Learning Proactive Control Strategies
for PTZ Cameras

Wiktor Starzyk[†]   Faisal Z. Qureshi[‡]
Faculty of Science
University of Ontario Institute of Technology
Oshawa, ON L1H 7K4 Canada

Abstract—This paper introduces a camera network capable of automatically learning proactive control strategies that enable a set of active pan/tilt/zoom (PTZ) cameras, supported by wide-FOV passive cameras, to provide persistent coverage of the scene. When a situation is encountered for the first time, a reasoning module performs PTZ camera assignments and handoffs. The results of this reasoning exercise are 1) generalized so as to be applicable to many other similar situations and 2) stored in a production system for later use. When a "similar" situation is encountered in the future, the production-system reacts instinctively and performs camera assignments and handoffs, bypassing the reasoning module. Over time the proposed camera network reduces its reliance on the reasoning module to perform camera assignments and handoffs, consequently becoming more responsive and computationally efficient.

I. INTRODUCTION

Automated video surveillance systems are increasingly relying upon passive wide field-of-view (FOV) and active pan/tilt/zoom (PTZ) cameras to provide high-resolution coverage of large areas. The respective strengths of passive and active cameras augment each other: wide FOV passive cameras provide low-resolution, wide-area coverage; whereas, active PTZ cameras can be controlled to record high-resolution video of one sub-area at a time. For example, a PTZ camera may be tasked to look at a particular individual whose position is estimated by passive cameras or a PTZ camera may be focused on a region with high activity as determined by passive wide-FOV cameras.

Manual control of PTZ cameras is infeasible when the number of activities of interest in the scene is more than the number of available PTZ cameras. It is therefore desirable to develop control and coordination strategies that enable PTZ cameras to work together in order to provide the "desired" coverage of a designated region with little or no human assistance.[1] A successful control strategy needs to be aware of both immediate and long-term consequences of its actions [1], while at the same time accounting for observation failures, assignment conflicts, sensing limitations, computation, energy and bandwidth budgets, and the dynamic nature of the tasks.

This paper develops a PTZ camera network that learns proactive control strategies by generalizing and storing the results of a reasoning process. The reasoning process—which considers both immediate and far-reaching consequences of different camera assignments when constructing a "plan" most likely to succeed (in a probabilistic sense) at the current observation task(s)—is capable of performing camera assignments and handoffs in order to provide persistent coverage of a region. We develop such a reasoning module elsewhere [1]. The results of this reasoning activity are then stored as rules in a *production system* [2]. Later when a similar situation is encountered, the production system bypasses the reasoning process and performs camera assignments. Initially, the camera network relies mostly upon the reasoning process to perform camera assignments; however, as the time goes by the camera network reduces its dependence on the reasoning process and camera assignments become instinctive.

To the best of our knowledge this is the first attempt at learning a proactive control strategy in PTZ camera networks. Our approach has a distinct advantage over existing schemes for PTZ camera control: it enables a camera network to adapt to its sensing environment without any human intervention. Over time this leads to a more capable, responsive and computationally efficient camera network.

The rest of the paper is organized as follows. Sec. II discusses related work. Next, we give a brief overview of production systems. The reasoning module is described in Sec. IV. Learning methodology is introduced in Sec. V. Sec. VI provides system overview and results are presented in Sec. VII. We conclude the paper with a summary and a brief discussion in Sec. VIII.

II. RELATED WORK

Research community is paying increasing attention to the problem of controlling or scheduling active cameras in order to capture high-resolution imagery of interesting events—high-resolution imagery is needed for biometric analysis that enhances the situational awareness of surveillance operators. In a typical setup, information gathered by stationary wide-FOV cameras is used to control one or more active cameras [3], [4], [5], [6], [7], [8]. Generally speaking, the cameras are assumed to be calibrated and the total coverage of the cameras is restricted to the FOV of the stationary camera. Nearly all PTZ scheduling schemes rely on site-wide multitarget, multicamera tracking.

