

DISTRIBUTED ATTENTION IN LARGE SCALE VIDEO SENSOR NETWORKS

Maurice Chu, James Reich, Feng Zhao

mchu@parc.com, jreich@parc.com, zhao@parc.com

Palo Alto Research Center (PARC), 3333 Coyote Hill Road, Palo Alto, CA, U.S.A.

Abstract

As distributed surveillance networks are deployed over larger areas and in increasingly busy environments, limiting the computation, bandwidth, and human attention burdens imposed is becoming critical. This paper describes a system addressing this problem which uses layered, in-network processing on each camera to filter out uninteresting events locally, avoiding disambiguation and tracking of irrelevant environmental distractors. Coupled with this is a factor-graph-based resource allocation algorithm which steers pan-tilt cameras to follow interesting targets while maintaining a “peripheral awareness” of emerging new targets. We describe this *distributed attention mechanism* and our implementation of this high-level architecture in a video sensor network.

1 Introduction

The increasing scale of deployment of surveillance systems in public and private areas makes it all too easy to overwhelm human operators. Automated systems [4,9] have been proposed to mitigate this, only reporting “interesting” events to operators. However, in complex, busy environments, with limited resources for sensing, processing, and communication, these algorithms are faced with a difficult problem: the computational complexity and communication bandwidth scale poorly in environments with large numbers of moving objects and with very large numbers of cameras.

In this paper, we present a *distributed attention mechanism*, inspired by the attention mechanism of biological visual systems, as investigated in [10], analogous to both the moving eyeball and the attentional focus of higher-level faculties of the brain. We propose an architecture and a demonstration scenario based on a large network of video sensor nodes of limited computational capability, each equipped with a pan-tilt camera (PTC) and linked together in a mesh network in which each link is of limited bandwidth.

The purpose of the network is to efficiently detect abnormal events and behaviors of a set of moving targets (which may be people, vehicles, or other phenomena). Targets which only exhibit “normal” behaviors, i.e., those which follow a learned or programmed model, are referred to hereafter as *distractors*. This model may be defined by human users, or relative to other targets, or based on historical trends. Other *targets of interest* are defined as those which exhibit abnormal behaviors. Targets of interest may not behave abnormally at all times, but the underlying state of being “interesting” is not

transient, although it may require several stages of inference to determine.

The challenges faced to implement a distributed attention mechanism fall into several categories, including:

1. (*Information Processing*) How do we efficiently represent and monitor the state of dynamically evolving phenomena while maintaining peripheral awareness of new emerging events?
2. (*Resource Allocation*) What is the appropriate evaluation of “interest” of a task, and how do nodes negotiate with neighbors to allocate resources in some near globally optimal fashion in a distributed network.
3. (*Distributed Implementation*) How do we implement algorithms in a distributed fashion taking into account bandwidth constraints, energy considerations, processing time, latency constraints, etc.? What is the nature of the exchange of information through the network?
4. (*Peripheral Awareness*) While we are monitoring one target of interest, how can we continue to watch for newly emerging targets of interest? How can one model the likelihood of a new target of interest emerging from a particular area, and can sensed data help determine this?
5. (*Adaptation*) What is considered abnormal behavior? How is this knowledge stored and used by the system to detect surprises and react appropriately?

2 High-level Distributed Attention Architecture

Our architecture, as pictured in Figure 1, consists of four major modules: (1) a layered information processing architecture, (2) a peripheral awareness module, (3) a resource allocation module, and (4) an adaptation module.

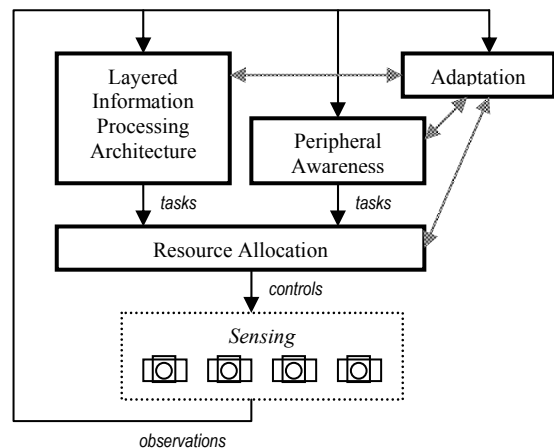


Figure 1. Distributed Attention Architecture.

