Distributed Coalition Formation in Visual Sensor Networks: A Virtual Vision Approach

Faisal Qureshi[1] and Demetri Terzopoulos[1,2]

[1] Dept. of Computer Science, University of Toronto, Toronto, ON, Canada
faisal@cs.toronto.edu
[2] Computer Science Dept., University of California, Los Angeles, CA, USA
dt@cs.ucla.edu

Abstract. We propose a distributed coalition formation strategy for collaborative sensing tasks in camera sensor networks. The proposed model supports task-dependent node selection and aggregation through an announcement/bidding/selection strategy. It resolves node assignment conflicts by solving an equivalent constraint satisfaction problem. Our technique is scalable, as it lacks any central controller, and it is robust to node failures and imperfect communication. Another unique aspect of our work is that we advocate visually and behaviorally realistic virtual environments as a simulation tool in support of research on large-scale camera sensor networks. Specifically, our visual sensor network comprises uncalibrated static and active simulated video surveillance cameras deployed in a virtual train station populated by autonomously self-animating pedestrians. The readily reconfigurable virtual cameras generate synthetic video feeds that emulate those generated by real surveillance cameras monitoring public spaces. Our simulation approach, which runs on high-end commodity PCs, has proven to be beneficial because this type of research would be difficult to carry out in the real world in view of the impediments to deploying and experimenting with an appropriately complex camera network in extensive public spaces.

Keywords: Camera sensor networks, Sensor coordination and control, Distributed coalition formation, Video surveillance.

1 Introduction

Camera sensor networks are becoming increasingly important to next generation applications in surveillance, in environment and disaster monitoring, and in the military. In contrast to current video surveillance systems, camera sensor networks are characterized by smart cameras, large network sizes, and ad hoc deployment.[1] These systems lie at the intersection of machine vision and sensor networks, raising issues in the two fields that must be addressed simultaneously. The effective visual coverage of extensive areas—public spaces, disaster zones, and battlefields—requires multiple cameras to collaborate towards common sensing goals. As the size of the camera network grows, it

[1] Smart cameras are self-contained vision systems, complete with image sensors, power circuitry, communication interfaces, and on-board processing capabilities [1].

J. Aspnes et al. (Eds.): DCOSS 2007, LNCS 4549, pp. 1–20, 2007.

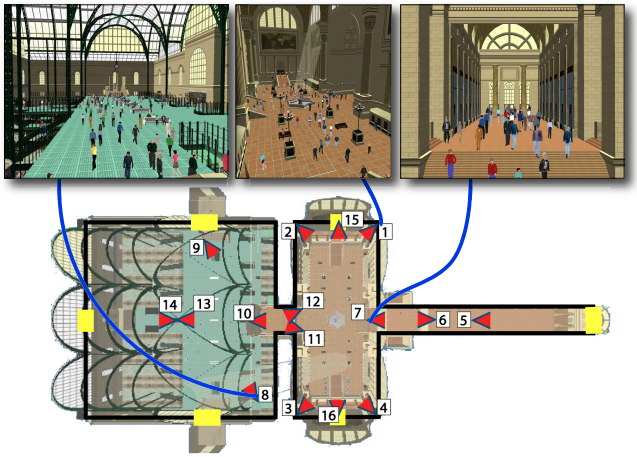


Fig. 1. Plan view of the virtual Penn Station environment with the roof not rendered, revealing the concourses and train tracks (left), the main waiting room (center), and the long shopping arcade (right). (The yellow rectangles indicate station pedestrian portals.) An example visual sensor network comprising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

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

In this paper, we demonstrate a camera network model comprising *uncalibrated* passive and active simulated video cameras that with minimal operator assistance can perform persistent surveillance of a large virtual public space—a train station populated by autonomously self-animating virtual pedestrians (Fig. 1). Once a human operator or an automated visual behavior analysis routine monitoring the surveillance video feeds identified a pedestrian of interest, the cameras decide amongst themselves how best to observe the subject. For example, a subset of the active pan/tilt/zoom (PTZ) cameras can collaboratively track the pedestrian as (s)he weaves through the crowd. The problem of assigning cameras to persistently monitor pedestrians becomes challenging when there are multiple pedestrians of interest. To deal with the numerous possibilities, the cameras must be able to *reason* about the dynamic situation. To this end, we propose a distributed camera network control strategy that is capable of dynamic task-driven node aggregation through local decision making and inter-node communication.

1.1 Virtual Vision

Deploying a large-scale camera sensor network in the real world is a major undertaking whose cost can easily be prohibitive for most researchers interested in designing and experimenting with sensor networks. Moreover, privacy laws generally restrict the monitoring of people in public spaces for experimental purposes. As a means of over-

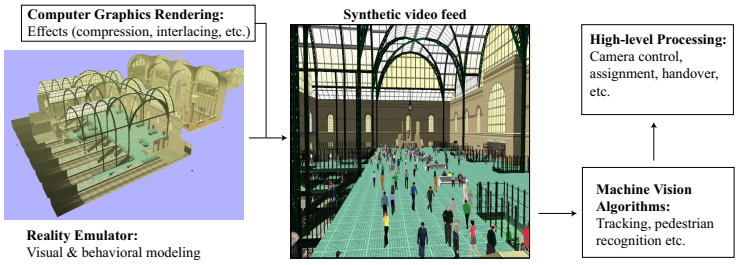


Fig. 2. Virtual vision paradigm (image from [2])

coming these impediments, we advocate the pursuit of camera sensor network research in the context of a unique synthesis of advanced computer graphics and vision simulation technologies. In particular, we demonstrate the design of simulated camera sensor network systems and meaningful experimentation with such systems within visually and behaviorally realistic virtual environments (Fig. 2).

