Multi-Camera Control Through Constraint Satisfaction for Persistent Surveillance

Faisal Z. Qureshi[1] and **Demetri Terzopoulos**[2,1]

[1]Department of Computer Science, University of Toronto, Toronto, ON, Canada
[2]Computer Science Department, University of California, Los Angeles, CA, USA

Abstract

We introduce a distributed camera coalition formation scheme for perceptive scene coverage and persistent surveillance by smart camera sensor networks. The proposed model supports task-dependent camera selection and grouping via a "contract net" task allocation protocol augmented with conflict resolution and error recovery mechanisms. Our technique avoids any central controller, and it is robust to node failures and imperfect communication. In the design and empirical evaluation of our camera networks, we exploit a visually and behaviorally realistic virtual environment simulator that is populated by autonomous, lifelike virtual pedestrians.

1. Introduction

Next-generation multi-camera systems will be smart camera networks. Smart cameras are self-contained vision systems, complete with image sensors, power circuitry, communication interfaces, and on-board processing and storage capabilities. Increasingly sophisticated networks of smart cameras provide new opportunities for the effective visual coverage of large areas—public spaces, disaster zones, battlefields, and even entire ecosystems. These multi-camera systems lie at the intersection of machine vision and sensor networks, raising issues in the two fields that must be addressed simultaneously. For large networks, the sheer volume of data renders human monitoring infeasible. Therefore, it is desirable to design smart camera networks that are capable of performing visual surveillance tasks autonomously, or at least with minimal human intervention.

Our work represents a first attempt to develop a multi-camera system that addresses *high-level camera control* issues relevant both to sensor networks and persistent surveillance. We present a camera sensor model that enables a collection of smart, *uncalibrated* passive and active cameras to provide persistent perceptive coverage of a large public space, such as an airport or train station, with minimal human intervention. Once a pedestrian of interest is identified by a human operator or by an automated detection routine monitoring the video feeds, the cameras decide among themselves how best to maintain the pedestrian under persistent surveillance during his presence in the public space. Typically, any single camera alone cannot adequately monitor the pedestrian, so the cameras must establish collaborations in order to carry out the persistent observation task. The problem of tasking cameras to observe pedestrians becomes even more complex when there are multiple pedestrians of interest. To this end, our contribution is a novel distributed camera network control strategy that is capable of dynamic task-driven node aggregation through local decision making and inter-node communication.

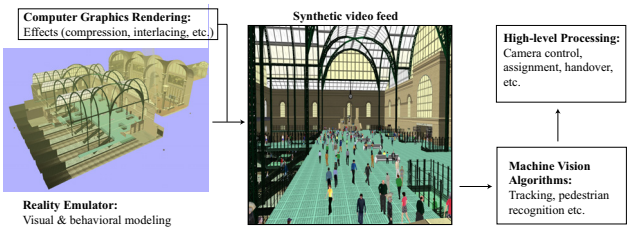


Figure 1: The *Virtual Vision* paradigm.

Cost and legal impediments make it all but infeasible for vision researchers to deploy large-scale physical camera networks in airports or train stations for experimental purposes. Hence, we embrace the *Virtual Vision* paradigm advocated in [19] and developed further in [13, 14], exploiting visually and behaviorally realistic virtual environments to develop and empirically evaluate our camera sensor network algorithms (Fig. 1). In particular, we employ a virtual train station environment populated by autonomous, lifelike virtual pedestrians, which was described in [16]. Easily reconfigurable virtual surveillance cameras situated within the station generate synthetic video feeds that emulate those acquired by real surveillance cameras monitoring public spaces. A few critics are prejudiced against the virtual vision approach simply because research results are obtained through simulations, regardless of how advanced the simulators might be, or might become in the future. Such is

shortsighted criticism, since the virtual vision paradigm not only enables, but greatly facilitates research into low-level and especially high-level vision/control problems that arise in extensive networks of smart cameras under realistic conditions. The obtained results are legitimate and valuable.