The critical problems of camera assignment and handoff have been studied in the context of smart camera networks. Park *et al.* construct a distributed look-up table to perform camera handoffs [9]. The look-up table encodes the suitability

---

† wiktorstarzyk@gmail.com
‡ faisal.qureshi@uoit.ca

[1]It is worth remembering that similar control and coordination problems arise and need to be addressed in any camera network comprising nodes with tunable sensory and processing parameters.
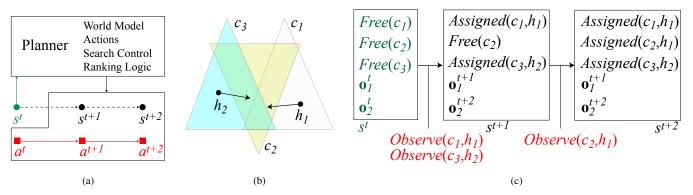
Fig. 1: (a) Anatomy of a planner, along with its inputs and outputs. State s^t shown in green is the current state of the world. The planner generates the action sequence (shown as red squares) that is most likely to achieve the goal. States s^{t+1} and s^{t+2} are the hypothesized states after executing actions a^t and a^{t+1}, respectively. The state sequence meets the goal requirements. (b) An example scenario comprising 3 PTZ cameras and 2 pedestrians. (c) A possible start state s^t for the scenario shown in (b) and the 2-step plan generated by the planner. The generated plan consists of an action sequence shown in red. We annotate the plan with the hypothesized states after each step of the plan.

of a camera to observe a specific location. [10] proposes the use of a handoff function for continuous tracking across multiple cameras. Li and Bhanu developed a game theoretic approach to achieve camera handoffs [11]. When a target is visible in multiple cameras, the best camera is selected based upon its expected utility. They also propose a number of criteria to construct the utility function, such as the number of pixels occupied by the selected target in an image. Their approach eschews spatial and geometric information. Kim and Kim develop a probabilistic framework for selecting the "dominant" camera for observing a pedestrian [12]. Song *et al.* present a game-theoretic strategy for cooperative control of a set of decentralized cameras [13]. The cameras work together to track every target in the area at acceptable image resolutions. The camera network can also be tasked to record higher resolution imagery of a selected target. [7] develops a PTZ camera scheduling strategy. [8] also tackles the problem of PTZ camera scheduling. It develops a scheme for generating camera schedules with the highest probability of success at the task of capturing close-up, facial imagery of individuals present in the scene. Previously, in [1], we develop a proactive PTZ camera control strategy that enables a suite of PTZ cameras to capture seamless video of pedestrians as they make their way through the designated region. [14] presents an occlusion-aware PTZ camera configuration strategy to observe areas of high activity.

Production systems have been studied extensively in cognitive sciences and artificial intelligence. These have been proposed as a viable model for mechanisms responsible for intelligent behavior: learning, development, and problem solving [15]. For an overview of production systems, we refer the reader to [2], [15], [16].

III. PRODUCTION SYSTEMS

A production system consists of three components described below.

Working memory: a database that stores knowledge representing the current state of the world. The production system can consult the working memory, add information to it, delete information from it, and modify the information currently present in it.

Production rules: condition-action behavior rules of the form,
if [*condition*] **then** [*action*].
The *condition* is expressed in terms of working memory elements and the corresponding *action* specify the changes that will be made to the working memory if this production rule is fired.

Interpreter: a control mechanism responsible for *triggering* relevant productions and selecting one of them to *fire*. A production rule is said to be triggered if its condition is supported by the current state of the memory. During each cycle many productions may be triggered, however, only one of these productions is allowed to change the contents of the working memory (fired). The process continues till no more productions are triggered.

The number of productions can quickly grow in large and complex systems and performance becomes an issue. On the bright side identifying production rules to trigger can occur in parallel. We conveniently avoid the performance issues associated with production systems due to the relatively small number of production rules (< 1000) that we encounter in our problem domain. This is in part due to the fact that our production rules are learnt through a deliberative process (planning), which must occur within higher levels of abstraction to remain tractable.