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

1.2 Distributed Control in Camera Sensor Networks

Our work deals with distributed control in camera sensor networks and many of the characteristic challenges associated with sensor networks are relevant. Task-based sensor selection is a fundamental issue in sensor networks [3]. The selection process must take into account the information contribution of each node against its resource consumption or potential utility in other tasks. Another key issue in sensor networks is node organization, which has been proposed by researchers as a means to limit the communication to those nodes that are relevant to the task at hand. Distributed approaches for node selection or node organization are preferable to centralized approaches and offer what are perhaps the greatest advantages of networked sensing—robustness and scalability. Also, in a typical sensor network, each sensor node has local autonomy and can communicate with a small number of neighboring nodes that are within radio communication range.

Mindful of these issues, we propose a novel camera sensor network control strategy that does not require camera calibration, a detailed world model, or a central controller. We model virtual cameras as nodes in a communication network that emulates those found in physical sensor networks: 1) nodes can communicate directly with their neighbours, 2) if necessary, a node can communicate with another node in the network through multi-hop routing, and 3) unreliable communication. The overall behavior of the network is the consequence of the local processing at each node and inter-node communication. The network is robust to node and communication link failures; moreover, it is scalable due to the lack of a central controller. Visual surveillance tasks are performed by groups of one or more camera nodes. These groups, which are created on the fly, define the information sharing parameters and the extent of collaboration between nodes. During the lifetime of the surveillance task, a group evolves—i.e., old nodes leave the group and new nodes join it. One node in each group acts as the group leader and is responsible for group-level decision making. We also present a new constraint satisfaction problem formulation for resolving group interactions.

1.3 Overview

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

2 Related Work

As was argued in [4, 5], computer graphics and virtual reality technologies are rapidly presenting viable alternatives to the real world for developing sensory systems (see also [6]). Our camera network is deployed and tested within the virtual train station simulator that was developed in [2]. The simulator incorporates a large-scale environmental model (of the original Pennsylvania Station in New York City) with a sophisticated pedestrian animation system. The simulator can efficiently synthesize well over 1000 self-animating pedestrians performing a rich variety of activities in the extensive indoor urban environment. Like real humans, the synthetic pedestrians are fully autonomous. They perceive the virtual environment around them, analyze environmental situations, make decisions and behave naturally within the train station. Standard computer graphics techniques enable a photorealistic rendering of the busy urban scene with considerable geometric and photometric detail (Fig. 1).

The problem of forming sensor groups based on task requirements and resource availability has received much attention within the sensor networks community [3]. Reference [1] argues that task-based grouping in ad hoc camera networks is highly advantageous. Collaborative tracking, which subsumes the above issue, is considered an

essential capability in many sensor networks [3]. Reference [7] introduces an information driven approach to collaborative tracking, which attempts to minimize the energy expenditure at each node by reducing inter-node communication. A node selects the next node by utilizing the information gain vs. energy expenditure tradeoff estimates for its neighbor nodes. In the context of camera networks, it is often difficult for a camera node to estimate the expected information gain by assigning another camera to the task without explicit geometric and camera-calibration knowledge, but such knowledge is tedious to obtain and maintain during the lifetime of the camera network. Therefore, our camera networks do without such knowledge; a node needs to communicate with nearby nodes in order to select new nodes.

Nodes comprising sensor networks are usually untethered sensing units with limited onboard power reserves. Consequently, a crucial concern in sensor networks is the energy expenditure at each node, which determines the life-span of a sensor network [8]. Node communications have large power requirements; therefore, sensor network control strategies attempt to minimize the inter-node communication [9, 7]. Presently, we do not address this issue. However, the communication protocol that we propose limits the communication to the active nodes and their neighbors.

Little attention has been paid in computer vision to the problem of controlling active cameras to provide visual coverage of an extensive public area, such as a train station or an airport [10,11]. 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 [12], and camera network calibration [13]. Our approach does not require calibration; however, we assume that the cameras can identify a pedestrian with reasonable accuracy. To this end, we employ color-based pedestrian appearance models.

IrisNet is a sensor network architecture tailored towards high-capability multimedia sensors connected via high-capacity communication channels [14]. It takes a *centralized* view of the network and models it as a distributed database, allowing efficient access to sensor readings. We consider this work to be orthogonal to ours. SensEye is a recent sensor-network inspired multi-camera systems [15]. It demonstrates the benefits of a multi-tiered network—each tier defines a set of sensing capabilities and corresponds to a single class of smart camera sensors—over single-tiered networks in terms of low-latencies and energy efficiency. However, SensEye does not deal with the distributed camera control issues that we address.