Building upon our earlier work [13, 14], which introduced the virtual vision paradigm and demonstrated its general usefulness in camera sensor network research, our specific contribution in the present paper is to address and solve a significant problem in multi-camera systems—camera coordination and collaboration for persistent surveillance. We do so via a new formulation, posed as a constraint satisfaction problem, for resolving camera assignment conflicts when *multiple persistent observation tasks are simultaneously active*.

In a typical sensor network each sensor node has local autonomy and can communicate with a small number of neighboring nodes within radio communication range. To elaborate, our work focuses on distributed, high-level camera control, which confronts many of the research challenges associated with visual sensor networks. Foremost among these are task-based sensor node selection and organization [21]. Distributed approaches to node selection and organization are preferable to centralized approaches and offer what are perhaps the greatest advantages of networked sensing—robustness and scalability.

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 local processing at each node and internode communication. The communication model emulates those found in real sensor networks: 1) nodes can communicate directly with their neighbors, 2) if necessary, a node can communicate with any other node in the network through multi-hop routing. Furthermore, we assume unreliability: 3) internode messages can be delayed, 4) messages can be lost, and 5) nodes can fail. We show that our network remains robust to node and communication link failures despite the lack of a central controller, a feature which also makes the network readily scalable. Visual surveillance tasks are performed by groups of one or more camera nodes. These groups, which are created on the fly, define the information sharing parameters and the extent of collaboration between nodes. A group evolves—i.e., old nodes leave the group and new nodes join it—during the lifetime of the surveillance task. One node in each group acts as the group supervisor and is responsible for group-level decision making. We also present a novel constraint satisfaction problem formulation for resolving group-group interactions.

2. Related Work

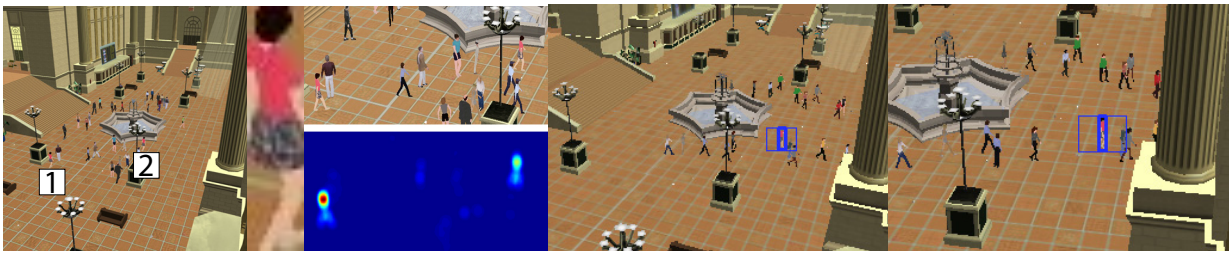
Previous work on camera networks in computer vision has dealt with issues related to low-level and mid-level vision, namely camera network calibration [3] and segmentation, tracking, and identification of moving objects [1]. Some multi-camera systems have addressed the problem of wide-area surveillance [6, 15]. In almost all cases, static cameras are employed and the network is assumed to be calibrated. Little attention has been paid to the problem of controlling active cameras to provide visual coverage of an extensive public space. Notable exceptions are [2, 8, 7]. Typically, master-slave assemblies are employed, where a passive camera controls one or more active cameras, and camera calibration is assumed. Our approach does not require calibration; however, we assume that the cameras can identify a pedestrian with reasonable accuracy, for which we employ color-based pedestrian appearance models.

The problem of forming sensor groups based on task requirements and resource availability has received much attention within the sensor networks community [21]. Mallett [9] argues that task-based grouping in ad hoc camera networks is highly advantageous. Zhao *et al*. [21] introduce 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 information gain versus 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 just communicates with nearby nodes before selecting new nodes.

Camera assignment to various tasks shares many features with Multi-Robot Task Allocation (MRTA) problems studied by the multi-agent systems community [5]. Task-based agent grouping is extensively studied and can be reduced to a Set Partitioning Problem (SPP), which is strongly NP-hard [4]. Fortunately, the sizes of MRTA problems, and by extension SPPs, encountered in our camera sensor network setting are small, because of the spatial/locality constraints inherent to the camera sensors.

