# A Virtual Reality Approach to Designing Visual Sensor Networks

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

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

***Abstract*— This paper presents sensor network research in the context of a unique synthesis of advanced computer graphics and vision simulation technologies. In particular, we propose the design of and experimentation with simulated sensor network systems within visually and behaviorally realistic virtual environments. Specifically, we demonstrate a visual sensor 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. The readily reconfigurable virtual cameras generate synthetic video feeds that emulate those generated by real surveillance cameras monitoring public spaces. With minimal reliance on a human operator, our new sensor network control strategy enables the collection of cameras to collaborate in performing various visual surveillance tasks, such as closely monitoring a pedestrian as (s)he locomotes in and out of the fields of view of individual cameras. The network supports task-dependent node selection and aggregation through local decision-making and inter-node communication. We propose a solution to the node selection problem that transforms it into an equivalent constraint satisfaction problem. Our technique is scalable and robust to node failures, as it lacks any central controller. Our simulation approach, which runs on a high-end commodity PC, 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.**

**Note to reviewers:** Please also see the supplemental multimedia material at `http://www.cs.toronto.edu/~faisal/ipsn06`

## I. Introduction

Deploying a large-scale visual 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 in this paper the pursuit of 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 sensor network systems and meaningful experimentation with such systems within visually and behaviorally realistic virtual environments. 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.

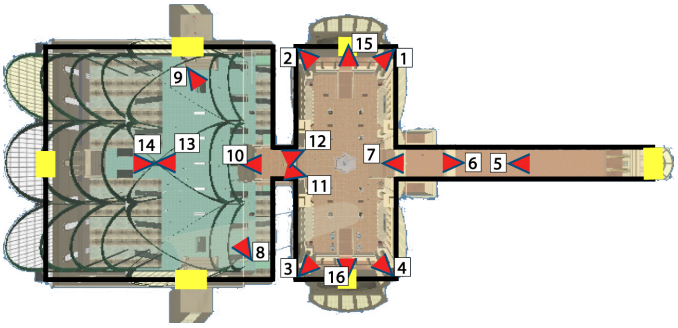



Fig. 1. Plan view of the virtual Penn Station environment with the roof not rendered, revealing the concourses and train tracks (left), the main waiting room (center), and the long shopping arcade (right). (The yellow rectangles indicate station pedestrian portals.) An example visual sensor network is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

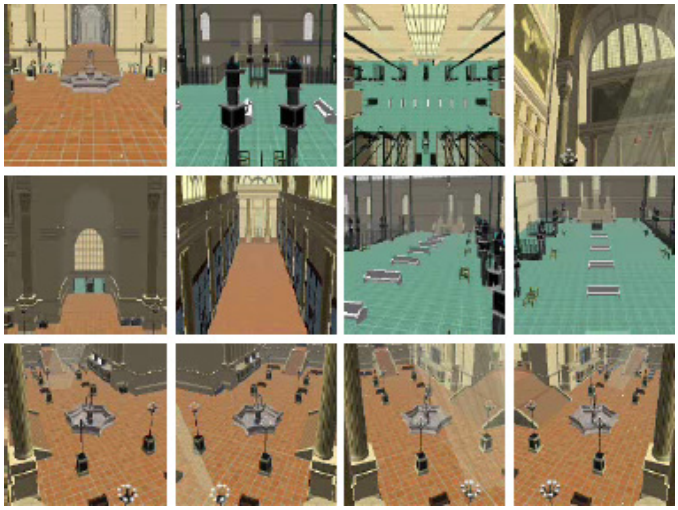


Fig. 2. Synthetic video feeds from multiple simulated video surveillance cameras situated in the (empty) Penn Station environment.

Specifically, we demonstrate visual sensor networks comprising static and active simulated video surveillance cameras that provide perceptive coverage of a large virtual public space; in our case, a train station (Fig. 1) populated by autonomously self-animating virtual pedestrians (Fig. 3). This type of research would be difficult to carry out in the real world given the cost of deploying and experimenting with an appropriately complex camera network a large public space the size of a train station. The multiple virtual cameras, which generate synthetic video feeds (Fig. 2) 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. Furthermore, 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.

Waiting room Concourses and platforms Arcade

Fig. 3. Views of the virtual train station populated by autonomously self-animating virtual pedestrians (from [1]).