IV. REASONING ABOUT CAMERA ASSIGNMENTS

Consider a camera network comprising N_p calibrated wide FOV passive cameras and N_a PTZ active cameras. Let $C = \{c_i | i \in [1, N_a]\}$ denote the set of active PTZ cameras. Each PTZ camera is described by a tuple $\langle \mathbf{p}, \alpha_{\min}, \alpha_{\max}, \beta_{\min}, \beta_{\max} \rangle$,
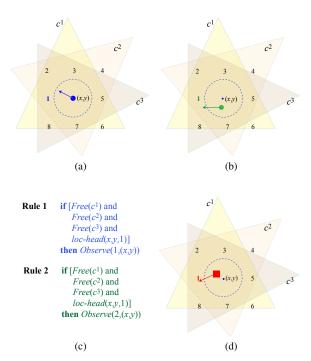
Fig. 2: State abstraction can generate multiple production rules with same conditional and different actions. (a)-(b) depict scenarios that lead the generation of (c) rules 1 and 2, respectively, through reasoning. (d) A new situation that triggers both rules.

where we assume that the 3D position $\mathbf{p}$ of each PTZ camera is known *a priori*, and where $[\alpha_{\min}, \alpha_{\max}]$ and $[\beta_{\min}, \beta_{\max}]$ represent pan and tilt limits, respectively, for each PTZ camera. Each PTZ camera is able to direct itself to look in the direction of a 3D location within its observational range.[2] The static cameras track and estimate the 3D positions and velocities of the observed pedestrians. Let $\mathcal{H} = \{h_j | j = 1, 2, \cdots\}$ denote the set of pedestrians observed during the operation of the camera network. At time instant t, the state of the pedestrians observed by the camera network is given by $\mathbf{o}_j^t = (\mathbf{x}_j^t, \mathbf{v}_j^t)$, where $\mathbf{x}_j$ and $\mathbf{v}_j$ represent the ground plane position and velocity, respectively, of observed pedestrian j.

We define the state of the system to include the status of all PTZ cameras and the locations and velocities of every observed pedestrian. Let tuple s^t represent the state of the system during time interval $[t, t+1)$ then $s^t = \langle s_i^t | i = 1, \cdots, N_a, \mathbf{o}_j^t | j, h_j \in \mathcal{H} \rangle$, where s_i^t denotes the status of the PTZ camera i at time t. Possible values for s_i^t are $Free(c_i)$ and $Assigned(c_i, h_j)$, for $h_j \in \mathcal{H}$, and for $c_i \in C$.

We assume that we have access to a planner that is able to compute camera assignments given the current state of the system [1]. The result of the planning activity is an action sequence $\mathcal{A}$ that encodes "optimal" camera assignments at time $t, t+1, t+2, \cdots, t+P_l - 1$, where P_l is the length of the plan. We use the term "optimal" cautiously and mean to

---

[2][7] describes a scheme for automatically learning a map between 3D world locations and the gaze direction parameters (α, β) of a PTZ camera.

say that our planner returns the actions sequence with highest probability of success among all the action sequences that the planar explored. We denote the actions available to our planar by the tuple $a = \langle a_i | i = 1, \cdots, N_a \rangle$, where a_i represents the action for PTZ camera i. The possible values for a_i are $Idle(c_i)$ and $Observe(c_i, h_j)$, where $h_j \in \mathcal{H}$. Neither actions have any preconditions and their effects are obvious from their names.

V. LEARNING PRODUCTION RULES

Production rules are represented as: **if** [*condition*] **then** [*action*], where *condition* is matched against the working memory, triggering the rule, and the *action* is executed if the rule is fired. We distinguish between rule trigger and firing: multiple rules can trigger simultaneously, but only one rule is allowed to fire. It is straightforward to create a production rule given an annotated plan. The annotated action sequence shown in Fig 1(c), for example, suggests the following two production rules.

Rule 1:
if [$Free(c_1) \wedge Free(c_2) \wedge Free(c_3) \wedge \mathbf{o}_1^t \wedge \mathbf{o}_2^t$]
then [$Observe(c_1, h_2), Observe(c_3, h_2), \mathbf{o}_j^t \to \mathbf{o}_j^{t+1}$]
Rule 2:
if [$Assigned(c_1, h_1) \wedge Free(c_2) \wedge Assigned(c_3, h_2) \wedge \mathbf{o}_1^{t+1} \wedge \mathbf{o}_2^{t+2}$]
then [$Observe(c_2, h_1), \mathbf{o}_j^t \to \mathbf{o}_j^{t+1}$]

Here j indices over pedestrians present in the scene. Henceforth we will drop $\mathbf{o}_j^t \to \mathbf{o}_j^{t+1}$, which simply refers to scene dynamics, in the *action* part of production rules.

