in designing and evaluating complex computer vision systems.

Envisioning a large computer-simulated world inhabited by virtual humans that look and behave like real humans, Terzopoulos [11] then proposed the idea of using such visually and behaviorally realistic environments, which he called reality emulators, to design machine vision systems, particularly surveillance systems. The work presented here is a significant step towards realizing this vision. Shao and Terzopoulos [1] developed a prototype reality emulator, comprising a reconstructed model of the original Pennsylvania Station in New York City populated by virtual pedestrians, autonomous agents with functional bodies and brains. The simulator incorporates a large-scale environmental model of the train station with a sophisticated pedestrian animation system including behavioral, perceptual, and cognitive human simulation algorithms. The simulator can efficiently synthesize well over 1000 self-animating pedestrians performing a rich variety of activities in the large-scale indoor urban environment. Like real humans, the synthetic pedestrians are fully autonomous. They perceive the virtual environment around them, analyze environmental situations, make decisions, and behave naturally within the train station. They can enter the station, avoiding collisions when proceeding through congested areas and portals, queue in lines as necessary, purchase train tickets at the ticket booths in the main waiting room, sit on benches when tired, obtain food/drinks from vending machines when hungry/thirsty, etc., and eventually proceed to the concourses and descend stairs to the train platforms. Standard computer graphics techniques render the busy urban scene with considerable

geometric and photometric detail (Fig. 1). Our camera network is deployed and tested within this virtual train station simulator.

In concordance with the virtual vision paradigm, Santuari *et al.* [12], [13] advocate the development and evaluation of pedestrian segmentation and tracking algorithms using synthetic video generated within a virtual museum simulator containing scripted animated characters. Synthetic video is generated via rendering, which supports global illumination, shadows, and visual artifacts like depth of field, motion blur, and interlacing. They have used their virtual museum environment to develop static background modeling, pedestrian segmentation, and pedestrian tracking algorithms. They focus on low-level computer vision, whereas our work goes beyond this to focus on high-level computer vision issues, especially multicamera control in large-scale camera networks. Previous work on multicamera systems has dealt with issues related to low- and medium-level computer vision, namely, identification, recognition, and tracking of moving objects [14]–[18]. The emphasis has been on tracking and on model transference from one camera to another, which is required for object identification across multiple cameras [19]. Multiple cameras have also been employed either to increase the reliability of the tracking algorithm [20] (by overcoming the effects of occlusion or by using three-dimensional (3-D) information for tracking) or to track an object as it moves through the fields of view (FOVs) of different cameras. In most cases, object tracking is accomplished by combining some sort of background subtraction strategy and an object appearance/motion model [21]. Numerous researchers have proposed camera network calibration to achieve robust object identification and classification from multiple viewpoints, and automatic camera network calibration strategies have been proposed for both stationary and actively controlled camera nodes [22]–[24]. Schemes for learning sensor (camera) network topologies have also been proposed [25]–[27].

Little attention has been paid, however, to the problem of controlling or scheduling active cameras when there are more objects to be monitored in the scene than there are active cameras. Some researchers employ a stationary wide-FOV camera to control an active camera [3], [28]–[30]. Generally speaking, the cameras are assumed to be calibrated and the total coverage of the cameras is restricted to the FOV of the stationary camera. In contrast, our approach does not require calibration; however, we assume that the cameras can identify a pedestrian with reasonable accuracy. To this end, we employ color-based pedestrian appearance models.

The problem of forming sensor groups based on task requirements and resource availability has received much attention within the sensor networks community [4]. Mallet [27] argues that task-based grouping in *ad hoc* camera networks is highly advantageous. Collaborative tracking, which subsumes this issue, is considered an essential capability in many sensor networks [4]. Zhao *et al.* [31] introduce an information driven approach to collaborative tracking that attempts to minimize the energy expenditure at each node by reducing internode communication. A node selects the next node by utilizing the information gain versus energy expenditure tradeoff estimates for its neighbor nodes. In the context of camera networks, it is often difficult without explicit geometric and camera calibration knowledge for a camera node to estimate the expected information gain of assigning another camera to the task, but such knowledge is tedious to obtain and maintain during the lifetime of the camera network. Therefore, our camera networks eschew such knowledge; a node need only communicate with nearby nodes before selecting new nodes.

The nodes in sensor networks are usually untethered sensing units with limited onboard power reserves. Hence, a crucial concern is the energy expenditure at each node, which determines the lifespan of a sensor network [32]. Node communications have large power requirements; therefore, sensor network control strategies attempt to minimize the internode communication [31]. Presently, we do not address this issue; however, the communication protocol that we propose limits the communication to the active nodes and their neighbors. IrisNet is a sensor network architecture tailored towards advanced sensors connected via high-capacity communication channels [33]. It takes a *centralized* view of the network, modeling it as a distributed database that allows efficient access to sensor readings. We consider this work to be orthogonal to ours. SensEye is a recent sensor-network inspired multicamera system [34]. It demonstrates the low latency and energy efficiency benefits of a multitiered network, where each tier defines a set of sensing capabilities and corresponds to a single class of smart camera sensors. However, SensEye does not deal with the distributed camera control issues that we address.

Our node grouping strategy is inspired by the ContractNet distributed problem solving protocol [35] and it realizes group formation via internode negotiation. Unlike Mallett's [27] approach to node grouping, where groups are defined implicitly via membership nodes, our approach defines groups explicitly through group leaders. This simplifies reasoning about groups; e.g., Mallett's approach requires specialized nodes for group termination. Our strategy handles group leader failures through group merging and group leader demotion operations.

Resolving group–group interactions requires sensor assignment to various tasks, which shares many features with Multi-Robot Task Allocation (MRTA) problems studied by the multiagent systems community [36]. Specifically, according to the taxonomy provided in [36], our sensor assignment formulation belongs to the single-task (ST) robots, multirobot (MR) tasks, instantaneous assignment (IA) category. ST–MR–IA problems are significantly more difficult than single-robot-task MRTA

problems. Task-based robot grouping arises naturally in ST–MR–IA problems, which are sometimes referred to as *coalition formation*. ST–MR–IA problems have been extensively studied and they can be reduced to a set partitioning problem (SPP), which is strongly NP-hard [37]. However, heuristics-based set partitioning algorithms exist that produce good results on large SPPs [38]. Fortunately, the sizes of MRTA problems, and by extension SPPs, encountered in our camera sensor network setting are small because of the spatial or locality constraints inherent to the camera sensors.

We model sensor assignments as a CSP, which we solve using "centralized" backtracking. Each sensor assignment that passes the hard constraints is assigned a weight, and the assignment with the highest weight is selected. We have intentionally avoided distributed constraint optimization techniques (e.g., [39] and [40]) because of their explosive communication requirements even for small sized problems. Additionally, it is not obvious how they handle node and communication failures. Our strategy lies somewhere between purely distributed and fully centralized schemes for sensor assignment—sensor assignment is distributed at the level of the network, whereas it is centralized at the level of a group.

III. SMART CAMERA NODES

Each virtual camera node in the sensor network is able to perform low-level visual processing and is an active sensor with a repertoire of camera behaviors. The virtual cameras also render the scene to generate synthetic video suitable for machine vision processing. Sections III-A–D describe each of these aspects of a camera node.

A. Synthetic Video

Virtual cameras use the OpenGL library and standard graphics pipeline [41] to render the synthetic video feed. Our imaging model emulates imperfect camera color response, compression artifacts, detector and data drop-out noise, and video interlacing; however, we have not yet modeled other imaging artifacts such as depth-of-field, vignetting, and chromatic aberration. Furthermore, the rendering engine does not yet support pedestrian shadows and specular highlights. More sophisticated rendering schemes would address these limitations. Noise is introduced during a post-rendering phase. The amount of noise introduced into the process determines the quality of the input to the visual analysis routines and affects the performance of the pedestrian segmentation and tracking module.

We model the variation in color response across cameras by manipulating the Hue, Saturation, Value (HSV) channels of the rendered image. Similarly, we can adjust the tints, tones, and shades of an image by adding the desired amounts of blacks, whites, and grays, respectively [42]. Our visual analysis routines rely on color-based appearance models to track pedestrians; hence, camera handovers are sensitive to variations in the color response of different cameras.