Our node grouping strategy is inspired by the ContractNet distributed problem solving protocol [16] and realizes group formation via inter-node negotiation. Unlike Mallett's [1] approach to node grouping where groups are defined implicitly via membership nodes, our approach defines groups explicitly through group leaders. This simplifies reasoning about groups; e.g., Mallett's approach requires specialized nodes for group termination. Our strategy handles group leader failures through group merging and group leader demotion operations.

Resolving group-group interactions requires sensor assignment to various tasks, which shares many features with Multi-Robot Task Allocation (MRTA) problems studied by the multi-agent systems community [17]. Specifically, according to the taxonomy provided in [17], our sensor assignment formulation belongs to the single-task

robots (ST), multi-robot tasks (MR), instantaneous assignment (IA) category. ST-MR-IA problems are significantly more difficult than single robot task MTRA problems. Task-based robot grouping arise naturally in ST-MR-IA problems, which are sometimes referred to as *coalition formation*. ST-MR-IA problems are extensively studied and can be reduced to a Set Partitioning Problem (SPP), which is strongly NP-hard [18]. However, many heuristics-based set partitioning algorithms exist that produce good results on large SPPs [19]. Fortunately, the sizes of MRTA problems, and by extension SPPs, encountered in our camera sensor network setting are small due to the spatial/locality constraints inherent to the camera sensors.

We model sensor assignments as a Constraint Satisfaction Problem (CSP), which we solve using "centralized" backtracking. Each sensor assignment that passes the hard constraints is assigned a weight, and the assignment with the highest weight is selected. We have intentionally avoided distributed constraint optimization techniques, such as [20] and [21], due to their explosive communication requirements even for small sized problems. Additionally, it is not obvious how they handle node and communication failures. Our strategy lies somewhere between purely distributed and fully centralized schemes for sensor assignments—sensor assignment is distributed at the level of the network, whereas it is centralized at the level of a group.

3 Camera Nodes

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

3.1 Local Vision Routines

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

We employ appearance-based models to track pedestrians. Pedestrians are segmented to construct unique and robust color-based signatures (appearance models), which are then matched across the subsequent frames. Color-based signatures have found widespread use in tracking applications [22], but they are sensitive to illumination changes. However, this shortcoming can be mitigated by operating in HSV space instead of RGB space. Furthermore, zooming can drastically change the appearance of a pedestrian, thereby confounding conventional appearance-based schemes. We employ a

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

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

modified *color-indexing* scheme [23] to tackle this problem. Thus, a distinctive characteristic of our pedestrian tracking routine is its ability to operate over a range of camera zoom settings. It is important to note that we do not assume camera calibration. See [24] for more details.

3.2 Camera Node Behaviors

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

In its default state, *Idle*, the camera node is not involved in any task. A camera node transitions into the *ComputingRelevance* state upon receiving a *queryrelevance* message from a nearby node. Using the description of the task that is contained within the *queryrelevance* message and by employing the LVRs, the camera node can 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

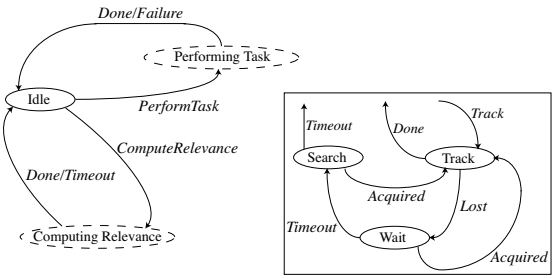


Fig. 4. Top-level camera controller. Dashed states contain the child finite state machine shown in the inset.

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

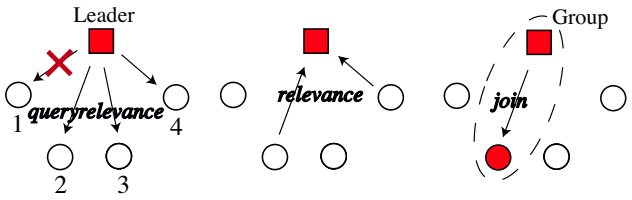
Each camera can *fixate* and *zoom* in on an object of interest. Fixation and zooming routines are image driven and do not require any 3D information, such as camera calibration or a global frame of reference. We discovered that traditional Proportional Derivative (PD) controllers generate unsteady control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

The *fixate* routine brings the region of interest—e.g., a pedestrian's bounding box—into the center of the image by tilting the camera about its local x and y axes (Fig. 3(b)). The *zoom* routine controls the FOV of the camera such that the region of interest occupies the desired percentage of the image (Fig. 3(c)). See [24] for the details.

A camera node returns to its default stance after finishing a task using the *reset* routine, which is a PD controller that attempts to minimize the error between the current zoom/tilt settings and the default zoom/tilt settings (Fig. 3(d)).