Effective visual coverage of large public spaces, such as a train station or an airport, requires multiple cameras to work together towards common sensing goals. As the size of the camera network grows and the level of activity in the public space increases, 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, such as closely monitoring a pedestrian of interest as (s)he meanders through the field-of-view (FOV) of multiple cameras, or zooming in on a particular subject to acquire one or more facial snapshots. It is, therefore, desirable to design camera networks that are capable of performing visual surveillance tasks autonomously, or at least with minimal human intervention.

Our simulation environment has been beneficial in developing a camera network capable of performing common visual surveillance tasks with minimal operator assistance. Once an operator monitoring surveillance video feeds spots a pedestrian involved in some suspicious activity, or a visual behavior analysis routine selects such a pedestrian automatically, 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 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 sensor network control strategy that is capable of dynamic task-driven node aggregation through local decision making and inter-node communication.[1]

### A. Distributed Sensor Network Control Strategy

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 [5]. The selection process must take into account the information contribution of each node against its resource consumption or potential utility in other uses. Distributed approaches for node selection are preferable to centralized approaches that compromise 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 the radio communication range.

[1]A node can communicate with neighboring nodes, where the neighborhood of a node A can be defined automatically as the set of nodes that are, e.g., within nominal radio communications distance of A [2]. References [3], [4] present schemes to learn sensor network topologies.

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 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. A group keeps evolving—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 new constraint satisfaction problem formulation for resolving group-group interactions.

### B. Overview

In this paper, we propose a sensor network framework particularly suitable for designing camera networks for surveillance applications and demonstrate the advantages of developing and evaluating this framework within our virtual reality simulation environment. The remainder of the paper is organized as follows: Section II covers relevant prior work. We explain the low-level vision emulation and behavior models for camera nodes in Section III. Section IV introduces the sensor network communication model. In Section V, we demonstrate the application of this model in the context of visual surveillance. We present our results in Section VI and our conclusions and future research directions in Section VII.

## II. Related Work

As was argued in [6], [7], computer graphics and virtual reality technologies are rapidly presenting viable alternatives to the real world for developing sensory systems. Our sensor 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. 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 to the train tracks. Standard computer graphics techniques enable a photo-realistic rendering of the busy urban scene with considerable geometric and photometric detail (Fig. 3).

The problem of forming sensor groups based on task requirements and resource availability has received much attention within the sensor networks community [5]. Collaborative tracking, which subsumes the above issue, is considered an essential capability in many sensor networks [5]. [8] 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. Within 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—such knowledge is tedious to obtain and maintain during the life-time of the camera network. The camera networks presented here, therefore, do without such knowledge, and a node needs to communicate with the 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 [9]. Node communications have large power requirements; therefore, sensor network control strategies attempt to minimize the inter-node communication [8], [10]. Presently, we do not address this issue; however, the communication protocol proposed here limits the communication to the active nodes and their neighbors.

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 [11]. The emphasis has been on model transference from one camera to another, which is required for object identification across multiple cameras [12]. Vision researchers have proposed camera network calibration to achieve robust object identification and classification from multiple viewpoints, and automatic camera network calibration strategies have been proposed for both stationary and actively controlled camera nodes [13], [14]. 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.

Little attention has been paid in computer vision to the problem of controlling/scheduling active cameras to provide visual coverage of a large public area, such as a train station or an airport. The authors in [15] use a stationary wide-FOV camera to control an active tilt-zoom camera. The cameras are assumed to be calibrated and the total coverage of the cameras is restricted to the FOV of the stationary camera. Reference [16] presents a scheme for scheduling available cameras in a task-dependent fashion. Here, the tasks are defined within a world model that consists of the ground plane, traffic pathways, and detailed building layouts. The scheduling problem is cast as a temporal logic problem that requires access to a central database consisting of current camera schedules and viewing parameters. Unlike ours, this scheme is not scalable due to the central decision-making bottleneck.

## III. Camera Nodes

Each virtual camera node in the sensor network is able to perform low-level visual processing is an active sensor with a repertoire of camera behaviors. The subsequent two sections describe each of these aspects of a camera node.