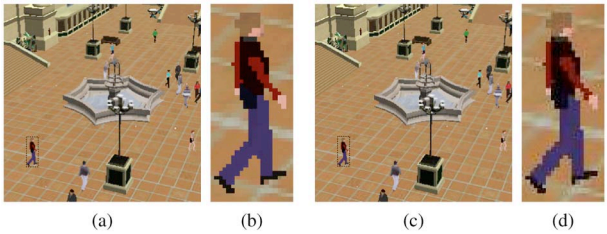


Fig. 4. *Compression artifacts in synthetic video. (a) Uncompressed image. (b) Enlarged region of the rectangular box in (a). (c) JPEG-compressed image. (d) Enlarged region of the rectangular box in (c).*

Bandwidth is generally at a premium in sensor networks, especially so in camera networks. In many instances, images captured by camera nodes are transmitted to a central location for analysis, storage, and monitoring purposes. Camera nodes routinely exchange information among themselves during camera handover, camera coordination, and multicamera sensing operations. The typical data flowing in a camera network is image/video data, which places much higher demands on a network infrastructure than, say, alphanumeric or voice data. Consequently, in order to keep the bandwidth requirements within acceptable limits, camera nodes compress the captured images and video before sending them off to other camera nodes or to the monitoring station.

Compression artifacts together with the low resolution of the captured images/video pose a challenge to visual analysis routines and are therefore relevant to camera network research. We introduce compression effects into the synthetic video by passing it through a JPEG compression/decompression stage before providing it to the pedestrian recognition and tracking module. Fig. 4 shows compressed and uncompressed versions of a 1000×1000 image. The compressed version (24 kb) is about 10 times smaller than the uncompressed version (240 kb). Notice the compression artifacts around the color region boundaries in Fig. 4(d).

We simulate detector noise as a data-independent, additive process with a zero-mean Gaussian distribution [Fig. 5(a)]. The standard deviation of the Gaussian distribution controls the amount of noise introduced into the image. Data dropout noise is caused by errors during data transmission within the imaging device [Fig. 5(b)]. The corrupted pixels are either set to the maximum value (snow) or have their bits flipped. Sometimes pixels are alternatively set to the maximum value or zero (salt and pepper noise). The amount of noise is determined by the percentage of corrupted pixels.

We simulate interlaced video by rendering frames at twice the desired frequency and interlacing the even and

Fig. 5. *Simulating noise in synthetic video. (a) Detector noise. (b) Data dropout noise.*

odd rows of sequential frames. Fig. 6 shows a 640×480 deinterlaced frame. The frame was generated by weaving two fields that were rendered 1/60th s apart. Pedestrians that are moving across the image plane appear jagged around the edges proportional to their speed. Interlacing effects also appear during panning and zooming operations in active PTZ cameras. Deinterlacing artifacts can be mitigated [43], but not removed entirely.

B. Visual Processing

The sensing capabilities of a camera node are determined by the low-level visual routines (LVR). The LVRs, such as pedestrian tracking and identification, are computer vision algorithms that directly operate upon the synthetic video generated by the virtual cameras. They mimic the performance of a state-of-the-art pedestrian segmentation and tracking module. In particular, pedestrian tracking can fail due to occlusions, poor segmentation, bad lighting, or crowding (Fig. 7). Tracking sometimes locks on the wrong pedestrian, especially if the scene contains multiple pedestrians with similar visual appearance; i.e., wearing similar clothes. Additionally, the virtual world affords us the benefit of fine tuning the performance of the recognition and tracking module by taking into consideration the ground truth data readily available from the virtual world.

We employ appearance-based models to track pedestrians. Pedestrians are segmented to compute robust color-based signatures, which are then matched across subsequent frames. Color-based signatures have found widespread use in tracking applications [44], but they are sensitive to illumination changes. This shortcoming can be mitigated, however, by operating in HSV color space instead of RGB color space. Furthermore, zooming can drastically change the appearance of a pedestrian, thereby confounding conventional appearance-based schemes. We employ a modified *color-indexing* scheme [45] to tackle this problem. Thus, a distinctive characteristic of our pedestrian tracking routine is its ability to operate over a range of camera zoom settings. It is important to note that we do not assume camera calibration.

Conventional pedestrian segmentation is difficult for active PTZ cameras due to the difficulty of maintaining a background model. We match pedestrian signatures across frames through *color indexing*. Proposed by Swain and Ballard [45], color indexing efficiently identifies objects present in an image using their color distributions in the presence of occlusions as well as scale and viewpoint changes. It was adapted by Terzopoulos and Rabie [8] for active vision in artificial animals. In color indexing, targets with similar color distributions are detected and localized through histogram backprojection, which finds the target in an image by emphasizing colors in the image that belong to the observed target.

For target histogram T and image histogram I, we define the ratio histogram as $R(i) = T(i)/I(i)$ for $i = 1, \ldots, n$, where n is the number of bins and $T(i)$, $I(i)$, and $R(i)$ are the number of samples in bin i of the respective histograms, and we set $R(i) = 0$ when $I(i) = 0$. Histogram R is backprojected into the image, which involves replacing the image pixel values by the values of R that they index: $B(x,y) = R(\mathrm{map}(\boldsymbol{c}(x,y)))$, where $B(x,y)$ is the value of the backprojected image at location (x,y), and where $\boldsymbol{c}(x,y)$ is the color of the pixel at location (x,y) and the function $\mathrm{map}(\boldsymbol{c})$ maps a 3-D HSV color value to the appropriate histogram bin. The backprojected image is then convolved with a circular disk of area equal to the expected area of the target in the image: $B_r = D_r * B$, where D_r is the disk of radius r. The peak in the convolved image gives the expected (x,y) location of the target in the image. We refer the reader to [45] for a thorough description of this process.

The last step of the color indexing procedure assumes that the area of the target in the image is known *a priori*. Active PTZ cameras violate this assumption, as the area covered by the target in the image can vary greatly depending on the current zoom settings of the camera. We propose a novel scheme to localize targets in a histogram

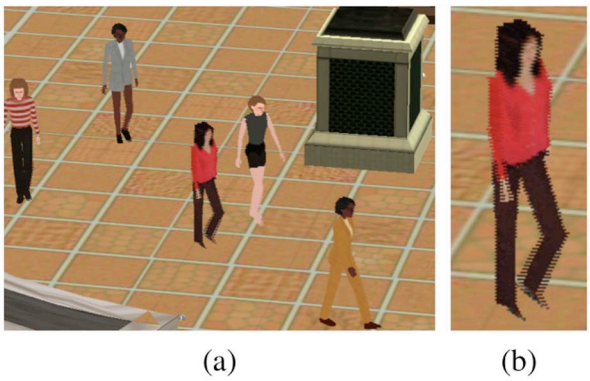


Fig. 6. *Simulating video interlacing effects. (a) A deinterlaced video frame computed by weaving two fields. (b) Close-up view of a pedestrian in (a).*

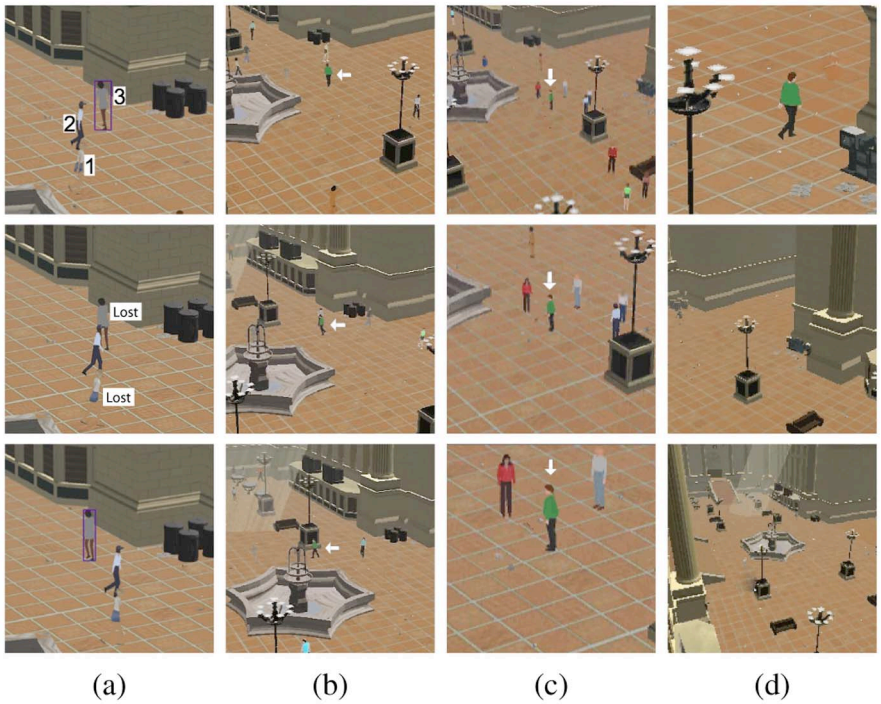


Fig. 7. *(a) The LVRs are programmed to track Pedestrians 1 and 3. Pedestrian 3 is tracked successfully; however, track is lost of Pedestrian 1 who blends into the background. The tracking routine loses Pedestrian 3 when she is occluded by Pedestrian 2, but it regains track of Pedestrian 3 when Pedestrian 2 moves out of the way. (b) Tracking while fixating on a pedestrian. (c) Tracking while zooming in on a pedestrian. (d) Camera returns to its default settings upon losing the pedestrian; it is now ready for another task.*

backprojected image when the size of the targets in the image is not known beforehand. Our scheme is based on the observation that when the size of the target is equal to the size of the localization kernel (i.e., the disk D_r), the filter response forms a peak at the "true" location of the target. On the other hand, the filter response forms a plateau centered at the "true" location of the target in the image for kernel sizes that are either too large or too small relative to the size of the target in the image. Fig. 8 illustrates this phenomenon. Fig. 9 details and demonstrates

