Distributed Coalition Formation in Visual Sensor Networks

A Virtual Vision Approach

Faisal Z. Qureshi
Department of Computer Science
University of Toronto
Toronto, ON, Canada
faisal@cs.toronto.edu

Demetri Terzopoulos
Computer Science Department
University of California
Los Angeles, CA, USA
dt@cs.ucla.edu

ABSTRACT

In this paper, we propose a distributed coalition formation strategy for collaborative sensing tasks in camera sensor networks. The proposed model supports task-dependent node selection and aggregation through an announcement/bidding/selection strategy combined with a constraint satisfaction problem based conflict resolution mechanism. Our technique is scalable, as it lacks any central controller, and it is robust to node failures and imperfect communication. Another unique aspect of this work is that we demonstrate the advantages of visually and behaviorally realistic virtual environments as a viable simulation and research tool for large scale camera sensor networks. Specifically, we demonstrate our visual sensor network comprising uncalibrated static and active simulated video surveillance cameras in a virtual public space, a train station populated by autonomously self-animating pedestrians. Here, the readily reconfigurable virtual cameras generate synthetic video feeds that emulate those generated by real surveillance cameras monitoring public spaces. Our simulation approach, which runs on high-end commodity PCs, has proven to be beneficial, since this type of research would be difficult to carry out in the real world given the prohibitive costs of deploying and experimenting with an appropriately complex camera network in large public spaces.

Categories and Subject Descriptors

I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Motion,Tracking*; I.5.4 [**Pattern Recognition**]: Applications—*Computer Vision*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems, Coherence and coordination*

General Terms

Design, Algorithms

Keywords

Camera Networks, Surveillance Systems, Virtual Vision, Smart Cameras, Camera Tasking, Sensor Coordination

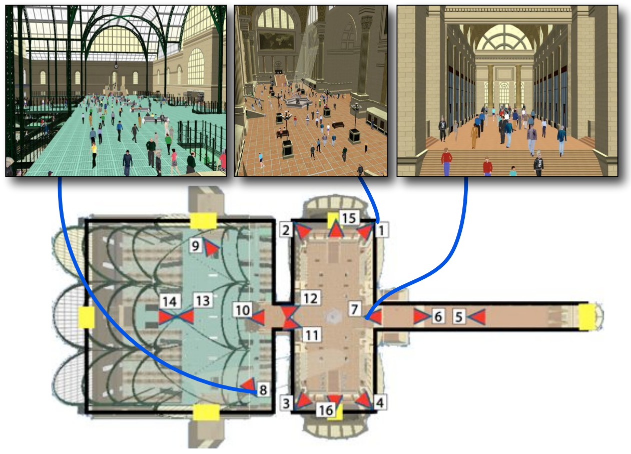

Figure 1: Plan view of the virtual Penn Station environment with the roof not rendered, revealing the concourses and train tracks (left), the main waiting room (center), and the long shopping arcade (right). (The yellow rectangles indicate station pedestrian portals.) An example visual sensor network is shown comprising 16 simulated active (pan-tilt-zoom) video surveillance cameras.

[**Note to Reviewer:** Supplementary material is available at http://www.cs.toronto.edu/∼faisal/ipsn07]

1. INTRODUCTION

Camera sensor networks are becoming increasingly important to next generation applications in surveillance, environment and disaster monitoring, and the military. In contrast to current video surveillance systems, camera sensor networks are characterized by smart cameras, large network sizes and ad hoc deployment. These systems lie at the intersection of machine vision and sensor networks, raising issues in the two fields that must be addressed simultaneously. The effective visual coverage of large areas—public spaces, disaster zones, and battlefields—requires multiple cameras to work together towards common sensing goals. As the size of the camera 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 in performing advanced surveillance tasks. Therefore, it is desirable to design camera sensor networks that are capable of performing visual surveillance tasks autonomously, or at least with minimal human intervention.

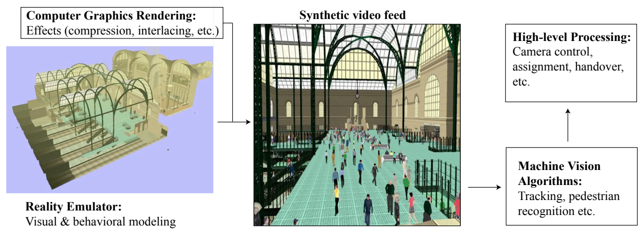
Computer Graphics Rendering:
Effects (compression, interlacing, etc.)
Synthetic video feed
High-level Processing:
Camera control, assignment, handover, etc.
Machine Vision Algorithms:
Tracking, pedestrian recognition etc.
Reality Emulator:
Visual & behavioral modeling

Figure 2: Virtual vision paradigm (image from [20]).

In this paper, we demonstrate a camera network model comprising *uncalibrated* passive and active simulated video surveillance cameras that provide perceptive coverage of a large virtual public space—a train station populated by autonomously self-animating virtual pedestrians (Fig. 1)—with minimal operator assistance. Once an operator monitoring the surveillance video feeds spots a suspicious pedestrian, or a visual behavior analysis routine automatically identifies such a pedestrian, 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 inter-node communication.

1.1 Virtual Vision

Deploying a large-scale camera sensor network in the real world is a major undertaking whose cost can easily be prohibitive for most researchers interested in designing and experimenting with sensor networks. Moreover, privacy laws generally restrict the monitoring of people in public spaces for experimental purposes. As a means of overcoming these impediments, we advocate the pursuit of camera sensor network research within the context of a unique synthesis of advanced computer graphics and vision simulation technologies. In particular, we demonstrate the design of simulated camera sensor network systems and meaningful experimentation with such systems within visually and behaviorally realistic virtual environments (Fig. 2).