Fig. 4. Color (hue) image pixel histograms of a tracked object change drastically with 'zooming' operations. A list of histograms is maintained for different zoom settings to address this issue. (c) The histogram of the (boxed) subject (a) when the FOV of the camera is set to 45 degrees. (d) The histogram of the same subject when the camera has zoomed in on the subject.

### A. 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 generated by virtual cameras and the information readily available from the 3D virtual world. They mimic the performance of a state-of-the-art pedestrian segmentation and tracking module. The virtual world affords us the benefit of fine tuning the performance of the recognition and tracking modules by taking into consideration the readily available ground truth. Our imaging model emulates camera jitter and imperfect color response; however, it does not yet account for such imaging artifacts as depth-of-field and image vignetting. More sophisticated rendering schemes would address this limitation.

We employ appearance-based models to track pedestrians. Pedestrians are segmented to construct unique and robust color-based pedestrian signatures (appearance models), which are then matched across the subsequent frames. Pedestrian segmentation is carried out using 3D geometric information and background modeling/subtraction. The quality of segmentation depends upon the amount of noise introduced into the process, and the noise is drawn from Gaussian distributions with appropriate means and variances. Color-based signatures, in particular, have found widespread use in tracking applications [17]. Color-based signatures are sensitive to illumination changes; however, this shortcoming can be mitigated by operating in HSV space instead of RGB space.

Zooming can drastically change the appearance of a pedestrian, thereby confounding conventional appearance-based schemes, such as color histogram signatures (Fig. 4). We tackle this problem by maintaining HSV color histograms for several camera zoom settings for each pedestrian. Thus, a distinctive characteristic of our pedestrian tracking routine is its ability to operate over a range of camera zoom settings.

The tracking module mimics the performance of a state-of-the-art tracking system. In particular, it can lose track due to occlusions, poor segmentation, or bad lighting. Tracking sometimes locks on the wrong pedestrian, especially if the the scene contains multiple pedestrians with similar visual appearance; i.e., wearing similar clothes. Tracking also fails in group settings when the pedestrian cannot be segmented properly.

### B. Camera Node Behaviors

Each camera node is an autonomous agent capable of communicating with nearby nodes. The LVRs determine the sensing capabilities

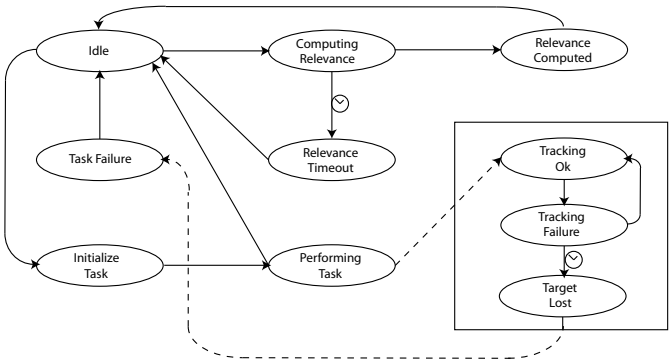



Fig. 5. Top-level camera controller.

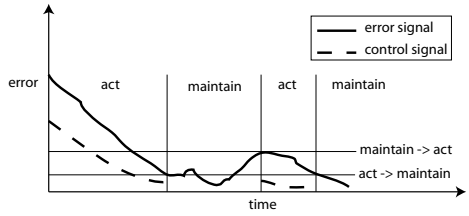



Fig. 6. Dual-state controller for fixation and zooming.

Fig. 7. Row 1: A fixate sequence. Row 2: A zoom sequence. Row 3: Camera returns to its default settings upon losing the pedestrian; it is now ready for another task.

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. 5).

In its default state, `Idle`, the camera node is not involved in any task. A camera node transitions into the `Computing Relevance` state upon receiving a `query_relevance` message from a nearby node. Using the description of the task that is contained within the `query_relevance` 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 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 due to the fact that it can not find a pedestrian that matches 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 leader (supervisor node) of the group, which decides whether or not to include the camera node in the group. The camera goes into `Performing Task` 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 short-duration sensing (tracking) failures. The camera node goes into `task_failure` state if it does not recover from the sensing failure within the set time limit. All states, except the `Performing Task` state, has built-in timers (not shown in Fig. 5) that allow the camera node to transition into the default state instead of being frozen in a state waiting forever for a message from another node. An expected message 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 (Fig. 6).