Image: $I = \boxed{0}\boxed{0}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}$

Filter response forms a plateau for kernel sizes that are too small:

$$I * \boxed{1} = \boxed{0}\boxed{0}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}$$

Filter response forms a peak for kernel sizes that match the target size:

$$I * \boxed{1}\boxed{1}\boxed{1} = \boxed{0}\boxed{1}\boxed{2}\boxed{3}\boxed{2}\boxed{1}\boxed{0}$$

Filter response forms a plateau for kernel sizes that are too large:

$$I * \boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1} = \boxed{1}\boxed{2}\boxed{3}\boxed{3}\boxed{3}\boxed{2}\boxed{1}$$

Fig. 8. *Multiscale target localization in histogram backprojected images: Convolving an idealized 7-pixel 1-D backprojected image I with 1-tap, 3-tap, and 5-tap summing kernels. The image is extended with 0 borders for convolution purposes.*

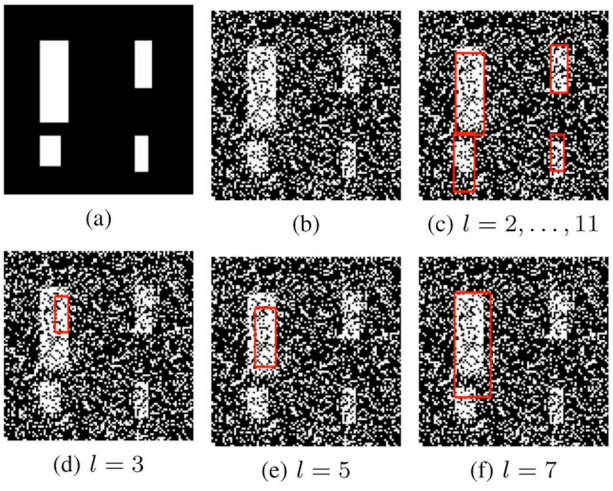


Fig. 9. *Target localization in backprojected images. The algorithm is detailed (top) and demonstrated on synthetic data using $(6l+1) \times (2l+1)$ rectangular summing kernels. (a) An ideal 2-D backprojected image that contains four different-sized targets. (b) Noise is added to the image to exacerbate the localization problem. (c) Our multiscale localization procedure successfully identifies all four regions, whereas the procedure in [45] yields poor localization results for kernel sizes 3 (d), 5 (e), and 7 (f).*

our algorithm for target localization in backprojected images. We use $(6l+1) \times (2l+1)$ rectangular kernels, where l is chosen as a fraction of $h/7$, with h the height of a video frame in pixels. Typically, $l = (0.1, 0.2, \ldots, 1) \times h/7$.

Algorithm for target localization in backprojected images

Step 1: Compute $B_l = B * K_l$, where B is the backprojected image, K_l is the kernel of size l, and $l = 1, \ldots, m$.

Step 2: Find $(x^*, y^*) = \text{argmax}_{(x,y)} \sum_{l=1}^{m} B_l$.

Step 3: Find $(x_l^*, y_l^*) = \text{argmax}_{(x,y) \in \mathcal{K}} B_l$, where $\mathcal{K}$ is the domain of K_l centered at (x^*, y^*).

Step 4: Find

$$l^* = \arg\max_{l} \sum_{(x,y) \in \mathcal{K}} (B_l(x_l^*, y_l^*) - B_l(x,y))^2 / |\mathcal{K}|.$$

Step 5: Construct the n-bin color histogram H of the region of size l centered at (x^*, y^*) using the original image.

If $\sum_i^n \min(T(i), H(i)) / \sum_i^n T(i) > \rho$, a user-specified threshold,

then output the region of size l at location (x^*, y^*), **else** *quit*.

Step 6: Remove from B_l a region of size l centered at (x^*, y^*) by setting the values of all the pixels in the region to 0.

Repeat steps 1 through 6.

Each camera can *fixate* and *zoom* in on an object of interest. The fixation and zooming routines are image-driven and do not require any 3-D information such as camera calibration or a global frame of reference. The *fixate* routine brings the region of interest—e.g., the bounding box of a pedestrian—into the center of the image by rotating the camera about its local x and y axes. The *zoom* routine controls the FOV of the camera such that the region of interest occupies the desired percentage of the image. Refer to [46] for the details.

C. Camera Node Behavioral Controller

Each camera node is an autonomous agent capable of communicating with nearby nodes. The camera controller determines the overall behavior of the camera node, taking into account the information gathered through visual analysis by the LVRs (bottom-up) and the current task (top-down). We model the camera controller as an augmented hierarchical finite state machine (Fig. 10).

In its default state, *Idle*, the camera node is not involved in any task. It transitions into the *Computing-Relevance* state upon receiving a *queryrelevance* message from a nearby node. Using the description of the task that

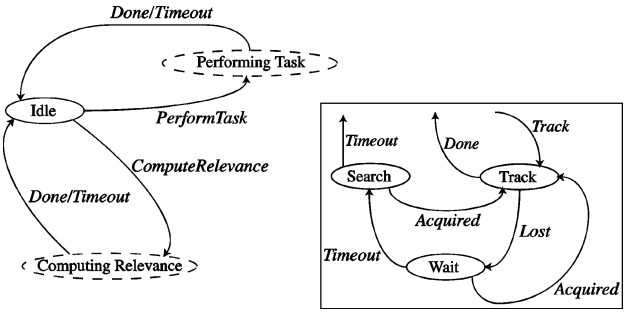


Fig. 10. *The top-level camera controller consists of a hierarchical FSM. The inset (right) represents the child FSM embedded within the PerformingTask and ComputingRelevance states in the top-level FSM.*

is contained within the *queryrelevance* message, and by employing the LVRs, the camera node can compute its *relevance* to the task (see Section III-D). For example, it can use visual search to find a pedestrian that matches the appearance-based signature forwarded by the querying node. The relevance encodes the expectation of how successful a camera node will be at a particular sensing task. The camera node returns to the *Idle* state if it fails to compute its relevance because it cannot find a pedestrian matching the description. Otherwise, when the camera successfully finds the desired pedestrian, it returns its relevance value to the querying node. The querying node passes the relevance value to the supervisor node of the group, which decides whether or not to include the camera node in the group. The camera goes into the *Performing-Task* state upon joining a group, where the embedded child finite state machine (FSM) hides the sensing details from the top-level controller and enables the node to handle transient sensing (tracking) failures. All states other than the *PerformingTask* state have built-in timers (not shown in Fig. 10) that allow the camera node to transition into the *Idle* state rather than wait indefinitely for a message from another node.

The child FSM [Fig. 10 (inset)] starts in *Track* state, where video frames are processed to track a target without panning and zooming a camera. *Wait* is entered when track is lost. Here camera zoom is gradually reduced in order to reacquire track. If a target is not reacquired during *Wait*, the camera transitions to the *Search* state, where it performs search sweeps in PTZ space to reacquire the target.

A camera node returns to its default state after finishing a task, using the *reset* routine, which is a PD controller that attempts to minimize the difference between the current zoom/tilt settings and the default zoom/tilt settings.

D. Computing Camera Node Relevance

The accuracy with which individual camera nodes are able to compute their relevance to the task at hand determines the overall performance of the network. The computation of the relevance of a camera to a video

Status	=	$s \in \{\text{busy}, \text{free}\}$
Quality	=	$c \in [0, 1]$
FOV	=	$\theta \in [\theta_{\min}, \theta_{\max}]$ degrees
XTurn	=	$\alpha \in [\alpha_{\min}, \alpha_{\max}]$ degrees
YTurn	=	$\beta \in [\beta_{\min}, \beta_{\max}]$ degrees
Time	=	$t \in [0, \infty)$ seconds
Task	=	$a \in \{a_i \mid i = 1, 2, \ldots\}$

Fig. 11. *Quantities associated with computing the relevance metric of a camera node relative to a surveillance task.*

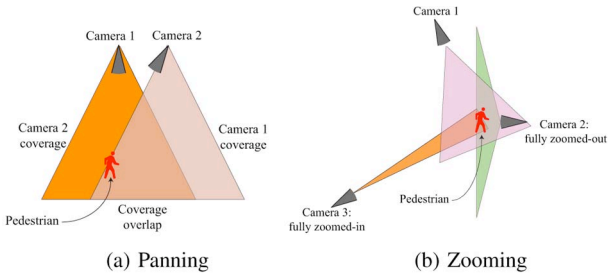


Fig. 12. *The effect of the pan and zoom settings of a camera on its relevance to a visual sensing task. (a) Both cameras can track the pedestrian; however, Camera 2 is at the limit of its pan angle, so (1) computes a lower relevance for it. (b) All three cameras can track the pedestrian, but 2 and 3 can do so only at the limits of their zoom settings; (1) computes a higher relevance for Camera 1.*

surveillance task encodes the intuitive observations that 1) a camera that is currently free should be chosen for the task, 2) a camera with better tracking performance with respect to the task at hand should be chosen, 3) the turn and zoom limits of cameras should be taken into account when assigning a camera to a task; i.e., a camera that has more leeway in terms of turning and zooming might be able to follow a pedestrian for a longer time, and 4) it is better to avoid unnecessary reassignments of cameras to different tasks, as doing so may degrade the performance of the underlying computer vision routines.