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



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

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

sensing request to one of the nodes. In response to this request, relevant nodes self-organize into a group with the aim of fulfilling the sensing task. The *group*, which formalizes the collaboration between member nodes, is a dynamic arrangement that evolves throughout the lifetime of the task. At any given time, multiple groups might be active, each performing its respective task. Group formation is determined by the local computation at each node and the communication between the nodes. Specifically, we employ the ContractNet protocol, which models auctions (announcement, bidding, and selection) for group formation [16] (see Fig. 5). The local computation at each node involves choosing an appropriate bid for the announced sensing task.

From the standpoint of user interaction, we have identified two kinds of sensing queries: 1) where the queried sensor itself can measure the phenomenon of interest—e.g., when a human operator selects a pedestrian to be tracked within a particular video feed—and 2) when the queried node might not be able to perform the required sensing and needs to route the query to other nodes. For instance, an operator can request the network to count the number of pedestrians wearing green shirts. To date we have experimented only with the first kind of queries, which are sufficient for setting up collaborative tracking tasks; however, this is by no means a limitation of the proposed communication model.

4.1 Coalition Formation

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

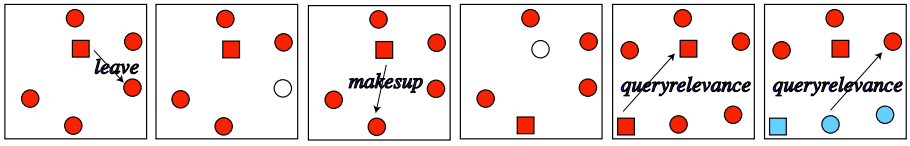


Fig. 6. (a)-(b) A node leaves a group after receiving a leave message from the group leader. (c)-(d) Old group leader selects a new group leader and leaves the group. (e) A leader node detects another leader performing the same task; leader/supervisor demotion commences. (f) Conflict detection between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

the next section. A node that is not already part of any group can join the group upon receiving a *join* message from the leader of that group. After receiving the *join* message, a subset M' of M elect to join the group.

For multinode groups, if a group leader decides to recruit more nodes for the task at hand, it instructs group nodes to broadcast task requirements. This is accomplished via sending *queryrelevance* to group nodes. The leader node is responsible for group-level decisions, so member nodes forward to the group leader all the group-related messages, such as the *relevance* messages from potential candidates for group membership. During the lifetime of a group, group nodes broadcast *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.[2] Similarly, a group node can choose to stop performing the task when it detects that its relevance value is below a certain threshold. When a leader detects that its own relevance value for the current task is below the predefined threshold, it selects a new leader from amongst the member nodes. The group vanishes when the last node leaves the group.

4.2 Conflict Resolution

A conflict resolution mechanism is needed when multiple groups require the same resources (Fig. 6(f)). The problem of assigning sensors to the contending groups can be treated as a Constraint Satisfaction Problem (CSP) [25]. Formally, a CSP consists of a set of variables $\{v_1, v_2, v_3, \cdots, v_k\}$, a set of allowed values Dom$[v_i]$ for each variable v_i (called the domain of v_i), and a set of constraints $\{C_1, C_2, C_3, \cdots, C_m\}$. The solution to the CSP is a set $\{v_i \leftarrow a_i | a_i \in \text{Dom}[v_i]\}$, where the a_is satisfy all the constraints.

We treat each group g as a variable, whose domain consists of the non-empty subsets of the set of sensors with relevance values (with respect to g) greater than a predefined

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

threshold. The constraints restrict the assignment of a sensor to multiple groups. Assume, for example, a group g and a set of nodes $\{n_1, n_2, n_3\}$ with relevance values $\{r_1, r_2, r_3\}$, respectively. If r_3 is less than the predefined threshold, the set of nodes that will be considered for assignment to g is $\{n_1, n_2\}$, and the domain of g is the set $\{\{n_1\}, \{n_2\}, \{n_1, n_2\}\}$. We define a constraint C_{ij} as $a_i \cap a_j = \{\Phi\}$, where a_i and a_j are sensor assignments to groups g_i and g_j, respectively; k groups give rise to $k!/2!(k-2)!$ constraints.

We can then define a CSP as $P = (G, D, C)$, where $G = \{g_1, g_2, \cdots, g_k\}$ is the set of groups (variables) with non-empty domains, $S = \{\text{Dom}[g_i] | i \in [1, k]\}$ is the set of domains for each group, and $C = \{C_{ij} | i, j \in [1, k], i \neq j\}$ is the set of constraints. To solve P, we employ *backtracking* to search systematically through the space of possibilities. We find all solutions, rank these solutions according to the relevance values for sensors (with respect to each group), and select the best solution to find the optimal assignments. The solution ranking procedure can easily incorporate other relevant concerns such as a preference for sensors that are positioned orthogonal to each other with respect to the pedestrian so as to increase the position estimate accuracy or using sensors that are within one hop distance of each other. When P has no solution, When P has no solution, we solve smaller CSPs by relaxing the node requirements for each task.

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.[3] The result is then conveyed to the other leader 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 and because it can easily cope with the size of problems encountered in the current setting. A key feature of our conflict resolution scheme is centralization, which requires that all the relevant information be gathered at the node that is responsible for solving the CSP. For smaller CSPs, the cost of centralization is easily offset by the speed and ease of solving the CSP.

