SMART CAMERA NETWORKS IN VIRTUAL REALITY

Faisal Qureshi

Department of Computer Science
University of Toronto
Toronto, ON, Canada

Demetri Terzopoulos

Computer Science Department
University of California
Los Angeles, CA, USA

ABSTRACT

This paper presents smart camera network research in the context of a unique new synthesis of advanced computer graphics and vision simulation technologies. We design and experiment with simulated camera networks within visually and behaviorally realistic virtual environments. Specifically, we demonstrate a smart camera network comprising static and active simulated video surveillance cameras that provides perceptive coverage of a large virtual public space, a train station populated by autonomously self-animating virtual pedestrians. In the context of human surveillance, we propose a camera network control strategy that enables a collection of smart cameras to provide perceptive scene coverage and perform persistent surveillance with minimal intervention. Our novel control strategy naturally addresses camera aggregation and camera handoff, it does not require camera calibration, a detailed world model, or a central controller, and it is robust against camera failures and communication errors.

Index Terms— Virtual Vision, Computer Vision, Persistent Surveillance, Smart Cameras, Camera Networks, Multi-Camera Coordination and Control

1. INTRODUCTION

Recently video surveillance has found numerous applications in search and rescue operations, crime fighting, environment monitoring, traffic and city management, urban sensing, etc. Multi-camera systems, or camera networks, are a critical component of any video surveillance infrastructure. As the size of the network grows, it becomes infeasible for human operators to monitor the numerous video streams and identify all events of possible interest. Therefore, it is desirable to have camera networks capable of carrying out sensing operations autonomously or with minimal human intervention. In this paper, we demonstrate a model 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).

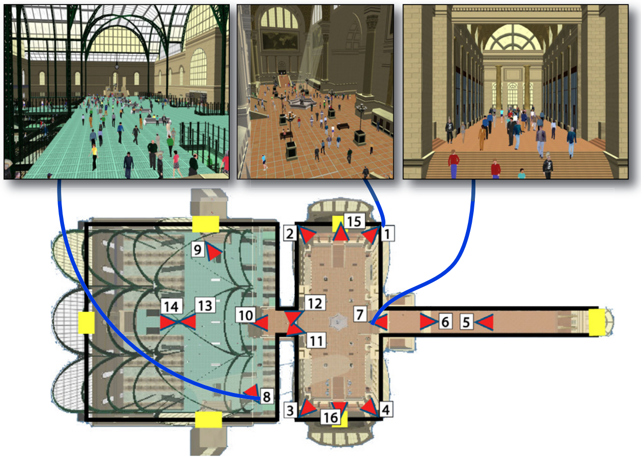


Figure 1: Plan view of the (roofless) virtual Penn Station environment, revealing the concourses and train tracks (left), the main waiting room (center), and the shopping arcade (right). (The yellow rectangles indicate pedestrian portals.) An example camera network is illustrated, comprising 16 simulated active (pan-tilt-zoom) video surveillance cameras. Synthetic images from simulated video surveillance cameras 1, 7, and 8 (from [1]).

Once a pedestrian of interest is selected either automatically or by an operator monitoring surveillance video feeds, the cameras decide amongst themselves how best to observe the subject. For example, a subset of the active pan/tilt/zoom (PTZ) cameras can collaboratively track the pedestrian as (s)he weaves through the crowd. The problem of assigning cameras to follow pedestrians becomes challenging when multiple pedestrians are involved. To deal with the myriad of 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 dynamic, task-driven node aggregation through local decision-making and internode communication.[1] The work presented here is an extension of that presented in [5].

[1] A node can communicate with 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 [2]. References [3, 4] present schemes for learning sensor network topologies.

1.1. Virtual Vision

This type of research would be very difficult to carry out in the real world given the cost of (and legal impediments to) deploying and experimenting with an appropriately complex camera network in a large public space the size of a train station. Despite its sophistication, our virtual vision simulator runs on high-end commodity PCs, obviating the need to grapple with special-purpose hardware and software (Fig. 2). The multiple *virtual* cameras, which generate synthetic video feeds that emulate those generated by real surveillance cameras, are very easily reconfigurable in the virtual space (Fig. 1). Moreover, the virtual world provides readily accessible ground-truth data for the purposes of camera network algorithm validation.