Upon receiving a task request, a camera node returns to the leader node a list of attribute-value pairs quantifying its relevance to the current task along multiple dimensions (Fig. 11). The leader node uses them to compute a relevance metric whose result is a scalar relevance value r, as shown in (1),

$$r = \begin{cases} \exp\left(-\frac{(\theta-\hat{\theta})^2}{2\sigma_\theta{}^2} - \frac{(\alpha-\hat{\alpha})^2}{2\sigma_\alpha{}^2} - \frac{(\beta-\hat{\beta})^2}{2\sigma_\beta{}^2}\right), & \text{if the camera is free} \\ 0, & \text{if the camera is busy} \end{cases} \tag{1}$$

where $\hat{\theta} = (\theta_{\min} + \theta_{\max})/2$, $\hat{\alpha} = (\alpha_{\min} + \alpha_{\max})/2$, and $\hat{\beta} = (\beta_{\min} + \beta_{\max})/2$. Here, $\theta_{\min}$ and $\theta_{\max}$ are extremal FOV settings, $\alpha_{\min}$ and $\alpha_{\max}$ are extremal vertical rotation angles around the x-axis, and $\beta_{\min}$ and $\beta_{\max}$ are extremal horizontal rotation angles around the y-axis. The values of the variances σ_θ, σ_α, and σ_β associated with each attribute are chosen empirically (in our experiments, we set $\sigma_\theta = \sigma_\alpha = \sigma_\beta = 5.0$), where α, β, and θ denote the camera pan, tilt, and zoom values, respectively, required to center the pedestrian in the field of view of the camera. This distance can be approximated by the declination angle, which may be estimated from α under a ground-plane assumption. Fig. 12 illustrates the relevance of cameras subject to their pan/zoom settings. See [46] for additional details.

IV. CAMERA NETWORK MODEL

The camera network communication scheme that enables task-specific node organization functions as follows: A human operator presents a particular sensing request to one of the nodes. In response to this request, relevant nodes self-organize into a group with the aim of fulfilling the sensing task. The *group*, which represents a collaboration between member nodes, is a dynamic arrangement that evolves throughout the lifetime of the task. At any given time, multiple groups might be active, each performing its respective task. Group formation is determined by the local computation at each node and the communication between the nodes. Specifically, we employ the ContractNet protocol, which models auctions (announcement, bidding, and selection) for group formation [35] (Fig. 13). The local computation at each node involves choosing an appropriate bid for the announced sensing task.

We distinguish between two kinds of sensing task initializations: 1) where the queried camera itself can measure the phenomenon of interest—e.g., when the operator selects a pedestrian to be observed in a particular video feed—and 2) when the queried camera node is unable to perform the required sensing and must route the query to other nodes—e.g., when the operator tasks the network to count the number of pedestrians wearing green tops. To date we have experimented only with the first kind of task initializations, which are sufficient for performing collaborative persistent observation tasks;

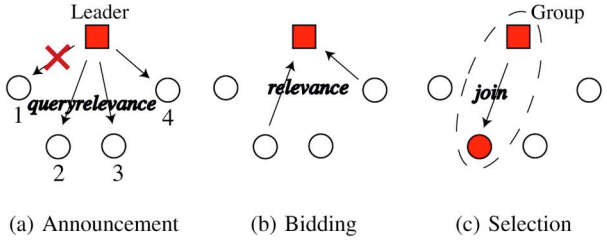


Fig. 13. *Task auction supports coalition formation. The red cross indicates a lost message.*

however, this is by no means a limitation of our proposed communication model.

A. Node Grouping

Node grouping commences when a node n receives a sensing query. In response to the query, the node sets up a named task and creates a single-node group. Initially, as node n is the only node in the group, it is chosen as the leader. To recruit new nodes to the current task, node n begins by sending *queryrelevance* messages to its neighboring nodes, N_n. This is akin to auctioning the task in the hope of finding suitable nodes. A subset N' of N_n respond by sending their relevance values for the current task (*relevance* message). This is the bidding phase. Upon receiving the relevance values, node n selects a subset M of N' to include in the group and sends *join* messages to the chosen nodes. This is the selection phase. When there is no resource contention between groups—e.g., when only one task is active, or when multiple tasks that do not require the same nodes for successful operation are active—the selection process is relatively straightforward; node n picks those nodes from N' that have the highest relevance values. Otherwise, when multiple groups vie for the same nodes, a conflict resolution mechanism is necessary. In Section IV-B, we present a conflict resolution method to handle this situation. A node that is not already part of any group can join the group upon receiving a *join* message from the leader of that group. After receiving the *join* message, a subset M' of M elect to join the group.

For groups comprising more than one node, if a group leader decides to recruit more nodes to the task at hand, it instructs group nodes to broadcast task requirements. This is accomplished by sending *queryrelevance* to group nodes. The leader node is responsible for group-level decisions, so member nodes forward to the group leader all the group-related messages, such as the *relevance* messages from potential candidates for group membership. During the lifetime of a group, member nodes broadcast *status* messages at regular intervals. Group leaders use these messages to update the relevance information of the group nodes. When a leader node receives a *status* message from another node performing the same task, the leader node includes that node into its group. The leader uses the most recent relevance values to decide when to drop a member node. A group leader also removes a node from the group if it has not received a *status* message from that node by some preset time limit.[2] Similarly, a group node can choose to stop performing the task when it detects that its relevance value is below a predefined threshold. When a leader detects that its own relevance value for the current task is below the threshold, it selects a new leader from among the member nodes. The group vanishes when the last member node leaves.

[2]The relevance value of a group node decays over time in the absence of new *status* messages from that node. Thus, we can conveniently model node-dependent timeouts; i.e., the time interval during which at least one *status* message must be received by the node in question.

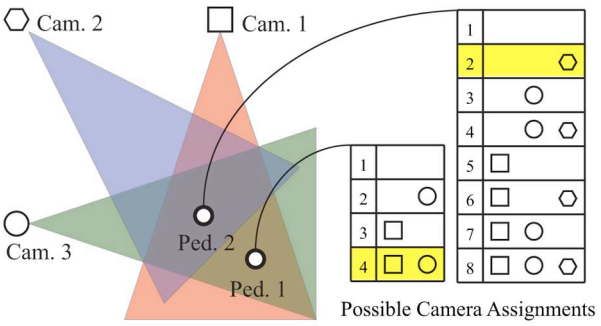


Fig. 14. *Conflict resolution for camera assignment.*

B. Conflict Resolution

A conflict resolution mechanism is needed when multiple groups require the same resources. The problem of assigning cameras to the contending groups can be treated as a Constraint Satisfaction Problem (CSP) [47]. Formally, a CSP consists of a set of variables $\{v_1, v_2, v_3, \ldots, v_k\}$, a set of allowed values $\mathrm{Dom}[v_i]$ for each variable v_i (called the domain of v_i), and a set of constraints $\{C_1, C_2, C_3, \ldots, C_m\}$. The solution to the CSP is a set $\{v_i \leftarrow a_i \mid a_i \in \mathrm{Dom}[v_i]\}$, where the assignments satisfy all the constraints.

We treat each group g as a variable whose domain consists of the nonempty subsets of the set of cameras with relevance values (with respect to the task associated with g) greater than a predefined threshold. The constraints restrict the assignment of a camera to multiple groups. We define a constraint C_{ij} as $a_i \cap a_j = \{\Phi\}$, where a_i and a_j are camera assignments to groups g_i and g_j, respectively; k groups give rise to $k(k-1)/2$ constraints. We can then define a CSP as $P = (G, D, C)$, where $G = \{g_1, g_2, \ldots, g_k\}$ is the set of groups (variables) with nonempty domains, $S = \{\mathrm{Dom}[g_i] \mid i \in [1, k]\}$ is the set of domains for each group, and $C = \{C_{ij} \mid i, j \in [1, k], i \neq j\}$ is the set of constraints.