Legal impediments and cost considerations aside, the use of realistic virtual environments in sensor network research offer significantly greater flexibility during the design and evaluation cycle, thus expediting the engineering process: The multiple virtual cameras, which generate synthetic video feeds that emulate those generated by real surveillance cameras monitoring public spaces, are very easily reconfigurable in the virtual space. The virtual world provides readily accessible ground-truth data for the purposes of visual sensor network algorithm validation. Experiments are perfectly repeatable in the virtual world, so we can readily modify algorithms and parameters and immediately determine their effect. The hard real-time constraints of the real world can easily be relaxed in the simulated world; i.e., simulation time can be prolonged relative to real, “wall-clock time”, in principle permitting arbitrary amounts of computational processing to be carried out during each unit of simulated time. Finally, despite its sophistication, our simulator runs on high-end commodity PCs, obviating the need to grapple with special-purpose hardware and software.

1.2 Distributed Control in Camera Sensor Networks

The work presented here deals with distributed control in camera sensor networks and many of the characteristics and challenges associated with sensor networks are relevant. Task-based sensor selection is a fundamental issue in sensor networks [26]. The selection process must take into account the information contribution of each node against its resource consumption or potential utility in other uses. Another key issue in sensor networks is node organization, which has been proposed by researchers as a means to limit the communication to those nodes that are relevant to the task at hand. Distributed approaches for node selection or node organization 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 sensor node has local autonomy and can communicate with a small number of neighboring nodes that are within radio communication range.

Mindful of these issues, we propose a novel camera sensor network control strategy that does not require camera calibration, a detailed world model, or a central controller. We model cameras as nodes in a communication network that emulates those found in real sensor networks: 1) nodes can communicate directly with their neighbours, 2) if necessary, a node can communicate with another node in the network through multi-hop routing, and 3) unreliable communication. The overall behavior of the network is the consequence of the local processing at each node and inter-node 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. During the lifetime of the surveillance task, a group evolves—i.e., old nodes leave the group and new nodes join it. One node in each group acts as the group leader and is responsible for group level decision making. We also present a new constraint satisfaction problem formulation for resolving group-group interactions.

1.3 Overview

The contributions of this paper are twofold. We introduce a novel camera sensor network framework suitable for next generation visual surveillance applications. Furthermore, we demonstrate the advantages of developing and evaluating camera sensor networks within a sophisticated virtual reality simulation environment. The remainder of the paper is organized as follows: Sec. 2 covers relevant prior work. We explain the low-level vision emulation and behavior models for camera nodes in Sec. 3. Sec. 4 introduces the sensor network communication model. In Sec. 5, we demonstrate the application of this model in the context of visual surveillance. We present our results in Sec. 6 and our conclusions and future research directions in Sec. 7.

2. RELATED WORK

As was argued in [23, 24], computer graphics and virtual reality technologies are rapidly presenting viable alternatives to the real world for developing sensory systems (see also [19]). Our camera network is deployed and tested within the virtual train station simulator that was developed in [20]. 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 extensive 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 though portals and congested areas, queue in lines as necessary, purchase train tickets at the ticket booths in the main waiting room, sit on benches when they are tired, purchase food/drinks from vending machines when they are hungry/thirsty, etc., and eventually proceed to the concourse area and down the stairs to the train tracks. Standard computer graphics techniques enable a photorealistic rendering of 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 [26]. Reference [14] 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 [26]. Reference [27] introduces an information driven approach to collaborative tracking, which attempts to minimize the energy expenditure at each node by reducing inter-node 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 do without such knowledge; a node needs to communicate with nearby nodes before selecting new nodes.

Nodes comprising sensor networks are usually untethered sensing units with limited onboard power reserves. A crucial concern in sensor networks, therefore, is that of energy expenditure at each node, which determines the life-span of a sensor network [2]. Node communications have large power requirements; therefore, sensor network control strategies attempt to minimize the inter-node communication [4, 27]. 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 [6,8]. Previous work on camera networks in computer vision has dealt with issues related to low-level and mid-level computer vision, namely segmentation, tracking, and identification of moving objects [5], and camera network calibration [9]. 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 high-capability multi-media sensors connected via high-capacity communication channels [3]. 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 systems [13]. It demonstrates the benefits of a multi-tiered network—each tier defines a set of sensing capabilities and corresponds to a single class of smart camera sensors—over single-tiered networks in terms of low-latencies and energy efficiency. SensEye, however, 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 [21] and realizes group formation via inter-node negotiation. Unlike Mallett's [14] 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 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 [12]. Specifically, according to the taxonomy provided in [12], our sensor assignment formulation belongs to the single-task robots (ST), multi-robot tasks (MR), 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 are extensively studied and can be reduced to a Set Partitioning Problem (SPP), which is strongly NP-hard [11]. However, many heuristics-based set partitioning algorithms exist that produce good results on large SPPs [1]. Fortunately, the sizes of MRTA problems, and by extension SPPs, encountered in our camera sensor network setting are small due to the spatial/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 [15] and [25], 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 assignments—sensor assignment is distributed at the level of the network, whereas it is centralized at the level of a group.

3. 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 next two sections describe each of these aspects of a camera node.