2.1 Layered Information Processing Architecture

This module is responsible for the processing of data into information about the environment, including tracking and estimation of phenomena states and detection of emerging abnormal behavior. We advocate a layered architecture as in [3] where different types of information are inferred from the data collected by the sensors. Our model consists of three layers: a pre-attentive layer, an attentive layer, and a cognitive layer. Each layer has a different set of information processing algorithms and a different pattern of information flow. For example, the pre-attentive layer might run low-level image processing routines to detect anomalies, e.g. computing optical flow of the sequence of images taken by a camera, and requires no cross-node communication. The attentive layer could be a tracking layer which segments out objects and combines observations over time to compute long-term tracks, requiring data handoff between nodes. The cognitive layer might use some rule-based system across widely separated network nodes to look for relations between tracks and infer group behavior and hypothesize intentions as in [5].

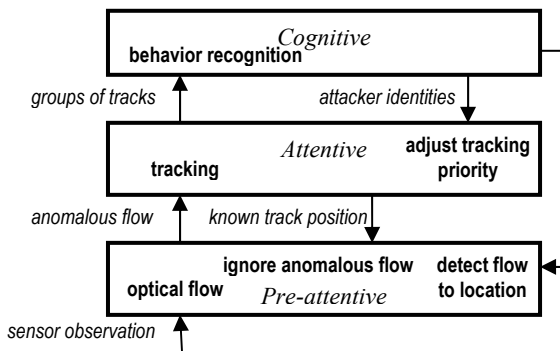


Figure 2. Example interactions in layered architecture.

But how do the different layers interact? Figure 2 shows an example of the interactions and computations for a three-layer information architecture. Lower layers act as *triggering mechanisms* for higher layers, spawning hypotheses in the layer above. For example, abnormal configurations of optical flow in the pre-attentive layer might trigger the instantiation of a new track in the attentive layer. A set of tracks in the attentive layer which are moving towards the same location could then trigger in the cognitive layer a hypothesis of a coordinated attack on that location (e.g., in a military application).

Higher layers interact with lower layers by priming the processing of the lower layer. A cognitive layer hypothesis that a location is under attack might prime the tracking layer to track targets near that region with higher frequency or estimation accuracy, at the cost of ignoring other tracks. The cognitive layer might also prime the pre-attentive layer to increase their sensitivity to abnormal flow in the direction of the landmark. This interaction of bottom-up triggering and top-down feedback allows low-level signal processing to be steered by the symbolic processing of the cognitive layer, while shielding the higher layers, which often scale badly

with high target count, from computing on many irrelevant stimuli.

These layers of processing must also ultimately be distributed in the network, and appropriate selection of information flow patterns will radically affect the system scalability. Basically, the lower layers of processing should require as little cross node information as possible, and computation should be relatively insensitive to target count. Thus, lower layers can process high bit-rate local data pertaining to the largest number of targets. As we proceed to higher layers, this may be reversed; the lower layers have decreased the number of interesting targets sufficiently that more scale-sensitive algorithms may be used, while reducing high bit-rate sensor data to more discrete forms which require less bandwidth. These reduced representations then be sent over more network hops to measure more global multitarget phenomena.

2.2 Peripheral Awareness Module

The peripheral awareness module models where abnormal behavior is most likely to emerge in the environment. The actual detection of suspicious behavior is not performed in this module, but rather it provides the resource allocator with a sense of which areas may be hot spots of suspicious activity, given a history of which regions have been observed. For example, consider a road section with a known direction of traffic flow as shown in Figure 3.

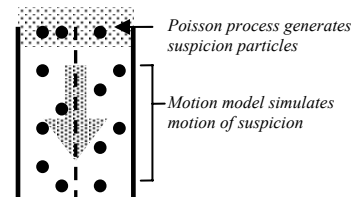


Figure 3. Example of the flow of suspicion particles.

We can simulate a probabilistic model of the flow of suspicious vehicles in an unobserved region by a Monte Carlo simulation. We can instantiate independent samples (particles) from the model at the entrance of the road section according to a Poisson arrival process. These are not real vehicles, but are simply representing the probability a suspicious vehicle appears at that location. We model the motion of each “particle of suspicion” according to an appropriate motion model. However, when a camera observes part of this road, the suspicion particles within its field of view can be deleted since any suspicious behavior in that region will have been detected and handled by detection mechanisms in the processing architecture. These suspicion particles compete with targets of interest to assure that the system attempts to detect emerging targets of interest instead of focusing entirely on those it has already detected.