A node initiates the conflict resolution procedure upon identifying a group–group conflict; e.g., when it intercepts a *queryrelevance* message from multiple groups, or when it already belongs to a group and it receives a *queryrelevance* message from another group. The conflict resolution procedure begins by *centralizing* the CSP in one of the supervisor nodes, which uses *backtracking* to solve the problem. The result is then conveyed to the other supervisor nodes.

Fig. 14 shows a camera network consisting of three cameras. The camera network is assigned two tasks: 1) observe Pedestrian 1 with at least two cameras and 2) observe Pedestrian 2 with one or more cameras.

Pedestrian 1 is visible in Cameras 1 and 3, and Pedestrian 2 is visible in all three cameras. Treating each task (or group) as a variable, we cast camera assignment as a CSP. The valid camera assignments listed in Fig. 14 define the domain of the two variables. The domain of each task (or group) is the powerset of the set of cameras that can carry out the task (i.e., that can be a member of the corresponding group). Since each camera can carry out only one task (or be a member of only one group) at any given time, a valid camera assignment will not assign any camera to more than one task (group). We express this restriction as a binary constraint by enforcing the intersection of the set of cameras assigned to any two tasks to be the null set. In the above scenario, Cameras 1 and 3 are assigned to Pedestrian 1 and Camera 2 is assigned to Pedestrian 2 (highlighted rows in Fig. 14).

CSPs have been studied extensively in the computer science literature and there exist several methods for solving them. We employ *backtracking* to search systematically through the space of possibilities in order to find an optimal camera assignment. The naive backtracking method, which we denote *AllSolv*, enumerates every solution in order to find the best solution. Instead, we store the currently best result and backtrack whenever the current partial solution is of poorer quality. We call this method *BestSolv*. Using this strategy, we can guarantee an optimal solution under the assumption that the quality of solutions increase monotonically as values are assigned to more variables. When P does not have a solution, we solve smaller CSPs by relaxing the node requirements for each task.

Table 1 compares our method (*BestSolv*) with naive backtracking (*AllSolv*). The problem is to assign three sensors each to two groups. The average number of relevant nodes for each group is 12 and 16. *AllSolv* finds all the solutions, ranks them, and picks the best one, whereas *BestSolv* computes the optimal solution by storing the currently best solution and backtracking when partial assignment yields a poorer solution. As expected, the *BestSolv* solver outperforms the *AllSolv* solver. Typically, *BestSolv* will outperform *AllSolv*, but equally importantly, *BestSolv* cannot do worse than *AllSolv*. Note that *AllSolv* and *BestSolv* explore the same solution space, so in the worst case both schemes will do the same amount of work. Typically, however, *BestSolv* can backtrack on partial solutions, thereby saving a potentially exponential amount of work.

Table 1 Finding an Optimal Sensor Node Assignment

Test case	1	2	3	4
Solver used	*AllSolv*	*BestSolv*	*AllSolv*	*BestSolv*
Number of groups	2	2	2	2
Number of sensors per group	3	3	3	3
Avg. number of relevant sensors	12	12	16	16
Average domain size	220	220	560	560
Number of solutions	29290	9	221347	17
Number of nodes explored	29511	175	221908	401
Number of Backtracks	48620	36520	314160	215040

Assumptions: Nodes n and m are leader nodes performing Task 1.

Case 1: Node n receives a *queryrelevance* or *status* message from node m.
if Node n is not in demotion negotiations with another node, **then** send a *demote* message to node m after a random interval.

Case 2: Node n receives a *demote* message from node m.
1) **if** Node n has not sent a *demote* message to another node, **then** demote node n and send a *demoteack* message to node m.
2) **if** Node n has sent a *demote* message to node m, **then** send a *demoteretry* message to node m and send a *demote* message to node m after a random interval.
3) **if** Node n has sent a *demote* message to another node, **then** send a *demotenack* message to node m.

Case 3: Node n receives a *demotenack* message from node m.
Terminate demotion negotiations with node m.

Case 4: Node n receives a *demoteack* message from node m.
Add node m to node n's group.

Case 5: Node n receives a *demoteretry* message from node m.
Send a *demote* message to node m after a random interval.

Fig. 15. ***Group merging via leader demotion.***

A key feature of our proposed conflict resolution method is centralization, which requires that all the relevant information be gathered at the node that is responsible for solving the CSP. For smaller CSPs, the cost of centralization is easily offset by the speed and ease of solving the CSP. One can perhaps avoid centralization by using a distributed constraint satisfaction scheme [40].

C. Node Failures and Communication Errors

The proposed communication model takes into consideration node and communication failures. Communication failures are perceived as camera failures. In particular, when a node is expecting a message from another node, and the message never arrives, the first node concludes that the second node is malfunctioning. A node failure is assumed when the supervisor node does not receive the node's *status* messages within a set time limit, and the supervisor node removes the problem node from the group. Conversely, when a member node fails to receive a *status* message from the supervisor node within a set time limit, it assumes that the supervisor node has experienced a failure and selects itself to be the supervisor of the group. An actual or perceived supervisor node failure can therefore give rise to multiple single-node groups performing the same task.

Multiple groups assigned to the same task are merged by demoting all the supervisor nodes of the constituent groups except one. Demotion is either carried out based upon the unique ID assigned to each node—among the conflicting nodes, the one with the highest ID is selected to be the group leader—or when unique node IDs are not guaranteed, demotion can be carried out via the procedure in Fig. 15. The following observations suggest that our leader demotion strategy is correct in the sense that only a

single leader node survives the demotion negotiations and every other leader node is demoted: 1) The demotion process for more than two nodes involves repeated (distributed and parallel) application of the demotion process between two nodes. 2) The demotion process between two leader nodes either succeeds or fails. It succeeds when one of the two nodes is demoted. Demotion between two nodes is based on the contention management scheme that was first introduced in the ALOHA network protocol [48], which was developed in the late 1960s and was a precursor to the ubiquitous Ethernet protocol (see [49] for the details). In its basic version, the ALOHA protocol states the following.

- If you have data to send, send it.
- If there is a collision, resend after a random interval.

The important thing to note here is that the demotion process between two nodes will eventually succeed and one of the two leader nodes will be demoted.

V. PERSISTENT SURVEILLANCE

Consider how a network of smart cameras may be used in the context of video surveillance (Fig. 16). Any two camera nodes that are within communication range of each other are considered neighbors. A direct consequence of this approach is that the network can easily be modified through removal, addition, or replacement of camera nodes.

A human operator spots one or more mobile pedestrians of interest in a video feed and, for example, requests the network to "zoom in on this pedestrian," "observe this pedestrian," or "observe the entire group." The successful execution and completion of these tasks requires an intelligent allocation of the available cameras. In particular, the network must decide which cameras should track the pedestrian and for how long.

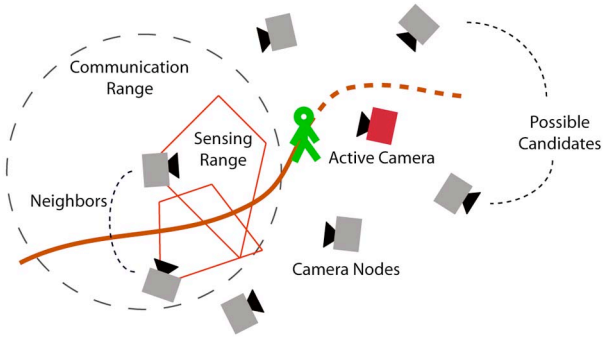


Fig. 16. *A camera network for video surveillance consists of camera nodes that can communicate with other nearby nodes. Collaborative, persistent surveillance requires that cameras organize themselves to perform camera handover when the observed subject moves out of the sensing range of one camera and into that of another.*

Table 2 Camera Network Simulation Parameters for Figs. 17–19

High-level camera controller parameters	
Wait state timeout	2 min
Search state timeout	5 min
At-node processing parameters	
Minimum acceptable relevance value	0.1
Heartbeat interval (inter-group *status* messages)	1000 ms
Group leader failure timeout	3000 ms
Group member failure timeout	5000 ms
Relevance threshold for task auction	0.5
Minimum duration for an assigned task	1 min
Minimum relevance threshold for group membership	0.1
Group merge timeout	5 min
Bidding timeout	2 min
Network communication parameters	
Probability of message loss	0.1
Minimum message delay	200 ms
Maximum message delay	500 ms
Probability of camera node failure	0.001
Maximum number of hops	3
Message timeout	1500 ms

The accuracy with which individual camera nodes are able to compute their relevance to the task at hand determines the overall performance of the network (see Section III-D and [46] for the details). The computed relevance values are used by the node selection scheme described above to assign cameras to various tasks. The supervisor node gives preference to the nodes that are currently free, so the nodes that are part of another group are selected only when an insufficient number of free nodes are available for the current task.