The above rules, however, are too specialized to be of any real use. These refer to pedestrian identities and exact locations, headings, and speeds, so these rules will only be triggered during a re-enactment of the scene. We desire production rules to be able generalize to similar situations. We are able achieve this via 1) state abstraction and 2) partial matching.

A. State abstraction

We replace pedestrian identities with Universally Unique Identifiers (UUIDs). UUIDs then refer to an activity with a persistent appearance (or behavior) signature. We also discretize locations, headings, and speed as follows:

Location: We divide the scene into regions defined around observed pedestrian locations; *e.g.*, in Fig. 2 the locations of pedestrians depicted as green hexagon (Fig. 2(b)) and red square (Fig. 2(c)) are described in terms of the location of the pedestrian (shown as a blue circle in Fig. 2(a)) who was observed earlier. This is because the actual locations of the green and red pedestrian are close (in the Euclidean sense) to the location of the blue pedestrian.

Heading: Headings are quantized into 8 bins; *e.g.*, the heading of the individual (blue circle) in Fig. 2(a) is set to 1.

Speed: We assume that an individual is either standing or walking. All individuals assume to have the same walking speed. Heading value for a stationary pedestrian is 0.

Abstracted location, heading, and speed of an activity (UUID) is represented using $loc\text{-}head/3$ predicate. State abstraction often gives rise to multiple productions with the same *condition* but different *actions*. Fig. 2 illustrates this phenomenon: (a) the planner assigns camera 1 to observe the individual shown as a blue circle, which gives rise to the first rule (shown in blue); (b) next, the planner assigns camera 2 to observe the individual shown as a green hexagon, resulting in rule number 2 (shown in green). (d) The production system interpreter triggers both rules when it encounters the situation where the individual shown as a red square is in the same region and has the same heading as the two individuals (blue circle and green hexagon) observed earlier. Clearly, the interpreter needs to select one of these two rules to fire. We discuss rule selection in Sec. V-C.

B. Partial Matching

Partial matching rules only apply to the $loc\text{-}head$ portion of a production rule *condition*. Currently, the camera state portion is always matched exactly. Fig. 3 illustrates partial matching.

Let $\mathcal{L} = \{loc\text{-}head(x,y,h)|x \in [1,n_x^c], y \in [1,n_y^c], h \in [0,8]\}$ represents the set of all possible values for $loc\text{-}head$ predicate. Here, n_x^c and n_y^c represents the number of regions (defined by first observations) along x and y dimensions, respectively. Define $\mathcal{L}^i \subset \mathcal{L}$ to be the location/heading part a production rule i (*condition*). We then define the following three partial matching rules:

Heading rule: Given $l_1 \in \mathcal{L}^i$ and $l_2 \in \mathcal{L}^j$, $l_1 = l_2$ if $heading(l_1) = heading(l_2)$ or $heading(l_k) = 0$ where $k \in [1,2]$. Operator $heading(.)$ returns the heading part of the $loc\text{-}head$ predicate.

Subset rule: $\mathcal{L}^i = \mathcal{L}^j$ if $\mathcal{L}^i \in \mathcal{L}^j$. In this case only those actions items of production rule i are used that refer to $location(l_k)$ where $l_k \in \mathcal{L}^j$. Operator $location(.)$ returns the location part of $loc\text{-}head$ predicate.

Intersection rule: $\mathcal{L}^i = \mathcal{L}^j$ if $\mathcal{L}^i \cup \mathcal{L}^j \neq \Phi$. In this case too only those actions items of production rule i are used that refer to $location(l_k)$ where $l_k \in \mathcal{L}^j$. Operator $location(.)$ returns the location part of $loc\text{-}head$ predicate.

In our current implementation, we only employ the heading rule during partial matching. Subset and intersection rules should only be considered if the reasoning module is not available. Since in situations where the number of individuals present in the scene are not the same as the number specified in a rule *condition* are best handled with planning.