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. 7, Row 1). The *zoom* routine controls the FOV of the camera such that the region of interest occupies the desired percentage of the image. This is useful in situations where, for example, the operator desires a closer look at a suspicious pedestrian (Fig. 7, Row 2). The details of the *fixate* and *zoom* routines can be found in [18].

A camera node uses the *reset* routine to return to its default stance after finishing a task (Fig. 7, Row 3). The *reset* routine is modeled as a PD controller that attempts to minimize the error between the current zoom/tilt settings and the default zoom/tilt settings. We expect that for real PTZ cameras this would not be enough due to parameter drifting: a common occurrence in real hardware where the "true" setting of a parameter is not necessarily equal to the "dial" setting. We do not expect it to be a major issue though, as the true and dial settings typically follow each other closely; moreover, an image-based approach can drive the last stages of the reset operation and discover the dependencies between the dial and true settings.

## IV. Sensor Network Model

We now explain the sensor network communication scheme that enables task-specific node organization. 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. We require each node to compute its relevance to a task in the same currency. Our approach draws inspiration from behavior-based autonomous agents where the popularly held belief is that the overall intelligent behavior is a consequence of the interaction between many simple processes, called behaviors, rather than being the result of some powerful central processing facility. We leverage the interaction between the individual nodes to generate global task-directed behavior.

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. Currently, the network supports only the first kind of queries, which are sufficient for setting up collaborative tracking tasks.

### A. Node Grouping

Node grouping commences when a node, $n$, receives a sensing query. In response to the query, the node sets up a named task and creates a single-node group. Initially, node $n$, being the only node in the group, is chosen as the supervisor node. To recruit new nodes for the current task, node $n$ begins by sending `query_relevance` messages to its neighboring nodes, $N_n$. A subset $N'$ of $N_n$ respond by sending their relevance values for the current task. 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. When there is no resource contention between groups (tasks)—e.g., when only one task is active, or when multiple tasks that do not rely upon the same nodes for successful operation are active—the selection process is relatively straightforward; node $n$ picks those nodes from $N'$ that have higher relevance values. On the other hand, a conflict resolution mechanism is required when multiple groups vie for the same nodes; we describe 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 supervisor of that group. After receiving the `join` message, a subset $M'$ of $M$ elect to join the group and respond with a `join_ok` message.

For groups of more than one node, every group member sends out `query_relevance` messages to the nearby nodes that are not already part of the group. The supervisor node is responsible for group-level decisions, so member nodes forward to the group supervisor all the group-related messages, such as the `relevance` messages from potential candidates for group membership. The supervisor node sends "heartbeat" messages to other member nodes, which respond with their current relevance values, at regular intervals. The supervisor node uses the most recent relevance values to decide when to drop a member node. When a supervisor detects that its own relevance value for the current task is below the predefined threshold, it selects a new supervisor from amongst the member nodes. The group vanishes when the last node leaves the group.

### B. Conflict Resolution

A conflict resolution mechanism is needed when multiple groups require the same resources (Fig. 8(e-f)). The problem of assigning sensors to the contending groups can be treated as a Constraint Satisfaction Problem (CSP) [19]. Formally, a CSP consists of a set of