Skeptics might argue that virtual vision relies on simulated data, which can lead to inaccurate results. In particular, they may fret that virtual video lacks the subtleties of real video and that meaningful evaluation of a machine vision system is impossible in the absence of real video. However, our high-level camera control routines do not directly process any raw video. Instead, low-level recognition and tracking routines, which mimic the performance (including failure modes) of a state-of-the-art pedestrian localization and tracking system, generate realistic input data for the high-level routines. Hence, our simulator enables us to develop and test camera network control algorithms under realistic assumptions derived from physical camera networks. We believe that these algorithms will readily port to the real world.

An important issue in smart camera networks is how to compare camera network algorithms. Two possible approaches are: 1) time-shared physical camera networks and 2) realistic simulation environments. It is a matter of simple video capture to gather benchmark data from time-shared physical networks comprising passive, fixed-zoom cameras, but gathering benchmark data for networks comprising active PTZ cameras requires scene reenactment for every run, which is clearly infeasible in most cases. Costello et al. [6], who compared various schemes for scheduling an active camera to observe pedestrians present in the scene, ran into this issue and resort to Monte Carlo simulation to evaluate camera scheduling approaches. They conclude that evaluating scheduling policies on an physical testbed comprising a single active camera is extremely complicated. Our virtual vision approach provides a viable alternative that, among other benefits compared to a physical camera network, offers convenient and unlimited repeatability.

1.2. Smart Camera Network

Many of the characteristics and challenges associated with sensor networks are relevant to the work presented here. A fundamental issue in sensor networks is the selection of sensor nodes that participate in a particular sensing task [7]. 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 that are within radio communication range. Message delay and message loss are common occurrences in sensor networks due to bandwidth limitations, interference, etc. One must also account for nonstationary network topology due to node failures, node additions, etc.

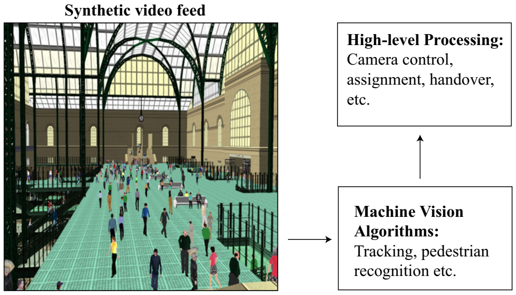

Synthetic video feed
High-level Processing:
Camera control, assignment, handover, etc.
Machine Vision Algorithms:
Tracking, pedestrian recognition etc.

Figure 2: The virtual vision paradigm (image from [1]).

Mindful of these issues, we propose a novel camera network control strategy that does not require camera calibration, 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 link failures. Moreover, it is scalable due to 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 group-group interactions.

2. RELATED WORK

As was argued in [8, 9], computer graphics and virtual reality technologies are rapidly presenting viable alternatives to the real world for developing sensory systems (see also [10]). Our camera network is deployed and tested within the virtual train station simulator that was developed in [1]. The simulator incorporates a large-scale environmental model (of the original Pennsylvania Station in New York City) with a sophisticated pedestrian animation system that combines 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. Standard computer graphics techniques render the busy urban scene with considerable geometric and photometric detail (Fig. 1).

The problem of forming sensor groups based on task requirements and resource availability has received much attention within the sensor networks community [7]. Reference [11] argues that task-based grouping in *ad hoc* camera networks is highly advantageous. Collaborative tracking, which subsumes the above issue, is considered an essential capability in many sensor networks [7]. Reference [12] introduces 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 vs. energy expenditure tradeoff estimates for its neighbor nodes. In the context of camera networks, it is often difficult for a camera node to estimate the expected information gain by assigning another camera to the task without explicit geometric and camera-calibration knowledge, and 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.

Nodes comprising sensor networks are usually untethered sensing units with limited onboard power reserves. Hence, a crucial concern in sensor networks is the energy expenditure at each node, which determines the lifespan of a sensor network [13]. Node communications have large power requirements; therefore, sensor network control strategies attempt to minimize the internode communication [12]. Presently, we do not address this issue; however, the communication protocol proposed here limits the communication to the active nodes and their neighbors.

Little attention has been paid in computer vision to the problem of controlling active cameras to provide visual coverage of an extensive public area, such as a train station or an airport [14, 6]. Previous work on camera networks in computer vision has dealt with issues related to low-level and mid-level vision, namely segmentation, tracking, and identification of moving objects [15], and camera network calibration [16]. 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.