C. Rule Selection

If no productions are triggered, it indicates a never before seen situation and the system resorts to planning to perform camera assignments and handoffs. On the other hand when one or more production rules are triggered, the interpreter needs to select one production rule to fire from the set of triggered rules (in Fig. 3(d), for example, two rules have been triggered). The rule selection behavior is modeled as a three stage filter, each stage rejects productions that do not meet the criteria needed to get to the next stage. The first stage only

(a) Production Rules

(b) Subset rule
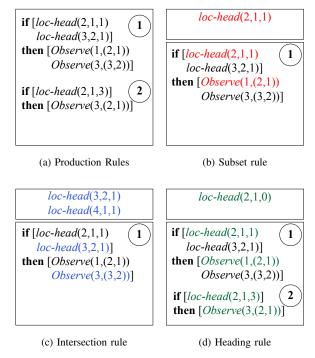
(c) Intersection rule

(d) Heading rule

Fig. 3: Partial matching rules to trigger relevant productions (a) given the current state of the working memory shown in (b)-(d). In the interest of clarity we have dropped the camera state part of the *conditions* and the working memory, assuming that camera state is always matched exactly.

accepts productions whose action results in minimal number of camera reassignments; thereby, minimizing camera handoffs and unnecessary task switches. The second stage accepts maximal rank productions. Rank encodes how successful the production rule has been in the past at carrying out the observation tasks. Rank value for a production is defined as the fraction of successful video frames (w.r.t. to an observation task) to the total number of video frames collected when the production rule is active. The last stage makes a random choice and selects one production to fire. Each of the previous two stages always let at least one production through.

VI. SYSTEM OVERVIEW

Whenever a new situation is encountered, the reasoning module initiates the planning activity. Upon completion the planning activity returns a plan containing a sequence of task assignments for every PTZ camera. These assignments are sent to the individual PTZ cameras. We model PTZ cameras as behavior-based agents, capable of carrying out the assigned tasks autonomously. The result of the planning activity is also intercepted by the production system, which stores this information for later use (Fig. 4).

Reasoning module is bypassed when a familiar situation is encountered. In this case, the production system is responsible for camera assignments. The working memory is updated to reflect the world state at each time step.
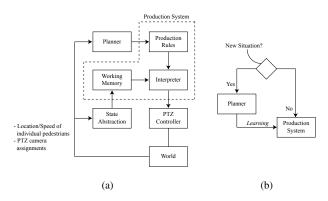
(a) (b)

Fig. 4: System overview (a) and the situation dependent interaction between the planner (reasoning module) and the production system.

(a) $s_e = 140, n_r = 16$ (b) $s_e = 0, n_r = 0$ (c) $s_e = 70, n_r = 9$
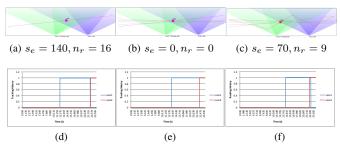

(d) (e) (f)

Fig. 5: (a) This is the first time the camera network has encountered the situation involving a single pedestrian walking across the room, so it has to plan to perform a successful handoff (d). (b) Another individual enters the scene from the left and walks across the room. The system did not need to plan in this case and the production system was able to perform a successful handoff (e). (c) The system only had to plan when the individual deviated from the paths (shown in gray) used to learn the rules. Here s_e refers to the number of states explored by the planner and n_r denotes the number of rules generated in the production system.

VII. RESULTS

Fig. 5 shows 2 PTZ cameras tasked to perform camera handoff in order to observe individuals as they walk through the area. Fig. 5(a) shows the situation when an individual (purple square) enters the scene from the right and walks across the room. This is the first time the camera network has encountered such a situation, so it plans to perform a successful handoff (Fig. 5(d)). The planner explored 140 states and 16 PTZ control rules were learnt. Sometime later another individual enters the scene from the right and walks across the room (Fig. 5(b)). The cameras are able perform a successful handoff (Fig. 5(e)) without planning by using the control rules learnt earlier. Afterwards another individual enters the scene from the right and moves across the room in a zig-zag fashion (Fig. 5(c)). In this case the cameras are able to perform successful handoff (Fig. 5(f)) by combining planning and the stored PTZ control rules. The planner explored 70 states and