3.1 Local Vision Routines

Each camera has its own suite of visual routines for pedestrian recognition, identification, and tracking, which we dub "Local Vision Routines" (LVRs). The LVRs are computer vision algorithms that directly operate upon the synthetic video captured by virtual cameras. LVRs do not have access to any 3D information available from the virtual world, and they mimic the performance of a state-of-the-art pedestrian segmentation and tracking module (Fig. 3 Row 1). In particular, pedestrian tracking can fail due to occlusions, poor segmentation, bad lighting, or crowding. Tracking sometimes locks on the wrong pedestrian, especially if the scene contains multiple pedestrians with similar visual appearance; i.e., wearing similar clothes. Our imaging model emulates artifacts that are of interest to camera network researchers, such as video compression and interlacing. It also models camera jitter and imperfect color response.

We employ appearance-based models to track pedestrians. Pedestrians are segmented to construct unique and robust color-based signatures (appearance models), which are then matched across the subsequent frames. Color-based signatures have found widespread

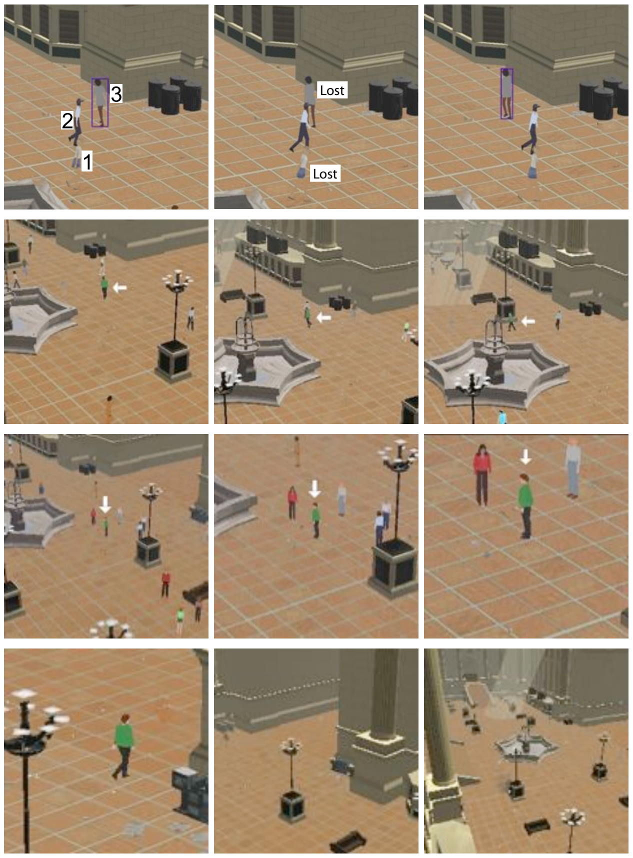

Figure 3: Row 1: The LVRs are programmed to track Pedestrians 1 and 3. Pedestrian 3 is tracked successfully; however, track is lost of Pedestrian 1 who blends in with the background. The tracking routine loses Pedestrian 3 when she is occluded by Pedestrian 2, but it regains track of Pedestrian 3 when Pedestrian 2 moves out of the way. Row 2: Tracking while fixating on a pedestrian. Row 3: Tracking while zooming in on a pedestrian. Row 4: Camera returns to its default settings upon losing the pedestrian; it is now ready for another task.

use in tracking applications [7], but they are sensitive to illumination changes. However, this shortcoming can be mitigated by operating in HSV space instead of RGB space. Furthermore, zooming can drastically change the appearance of a pedestrian, thereby confounding conventional appearance-based schemes. We employ a modified *color-indexing* scheme [22] 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. See [17] for more details.

3.2 Camera Node Behaviors

Each camera node is an autonomous agent capable of communicating with nearby nodes. The LVRs determine the sensing capabilities of a camera node, whose overall behavior is determined by the LVR (bottom-up) and the current task (top-down). We model the camera controller as an augmented hierarchical finite state machine (Fig. 4).

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

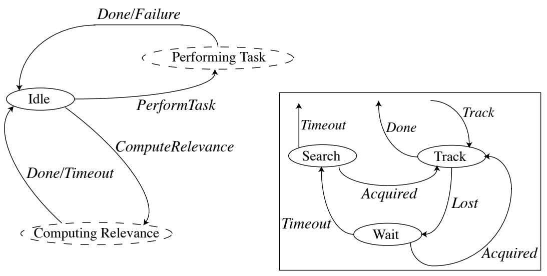

Figure 4: Top-level camera controller. Dashed states contain the child finite state machine shown in the inset.

compute its relevance to the task. For example, a camera can use visual search to find a pedestrian that matches the appearance-based signature passed 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 when it fails to compute the relevance because it cannot find a pedestrian that matches the description. When the camera successfully finds the desired pedestrian, however, it returns the relevance value to the querying node. The querying node passes the relevance value to the leader (leader node) of the group, which decides whether or not to include the camera node in the group. The camera goes into *PerformingTask* state upon joining a group where the embedded child finite state machine (Fig. 4 inset) hides the sensing details from the top-level controller and enables the node to handle short-duration sensing (tracking) failures. Built-in timers allow the camera node to transition into the default state instead of hanging in some state waiting for a message from another node, which might never arrive due to a communication error or node failure.

Each camera can *fixate* and *zoom* in on an object of interest. Fixation and zooming routines are image driven and do not require any 3D information, such as camera calibration or a global frame of reference. We discovered that traditional Proportional Derivative (PD) controllers generate unsteady control signals, resulting in jittery camera motion. The noisy nature of tracking forces the PD controller to try continuously to minimize the error metric without ever succeeding, so the camera keeps servoing. Hence, we model the fixation and zooming routines as dual-state controllers. The states are used to activate/deactivate the PD controllers. In the *act* state the PD controller tries to minimize the error signal; whereas, in the *maintain* state the PD controller ignores the error signal altogether and does nothing.