IrisNet is a sensor network architecture tailored towards advanced sensors connected via high-capacity communication channels [17]. It takes a *centralized* view of the network and models it as a distributed database, allowing efficient access to sensor readings. We consider this work to be orthogonal to ours. SensEye is a recent sensor-network inspired multi-camera system [18]. It demonstrates the benefits in terms of low-latencies and energy efficiency of a multi-tiered 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 [19] and it realizes group formation via internode negotiation. Unlike Mallett's [11] 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 multi-agent systems community [20]. Specifically, according to the taxonomy provided in [20], our sensor assignment formulation belongs to the single-task (ST) robots, multi-robot (MR) tasks, instantaneous assignment (IA) category. ST-MR-IA problems are significantly more difficult than single robot task MTRA problems. Task-based robot grouping arise 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 [21]. However, heuristics-based set partitioning algorithms exist that produce good results on large SPPs [22]. Fortunately, the sizes of MRTA problems, and by extension SPPs, encountered in our camera sensor network setting are small due to the spatial or locality constraints inherent to the camera sensors.

We model sensor assignments as a Constraint Satisfaction Problem (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, such as [23] and [24], due to 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.

3. SMART CAMERA NODES

Each virtual camera node is an autonomous agent capable of communicating with nearby nodes.

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 as well as upon the information readily available from the 3D virtual world. They mimic the

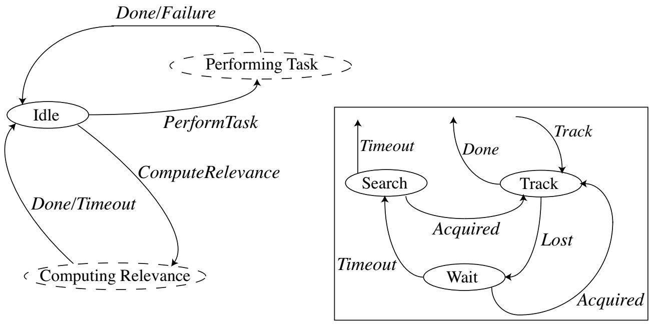


Figure 3: The top-level camera controller.

performance of a state-of-the-art pedestrian segmentation and tracking module. Each camera can *fixate* and *zoom* on an object of interest. The fixation and zooming routines are image driven and do not require any 3D information such as camera calibration or a global frame of reference. Refer to [25] for the details.

An augmented hierarchical finite state machine implements the top-level controller of the camera node (Fig. 3). The camera controller is responsible for the overall behavior of the camera node and it takes into account the information gathered through visual analysis (bottom-up) and the current task (top-down).

In its default state, *Idle*, the camera node is not involved in any task. A camera node transitions into the *ComputingRelevance* state upon receiving a *queryrelevance* message from a nearby node. Using the description of the task that is contained within the *queryrelevance* message, and by employing the LVRs, the camera node can compute its relevance to the task. For example, a camera 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 returns to the *Idle* state if it fails to compute the relevance due to the fact that it cannot find a pedestrian matching the description. On the other hand, when the camera successfully finds the desired pedestrian, it returns the 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 *PerformingTask* state upon joining a group, where the embedded child finite state machine 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. 3) that allow the camera node to transition into the default state rather than wait indefinitely for a message from another node.

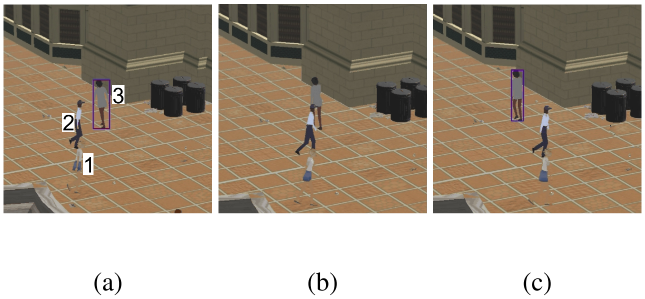


Figure 4: The LVRs are programmed to track pedestrians 1 and 3. Pedestrian 3 is tracked successfully; however, (a) track is lost of pedestrian 1 who blends in with the background. (b) The tracking routine loses pedestrian 3 when she is occluded by pedestrian 2, but it regains track of pedestrian 3 when pedestrian 2 moves out of the way (c).