Solving the CSP. Any solution of the above CSP P is a valid sensor node assignment; however, some solutions are better than others as not all nodes are equally suitable for any given sensing task. The node relevance value with respect to a group quantifies the suitability of the node to the task performed by that group, and we can view the quality of a solution as a function of the quality of sensor assignments to different groups. In a restrictive setting, we can define the quality of a solution to be the sum of the quality of sensor assignments to individual groups.

When it is possible to compare the quality of a partial solution to that of a full solution, we can store the currently best result and backtrack whenever the current partial solution is of poorer quality. Using this strategy, we can guarantee an optimal solution

[3] Leader node where centralization occurs is selected using a strategy similar to that used for group merging (Fig. 7).

Table 1. Finding an 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. *AllSolu* finds all solutions, ranks them, and picks the best one, whereas *BestSolu* computes the optimal solution by storing the currently best solution and backtracking when partial assignment yields a poorer solution. As expected, the *BestSolu* solver outperforms the *AllSolu*solver.

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

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, *BestSolu*, which employs backtracking based on the quality of the partial solution, visited only 175 nodes to find the optimal solution, as opposed to *AllSolu*, which visited 29290 nodes. The *AllSolu* solver enumerates every solution to find the optimal sensor assignment. The same trend is observed in columns 3 and 4 in the table. The *BestSolu* solver clearly outperforms the *AllSolu* solver in finding the optimal node assignment. Of course, when operating under time/resource constraints, we can always choose the first solution or pick the best solution after a predetermined number of nodes have been explored.

4.3 Node Failures and Communication Errors

The purposed communication model takes into consideration node and communication failures. Communication failures are perceived as sensor failures; for example, when a node is expecting a message from another node, and the message never arrives, the first node concludes that the second node is malfunctioning. A node failure is assumed when the leader node does not receive a *status* from the node during some predefined interval, and the leader node removes the problem node from the group. On the other hand, when a member node does not receive any message (*status* or *queryrelevance*) from the leader node during a predefined interval, it assumes that the leader node has experienced a failure and selects itself to be the leader of the group. An actual or perceived leader node failure can therefore give rise to multiple single-node groups performing the same task. Multiple groups assigned to the same task are merged by demoting all of the leader nodes of the constituent groups, except one. Consider, for example, a group comprising three nodes a, b, and c, with node a being the leader node. When a fails, b and c form two single-node groups and continue to perform the sensing task. In due course, nodes b

Assumptions: Nodes n and m are two leader nodes performing Task 1.
Case 1: Node n receives a *queryrelevance* or *status* message from node m.
if Node n is not involved in demotion negotiations with another node **then** send *demote* message to node m after a random interval.
Case 2: Node n receives a *demote* message from node m.
a) **if** Node n has not sent a *demote* message to another node **then** demote node n and send *demoteack* message to node m.
b) **if** Node n has sent a *demote* message to node m **then** send *demoteretry* message to node m and send a *demote* message to node m after a random interval.
c) **if** Node n has sent a *demote* message to another node **then** send a *demotenack* message to node m.
Case 3: Node n receives a *demotenack* message from node m.
Terminate demotion negotiations with node m.
Case 4: Node n receives a *demoteack* message from node m.
Add m to node n's group.
Case 5: Node n receives a *demoteretry* message from node m.
Send a *demote* message to node m after a random interval.

Fig. 7. Group merging via leader demotion

and c discover each other—e.g., when b intercepts a *queryrelevance* or a *status* message from c—and they form a new group comprising b and c, demoting node c in the process. Thus, our proposed communication model is able to handle node failures.

Demotion is either carried out based upon the unique ID assigned to each node—among the conflicting nodes, the one with the highest ID is selected to be the group leader—or, when unique node IDs are not guaranteed, demotion can be carried out via the process shown in Fig. 7. The following observations suggest that our leader demotion strategy is correct; i.e., only a single leader node survives the demotion negotiations and every other leader node is demoted.

- **Observation 1:** The demotion process between two leader nodes either succeeds or fails. It succeeds when one of the two nodes is demoted. Demotion between two nodes is based on the contention management scheme that was first introduced in the ALOHA network protocol [26]. The ALOHA network protocol was developed in the late 60s and it is a precursor to the widely used Ethernet protocol. In its basic version, the ALOHA protocol states
  - if you have data to send, send it.
  - if there is a collision, resend after a random interval.

  We point the interested reader to [27] for the details. What is important here is to note that eventually one of the two leader nodes will be demoted; i.e., the demotion process between two nodes will eventually succeed.
- **Observation 2:** The demotion process between more than two nodes involves repeated (distributed and parallel) application of the demotion process between two nodes.

5 Video Surveillance

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

A detailed world model that includes the location of cameras, their fields of view, pedestrian motion prediction models, occlusion models, and pedestrian movement pathways may allow (in some sense) *optimal* allocation and scheduling of cameras; however, it is cumbersome and in most cases infeasible to acquire such a world model. Our approach does not require such a knowledge base. We assume only that a pedestrian can be identified by different cameras with reasonable accuracy and that the camera network topology is known *a priori*. A direct consequence of this approach is that the network can easily be modified through removal, addition, or replacement of camera nodes.