Our node grouping strategy is inspired by the "Contract Net" distributed problem solving protocol [17] and realizes group formation via inter-node negotiation. Unlike Mallett's [9] 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 scheme requires specialized nodes for group termination, whereas our strategy handles group leader failures through group merging and group leader demotion operations.

We model conflicts that may arise during camera assignment as an equivalent Constraints Satisfaction Problem, 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 [10] and [20], due to their explosive communication requirements even for small sized



(a) (b) (c) (d) (e)

Figure 2: The operator selects the pedestrian labeled 1 in (a) to construct the pedestrian signature. (b) The operator's selection. Note that there is another pedestrian in the scene (labeled 2) with a similar color distribution as the selected pedestrian. The signature of Pedestrian 1 is constructed without first segmenting the pedestrian, so it contains contributions from the background. (c) Shows a cropped source image (top) and the resulting histogram backprojected image (bottom). Note that Pedestrian 1 is successfully localized in the backprojected image (red hues indicate higher values; blue hues indicate lower values). (d)–(e) Persistently observing the pedestrian during her stay in the main waiting room.

problems. Consequently, our strategy lies somewhere between purely distributed and fully centralized schemes for sensor assignments—sensor assignment is distributed at the scale of the entire network, whereas it is centralized at the scale of a node 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 (LVRs)

The performance of the camera network is ultimately tied to the capabilities of the low-level machine vision routines responsible for gathering the sensory data. Consequently, when working with camera networks within the virtual vision paradigm, it is important to make accurate assumptions about the capabilities and limitations of the low-level visual sensing processes. We ensure that our assumptions about the low-level visual sensing are qualitatively correct by implementing a pedestrian tracking system that operates solely upon the synthetic video captured by the virtual cameras. Local vision routines mimic the performance of a state-of-the-art pedestrian segmentation and tracking module. In particular, pedestrian tracking can fail due to occlusions, poor segmentation, bad lighting, or crowding. 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. We match pedestrian signatures across frames through color indexing [18]. Color indexing efficiently identifies objects present in an image using object color distributions in the presence of occlusions as well as scale and viewpoint changes. In color indexing, targets with similar color distributions are detected and localized through histogram backprojection, which finds the target in an image by emphasizing colors in the image that belong to the target being observed (Fig. 2(c)). The last step of the color indexing procedure assumes that the area of the target in the image is known a priori. Active PTZ cameras violate the above assumption, as the area covered by the target in the image can vary drastically depending upon the current zoom settings of the camera. 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.

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. 3).

In its default state, *Idle*, the camera node is not involved in any task. A camera node transitions into the *ComputingRelevance* state upon receiving a *queryrelevance* message from a nearby node. Using the description of the task that is contained within the *queryrelevance* message and by employing the LVRs, the camera node can compute its relevance to the task, as we will explain in the next section. For example, a camera can use visual search to find a pedestrian that matches the appearance-based signature transmitted 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

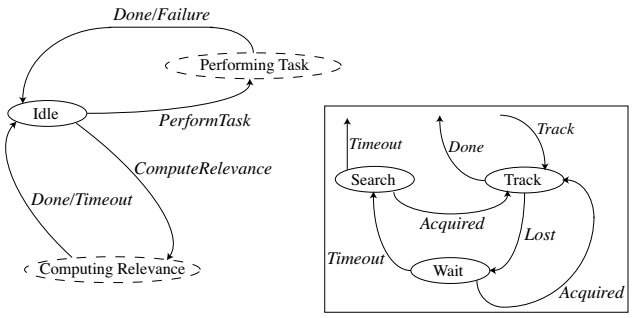


Figure 3: Top-level camera controller. Dashed states contain the child finite state machine shown in the inset (lower right).

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 the *Performing-Task* state upon joining a group, where the embedded child finite state machine (Fig. 3 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 that 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. 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. The *zoom* routine controls the field of view (FOV) of the camera such that the region of interest occupies the desired percentage of the image. See [12] 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 tilt/zoom settings and the default tilt/zoom settings.

3.3. 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 to turn and zoom might be able to follow a pedestrian for a longer time, and 4) it is better to avoid unnecessary reassignments of cameras to different tasks, as doing so may degrade the performance of the underlying computer vision routines.