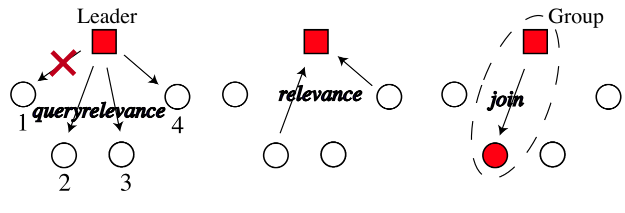
4. CAMERA NETWORK MODEL

The camera network communication scheme that enables task-specific node organization is as follows: An 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 formalizes the 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 [19] (Fig. 5). The local computation at each node involves choosing an appropriate bid for the announced sensing task.

From the standpoint of user interaction, we have identified two kinds of sensing queries: 1) where the queried camera itself can measure the phenomenon of interest—e.g., when a human operator selects a pedestrian to be tracked within a particular video feed—and 2) when the queried node might not be able to perform the required sensing and needs to route the query to other nodes. For instance, an operator can task the network to count the number of pedestrians wearing green tops. Currently, the network supports only the first type of query, which suffices for initiating collaborative tracking tasks.

4.1. 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



(a) Announcement (b) Bidding (c) Selection

Figure 5: Task auction supports coalition formation. The red cross indicates a lost message.

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 (tasks)—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. On the other hand, a conflict resolution mechanism is required when multiple groups vie for the same nodes. We present a scheme to handle this situation in the next section. 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 certain threshold. When a leader detects that its own relevance value for the current task is below the predefined threshold, it selects a new leader from among the member nodes. The group vanishes when the last node leaves.

4.2. 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) [26]. Formally, a CSP consists of a set of variables $\{v_1, v_2, v_3, \cdots, v_k\}$, a set of allowed values $\text{Dom}[v_i]$ for each variable v_i (called the domain of v_i), and a set of constraints $\{C_1, C_2, C_3, \cdots, C_m\}$. The solution to the CSP is a set $\{v_i \leftarrow a_i \,|\, a_i \in \text{Dom}[v_i]\}$, where the a_is satisfy all the constraints.

We treat each group g as a variable, whose domain consists of the non-empty subsets of the set of cameras with relevance values (with respect to 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 problem $P = (G, D, C)$, where $G = \{g_1, g_2, \cdots, g_k\}$ is the set of groups (variables) with non-empty domains, $S = \{\text{Dom}[g_i] \,|\, i \in [1, k]\}$ is the set of domains for each group, and $C = \{C_{ij} \,|\, i, j \in [1, k], i \neq j\}$ 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.

CSPs have been studied extensively in the computer science literature and there exist many schemes for solving CSPs. We employ *backtracking* to search systematically through the space of possibilities in order to find an optimal camera assignment. We store the currently best result and backtrack whenever the current partial solution is of poorer quality. 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 (Table 1). When P does not have a solution, we solve smaller CSPs by relaxing the node requirements for each task.

A key feature of our proposed conflict resolution scheme 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 scheme for distributed CSPs [24].

[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 duration during which at least one *status* message must be received by the node in question.

Test cases	1	2	3	4
Number of groups	2	2	2	2
Number of sensors per group	3	3	3	3
Average 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
Solver used	*AllSolu*	*BestSolu*	*AllSolu*	*BestSolu*

Table 1: Finding an optimal sensor node assignment. We compare our scheme (*BestSolu*) with naive backtracking (*AllSolu*). The problem is to assign three sensors each to two groups. The average number of relevant nodes for each group is 12 and 16. *AllSolu* finds all solutions, ranks them, and picks the best one, whereas *BestSolu* computes the optimal solution by storing the currently best solution and backtracking when partial assignment yields a poorer solution. As expected, the *BestSolu* solver outperforms the *AllSolu*solver.

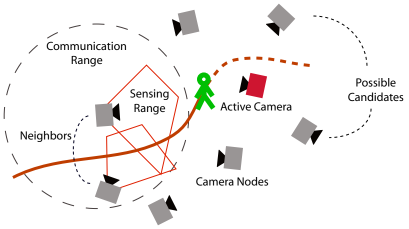


Figure 6: A camera network for video surveillance consists of camera nodes that can communicate with other nearby nodes. Collaborative, persistent surveillance requires that cameras organize themselves to perform camera handover when the observed subject moves out of the sensing range of one camera and into that of another.