5.1 Computing Camera Node Relevance

The accuracy with which individual camera nodes are able to compute their relevance to the task at hand determines the overall performance of the network. Our scheme for computing the relevance of a camera to a video surveillance task encodes the intuitive observations that 1) a camera that is currently free should be chosen for the task, 2) a camera with better tracking performance with respect to the task at hand should be chosen, 3) the turn and zoom limits of cameras should be taken into account when assigning a camera to a task; i.e., a camera that has more leeway in terms of turning and zooming might be able to follow a pedestrian for a longer time, and 4) it is better to avoid unnecessary reassignments of cameras to different tasks, as that might degrade the performance of the underlying computer vision routines.

Upon receiving a task request, a camera node returns to the leader node a relevance metric—a list of attribute-value pairs describing its relevance to the current task along multiple dimensions (Fig. 8). The leader node converts this metric into a scalar relevance value r as follows:

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

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

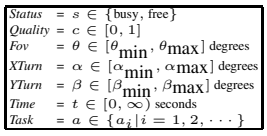


Fig. 8. The relevance metric returned by a camera node relative to a new task request. The leader node converts the metric into a scalar value representing the relevance of the node for the particular surveillance task.

The computed relevance values are used by the node selection scheme described above to assign cameras to various tasks. The leader node gives preference to the nodes that are currently free, so the nodes that are part of another group are selected only when an insufficient number of free nodes are available for the current task.

5.2 Surveillance Tasks

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

6 Results

To date, we have tested our visual sensor network system with up to 16 stationary and pan-tilt-zoom cameras, and we have populated the virtual Penn Station environment with up to 100 pedestrians. The sensor network correctly assigned cameras in most of the 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. 9, we placed 16 active PTZ cameras in the train station, as shown in Fig. 1. The operator selects the pedestrian with the red shirt in Camera 7 (Fig. 9(e)) and initiates the "follow" task. Camera 7 forms the task group and begins tracking the pedestrian. Subsequently, Camera 7 recruits Camera 6, which in turn recruits Cameras 2 and 3 to track the pedestrian. Camera 6 becomes the leader of the group when Camera 7 loses track of the pedestrian and leaves the group. Subsequently, Camera 6 experiences a tracking failure, sets Camera 3 as the group leader, and leaves the group. Cameras 2 and 3 track the pedestrian during her stay in the main waiting room, where she also visits a vending machine. When the pedestrian starts walking towards the concourse, Cameras 10 and 11 take over the group from Cameras 2 and 3.

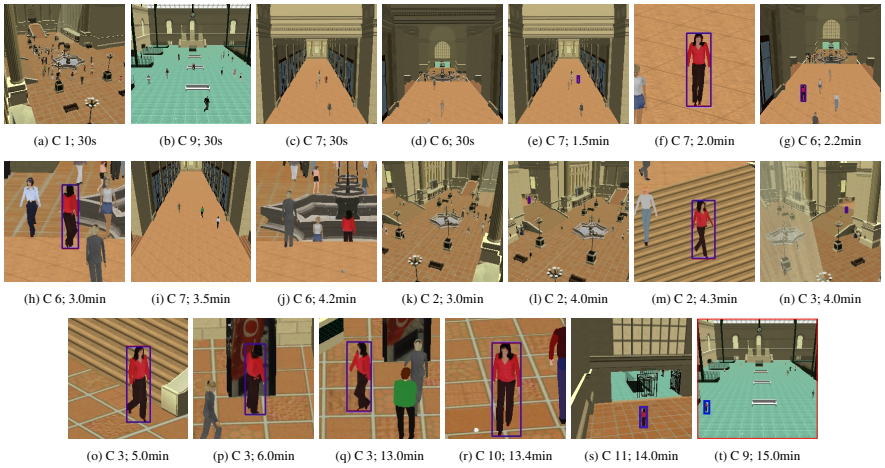


Fig. 9. A pedestrian is successively tracked by Cameras 7, 6, 2, 3, 10, and 9 (see Fig. 1) as she makes her way through the station to the concourse. (a-d) Cameras observing the station. (e) The operator selects a pedestrian in the video feed from Camera 7. (f) Camera 7 has zoomed in on the pedestrian, (g) Camera 6, which is recruited by Camera 7, acquires the pedestrian. (h) Camera 6 zooms in on the pedestrian. (i) Camera 7 reverts to its default mode after losing track of the pedestrian and is now ready for another task (j) Camera 6 has lost track of the pedestrian. (k) Camera 2. (l) Camera 2, which is recruited by Camera 6, acquires the pedestrian. (m) Camera 2 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) 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.

Fig. 10. "Follow" sequence. (a) The operator selects a pedestrian in Camera 1 (upper row). (b) and (c) Camera 1 and Camera 2 (lower row) are tracking the pedestrian. (d) Camera 2 loses track. (e) Camera 1 is still tracking; Camera 2 has returned to its default settings.