Upon receiving a task request, a camera node returns to

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

Figure 4: 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 to the particular surveillance task.

the leader node a relevance metric—a list of attribute-value pairs describing its relevance to the current task along multiple dimensions (Fig. 4). The leader node converts this metric into a scalar relevance value r as follows:

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

Here $\hat{\theta} = (\theta_{\min} + \theta_{\max})/2$, $\hat{\alpha} = (\alpha_{\min} + \alpha_{\max})/2$, and $\hat{\beta} = (\beta_{\min} + \beta_{\max})/2$, where $\theta_{\min}$ and $\theta_{\max}$ are extremal field of view settings, $\alpha_{\min}$ and $\alpha_{\max}$ are extremal (tilt) rotation angles around the x-axis, and $\beta_{\min}$ and $\beta_{\max}$ are extremal (pan) rotation angles around the y-axis, $0.3 \leq \sigma_c \leq 0.33$, $\sigma_\theta = (\theta_{\max} - \theta_{\min})/6$, $\sigma_\alpha = (\alpha_{\max} - \alpha_{\min})/6$, and $\sigma_\beta = (\beta_{\max} - \beta_{\min})/6$. The value of γ is chosen empirically ($\gamma = 1000$ in our experiments).

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 the contract net protocol that models auctions (announcement, bidding, and selection) for group formation [17]. Local computation at each node involves choosing an appropriate bid for the announced sensing task.

From the standpoint of operator interaction, we distinguish between 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 our 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 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 auctions the task in the hope of finding suitable nodes. A subset N' of N_n respond by sending their relevance values for the current task (*relevance* message). This is the bidding phase. Upon receiving the relevance values, node n selects a subset M of N' to include in the group, and sends *join* messages to the chosen nodes. This is the selection phase. When there is no resource contention between groups—e.g., when only one task is active, or when multiple tasks that do not require the same nodes for successful operation are active—the selection process is relatively straightforward; node n picks those nodes from N' that have the highest relevance values. 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 by sending a *queryrelevance* message to the 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 includes 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 among the member nodes. The group vanishes when the last node leaves the group.

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

4.2. Conflict Resolution

A conflict resolution mechanism is needed when multiple groups require the same resources. We treat the problem of assigning sensors to the contending groups as a Constraint Satisfaction Problem (CSP) [11]. Formally, a CSP consists of a set of variables $\{v_1, v_2, v_3, \ldots, v_k\}$, 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 $\{C_1, C_2, C_3, \ldots, C_m\}$. The solution to the CSP is a set $\{v_i \leftarrow a_i \,|\, a_i \in \text{Dom}[v_i]\}$, where the a_i 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. Consider, 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 $G = \{g_1, g_2, \ldots, g_k\}$ 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 solve smaller CSP problems by relaxing the node requirements for each task.

When it is possible to compare the quality of a partial solution to that of a full solution, we can store the best result so far and backtrack whenever the current partial solution is of poorer quality. Using this strategy, we can guarantee an optimal solution under the assumption that the quality of solutions increase monotonically as values are assigned to more variables. For example, compare test cases 1 and 2 in Table 1. The goal was to assign 3 sensors each to the two groups. Optimal assignments were found in both cases; however, *BestSolv*, which employs backtracking based on the quality of the partial solution, visited only 175 nodes to find the optimal solution as opposed to *AllSolv*, which visited 29290 nodes. *AllSolv* enumerates every solution to find the optimal sensor assignment. The same trend is observed in columns 3 and 4 in Table 1. The *BestSolv* solver clearly outperforms the *AllSolv* solver in finding the opti-

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

Table 1: Finding optimal sensor node assignment. The problem is to assign three sensors each to two groups. The average number of relevant nodes for each group is 12 and 16. *AllSolv* finds all solutions, ranks them, and picks the best one, whereas *BestSolv* computes the optimal solution by storing the best solution so far and backtracking when partial assignment yields a poorer solution. As expected, *BestSolv* outperforms *AllSolv*.

mal node assignment. Of course, when operating within time/resource limitations, we can always choose the first solution or pick the best solution after a predetermined number of nodes have been explored.