2.3 Resource Allocation

The resource allocation module allocates resources needed to sense known phenomena and detect emerging ones. This module computes a functional representing *interest level* based on the predicted state estimates in the information

processing architecture and the predicted suspicion particles. Depending on how the two are weighted, we can elicit a range of resource allocation behavior, from focusing completely on currently known phenomena and ignoring detection of new emergent events to ignoring currently known phenomena and focusing completely on detecting newly emerging events. To give an anthropomorphic interpretation, the system exhibits behavior from focused concentration on known phenomena to a paranoid hunt for suspicious events.

In our testbed, we consider cameras as sensing resources, but one might also allocate CPU cycles or communication bandwidth in the same fashion. A distributed algorithm implementing resource allocation requires negotiations between a subset of cameras to determine the optimal joint set of camera pan-tilt parameters that maximizes the interest level. This will be discussed in more detail in Section 3.1. Other algorithms include the market inspired auction algorithm [1].

2.4 Adaptation

The adaptation module allows system behavior to evolve as the environment changes and more knowledge about the world is accumulated. The result is a system that adapts to the environment to operate more efficiently and robustly, refine the model of normalcy and learn which events matter to human operators. This affects several modules.

From learning what to ignore and what to pay attention to, the triggering mechanism in the layered information architecture at various layers can be adapted appropriately. This affects the kinds of processing the system will ultimately perform and the information that will be extracted. Adaptation of the triggering and feedback mechanism has the two-fold effect of guaranteeing good detection and monitoring of suspicious phenomena as well as improving overall system efficiency by not wasting computational resources on uninteresting events.

The models of suspicion in the peripheral awareness module can also be learned based on observations of the environment and feedback of whether suspicious events have occurred.

The resource allocation module allocates resources to maximize an interest functional derived from the state estimates of the various layers of the information processing architecture and the suspicion particles simulated by the peripheral awareness module. The interest functional can be learned based on feedback of system performance, connectivity and availability of particular nodes.

3 Distributing the High-Level Architecture

3.1 Distributed Tracking

A distributed algorithm for tracking in a sensor network has been presented in previous work [7]. The idea is that the set of tracks and their state distribution can be distributed in the network by allowing the tracks to hop around the network according to estimate of its position. A further problem when

there are multiple targets is the ambiguity that arises when targets cross. If there exists no discriminant to tell the two targets apart, then the system must maintain that the identity of the targets has been mixed when they diverge after crossing. This problem of identity management has also been dealt with in previous work [8].

3.2 Distributed resource allocation via factor graphs

We derive the distributed algorithm for near optimally allocating resources by representing the interest level functional in terms of a factor graph [6]. The max-sum algorithm, which is a variant of the sum-product algorithm, on this factor graph decomposes the global optimization of the interest level functional into local computations of summaries and message passing which can be easily mapped to computations of summaries within nodes of the network with communications between them.

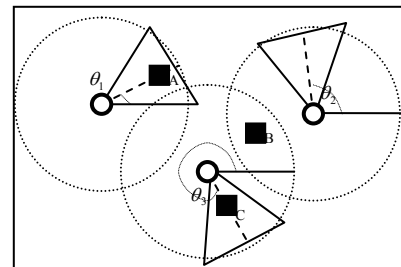


Figure 4. Three camera network with three targets.

Let us explain how the interest level functional is represented as a factor graph. Consider the three camera network shown in Figure 4. The small circles represent cameras, the wedge with a dotted line represents the field of view of the camera for given pan-tilt parameter θ_i , and the dotted circle around each camera represents the union of all field of views that the camera can view. There are three targets depicted in the figure by a black square and labelled by a letter.

The interest functional, $g(\theta_1, \theta_2, \theta_3)$, is a function of the three camera pan-tilt parameters θ_1 , θ_2 , and θ_3 . Assuming the interest functional is additive in the interest contributions from each target, we can express the interest functional, g , as the sum of the interest returned by each target A , B , and C

$$g(\theta_1, \theta_2, \theta_3) = f_A(\theta_1) + f_B(\theta_2, \theta_3) + f_C(\theta_3) .$$

The interest returned by target A and target C are a function of only one camera because only cameras 1 and 3 respectively can see these targets. The interest returned by target B is a function of the pan-tilt parameters of both camera 2 and camera 3 because if either or both look at the target we get an interest level of one target. The factor graph representation of the function g is shown in Figure 5.

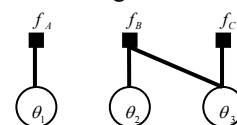


Figure 5. Factor graph representation of interest functional.