The *fixate* routine brings the region of interest—e.g., the bounding box of a pedestrian—into the center of the image by tilting the camera about its local x and y axes (Fig. 3, Row 2). The *zoom* routine controls the FOV of the camera such that the region of interest occupies the desired percentage of the image (Fig. 3 Row 3). See [17] for details.

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

4. SENSOR NETWORK MODEL

We now explain the sensor network communication scheme that enables task-specific coalition formation. The idea is 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 formalizes the collaboration between member nodes, is a dynamic arrangement that keeps evolving 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 ContractNet protocol that models auctions (announcement, bidding, and selection) for group formation [21] (see Fig. 5). 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 sensor 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 request the network to count the number of pedestrians wearing green shirts. To date we have experimented only with the first kind of queries, which are sufficient for setting up collaborative tracking tasks; however, this is by no means a limitation of the proposed communication model.

4.1 Coalition Formation

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 node. To recruit new nodes for 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 (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 multinode groups, if a group leader decides to recruit more nodes for the task at hand, it instructs group nodes to broadcast task requirements. This is accomplished via 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, group nodes broadcasts *status* messages at regular intervals. Group leaders use *status* 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 include that node into its group. The leader node 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 the node in some preset time limit.[1] 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 amongst the member nodes. The group vanishes when the last node leaves the group.

[1]The relevance value of a group node is decayed 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.

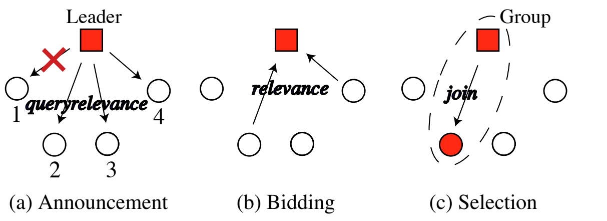

Figure 5: Task auction supports coalition formation. The red cross indicates a lost message.

4.2 Conflict Resolution

A conflict resolution mechanism is needed when multiple groups require the same resources (Fig. 6(f)). The problem of assigning sensors to the contending groups can be treated as a Constraint Satisfaction Problem (CSP) [16]. Formally, a CSP consists of a set of variables $\{v_1, v_2, v_3, \cdots, v_k\}$, a set of allowed values 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 sensors with relevance values (with respect to g) greater than a predefined threshold. The constraints restrict the assignment of a sensor to multiple groups. Assume, for example, a group g and a set of nodes $\{n_1, n_2, n_3\}$ with relevance values $\{r_1, r_2, r_3\}$, respectively. If r_3 is less than the predefined threshold, the set of nodes that will be considered for assignment to g is $\{n_1, n_2\}$, and the domain of g is the set $\{\{n_1\}, \{n_2\}, \{n_1, n_2\}\}$. We define a constraint C_{ij} as $a_i \cap a_j = \{\Phi\}$, where a_i and a_j are sensor assignments to groups g_i and g_j, respectively; k groups give rise to $k!/2!(k-2)!$ 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. To solve P, we employ *backtracking* to search systematically through the space of possibilities. We find all solutions, rank these solutions according to the relevance values for sensors (with respect to each group), and select the best solution to find the optimal assignments. The solution ranking procedure can easily incorporate other relevant concerns such as a preference for sensors that are positioned orthogonal to each other with respect to the pedestrian so as to increase the position estimate accuracy or using sensors that are within one hop distance of each other. When P has no solution, we recursively subdivide and solve smaller CSP problems $P' = (G', D', C')$, where $G' \subset G$ and D' and C' are defined accordingly, until we find a solution to a smaller problem P'.

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 leader nodes that uses *backtracking* to solve the problem.[2] The result is then conveyed to the other leader nodes.

[2]Leader node where centralization occurs is selected using a strat-

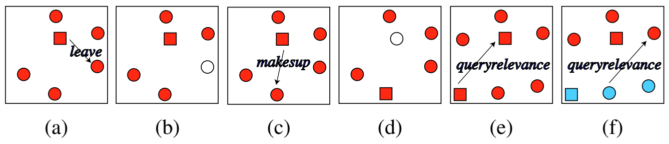

Figure 6: (a)-(b) A node leaves a group after receiving a leave message from the group leader. (c)-(d) Old group leader selects a new group leader and leaves the group. (e) A leader node detects another leader performing the same task; leader/supervisor demotion commences. (f) Conflict detection between two resources.

CSPs have been studied extensively in the computer science literature and there exist more powerful variants of the basic backtracking method; however, we employ the naive backtracking approach in the interest of simplicity and because it can easily cope with the size of problems encountered in the current setting. A key feature of our 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.

4.3 Node Failures & Communication Errors

The purposed communication model takes into consideration node and communication failures. Communication failures are perceived as sensor 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 leader node does not receive a *status* from the node during some predefined interval, and the leader node removes the problem node from the group. On the other hand, when a member node does not receive any message (*status* or *queryrelevance*) from the leader node during a predefined interval, it assumes that the leader node has experienced a failure and selects itself to be the leader of the group. An actual or perceived leader 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 of the leader nodes of the constituent groups, except one. Consider, for example, a group comprising three nodes a, b, and c; node a being the leader node. When a fails, b and c form two single-node groups and continue to perform the sensing task. In due course, nodes b and c discover each other—e.g., when b intercepts a *queryrelevance* or a *status* message from c—and they form a new group comprising b and c, demoting node c in the process. Thus, our proposed communication model is able to handle node failures.