(a) $s_e = 11799, n_r = 29$ (b) $s_e = 0, n_r = 0$
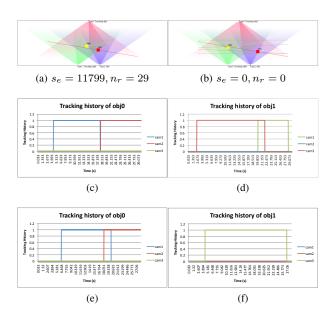

(c) (d)

(e) (f)

Fig. 6: 3 PTZ camera network is tasked with observing two pedestrians as the move across the area under surveillance. The cameras learn a swap strategy (a) and use it to perform swaps without planning in (b). Here s_e refers to the number of states explored by the planner and n_r denotes the number of rules generated in the production system. Gray trajectories in (b) indicate paths followed by pedestrians in previous situations.

9 rules were added to the production system.

Fig. 6 represents a scenario involving three cameras that need to keep two pedestrians in view at all times. By necessity the cameras need to perform a swap when these two pedestrians cross each other. The two cameras can use the third camera temporarily during the swap to ensure that both pedestrians are viewed by at least one of the cameras. When two individuals walk across the room for the first time the camera network relies upon the planner to construct a swap strategy (Fig. 6(a)). The planner explored 11799 nodes and 29 control rules were learnt. Afterwards when two other individuals enter the scene and walk across the room the camera network successfully performed the swap by relying upon the rules learnt previously (Fig. 6(b)). No planning activity took place.

Next, we set up a network comprising 4 PTZ cameras that is tasked to observe all 4 individuals present in the scene (Fig. 7). In the beginning the network had to rely on reasoning to come up with an appropriate control strategy. During the first run the planner explored 127500 states, generating 10 rules. In the next run the planner explored 63125 states and generated 12 more PTZ control rules. On the third run the system had to perform some planning. This time the planner explored 33750 states, and only 2 new rules were generated bringing the total number of learnt rules to 24. The camera network successfully completed the observation tasks for all three runs.

All three scenarios presented above suggest that the camera network relies less on the planner to successfully carry out the observation task as the time goes by.

(a) $s_e = 127500, n_r = 10$ (b) $s_e = 63125, n_r = 12$ (c) $s_e = 33750, n_r = 2$
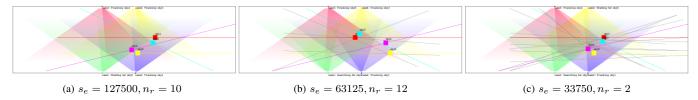
Fig. 7: 4 camera PTZ network tasked to observe all 4 individuals present in the scene. (a), (b), and (c) represent the first, second and the third run of the system, respectively. Gray trajectories indicate paths followed by pedestrians in previous runs. Here s_e refers to the number of states explored by the planner and n_r denotes the number of rules generated in the production system.

VIII. SUMMARY AND DISCUSSION

We propose a camera network—comprising active and passive sensors—that learns control strategies to carry out persistent coverage of a designated region. When a new situation is encountered, a centralized *reasoning* module constructs a plan containing camera assignments most likely to succeed (in a probabilistic sense) at the current observation task(s). The reasoning engine relies upon sensory inputs captured by the wide-FOV passive cameras, current PTZ camera assignments, and a dynamic model of the world to compute the "optimal" camera assignments. Since the reasoning module considers both short-term and long-term consequences of different assignments, it is able to avoid assignments that might appear optimal now but will eventually lead to observation failures. The results from this reasoning activity are 1) generalized and 2) stored as rules in a *production system*. When a similar situation arises in the future, the production system bypasses the reasoning module and performs camera assignments. Over time the proposed camera network reduces its reliance on the reasoning module to perform camera assignments and handoffs, consequently becoming more responsive and computationally efficient.