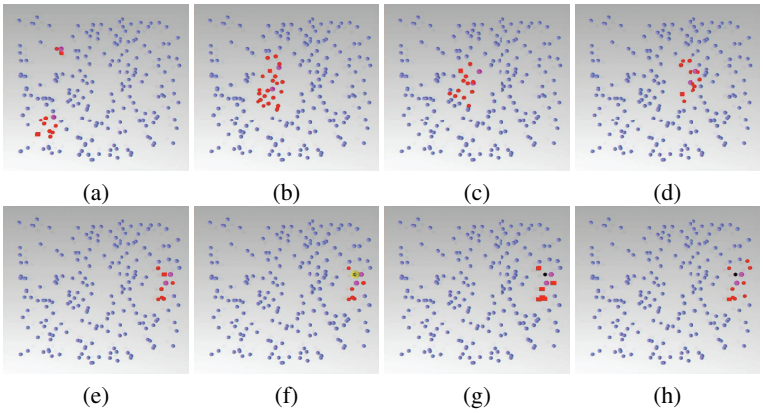


Fig. 11. Group merging and leader failure. Blue nodes are idle. Red nodes are following the targets shown as pink cones. Square nodes represent group leaders and black nodes indicate node failures.

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

Fig. 10 illustrates a "follow" task sequence. An operator selects the pedestrian with the green shirt in Camera 1 (top row). Camera 1 forms a group with Camera 2 (bottom row) to follow and zoom in on the pedestrian. At some point, Camera 2 loses the pedestrian (due to occlusion), and it invokes a search routine, but it fails to reacquire the pedestrian. Camera 1, however, is still tracking the pedestrian. Camera 2 leaves the group and returns to its default settings.

Fig. 11 presents a simulation of larger sensor networks outside our virtual vision simulator. It shows a sensor network of 50 nodes placed randomly in a 25 m^2 area. The nodes that are within 5 m of each other can directly communicate with each other. Each node can communicate with another node in the network through multi-hop routing. Fig. 11(a)–(e) show group merging. When the leader of the group fails (Fig. 11(f)), multiple member nodes assume leadership (Fig. 11(g)). These nodes negotiate each other to select a single leader (Fig. 11(h)).

6.1 Discussion

Given the above results, we make the following observations about the proposed scheme:

– The proposed protocol successfully forms camera groups to carry out various observation tasks. Cameras that belong to a single group collaborate with each other for the purposes of carrying out the observation task. Currently we support a few observation tasks that are of interest to the visual surveillance community. These are 1) taking snapshots of a pedestrian, 2) closely observing a pedestrian during his/her stay in the designated region, and 3) following a pedestrian across multiple cameras.

- Camera grouping does not require camera calibration or camera network topology information, which makes our system suitable for ad hoc deployment. This is not to say that the proposed protocol cannot take advantage of camera calibration and/or camera network topology information if such information were available.
- Camera groups are dynamic and transient arrangements that evolve in order to perform an observation task. Like group formation, group evolution is a negotiation between the relevant nodes.
- The proposed protocol can deal with node and message failures. This suggests that the network protocol can handle addition and removal of camera nodes during the lifetime of an observation task.
- Camera hand off occurs naturally during negotiations.
- Smaller group sizes are preferable. Larger groups have slower responses and higher maintenance costs. The proposed protocol might fail to carry out an observation task even when each (camera) node is assumed to be a perfect sensor if the group evolution cannot keep up with a fast changing observation task.
- Assuming that each (camera) node is a perfect sensor, the proposed protocol still might fail to carry out an observation task if a large fraction of nodes fail or a significant fraction of messages are lost.
- Camera node aggregation is fully distributed and lacks a central controller, so it is scalable. Sensor assignment in the presence of conflicts, however, is centralized over the involved groups. Therefore, our scheme lies somewhere between a fully distributed and a fully centralized system. In the interest of scalability, group sizes should be kept small.

7 Conclusion

We envision future video surveillance systems to be networks of stationary and active cameras capable of providing perceptive coverage of extensive 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 these new sensor networks and our initial results are 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 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 is generally hard to acquire. We expect the proposed approach to be robust and scalable.

We are currently pursuing a *Cognitive Modeling* [28] 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 prolonged stay in the train station.

Acknowledgments

The research reported herein was supported in part by a grant from the Defense Advanced Research Projects Agency (DARPA) of the Department of Defense. We thank Tom Strat, formerly of DARPA, for his generous support and encouragement. We also thank Wei Shao and Mauricio Plaza-Villegas for their invaluable contributions to the implementation of the Penn Station simulator. Deborah Estrin provided helpful advice and pointers into the sensor networks literature.