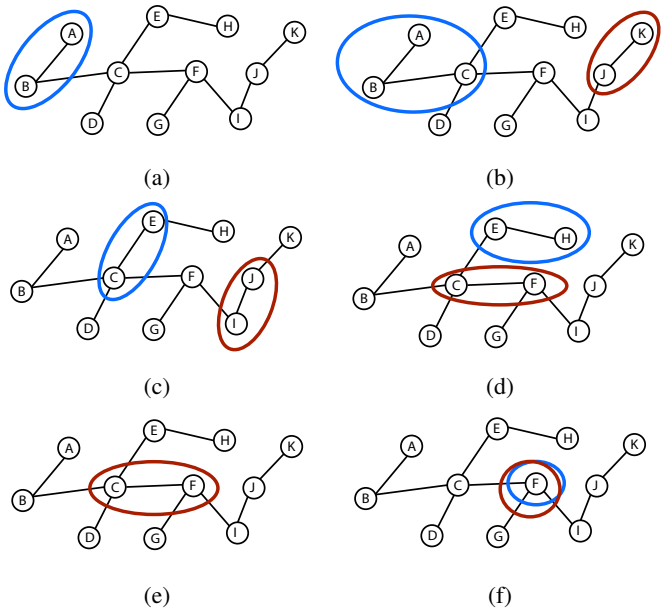



Fig. 8. Grouping and conflict resolution. (a) Group 1: A and B; possible candidate: C (b) Group 1: A, B, and C; possible candidates: E, F, and D. Group 2: J and K; possible candidate: I. (c) Group 1: E and C; possible candidates: H, F, D, and B. Group 2: J and I; possible candidate: F. (d) Group 1: E and H; possible candidate C. Group 2: C and F; possible candidates: B, D, G, I, and E. (e) Group 1 and 2 require the same resources, so Group 1 vanished; task failure. (f) A unique situation where both groups successfully use the same nodes; e.g., imagine two groups tracking two pedestrians that started walking together.

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 $a_i$s 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 (w.r.t. $g$) 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 (w.r.t. $g$) $\{r_1, r_2, r_3\}$, 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 (w.r.t. the each group), and select the best solution to find the optimal assignments. When $P$ does not have a solution, we recursively solve smaller CSP problems $P' = (G', D', C')$, where $G' \subset G$ and $D'$ and $C'$ are defined accordingly, until we found 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 `query_relevance` message from multiple groups, or when it already belongs to a group and it receives a `query_relevance` message from another group. The conflict resolution procedure begins by *centralizing* the CSP in one of the supervisor nodes that uses *backtracking* to solve the

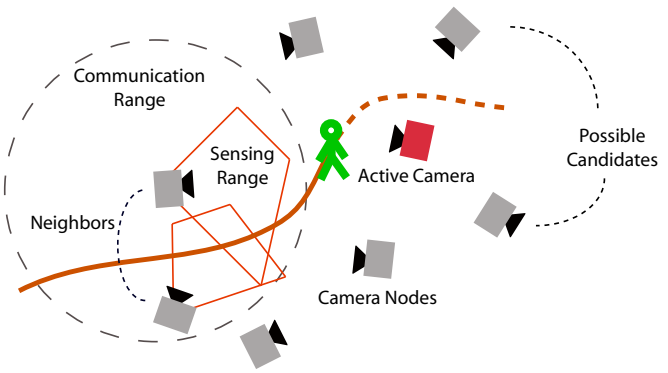



Fig. 9. A camera network for video surveillance consists of camera nodes that can communicate with other nearby nodes. Collaborative tracking requires that cameras organize themselves to perform camera handover when the tracked subject moves out of the sensing range of one camera and into that of another.

problem. The result is then conveyed to the other supervisor nodes.

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. Naive backtracking can easily cope with the size of problems encountered in the current setting. A key feature of the conflict resolution scheme proposed here is centralization, which requires that all the relevant information is 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 [20].[2]

### C. 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 supervisor node does not receive the node's response to the heartbeat message, and the supervisor node removes the problem node from the group. On the other hand, when a member node does not receive a heartbeat message from the supervisor node, 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 of the supervisor nodes of the constituent groups, except one. Consider, for example, a group comprising three nodes $a$, $b$, and $c$; node $a$ being the supervisor 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 `query_relevance` message from $c$—and they form a new group comprising $b$ and $c$, demoting node $c$ in the process. Thus, the proposed communication model is able to handle node failures.

## V. Video Surveillance

Let us now consider how a sensor network of dynamic cameras may be used in the context of video surveillance (Fig. 9). A human operator spots one or more suspicious pedestrians in one of the video feeds and, for example, requests the network to "track this pedestrian," "zoom in on this pedestrian," or "track 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 pedestrian and for how long.

[2]Methods for solving distributed constraint satisfaction problems are in general more difficult to implement and have higher communication requirements.

| | | |
|---|---|---|
| `status` | = | $s \subset \{\text{busy}, \text{free}\}$ |
| `quality` | = | $c \in [0, 1]$ |
| `fov` | = | $\theta \in [\theta_{\min}, \theta_{\max}]$ degrees |
| `x_turn` | = | $\alpha \in [\alpha_{\min}, \alpha_{\max}]$ degrees |
| `y_turn` | = | $\beta \in [\beta_{\min}, \beta_{\max}]$ degrees |
| `time` | = | $t \in [0, \infty)$ seconds |
| `task` | = | $a \subset \{a_i \mid i = 1, 2, \cdots\}$ |

Fig. 10. The relevance metric returned by a camera node relative to a new task request. The supervisor node converts the metric into a scalar value representing the relevance of the node for the particular surveillance task.