Planning ahead to perform PTZ control is advantageous in many settings, but also computationally expensive. On the other end of the spectrum are purely reactive schemes that are computationally less demanding, but prone to failures. The work presented here is an attempt to combine the strengths of both these approaches. Reactive rules that represent network-specific control strategy are automatically learnt via planning. These rules are learnt by considering both long-term and short-term consequences of various control decisions. The ability to plan enable the proposed system to adapt to new situations (deployments) and the ability to learn makes the system more responsive as the time goes on.

Currently we assume a centralized production system that controls every PTZ camera. This design choice clearly has scalability ramifications, which we intend to address by distributing production rule storage, matching, and selection across the network. We have prototyped our surveillance system in a virtual environment and evaluated it on simple configurations. In the future, we intend to evaluate it on more complex scenarios. We also aim to evaluate the proposed system using a physical camera network.

REFERENCES

[1] F. Z. Qureshi and D. Terzopoulos, "Planning Ahead for PTZ Camera Assignment and Control," in *Proc. Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 09)*, Como, Italy, Aug. 2009, pp. 1–8.

[2] N. J. Nilsson, *Principles of Artificial Intelligence*. Springer-Verlag, 1980.

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

[4] X. Zhou, R. T. Collins, T. Kanade, and P. Metes, "A master-slave system to acquire biometric imagery of humans at distance," in *Proc. ACM SIGMM International Workshop on Video Surveillance*. 2003, pp. 113–120.

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

[6] A. Hampapur, S. Pankanti, A. Senior, Y.-L. Tian, L. Brown, and R. Bolle, "Face cataloger: Multi-scale imaging for relating identity to location," in *Proc. IEEE Conference on Advanced Video and Signal Based Surveillance*, Washington, DC, 2003, pp. 13–21.

[7] F. Z. Qureshi and D. Terzopoulos, "Surveillance Camera Scheduling: A Virtual Vision Approach," *ACM Multimedia Systems Journal*, vol. 12, no. 3, pp. 269–283, Dec. 2006.

[8] N. O. Krahnstoever, T. Yu, S. N. Lim, K. Patwardhan, and P. H. Tu, "Collaborative Real-Time Control of Active Cameras in Large-Scale Surveillance Systems." in *Proc. ECCV Workshop on Multi-camera and Multi-modal Sensor Fusion*, Marseille, France, Oct. 2008, pp. 1–12.

[9] J. Park, P. C. Bhat, and A. C. Kak, "A Look-up Table Based Approach for Solving the Camera Selection Problem in Large Camera Networks," in *Working Notes of the International Workshop on Distributed Smart Cameras (DSC 2006)*, B. Rinner and W. Wolf, Eds., Boulder, CO, Oct. 2006, pp. 72–76.

[10] Y. Jo and J. Han, "A new approach to camera hand-off without camera calibration for the general scene with non-planar ground," in *Proc. of the 4th ACM international workshop on Video surveillance and sensor networks (VSSN06)*. Santa Barbara, CA: ACM, Oct. 2006, pp. 195–202.

[11] Y. Li and B. Bhanu, "Utility-based Dynamic Camera Assignment and Hand-off in a Video Network," in *Proc. of the Second International Conference on Distributed Smart Cameras (ICDSC08)*, Menlo Park, CA, Sep. 2008, pp. 1–9.

[12] J. Kim and D. Kim, "Probabilistic camera hand-off for visual surveillance," in *Proc. Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC08)*, Stanford, CA, Sep. 2008, pp. 1–8.

[13] B. Song, C. Soto, A. K. Roy-Chowdhury, and J. A. Farrell, "Decentralized camera network control using game theory," in *Proc. of the Second IEEE/ACM International Conference on Distributed Smart Camers (ICDSC08)*, Menlo Park, CA, Sep. 2008, pp 1–8.

[14] C. Piciarelli, C. Micheloni, and G. L. Foresti, "Occlusion-aware multiple camera reconfiguration," in *Proc. of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 10)*. Atlanta, GA: ACM Press, Aug. 2010, pp. 1–8.

[15] D. Klahr, P. Langley, and R. Neches, *Production System Models of Learning and Development*. MIT Press, Cambridge MA, 1987.

[16] J. Laird, "SOAR: an architecture for general intelligence," *Artificial Intelligence*, vol. 33, no. 1, pp. 1–64, Sep. 1987.