4.3. Node Failures and Communication Errors

The proposed communication model takes into consideration node and communication failures. Communication failures are perceived as camera failures. For example, 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 response to successive heartbeat messages, and the supervisor node removes the problem node from the group. Conversely, when a member node does not receive a heartbeat 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 a contention management scheme that was first introduced in ALOHA network [27]. See [25] for the details.

5. PERSISTENT VIDEO SURVEILLANCE

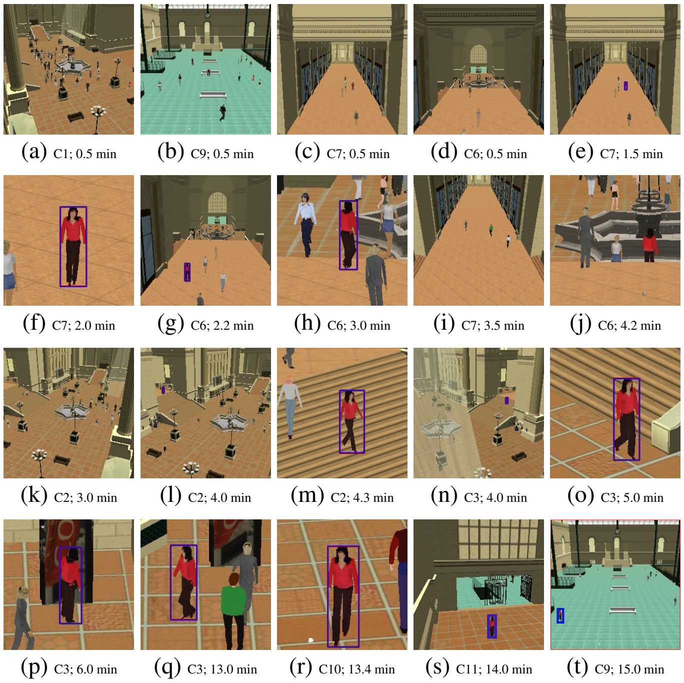
Now, consider how a network of dynamic cameras may be used in the context of video surveillance (Fig. 6). A human operator spots one or more mobile pedestrians of interest in a video feed and, for example, requests the network to "zoom in on this pedestrian," "observe this pedestrian," or "observe the 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.

A detailed world model that includes the location of cameras, their fields of view, pedestrian motion prediction models, occlusion models, and pedestrian movement pathways may allow (in some sense) *optimal* allocation of cameras; however, it is cumbersome and in most cases infeasible to acquire such a world model. Our approach eschews such a knowledge base. We assume only that a pedestrian can be identified by camera nodes with reasonable accuracy. Any two 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.

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 [25] 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.

6. RESULTS

To date, we have tested our smart camera network system with up to 16 stationary and pan-tilt-zoom virtual cameras, and we have populated the virtual Penn station with up to 100 pedestrians. The camera network correctly assigned cameras


(a) C1; 0.5 min (b) C9; 0.5 min (c) C7; 0.5 min (d) C6; 0.5 min (e) C7; 1.5 min

(f) C7; 2.0 min (g) C6; 2.2 min (h) C6; 3.0 min (i) C7; 3.5 min (j) C6; 4.2 min

(k) C2; 3.0 min (l) C2; 4.0 min (m) C2; 4.3 min (n) C3; 4.0 min (o) C3; 5.0 min

(p) C3; 6.0 min (q) C3; 13.0 min (r) C10; 13.4 min (s) C11; 14.0 min (t) C9; 15.0 min

Figure 7: For 15 minutes, a pedestrian of interest is successively observed by Cameras 7, 6, 2, 3, 10, and 9 (refer to Fig. 1) as she makes her way through the station from the arcade through the main waiting room and into the concourse. (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 7 reverts to its default mode after losing track of the pedestrian and is now ready for another task (j) Camera 6 has lost track of the pedestrian. (k) Camera 2. (l) Camera 2, which is recruited by Camera 6, acquires the pedestrian. (m) Camera 2 observing the pedestrian. (n) Camera 3 is recruited by Camera 6; Camera 3 has acquired 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.

in most cases. Some of the problems that we encountered are related to pedestrian identification and tracking. As we increase the number of virtual pedestrians in the train station, the identification and tracking module experiences increasing difficulty, so the surveillance task fails (and the cameras return to their default settings).