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.

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 and Communication Errors

The proposed 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, it concludes that the other node is malfunctioning. A node failure is assumed when the leader node does not receive a *status* message 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 group leader.

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 but one of the group leader nodes. Demotion is either carried out based upon the unique ID assigned to each node—among the conflicting nodes, the one with the highest ID is selected to be the group leader—or when unique node IDs are not guaranteed, demotion can be carried out via a contention management scheme based on the ALOHA network protocol. See [12] for the details of the demotion process.

[2]The leader node where centralization occurs is selected using a strategy similar to that used for group merging (see [12] for the details).

5. Persistent Surveillance

We now consider how a sensor network of dynamic cameras may be used in the context of video surveillance.

We have implemented an interface that presents to the operator a display of the synthetic video feeds from the 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 of the pedestrian, or 3) zoom-in and follow the pedestrian. The successful execution and completion of these tasks requires the intelligent allocation and scheduling of the available cameras; in particular, the network must decide which cameras should track the pedestrian and for how long. The network automatically assigns cameras to fulfill the task requirements. If the operator initiates multiple tasks, either cameras that are not currently occupied are chosen to fulfill the new task or some currently occupied cameras are reassigned to the new task.

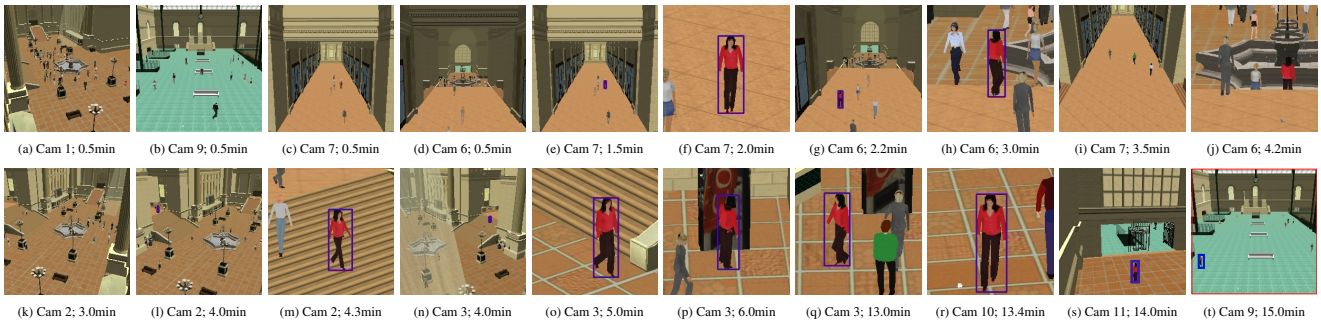
A detailed world model that includes the location of cameras, their fields of view, pedestrian motion prediction models, occlusion models, and pedestrian movement pathways may enable, in some sense, the *optimal* allocation and scheduling of cameras; however, it is cumbersome and in most cases infeasible 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. A direct consequence of this approach is that the network can easily be modified through removal, addition, or replacement of camera nodes.

The computed relevance values (Section 3.3) 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 too few free nodes are available for the current task.

5.1. 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 train 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).

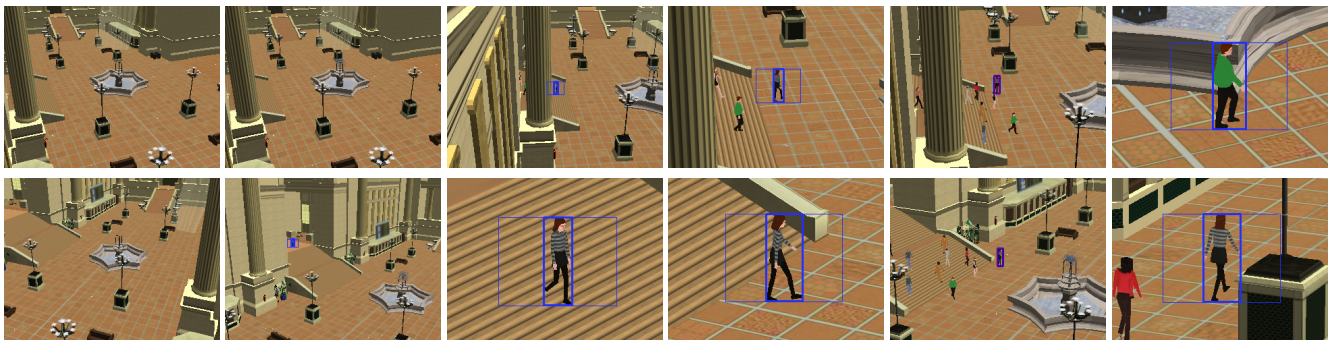
For the example shown in Fig. 5, we placed 16 active PTZ cameras in the train station, as shown in Fig. 6. An op-