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 process shown in Fig. 7.

5. VIDEO SURVEILLANCE

We now consider how a sensor network of dynamic cameras may be used in the context of video surveillance (Fig. 8). A human operator spots one or more suspicious pedestrians in one of the video feeds and, for example, requests the network to "observe this pedestrian," "zoom in on this pedestrian," or "observe the entire group." The successful execution and completion of these tasks requires intelligent allocation and scheduling of the available cameras; in particular, the network must decide which cameras should track the

egy similar to that used for group merging (Fig. 7).

Assumptions: Nodes n and m are two leader nodes performing Task 1.
Case 1: Node n receives a *queryrelevance* or *status* message from node m.
Node n is not involved in demotion negotiations with another node **then** send *demote* message to node m after a random interval.
Case 2: Node n receives a *demote* message from node m.
a) Node n has not sent a *demote* message to another node **then** demote node n and send *demoteack* message to node m.
b) Node n has sent a *demote* message to node m **then** send *demoteretry* message to node m and send a *demote* message to node m after a random interval.
c) 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 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.

Figure 7: Group merging via leader demotion.

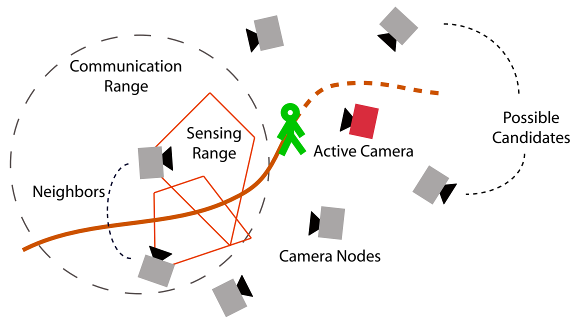

Figure 8: A camera network for video surveillance consists of camera nodes that can communicate with other nearby nodes. Collaborative tracking requires that cameras organize themselves to perform camera handover when the tracked subject moves out of the sensing range of one camera and into that of another.

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 and scheduling of cameras; however, it is cumbersome and in most cases infeasible to acquire such a world model. Our approach does not require such a knowledge base. We assume only that a pedestrian can be identified by different cameras with reasonable accuracy and that the camera network topology is known *a priori*. A direct consequence of this approach is that the network can easily be modified through removal, addition, or replacement of camera nodes.

5.1 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. Our scheme for computing the relevance of a camera to a video 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

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

Figure 9: The relevance metric returned by a camera node relative to a new task request. The leader node converts the metric into a scalar value representing the relevance of the node for the particular surveillance task.

longer time, and 4) it is better to avoid unnecessary reassignments of cameras to different tasks, as that might degrade the performance of the underlying computer vision routines.

Upon receiving a task request, a camera node returns to the leader node a relevance metric—a list of attribute-value pairs describing its relevance to the current task along multiple dimensions (Fig. 9). The leader node converts this metric into a scalar relevance value r as follows:

$$r = \begin{cases} \exp\left(-\frac{(c-1)^2}{2{\sigma_c}^2} - \frac{(\theta-\hat{\theta})^2}{2{\sigma_\theta}^2} - \frac{(\alpha-\hat{\alpha})^2}{2{\sigma_\alpha}^2} - \frac{(\beta-\hat{\beta})^2}{2{\sigma_\beta}^2}\right) & \text{when } s = \text{free} \\ \frac{t}{t+\gamma} \quad \text{when } s = \text{busy} \end{cases} \qquad (1)$$

where $\hat{\theta} = (\theta_{\text{min}} + \theta_{\text{max}})/2$, $\hat{\alpha} = (\alpha_{\text{min}} + \alpha_{\text{max}})/2$, and $\hat{\beta} = (\beta_{\text{min}} + \beta_{\text{max}})/2$, and where θ_{min} and θ_{max} are extremal field of view settings, α_{min} and α_{max} are extremal rotation angles around the x-axis (up-down), and β_{min} and β_{max} are extremal rotation angles around the y-axis (left-right). Here, $0.3 \leq \sigma_c \leq 0.33$, $\sigma_\theta = (\theta_{\text{max}} - \theta_{\text{min}})/6$, $\sigma_\alpha = (\alpha_{\text{max}} - \alpha_{\text{min}})/6$, and $\sigma_\beta = (\beta_{\text{max}} - \beta_{\text{min}})/6$. The value of γ is chosen empirically (for our experiments we have selected γ to be 1000).

The computed relevance values are used by the node selection scheme described above to assign cameras to various tasks. The leader 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.

5.2 Surveillance Tasks

We have implemented an interface that presents the operator a display of the synthetic video feeds from multiple virtual surveillance cameras. The operator can select a pedestrian in any video feed and instruct the camera network to perform one of the following tasks: 1) follow the pedestrian, 2) capture a high-resolution snapshot, or 3) zoom-in and follow the pedestrian. The network then automatically assigns cameras to fulfill the task requirements. The operator can also initiate multiple tasks, in which case either cameras that are not currently occupied are chosen for the new task or some currently occupied cameras are reassigned to the new task.

6. RESULTS

To date, we have tested our visual sensor network system with up to 16 stationary and pan-tilt-zoom cameras, and we have populated the virtual Penn station with up to 100 pedestrians. The sensor network correctly assigned cameras 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 has increasing difficulty following the correct pedestrian, so the probability increases that the surveillance task fails (and the cameras just return to their default settings).