A factor graph is a bipartite graph consisting of variable nodes (i.e. θ_1 , θ_2 , and θ_3) and function nodes (i.e. f_A , f_B , and f_C) with edges connecting function nodes to those variable nodes that the function depends on. The optimization to be performed for choosing the best set of camera directions is $\max_{\theta_1, \theta_2, \theta_3} g(\theta_1, \theta_2, \theta_3)$.

The max-sum algorithm is a message passing algorithm which computes the above maximization by computing summaries at the variable and function nodes and sends appropriate messages along the edges of the factor graph. It is known that for factor graphs which are trees, this computation results in the optimal solution. For loopy graphs, the solution is suboptimal although it is known to produce near optimal results in certain cases.

The features of this factor graph solution are that the iterations of computing summaries and message passing can be easily mapped to a distributed algorithm where cameras compute summaries and send appropriate messages to their neighbors. Furthermore, since the global interest functional can be expressed as a sum of the interest contributions by each target as well as the interest contributions from the suspicion particles of the suspicion module, the global interest functional can be constructed in a distributed fashion. If each camera is only knowledgeable of the targets and cameras in its local neighborhood, the maximization of this local part of the global interest functional will give a near optimal allocation for this camera.

4 Experimental Testbed

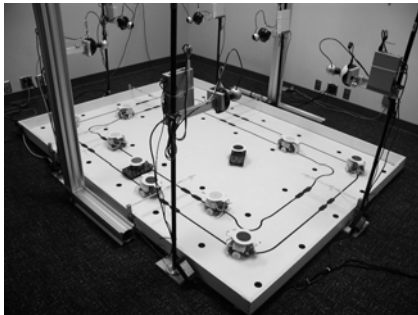


Figure 6. The Distributed Attention Testbed

4.1 Testbed Hardware

Experiments were carried out on the testbed shown in Figure 6 as an exploratory prototype of a two-layered inference architecture with resource allocation. Two types of vehicles are present: distractors, represented by line-tracking robots which automatically follow tracks drawn on the testbed surface, and vehicles of interest, represented by radio controlled tanks. To simplify image processing, vehicles were marked with either a disk of a single color (giving 4 appearance models) or a disk surrounded by a ring of a different color (16 appearance models). No *a priori* knowledge of the colors assigned to distractors or vehicles of interest is used by the system.

The sensor nodes are OpenBricks, 300 MHz x86-class tiny PCs with TrackerPod pan-tilt cameras (PTCs) mounted on each. At each timestep, each PTC points at the center of one of 4 neighboring regions on the testbed, with 0-2 regions overlapping among each pair of neighboring nodes.

4.2 Algorithms

4.2.1 Video Processing

Since this work was focused on the attentional processing, the low-level vision algorithms used here were simple. Two frames are captured at each iteration (every 2 seconds), separated by 75 msec. The YUV4:2:0-formatted images are separated into four color planes in {Red, Green, Blue, Yellow} by thresholding their Euclidean distance in UV space (i.e. using only chrominance) from four calibrated prototype colors. A simple connected-component blob tracker finds the centroids of blobs in the image, and velocity estimates are made by back-differencing centroid positions of nearest-neighbor blobs of the same color between the two frames.

Note that although this approach does individually detect all moving vehicles in the frame, future work on more complex scenes with larger numbers of vehicles could derive a velocity field directly via optical flow, only tracking individual vehicles once the attentive layer is triggered.

4.2.2 Pre-Attentive Layer (Normalcy)

Pre-attentive processing (see Figure 7) is performed by comparing the velocity field output by the video processing to a 2D histogram of heading angle and velocity, learned during a training period. Each FOV is divided into grid squares, and the velocity estimates from the video processing are mapped into their respective squares. These velocity estimates are compared to the histogram for the appropriate grid square, and if the frequency of occurrence of that velocity is below a threshold at that location, the square is flagged as “abnormal” and attentive processing is triggered at that location.

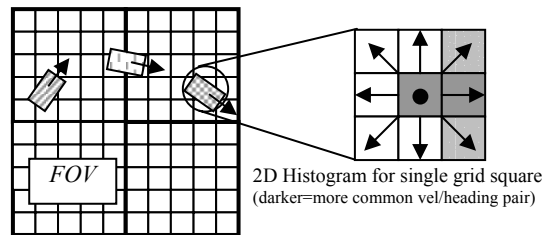


Figure 7. Histogram-Based Normalcy

The 2D histogram for each grid point is learned during a training period, during which only distractors are present. Observations are collected and placed into histogram bins. Due to the large number of histogram bins in the 4D space, collected data is “generalized” by adding multiple copies of each observation with Gaussian heading/velocity noise added.