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.

### A. 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 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 supervisor node a relevance metric—a list of attribute-value pairs describing its relevance to the current task along multiple dimensions (Fig. 10). The supervisor node converts this metric into a scalar relevance value $r$ using the following equation:

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

where $\hat{\theta} = (\theta_{\min} + \theta_{\max})/2$, $\hat{\alpha} = (\alpha_{\min} + \alpha_{\max})/2$, and $\hat{\beta} = (\beta_{\min} + \beta_{\max})/2$, and where $\theta_{\min}$ and $\theta_{\max}$ are extremal field of view settings, $\alpha_{\min}$ and $\alpha_{\max}$ are extremal rotation angles around the $x$-axis (up-down), and $\beta_{\min}$ and $\beta_{\max}$ are extremal rotation angles around the $y$-axis (left-right). The values of the variances $\sigma_c$, $\sigma_\theta$, $\sigma_\alpha$, and $\sigma_\beta$ associated with each attribute are chosen empirically (for our experiments, we assigned $\sigma_\theta = \sigma_\alpha = \sigma_\beta = 5.0$ and $\sigma_c = 0.2$).

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.

### B. Surveillance Tasks

We have implemented an interface that presents the operator a display of the synthetic video feeds from multiple virtual surveillance cameras (c.f., Fig. 2). 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 cameras are reassigned to the new task.

## VI. 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 surveillance task fails (and the cameras just return to their default settings).

For the example shown in Fig. 11, 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. 11(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 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 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 supervisor, recruits camera 9, which tracks the pedestrian as she enters the concourse.

## VII. 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* [21], [22] 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 Dr. Tom Strat 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. Mauricio Plaza-Villegas implemented a prototype virtual vision infrastructure within the simulator. We thank Deborah Estrin for introducing us to the sensor networks literature and for her comments on a draft.

## 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, February 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, April 2004, pp. 225–233. [Online]. Available: citeseer.ist.psu.edu/ihler04nonparametric.html

[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, April 2005.

[5] 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.

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

[7] 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, September 1997.