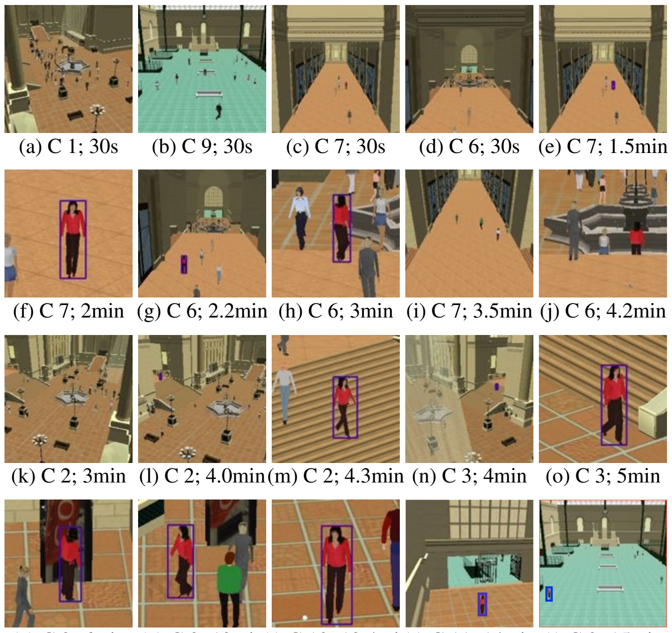
(a) C 1; 30s (b) C 9; 30s (c) C 7; 30s (d) C 6; 30s (e) C 7; 1.5min
(f) C 7; 2min (g) C 6; 2.2min (h) C 6; 3min (i) C 7; 3.5min (j) C 6; 4.2min
(k) C 2; 3min (l) C 2; 4.0min (m) C 2; 4.3min (n) C 3; 4min (o) C 3; 5min
(p) C 3; 6min (q) C 3; 13min (r) C 10; 13.4min (s) C 11; 14min (t) C 9; 15min

Figure 10: A pedestrian is successively tracked by Cameras 7, 6, 2, 3, 10, and 9 (see Fig. 1) as she makes her way through the station to the concourse. (a-d) Cameras observing the station. (e) Operator selects a pedestrian in feed 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—it 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 tracking the pedestrian. (n) Camera 3 is recruited by the 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 tracking the pedestrian. (s) Camera 11 is recruited by Camera 10. (t) Camera 9 is recruited by Camera 10.

For the example shown in Fig. 10, we placed 16 active PTZ cameras in the train station, as shown in Fig. 1. An operator selects the pedestrian with the red shirt in Camera 7 (Fig. 10(e)) and initiates the "follow" 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 track the pedestrian. Camera 6 becomes the leader 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 leader, and leaves the group. Cameras 2 and 3 track 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 leader, recruits Camera 9, which tracks the pedestrian as she enters the concourse.

Fig. 11 illustrates a "follow" task sequence. An operator selects the pedestrian with the green shirt in Camera 1 (top row). Camera

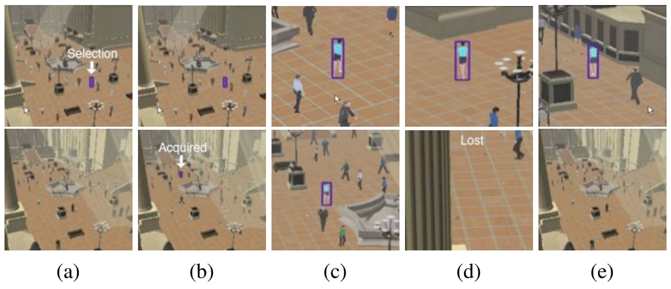

Figure 11: "Follow" sequence. (a) The operator selects a pedestrian in Camera 1 (upper row). (b) and (c) Camera 1 and Camera 2 (lower row) are tracking the pedestrian. (d) Camera 2 loses track. (e) Camera 1 is still tracking; Camera 2 has returned to its default settings.

1 forms a group with Camera 2 (bottom row) to follow and zoom in on the pedestrian. At some point, Camera 2 loses the pedestrian (due to occlusion), and it invokes a search routine, but it fails to reacquire the pedestrian. Camera 1, however, is still tracking the pedestrian. Camera 2 leaves the group and returns to its default settings.

The appendix presents several simulations of larger sensor networks outside our virtual vision simulator.

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 sensor network methodologies. The work presented in this paper is a step toward the realization of new sensor networks. Our initial results appear to be promising.

A unique and, in our view, important aspect of our work, is that we have developed and demonstrated our prototype video surveillance system in a realistic virtual train station environment populated by lifelike, autonomously self-animating virtual pedestrians. Our sophisticated sensor 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 sensor 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 pursuing a *Cognitive Modeling* [10, 18] approach to node organization and camera scheduling. We are also investigating scalability and node failure issues. Moreover, we are 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 when it is required to follow multiple pedestrians during their entire stay in the train station.

Acknowledgments

The research reported herein was supported 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 Penn Station simulator. We thank Deborah Estrin for introducing us to the sensor networks literature.