A detailed world model that includes the location of cameras, their fields of view, pedestrian motion prediction

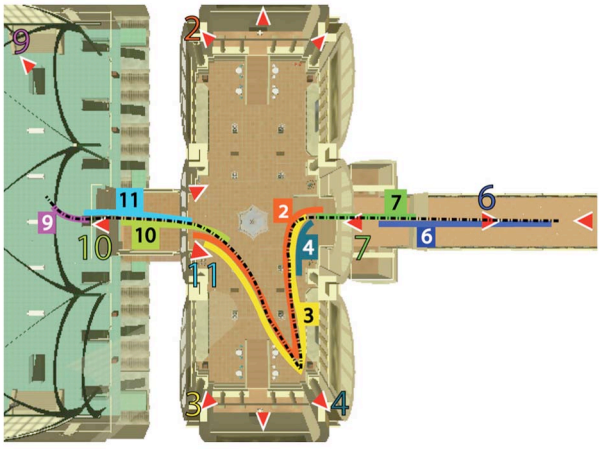


Fig. 17. *A pedestrian of interest walking through the train station for 15 min is automatically observed successively by Cameras 7, 6, 2, 3, 10, and 9 (refer to Fig. 1) as she makes her way from the arcade through the main waiting room and into the concourse. The dashed contour shows the pedestrian's path. The camera numbers are color coded and the portion of the path walked while the pedestrian is being observed by a particular camera is highlighted with the associated color.*

models, occlusion models, and pedestrian movement pathways may allow (in some sense) optimal allocation of camera resources; however, it is cumbersome and in most cases infeasible to acquire such a world model. Our approach eschews such detailed knowledge. We assume only that a pedestrian can be identified with reasonable accuracy by the camera nodes.

A. Results

To date, we have simulated our smart camera network with up to 16 stationary and/or PTZ virtual cameras in the virtual train station populated with up to 100 autonomous pedestrians, with network simulation parameters per Table 2.

For the 15-min simulation illustrated in Figs. 17 and 18, with 16 active PTZ cameras in the train station as indicated in Fig. 1, an operator selects the female pedestrian with the red top visible in Camera 7 [Fig. 18(e)] and initiates an *observe* task. Camera 7 forms a task group and begins tracking the pedestrian. Subsequently, Camera 7 recruits Camera 6, which in turn recruits Cameras 2 and 3 to observe the pedestrian. Camera 6 becomes the supervisor of the group when Camera 7 loses track of the pedestrian and leaves the group. Subsequently, Camera 6 experiences a tracking failure, sets Camera 3 as the group supervisor, and leaves the group. Cameras 2 and 3 persistently observe the pedestrian during her stay in the main waiting room, where she also visits a vending machine. When the pedestrian enters the portal connecting the main waiting room to the concourse, Cameras 10 and 11 are recruited and they take over the group from Cameras 2 and 3.

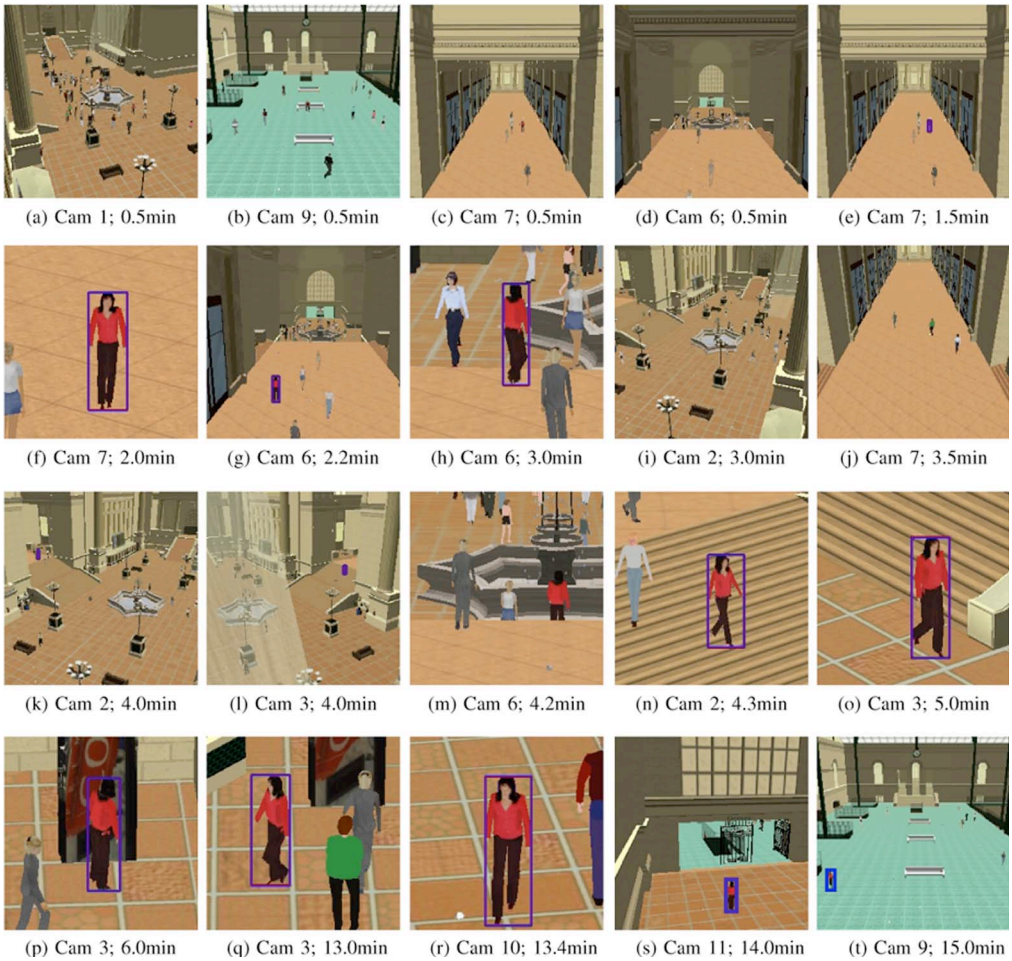


Fig. 18. *15-min persistent observation of a pedestrian of interest as she makes her way through the train station (refer to Fig. 17). (a)–(d) Cameras 1, 9, 7, and 8 monitoring the station. (e) The operator selects a pedestrian of interest in the video feed from Camera 7. (f) Camera 7 has zoomed in on the pedestrian, (g) Camera 6, which is recruited by Camera 7, acquires the pedestrian. (h) Camera 6 zooms in on the pedestrian. (i) Camera 2. (j) Camera 7 reverts to its default mode after losing track of the pedestrian and is now ready for another task. (k) Camera 2, which is recruited by Camera 6, acquires the pedestrian. (l) Camera 3 is recruited by Camera 6; Camera 3 has acquired the pedestrian. (m) Camera 6 has lost track of the pedestrian. (n) Camera 2 observing the pedestrian. (o) Camera 3 zooming in on the pedestrian. (p) Pedestrian is at the vending machine. (q) Pedestrian is walking towards the concourse. (r) Camera 10 is recruited by Camera 3; Camera 10 is observing the pedestrian. (s) Camera 11 is recruited by Camera 10. (t) Camera 9 is recruited by Camera 10.*

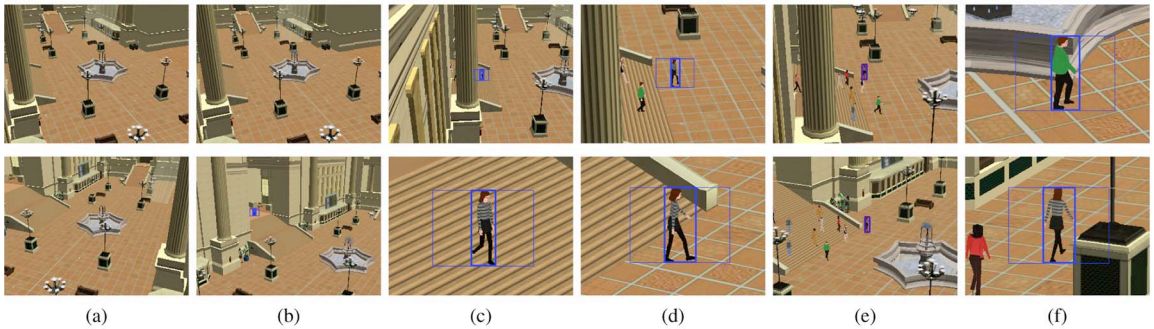


Fig. 19. *Camera assignment and conflict resolution. (a) Camera 1 (top row) and Camera 2 (bottom row) observe the main waiting room. (b) Camera 2 starts observing a pedestrian as soon as she enters the scene. (c)–(d) Camera 1 recognizes the target pedestrian by using the pedestrian signature computed by Camera 2. Cameras 1 and 2 form a group to observe the first pedestrian. (e) The operator issues a second task for the camera network, to observe the pedestrian wearing green. The two cameras pan out to search for the latter. They decide between them which will carry out each of the two tasks. (f) Camera 1 is better suited to observing the pedestrian in the green top while Camera 2 continues observing the original pedestrian.*

Cameras 2 and 3 leave the group and return to their default states. Later, Camera 11, which is now acting as the group's supervisor, recruits Camera 9, which observes the pedestrian as she enters the concourse.