(a) Cam 1; 0.5min (b) Cam 9; 0.5min (c) Cam 7; 0.5min (d) Cam 6; 0.5min (e) Cam 7; 1.5min (f) Cam 7; 2.0min (g) Cam 6; 2.2min (h) Cam 6; 3.0min (i) Cam 7; 3.5min (j) Cam 6; 4.2min

(k) Cam 2; 3.0min (l) Cam 2; 4.0min (m) Cam 2; 4.3min (n) Cam 3; 4.0min (o) Cam 3; 5.0min (p) Cam 3; 6.0min (q) Cam 3; 13.0min (r) Cam 10; 13.4min (s) Cam 11; 14.0min (t) Cam 9; 15.0min

Figure 5: A pedestrian is successively tracked by Cameras 7, 6, 2, 3, 10, and 9 (see Fig. 6) as she makes her way through the station to the concourses. (a-d) Four cameras observing the station. (e) The operator selects a pedestrian in video 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 Camera 6; Camera 3 has acquired the pedestrian. (o) Camera 3 zooming in on the pedestrian. (p) The pedestrian is at the vending machine. (q) The pedestrian is walking towards the concourses. (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.



(a) (b) (c) (d) (e) (f)

Figure 7: Camera assignment and conflict resolution. Cameras 1 (upper row) and 2 (lower row) are set to observe the first pedestrian who enters the main waiting room (a). Camera 2 starts observing the pedestrian as soon as she enters the scene (b). (c)-(d) Camera 1 recognizes the target pedestrian by using the pedestrian signature computed by Camera 2. Cameras 1 and 2 successfully form a group to observe the first pedestrian. (e) The operator initiates another goal for the camera network, which is to observe the pedestrian wearing green. The two cameras then pan out to visually search for the green pedestrian and decide between them each to carry out a different task. (f) Camera 1 is deemed more suitable for observing the second pedestrian wearing green, whereas Camera 2 continues observing the first pedestrian.

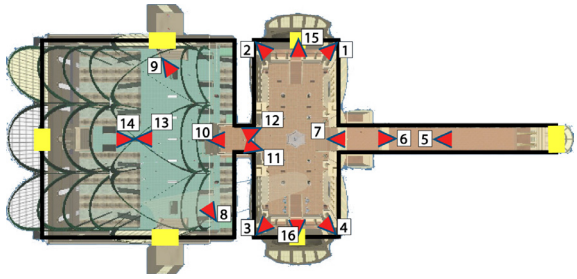


Figure 6: Plan view of the virtual Penn Station environment with the roof not rendered, revealing the concourses and train tracks (left), the main waiting room (center), and the long shopping arcade (right); cf. the left image in Fig. 1. (The yellow rectangles indicate station pedestrian portals.) An example camera network is shown comprising 16 simulated active (pan-tilt-zoom) video surveillance cameras.

erator selects the pedestrian with the red shirt in Camera 7 (Fig. 5(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 of the station, where she also visits a vending machine. When the pedestrian starts walking towards the concourses section of the station, 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, now acting as the group's leader, recruits Camera 9, which tracks the pedestrian as she enters the concourses.