APPENDIX

A. OTHER SIMULATIONS

We have also tested the sensor network communication model under various conditions with up to 100 sensor nodes. Fig. 12 shows a sensor network consisting of 100 nodes on a 10 by 10 uniform grid. Each node can communicate with its 4-neighbours. A node can communicate with any other node in the network via multi-hop messaging. The user task the bottom-left node to follow the target shown as a cyan cone.[3] The bottom left node elects itself as the leader and begins recruiting other sensor nodes (Fig. 12(a)). The group evolves by recruiting new nodes, dropping member nodes that are no longer relevant, and selecting new leaders in response to the change in target's location (Fig. 12(b)). When the target splits into two, the group successfully splits into two groups, each with its own leader, to follow the target (Fig. 12(c)–(e)). It is important to note that group splitting occurs naturally in our protocol and does not require any special handling. Furthermore, the two groups do not need to communicate with each other. As targets converge upon each other, the two groups discover each other and merge demoting one of the leaders (Fig. 12(f)–(h)). The protocol successfully handled both leader and member node failures and kept evolving to follow the target (Fig. 12(i)–(l)).

Fig. 13 shows a sensor network of 50 nodes placed randomly in a 25 square m area. The nodes that are within 5 m of each other can directly communicate with each other. Each node can communicate with another node in the network through multi-hop routing. Fig. 13(a)–(e) can show group merging. When the leader of the group fails (Fig. 13(f)), multiple member nodes assume leadership (Fig. 13(g)). These nodes negotiate each other to select a single leader (Fig. 13(h)).

B. REFERENCES

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

[2] M. Bhardwaj, A. Chandrakasan, and T. Garnett. Upper bounds on the lifetime of sensor networks. In *IEEE International Conference on Communications*, number 26, pages 785 – 790, 2001.

[3] 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 (MULTIMEDIA '05)*, pages 81–88, New York, NY, USA, 2005. ACM Press.

[4] J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless adhoc networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 22–31, Tel Aviv, Israel, Mar. 2000.

[3]It appears as the cyan circle in Fig. 12.

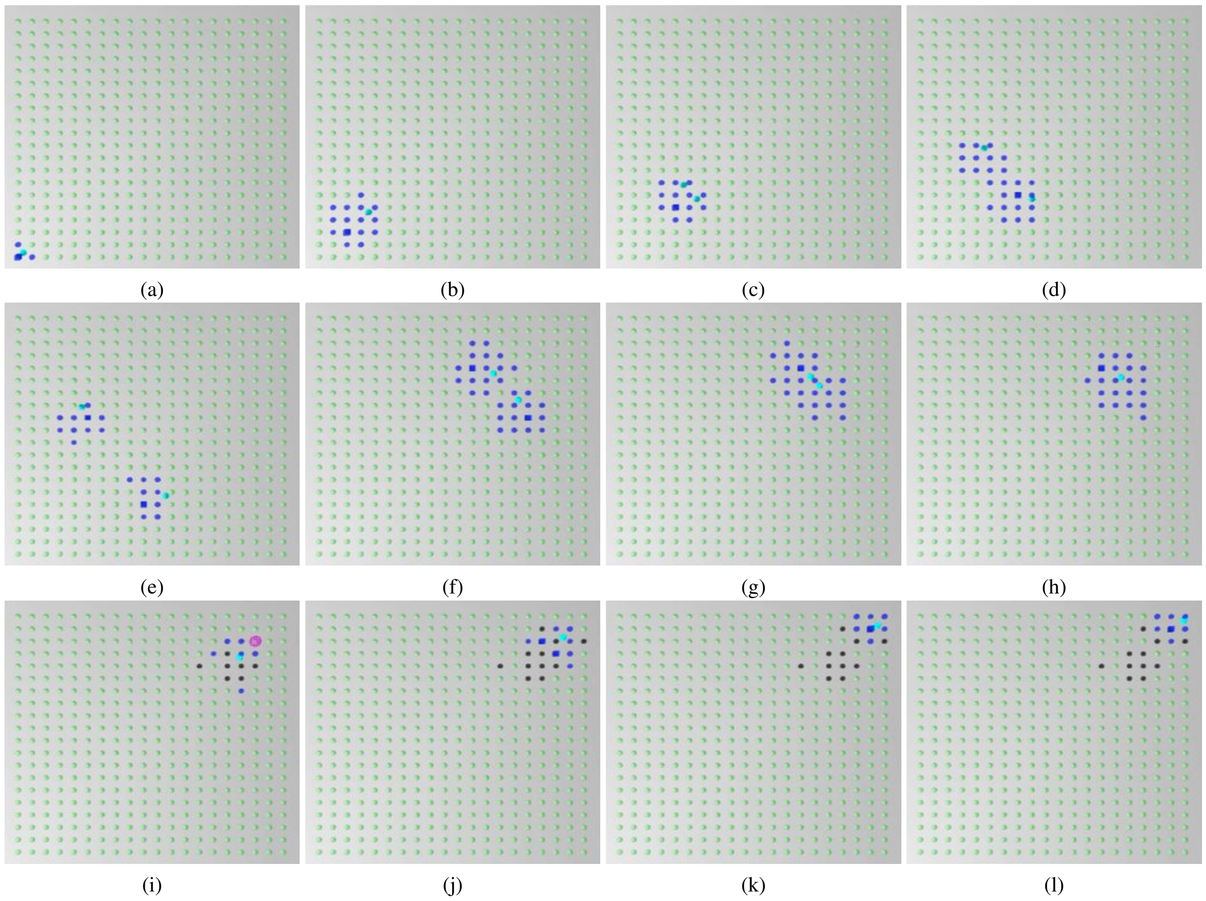

Figure 12: Group splitting and merging. Green nodes represent idle sensor nodes, blue nodes represent sensor nodes that are currently engaged in the task of following the target denoted by a cyan cone. The target appears as a cyan circle in the top view of the sensor grid shown here. Square nodes represent group leaders and black nodes represent failed nodes. Each node can communicate with its 4-neighbours.