For the example in Fig. 7, we placed 16 active PTZ cameras in the train station as indicated in Fig. 1. An operator selects the pedestrian with the red top in Camera 7 (Fig. 7(e)) and initiates an “observe” task. Camera 7 forms the 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 starts walking towards the concourse, Cameras 10 and 11 take over the group from Cameras 2 and 3. Cameras 2 and 3 leave the group and return to their default modes. Later, Camera 11, which is now acting as the group’s supervisor, recruits Camera 9, which observes the pedestrian as she enters the concourse.

7. CONCLUSION

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

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 experiment with them on commodity personal computers.

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 new insofar as it does not require camera calibration, a detailed world model, or a central controller. We have intentionally avoided multi-camera 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 the proposed approach to be robust and scalable.

We are currently constructing 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 stay in the train station.

8. ACKNOWLEDGMENTS

The research reported herein was made possible in part by a grant from the Defense Advanced Research Projects Agency (DARPA) of the Department of Defense. We thank Tom Strat, formerly of DARPA, for his generous support and encouragement. We also thank

Wei Shao and Mauricio Plaza-Villegas for their invaluable contributions to the implementation of the train station simulator.

9. REFERENCES

[1] W. Shao and D. Terzopoulos, "Autonomous pedestrians," in *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, Los Angeles, CA, July 2005, pp. 19–28.

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

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

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

[5] F. Qureshi and D. Terzopoulos, "Virtual vision and smart camera networks," in *Working Notes of the International Workshop on Distributed Smart Cameras (DSC 2006)*, B. Rinner and W. Wolf, Eds., Boulder, CO, Oct. 2006, pp. 62–66, Held in conjunction with ACM SenSys 2006.

[6] C.J. Costello, C.P. Diehl, A. Banerjee, and H. Fisher, "Scheduling an active camera to observe people," in *Proc. 2nd ACM International Workshop on Video Surveillance and Sensor Networks*, New York, NY, 2004, pp. 39–45, ACM Press.

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

[8] D. Terzopoulos and T. Rabie, "Animat vision: Active vision in artificial animals," *Videre: Journal of Computer Vision Research*, vol. 1, no. 1, pp. 2–19, Sept. 1997.

[9] D. Terzopoulos, "Perceptive agents and systems in virtual reality," in *Proc. 10th ACM Symposium on Virtual Reality Software and Technology*, Osaka, Japan, Oct. 2003, pp. 1–3.

[10] A. Santuari, O. Lanz, and R. Brunelli, "Synthetic movies for computer vision applications," in *Proc. 3rd IASTED International Conference: Visualization, Imaging, and Image Processing (VIIP 2003)*, Spain, Sept. 2003, no. 1, pp. 1–6.

[11] J. Mallett, *The Role of Groups in Smart Camera Networks*, Ph.D. thesis, Program of Media Arts and Sciences, School of Architecture, Massachusetts Institute of Technology, Feb. 2006.

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

[13] M. Bhardwaj, A. Chandrakasan, and T. Garnett, "Upper bounds on the lifetime of sensor networks," in *IEEE International Conference on Communications*, 2001, no. 26, pp. 785–790.

[14] Robert Collins, Alan Lipton, Hironobu Fujiyoshi, and Takeo Kanade, "Algorithms for cooperative multisensor surveillance," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1456–1477, Oct. 2001.

[15] R. Collins, O. Amidi, and T. Kanade, "An active camera system for acquiring multi-view video," in *Proc. International Conference on Image Processing*, Rochester, NY, Sept. 2002, pp. 517–520.

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

[17] J. Campbell, P.B. Gibbons, S. Nath, P. Pillai, S. Seshan, and R. Sukthankar, "Irisnet: An internet-scale architecture for multimedia sensors," in *Proc. of the 13th annual ACM international conference on Multimedia*, New York, NY, 2005, pp. 81–88, ACM Press.

[18] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu, "Senseye: a multi-tier camera sensor network," in *Proc. of the 13th annual ACM international conference on Multimedia*, New York, NY, 2005, pp. 229–238, ACM Press.

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

[20] B. Gerkey and M. Matari, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

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

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

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

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

[25] F.Z. Qureshi, *Intelligent Perception in Virtual Camera Networks and Space Robotics*, Ph.D. thesis, Department of Computer Science, University of Toronto, Canada, January 2007.

[26] J.K. Pearson and P.G. Jeavons, "A survey of tractable constraint satisfaction problems," Tech. Rep. CSD-TR-97-15, Royal Holloway, University of London, July 1997.

[27] F.F. Kuo, "The ALOHA system," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 41–44, 1995,