References

1. Mallett, J.: The Role of Groups in Smart Camera Networks. PhD thesis, Program of Media Arts and Sciences, School of Architecture, Massachusetts Institute of Technology (February 2006)
2. Shao, W., Terzopoulos, D.: Autonomous pedestrians. In: Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation, Los Angeles, CA, pp. 19–28 (July 2005)
3. Zhao, F., Liu, J., Liu, J., Guibas, L., Reich, J.: Collaborative signal and information processing: An information directed approach. Proceedings of the IEEE 91(8), 1199–1209 (2003)
4. Terzopoulos, D., Rabie, T.: Animat vision: Active vision in artificial animals. Videre: Journal of Computer Vision Research 1(1), 2–19 (1997)
5. Terzopoulos, D.: Perceptive agents and systems in virtual reality. In: Proc. 10th ACM Symposium on Virtual Reality Software and Technology, Osaka, Japan, pp. 1–3 (October 2003)
6. Santuari, A., Lanz, O., Brunelli, R.: Synthetic movies for computer vision applications. In: Proc. 3rd IASTED International Conference: Visualization, Imaging, and Image Processing (VIIP 2003) Number 1, Spain, pp. 1–6 (September 2003)
7. Zhao, F., Shin, J., Reich, J.: Information-driven dynamic sensor collaboration for tracking applications. IEEE Signal Processing Magazine 19, 61–72 (2002)
8. Bhardwaj, M., Chandrakasan, A., Garnett, T.: Upper bounds on the lifetime of sensor networks. In: IEEE International Conference on Communications. Number 26, pp. 785–790 (2001)
9. Chang, J.H., Tassiulas, L.: Energy conserving routing in wireless adhoc networks. In: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), Tel Aviv, Israel, pp. 22–31 (March 2000)
10. Collins, R., Lipton, A., Fujiyoshi, H., Kanade, T.: Algorithms for cooperative multisensor surveillance. Proceedings of the IEEE 89(10), 1456–1477 (2001)
11. Costello, C.J., Diehl, C.P., Banerjee, A., Fisher, H.: Scheduling an active camera to observe people. In: Proc. 2nd ACM International Workshop on Video Surveillance and Sensor Networks, pp. 39–45. ACM Press, New York (2004)
12. Collins, R., Amidi, O., Kanade, T.: An active camera system for acquiring multi-view video. In: Proc. International Conference on Image Processing, Rochester, NY, USA, pp. 517–520 (September 2002)

13. Devarajan, D., Radke, R.J., Chung, H.: Distributed metric calibration of ad hoc camera networks. ACM Transactions on Sensor Networks 2(3), 380–403 (2006)
14. Campbell, J., Gibbons, P.B., Nath, S., Pillai, P., Seshan, S., Sukthankar, R.: Irisnet: An internet-scale architecture for multimedia sensors. In: MULTIMEDIA '05. Proc. of the 13th annual ACM international conference on Multimedia, pp. 81–88. ACM Press, New York (2005)
15. Kulkarni, P., Ganesan, D., Shenoy, P., Lu, Q.: Senseye: a multi-tier camera sensor network. In: MULTIMEDIA '05. Proc. of the 13th annual ACM international conference on Multimedia, pp. 229–238. ACM Press, New York (2005)
16. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transctions on Computers C-29(12), 1104–1113 (1980)
17. Gerkey, B., Matari, M.: A formal analysis and taxonomy of task allocation in multi-robot systems. International Journal of Robotics Research 23(9), 939–954 (2004)
18. Garey, M.R., Johnson, D.S.: "strong" npcompleteness results: Motivation, examples, and implications. Journal of the ACM 25(3), 499–508 (1978)
19. Atamturk, A., Nemhauser, G., Savelsbergh, M.: A combined lagrangian, linear programming and implication heuristic for large-scale set partitioning problems. Journal of Heuristics 1, 247–259 (1995)
20. Modi, P.J., Shen, W.S., Tambe, M., Yokoo, M.: Adopt: asynchronous distributed constraint optimization with quality guarantees. In: Artificial Intelligence, vol. 161(1–2), pp. 149–180. Elsevier, North-Holland (2006)
21. Yokoo, M.: Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-agent Systems. Springer, Heidelberg (2001)
22. Comaniciu, D., Ramesh, V., Meer, P.: Real-time tracking of non-rigid objects using mean shift. In: Proc. of the, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 00). vol. 2. Hilton Head Island, South Carolina, USA, pp. 142–151 (2000)
23. Swain, M.J., Ballard, D.H.: Color indexing. International Journal of Computer Vision 7(1), 11–32 (1991)
24. Qureshi, F.Z.: Intelligent Perception in Virtual Camera Networks and Space Robotics. PhD thesis, Department of Computer Science, University of Toronto (January 2007)
25. Pearson, J.K., Jeavons, P.G.: A survey of tractable constraint satisfaction problems. Technical Report CSD-TR-97-15, Royal Holloway, University of London (July 1997)
26. Kuo, F.F.: The aloha system. ACM SIGCOMM Computer Communication Review, vol. 25(1), pp. 41–44 Special twenty-fifth anniversary issue. Highlights from 25 years of the Computer Communication Review (1995)
27. Murthy, C., Manoj, B.: Ad Hoc Wireless Networks Architectures and Protocols. Prentice Hall, Englewood Cliffs (2004)
28. Qureshi, F., Terzopoulos, D., Jaseiobedzki, P.: Cognitive vision for autonomous satellite rendezvous and docking. In: Proc. IAPR Conference on Machine Vision Applications, Tsukuba Science City, Japan, pp. 314–319 (May 2005)