Fig. 7 illustrates camera assignment and conflict reso-

lution. Cameras 1 and 2 successfully form a group to observe the first pedestrian who enters the scene, as there is only one active task. On the other hand, when the operator specifies a second task—follow the pedestrian wearing the green sweater—the cameras decide to break the group and reassign themselves. Among themselves, the cameras decide that Camera 1 is better suited for observing the pedestrian wearing green. Camera 2 continues observing the first pedestrian who entered the scene. It bears repeating that the cameras are able to handle the two observation tasks completely autonomously. Additionally, the interaction between the two cameras is strictly local—other cameras present in the camera network (Fig. 6) are not involved.

6. Conclusion

We envision future video surveillance systems to involve networks of stationary and active cameras capable of providing persistent perceptive coverage of extended environments with minimal reliance on human operators. Such highly-automated, intelligent surveillance 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, with promising results.

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 innovative 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. Moreover, our approach does not assume a detailed world model, which is generally hard to acquire. Since it lacks any central controller, we expect our proposed approach to be robust and scalable.

We are currently pursuing a "Cognitive Modeling" approach to node organization and camera scheduling. We are also investigating scalability and node failure issues. Moreover, we are constructing more elaborate scenarios involving more cameras situated in different locations within the train station, with which we would like to study the performance of the camera sensor network when it is required to persistently observe multiple pedestrians throughout their entire stay in the train station.

References

[1] R. Collins, O. Amidi, and T. Kanade. An active camera system for acquiring multi-view video. In *Proc. International Conf. on Image Processing*, pages 517–520, Rochester, NY, Sep 2002. 2

[2] R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE*, 89(10):1456–1477, Oct 2001. 2

[3] D. Devarajan, R. J. Radke, and H. Chung. Distributed metric calibration of ad hoc camera networks. *ACM Transactions on Sensor Networks*, 2(3):380–403, 2006. 2

[4] M. R. Garey and D. S. Johnson. "Strong" NP-completeness results: Motivation, examples, and implications. *Journal of the ACM*, 25(3):499–508, 1978. 2

[5] B. Gerkey and M. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954, 2004. 2

[6] D. Hall, et al. Comparison of target detection algorithms using adaptive background models. In *Proc. Joint IEEE International Workshop on Visual Surveillance... (VS-PETS05)*, pages 113–120, Beijing, China, Oct 2005. 2

[7] A. Jain, D. Kopell, K. Kakligian, and Y. Wang. Using stationary-dynamic camera assemblies for wide-area video surveillance and selective attention. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 06)*, volume 1, pages 537–544, New York, NY, Jun 2006. 2

[8] S.-N. Lim, L. S. Davis, and A. Elgammal. A scalable image-based multi-camera visual surveillance system. In *Proc. IEEE Conf. on Advanced Video and Signal Based Surveillance*, pages 205–212, Washington, DC, 2003. 2

[9] J. Mallett. *The Role of Groups in Smart Camera Networks*. PhD thesis, Dept. of Media Arts and Sciences, Massachusetts Institute of Technology, Feb 2006. 2

[10] 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. 2

[11] 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, Jul 1997. 5

[12] F. Qureshi, *Intelligent Perception in Virtual Camera Networks and Space Robotics*, PhD thesis, Dept. of Computer Science, University of Toronto, Jan 2007. 4, 6

[13] F. Qureshi and D. Terzopoulos. Surveillance camera scheduling: A virtual vision approach. *ACM Multimedia Systems Journal*, 12:269–283, Dec 2006. 1, 2

[14] F. Qureshi and D. Terzopoulos. Surveillance in virtual reality: System design and multicamera control. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 07)*, pages 1–8, Minneapolis, MN, Jun 2007. 1, 2

[15] M. Shah, O. Javed, and K. Shafique. Automated visual surveillance in realistic scenarios. In *IEEE Multimedia*, volume 14, pages 30–39, Mar 2007. 2

[16] W. Shao and D. Terzopoulos. Autonomous pedestrians. *Graphical Models*, 69(5-6):246–274, Sep/Nov 2007. 1

[17] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. on Computers*, C-29(12):1104–1113, Dec 1980. 2, 4

[18] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, Nov 1991. 3

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

[20] M. Yokoo. *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent Systems*. Springer-Verlag, Berlin, Germany, 2001. 2

[21] 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. 2