Fig. 19 illustrates camera assignment and conflict resolution. First, Cameras 1 and 2 situated in the main waiting room successfully form a group to observe the first pedestrian that enters the scene, and there is only one active task. When the user specifies a second task—follow the pedestrian wearing the green top—the cameras decide to break the group and reassign themselves. They decide among themselves that Camera 1 is more suitable for observing the pedestrian in the green top. Camera 2 continues observing the first pedestrian that entered the scene. Note that the cameras are able to handle the two observation tasks completely autonomously and also that the interaction between them is strictly local—the other 14 cameras present in the network (Fig. 1) are not involved.

We have observed that the camera network correctly assigns cameras in most cases. The problems that we encountered are usually related to pedestrian identification and tracking. The task of persistently observing a pedestrian moving through an extensive space will fail if the low-level visual analysis routines cannot correctly identify the pedestrian from camera to camera. As we increase the number of virtual pedestrians in the train station, the identification and tracking module experiences increasing difficulty, which increases the chances that the persistent surveillance task will fail. Note that in the absence of global 3-D information from calibrated cameras, our proposed approach is unable to assist the low-level visual analysis routines. Similarly, the high-level task has no way of knowing if the visual analysis routines are performing satisfactorily. While it is beyond the scope of our current model, information about the 3-D location of pedestrians (which our simulator can readily provide) can be utilized to detect pedestrian identification errors. More sophisticated visual analysis routines should be developed to improve pedestrian identification in multiple cameras. We refer the reader to [46] for a more detailed discussion.

VI. CONCLUSION

We envision future video surveillance systems to be networks of stationary and active cameras capable of maintaining extensive urban environments under persistent surveillance with minimal reliance on human operators. Such systems will require not only robust, low-level vision routines, but also new camera network methodologies. The work presented in this paper is a step toward the realization of such smart camera networks and our initial results appear promising.

The overall behavior of our prototype smart camera network is governed by local decision making at each node and communication between the nodes. Our approach is novel insofar as it does not require camera calibration, a detailed world model, or a central controller. We have intentionally avoided multicamera tracking schemes that assume prior camera network calibration which, we believe, is an unrealistic goal for a large-scale camera network consisting of heterogeneous cameras. Similarly, our approach does not expect a detailed world model which, in general, is hard to acquire. Since it lacks any central controller, we expect our approach to be robust and scalable.

A unique and important aspect of our work is that we have developed and demonstrated our prototype video surveillance system in virtual reality—a realistic train station environment populated by lifelike, autonomously self-animating virtual pedestrians. Our sophisticated camera network simulator should continue to facilitate our ability to design such large-scale networks and conveniently experiment with them on commodity personal computers.

We are currently experimenting with more elaborate scenarios involving multiple cameras situated in different locations within the train station, with which we would like to study the performance of the network in persistently observing multiple pedestrians during their extended presence in the train station. ■

Acknowledgment

The authors wish to thank W. Shao for developing and implementing the train station simulator and M. Plaza-Villegas for his valuable contributions. They also wish to thank T. Strat, formerly of DARPA, for his generous support and encouragement.

REFERENCES

[1] W. Shao and D. Terzopoulos, "Autonomous pedestrians," *Graph. Models*, vol. 69, no. 5–6, pp. 246–274, Sep./Nov. 2007.

[2] W. H. Widen, "Smart cameras and the right to privacy," *Proc. IEEE*, vol. 96, no. 10, Oct. 2008.

[3] C. J. Costello, C. P. Diehl, A. Banerjee, and H. Fisher, "Scheduling an active camera to observe people," in *Proc. ACM Int. Workshop on Video Surveillance and Sensor Networks*, New York, 2004, pp. 39–45.

[4] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich, "Collaborative signal and information processing: An information directed approach," *Proc. IEEE*, vol. 91, no. 8, pp. 1199–1209, Aug. 2003.

[5] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, Feb. 2003.

[6] F. Qureshi and D. Terzopoulos, "Virtual vision and smart cameras," in *Proc. 1st ACM/IEEE Int. Conf. Distributed Smart Cameras*, Vienna, Austria, Sep. 2007, pp. 87–94.

[7] F. Qureshi and D. Terzopoulos, "Surveillance camera scheduling: A virtual vision approach," *ACM Multimedia Syst. J.*, vol. 12, pp. 269–283, Dec. 2006.

[8] D. Terzopoulos and T. Rabie, "Animat vision: Active vision in artificial animals," *Videre: J. Comp. Vision Res.*, vol. 1, no. 1, pp. 2–19, Sep. 1997.

[9] T. Rabie and D. Terzopoulos, "Active perception in virtual humans," in *Vision Interface*, Montreal, QC, Canada, May 2000, pp. 16–22.

[10] T. Rabie, A. Shalaby, B. Abdulhai, and A. El-Rabbany, "Mobile vision-based vehicle tracking and traffic control," in *Proc. IEEE Int. Conf. Intelligent Transportation Systems*, Singapore, Sep. 2002, pp. 13–18.

[11] D. Terzopoulos, "Perceptive agents and systems in virtual reality," in *Proc. ACM Symp. Virtual Reality Software and Technology*, Osaka, Japan, Oct. 2003, pp. 1–3.

[12] A. Santuari, O. Lanz, and R. Brunelli, "Synthetic movies for computer vision applications," in *Proc. IASTED Int. Conf. Visualization, Imaging, and Image Processing*, pp. 1–6.

[13] F. Bertamini, R. Brunelli, O. Lanz, A. Roat, A. Santuari, F. Tobia, and Q. Xu, "Olympus: An ambient intelligence architecture on the verge of reality," in *Proc. Int. Conf. Image Analysis and Processing*, Mantova, Italy, Sep. 2003, pp. 139–145.

[14] R. Collins, O. Amidi, and T. Kanade, "An active camera system for acquiring multi-view video," in *Proc. Int. Conf. Image Processing*, Rochester, NY, Sep. 2002, pp. 517–520.

[15] D. Comaniciu, F. Berton, and V. Ramesh, "Adaptive resolution system for distributed surveillance," *Real Time Imag.*, vol. 8, no. 5, pp. 427–437, Oct. 2002.

[16] M. Trivedi, K. Huang, and I. Mikic, "Intelligent environments and active camera networks," in *Proc. IEEE Int. Conf. Systems, Man and Cybernetics*, Nashville, TN, Oct. 2000, vol. 2, pp. 804–809.

[17] S. Stillman, R. Tanawongsuwan, and I. Essa, "A System for Tracking and Recognizing Multiple People With Multiple Cameras," Georgia Institute of Technology, GVU Center, 1998, Tech. Rep. GIT-GVU-98-25.

[18] O. Javed, Z. Rasheed, O. Alatas, and M. Shah, "Knight: A real time surveillance system for multiple and non-overlapping cameras," in *Proc. Int. Conf. Multimedia and Expo*, Baltimore, MD, Jul. 2003, vol. 1, pp. 649–652.

[19] S. Khan and M. Shah, "Consistent labeling of tracked objects in multiple cameras with overlapping fields of view," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 10, pp. 1355–1360, Oct. 2003.

[20] J. Kang, I. Cohen, and G. Medioni, "Multi-view tracking within and across uncalibrated camera streams," in *Proc. ACM SIGMM Int. Workshop on Video Surveillance*, New York, 2003, pp. 21–33.

[21] N. T. Siebel, "Designing and implementing people tracking applications for automated visual surveillance," Ph.D. dissertation, Dept. Comp. Sci., Univ. Reading, Reading, U.K., 2003.

[22] F. Pedersini, A. Sarti, and S. Tubaro, "Accurate and simple geometric calibration of multi-camera systems," *Signal Process.*, vol. 77, no. 3, pp. 309–334, 1999.

[23] T. Gandhi and M. M. Trivedi, "Calibration of a reconfigurable array of omnidirectional cameras using a moving person," in *Proc. ACM Int. Workshop on Video Surveillance and Sensor Networks*, New York, , 2004, pp. 12–19.

[24] D. Devarajan, R. J. Radke, and H. Chung, "Distributed metric calibration of ad hoc camera networks," *ACM Trans. Sensor Netw.*, vol. 2, no. 3, pp. 380–403, 2006.