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

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

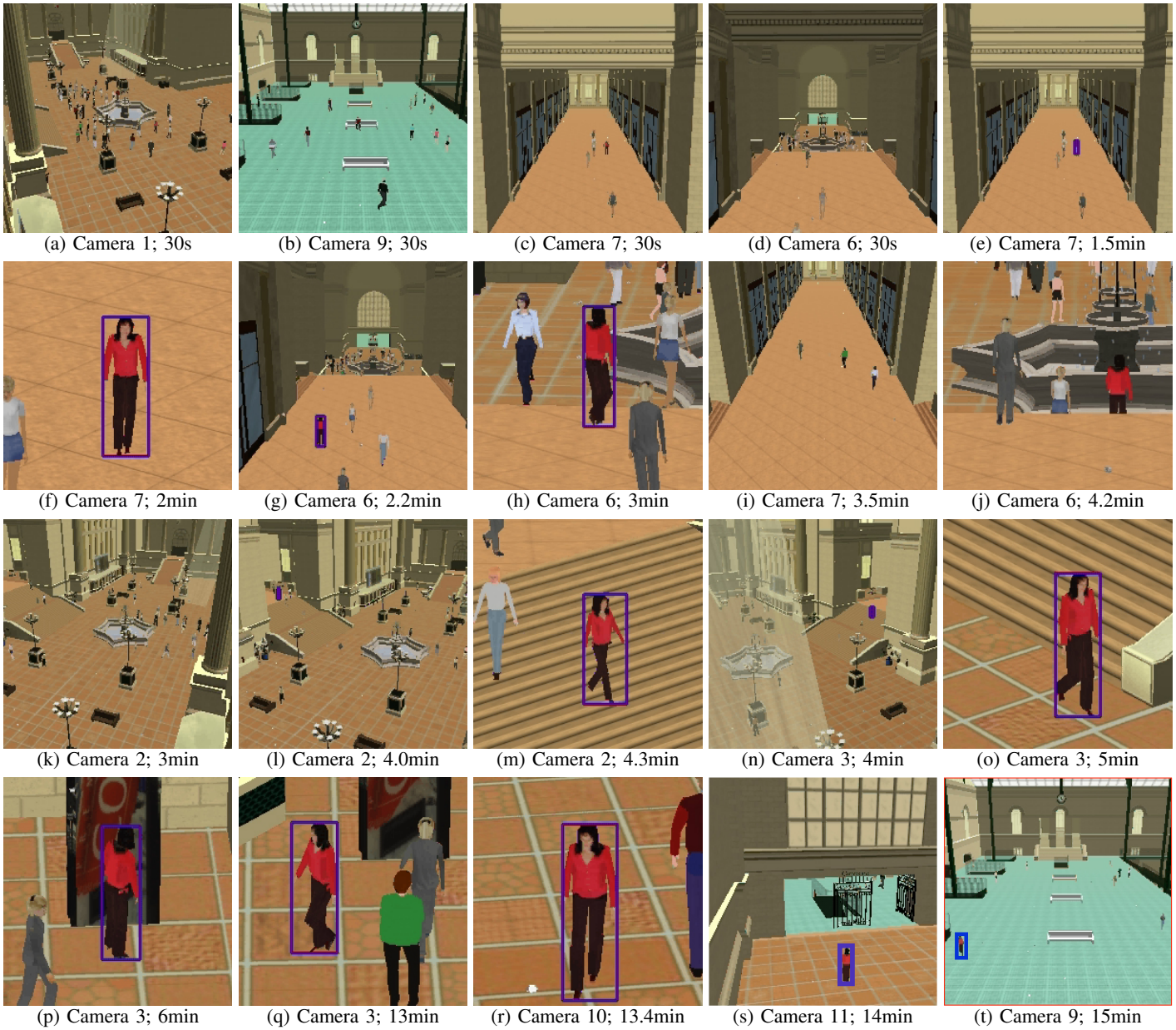
[10] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless adhoc networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Tel Aviv, Israel, March 2000, pp. 22–31.

[11] 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, September 2002, pp. 517–520.

[12] S. Khan and M. Shah, "Consistent labeling of tracked objects in multiple cameras with overlapping fields of view," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1355–1360, October 2003.

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

[14] T. Gandhi and M. M. Trivedi, "Calibration of a reconfigurable array of omnidirectional cameras using a moving person," in *Proc. 2nd ACM International Workshop on Video Surveillance and Sensor Networks*. New York, NY: ACM Press, 2004, pp. 12–19.



(a) Camera 1; 30s (b) Camera 9; 30s (c) Camera 7; 30s (d) Camera 6; 30s (e) Camera 7; 1.5min

(f) Camera 7; 2min (g) Camera 6; 2.2min (h) Camera 6; 3min (i) Camera 7; 3.5min (j) Camera 6; 4.2min

(k) Camera 2; 3min (l) Camera 2; 4.0min (m) Camera 2; 4.3min (n) Camera 3; 4min (o) Camera 3; 5min

(p) Camera 3; 6min (q) Camera 3; 13min (r) Camera 10; 13.4min (s) Camera 11; 14min (t) Camera 9; 15min

Fig. 11. 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.


[15] R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multisensor surveillance," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1456–1477, October 2001.

[16] 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: ACM Press, 2004, pp. 39–45.

[17] N. T. Siebel, "Designing and implementing people tracking applications for automated visual surveillance," Ph.D. dissertation, Dept. of Computer Science. The University of Reading., UK, March 2003.

[18] F. Qureshi and D. Terzopoulos, "Towards intelligent camera networks: A virtual vision approach," in *Proc. The Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS05)*, Beijing, October 2005, pp. 177–184.

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

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

[21] J. Funge, X. Tu, and D. Terzopoulos, "Cognitive modeling: Knowledge, reasoning and planning for intelligent characters," in *Proc. ACM SIGGRAPH 99*, A. Rockwood, Ed., Aug. 1999, pp. 29–38.

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