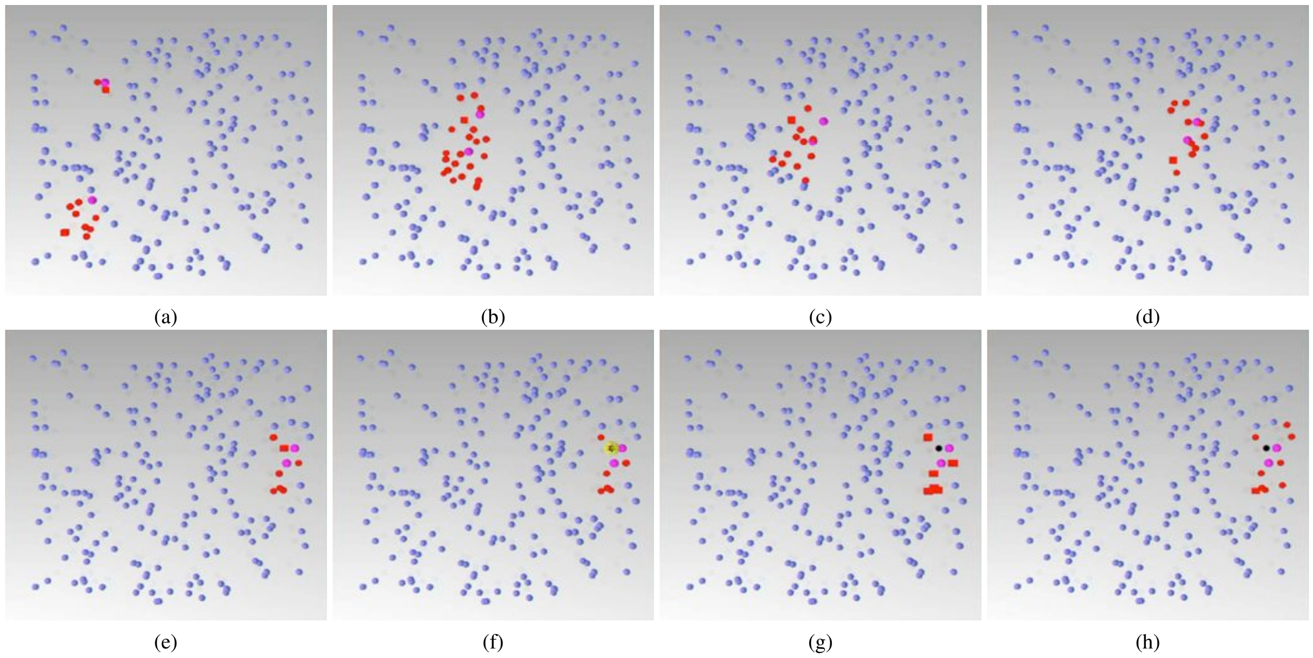

Figure 13: Group merging and leader failure. Blue nodes are idle. Red nodes are following the targets shown as pink cones. Square nodes represent group leaders and black nodes indicate node failures.

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

[6] R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE*, 89(10):1456–1477, Oct. 2001.

[7] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. of the 2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 00)*, volume 2, pages 142–151, Hilton Head Island, South Carolina, USA, 2000.

[8] 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*, pages 39–45, New York, NY, 2004. ACM Press.

[9] D. Devarajan, R. J. Radke, and H. Chung. Distributed metric calibration of ad-hoc camera networks. To appear in ACM Transactions on Sensor Networks, 2006.

[10] J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In A. Rockwood, editor, *Proc. ACM SIGGRAPH 99*, pages 29–38, Aug. 1999.

[11] M. R. Garey and D. S. Johnson. “strong” npcompleteness results: Motivation, examples, and implications. *Journal of the ACM*, 25(3):499–508, 1978.

[12] B. Gerkey and M. Matari. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954, 2004.

[13] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu. Senseye: a multi-tier camera sensor network. In *MULTIMEDIA ’05: Proc. of the 13th annual ACM international conference on Multimedia*, pages 229–238, New York, NY, USA, 2005. ACM Press.

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

[15] P. J. Modi, W.-S. Shen, M. Tambe, and M. Yokoo. Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1–2):149–180, Mar 2006. Elsevier.

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

[17] F. Qureshi. Intelligent perception in virtual camera networks and space robotics. Ph. D. Thesis (under preparation). Dept. of Computer Science, University of Tronto, 2006.

[18] F. Qureshi, D. Terzopoulos, and P. Jaseiobedzki. Cognitive vision for autonomous satellite rendezvous and docking. In *Proc. IAPR Conference on Machine Vision Applications*, pages 314–319, Tsukuba Science City, Japan, May 2005.

[19] 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)*, number 1, pages 1–6, Spain, Sept. 2003.

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

[21] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transctions on Computers*, C-29(12):1104–1113, Dec 1980.

[22] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, Nov 1991.

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

[24] D. Terzopoulos and T. Rabie. Animat vision: Active vision in artificial animals. *Videre: Journal of Computer Vision Research*, 1(1):2–19, Sept. 1997.

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

[26] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich. Collaborative signal and information processing: An information directed approach. *Proceedings of the IEEE*, 91(8):1199–1209, 2003.

[27] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. In *IEEE Signal Processing Magazine*, volume 19, pages 61–72. Mar. 2002.