[25] A. Ihler, J. Fisher, R. Moses, and A. Willsky, "Nonparametric belief propagation for self-calibration in sensor networks," in *Proc. Int. Symp. Information Processing in Sensor Networks*, Berkeley, CA, Apr. 2004, pp. 225–233.

[26] D. Marinakis, G. Dudek, and D. Fleet, "Learning sensor network topology through Monte Carlo expectation maximization," in *Proc. IEEE Intl. Conf. Robotics and Automation*, Barcelona, Spain, Apr. 2005, pp. 4581–4587.

[27] J. Mallett, "The role of groups in smart camera networks," Ph.D. dissertation, Program of Media Arts and Sciences, School of Architecture, MIT, Cambridge, MA, 2006.

[28] R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multisensor surveillance," *Proc. IEEE*, vol. 89, no. 10, pp. 1456–1477, Oct. 2001.

[29] X. Zhou, R. T. Collins, T. Kanade, and P. Metes, "A master–slave system to acquire biometric imagery of humans at distance," in *Proc. ACM SIGMM Inte. Workshop on Video Surveillance*, New York, 2003, pp. 113–120.

[30] A. Hampapur, S. Pankanti, A. Senior, Y.-L. Tian, L. Brown, and R. Bolle, "Face cataloger: Multi-scale imaging for relating identity to location," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, Washington, DC, 2003, pp. 13–21.

[31] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Process. Mag.*, vol. 19, pp. 61–72, Mar. 2002.

[32] M. Bhardwaj, A. Chandrakasan, and T. Garnett, "Upper bounds on the lifetime of sensor networks," in *Proc. IEEE Int. Conf. Communications*, 2001, pp. 785–790.

[33] J. Campbell, P. B. Gibbons, S. Nath, P. Pillai, S. Seshan, and R. Sukthankar, "Irisnet: An internet-scale architecture for multimedia sensors," in *Proc. ACM Int. Conf. Multimedia*, New York, 2005, pp. 81–88.

[34] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu, "Senseye: A multi-tier camera sensor network," in *Proc. ACM Int. Conf. Multimedia*, New York, 2005, pp. 229–238.

[35] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. C-29, no. 12, pp. 1104–1113, Dec. 1980.

[36] B. Gerkey and M. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robotics Res.*, vol. 23, no. 9, pp. 939–954, 2004.

[37] M. R. Garey and D. S. Johnson, "Strong NP-completeness results: Motivation, examples, and implications," *J. ACM*, vol. 25, no. 3, pp. 499–508, 1978.

[38] A. Atamturk, G. Nemhauser, and M. Savelsbergh, "A combined lagrangian, linear programming and implication heuristic for large-scale set partitioning problems," *J. Heuristics*, vol. 1, pp. 247–259, 1995.

[39] P. J. Modi, W.-S. Shen, M. Tambe, and M. Yokoo, "Adopt: Asynchronous distributed constraint optimization with quality guarantees," *Artificial Intell.*, vol. 161, no. 1–2, pp. 149–180, Mar. 2006.

[40] M. Yokoo, *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent Systems*. Berlin, Germany: Springer-Verlag, 2001.

[41] J. D. Foley, A. van Dam, S. Feiner, and J. K. Hughes, *Computer Graphics: Principles and Practice*. Boston, MA: Addison-Wesley, 1990.

[42] F. Birren, *Color Perception in Art.*. New York: Van Nostrand Reinhold, 1976.

[43] G. D. Haan and E. B. Bellers, "Deinterlacing—An overview," *Proc. IEEE*, vol. 86, no. 9, pp. 1839–1857, Sep. 1998.

[44] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects

using mean shift," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Hilton Head Island, SC, Jun. 2000, vol. 2, pp. 142–151.

[45] M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comp. Vision*, vol. 7, no. 1, pp. 11–32, Nov. 1991.

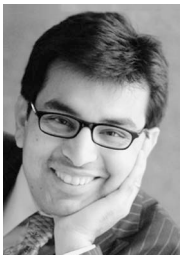
[46] F. Z. Qureshi, "Intelligent perception in virtual camera networks and space robotics," Ph.D. dissertation, Dept. Comp. Science, Univ. Toronto, Toronto, ON, Canada, 2007.

[47] J. K. Pearson and P. G. Jeavons, "A Survey of Tractable Constraint Satisfaction Problems," University of London, Royal Holloway, 1997, Tech. Rep. CSD-TR-97-15.

[48] F. F. Kuo, "The ALOHA system," *ACM SIGCOMM Computer Commun. Revi.*, vol. 25, no. 1, pp. 41–44, 1995.

[49] C. Murthy and B. Manoj, *Ad Hoc Wireless Networks Architectures and Protocols*. Upper Saddle River, NJ: Prentice-Hall, 2004.

ABOUT THE AUTHORS



Faisal Qureshi (Member, IEEE) received the B.Sc. degree in Mathematics and Physics from Punjab University, Lahore, Pakistan, in 1993, the M.Sc. degree in Electronics from Quaid-e-Azam University, Islamabad, Pakistan, in 1995, and the M.Sc. and Ph.D. degrees in Computer Science from the University of Toronto, Toronto, Canada, in 2000 and 2007, respectively.

He is an Assistant Professor of Computer Science at the University of Ontario Institute of Technology (UOIT), Oshawa, Canada. His research interests include sensor networks, computer vision, and computer graphics. He has also published papers in space robotics. He has interned at ATR Labs (Kyoto, Japan), AT&T Research Labs (Red Bank, NJ), and MDA Space Missions (Brampton, Canada). He joined the UOIT in 2008 from Autodesk Canada Co. in Toronto, where he was a Software Developer on the AutoStudio modeling team.

Dr. Qureshi is a member of the ACM.

Demetri Terzopoulos (Fellow, IEEE) received the B.Eng. and M.Eng. degrees in Electrical Engineering from McGill University in 1978 and 1980, respectively, and the Ph.D. degree in Artificial Intelligence from the Massachusetts Institute of Technology (MIT) in 1984.

He is the Chancellor's Professor of Computer Science at the University of California, Los Angeles. He joined UCLA in 2005 from New York University, where he held the Lucy and Henry Moses Professorship in Science and was Professor of Computer Science and Mathematics at NYU's Courant Institute. He was also Professor of Computer Science and Professor of Electrical and Computer Engineering at the University of Toronto, where he currently retains status-only faculty appointments. Prior to joining Toronto in 1989, he was a program leader at Schlumberger corporate research centers in Palo Alto, California, and Austin, Texas. In 1984–85 he was a research scientist at the MIT Artificial Intelligence Lab. He has been a Visiting Professor at the University of Paris and at IBM, Intel, Schlumberger, Digital, and other corporations.

Prof. Terzopoulos is a Fellow of the IEEE, a Fellow of the ACM, a Fellow of the Royal Society of Canada, and a member of the European Academy of Sciences. His many awards and honors include an Academy Award for Technical Achievement from the Academy of Motion Picture Arts and Sciences for his pioneering work on physics-based computer animation, and the inaugural Computer Vision Significant Researcher Award from the IEEE for his pioneering and sustained research on deformable models and their applications. He is listed by ISI and other citation indexes as one of the most highly cited authors in engineering and computer science, with more than 300 published research papers and several volumes, primarily in computer graphics, computer vision, medical imaging, computer-aided design, and artificial intelligence/life. His published work has received about a dozen outstanding paper recognitions from *Computers and Graphics*, IEEE, AAAI, ACM, AMPAS, NICOGRAPH, MICCAI, IMIA, SAE, and other organizations. He has delivered over 400 invited talks around the world, including more than 80 distinguished lectures and keynote or plenary addresses. While at Toronto, he held three of Canada's most distinguished research fellowships (Killam, NSERC Steacie, CIFAR) and received six Faculty of Arts and Science Excellence Awards, as well as awards from Ars Electronica and the International Digital Media Foundation, plus a citation from the Canadian Image Processing and Pattern Recognition Society for his "outstanding contributions to research and education in Image Understanding." The PhD thesis of one of his Toronto students (Tu) won the 1996 ACM Doctoral Dissertation Award.

Prof Terzopoulos serves as a series editor of Springer-Verlag's *Lecture Notes in Computer Science* and is or has been a founding member of the editorial boards of 7 journals spanning vision, graphics, medical imaging, and applied math. He serves on the Presidential Scientific Advisory Board of the Max Plank Institute for Informatics in Germany and has served on DARPA, NIH, NSF, and NSERC-Canada advisory committees. He has served on the program committees of all the major conferences in his fields of expertise, and was a program area chair of ICCV 2007, a program chair of CVPR 1998 and Pacific Graphics 2004, and a conference chair of the 2005 ACM SIGGRAPH/EG Symposium on Computer Animation. He is a member of the New York Academy of Sciences and Sigma Xi.