Peripheral awareness is implemented here with a single suspicion particle fixed at the center of each FOV with no suspicion propagation between FOVs. Each FOV is statically

assigned an owning node, which maintains its suspicion level and other nodes notify it of any relevant observations.

4.2.3 Attentive Layer (Tracking)

Once the pre-attentive layer detects an abnormal velocity vector, a target record is instantiated including the detected initial state and appearance model (i.e. color) of the vehicle. The sensor node nearest the target centroid is assigned ownership of the target state, which is handed off to neighboring nodes as the target moves. Nodes with FOVs containing possible target positions are informed of the target's owner during the factor graph construction process, and pass any target sightings to the target's owner.

The actual position/velocity estimation is accomplished using a particle filter [2] using a delta function as a likelihood function and a simple dynamics model consisting of integrated uniform random velocity inputs on each axis. Some regions of the testbed are defined to be inaccessible, and in these cases, velocities are limited to the valid range. In this testbed, appearance models are assumed to be unique with perfect classification, but we have addressed methods for dealing with ambiguity in [8].

4.2.4 Resource Allocation

The basic algorithm of formulating the optimization for resource allocation as an appropriate factor graph was described earlier in Section 3.2. This section will elaborate on how to implement the factor graph formulation in a distributed way in our system.

Each camera owns a subset of tracks and suspicion regions. Furthermore, we assume each camera is aware of a local subset of neighboring cameras. Thus, each camera can compute the factor graph functions corresponding to the interest contributions of its set of owned tracks and suspicion regions. As long as each camera can own only those tracks and suspicion regions which are spatially local, the computed factor graph functions are functions of the pan-tilt parameters of only those cameras that are within its local neighborhood. Then, each camera communicates the factor graph functions that it has computed to its local neighbors. The result of this exchange is that each camera now has a local view of the complete, global factor graph which has the most influence on deciding this camera's pan-tilt configuration. Each camera can then run the max-sum algorithm on its local view of the global factor graph and allocate its pan-tilt configuration accordingly.

5.3 Results

In a series of tests (illustrative frames in Figure 8), the system ignores vehicles being driven normally (a), but detects a vehicle being driven abnormally (large dot in system output of (b)). When the vehicle is driven into the unobservable central area (c), the cameras successfully determine that the vehicle is in the unobservable zone, and, as expected, move to "guard" the borders of that zone. Furthermore, even while a known vehicle is present, the cameras can be seen to "trade off" the task of looking at a vehicle in an overlapping region,

freeing the camera up to make sure no new vehicles have begun behaving abnormally in other "suspicious" regions.

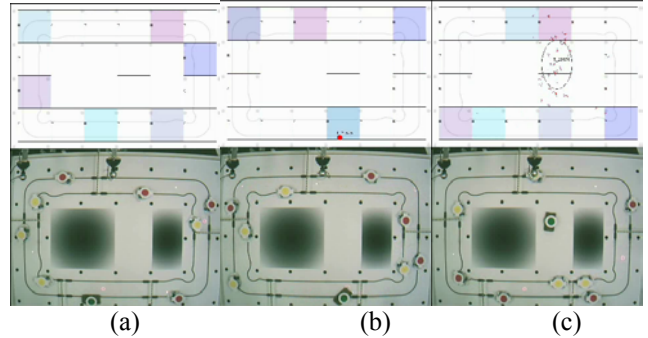


Figure 8. Detection and Tracking Sequence. Top shows system output: gray squares indicate camera directions. Bottom shows overhead ground truth view.

Further simulations have shown that this system scales up well, handling several hundred distractors without requiring additional inter-node communication or additional processing beyond the basic pre-attentive video processing functionality.

Acknowledgements

This work is partially supported by the DARPA Information Processing Technology Office under contract F30602-00-C-0139. We acknowledge the significant contributions of Patrick Cheung, Jie Liu, Qingfeng Huang, Juan (Julia) Liu, Jed Beach, and Amanda Williams in developing this system.

References

- [1] D. P. Bertsekas, D. A. Castanon, H. Tsaknakis, "Reverse auction and the solution of inequality constrained assignment problems", *SIAM J. on Optimization*, **3**, pp. 268-299, 1993.
- [2] M. Isard, A. Blake, "Condensation – conditional density propagation for visual tracking", *Int'l J. of Computer Vision*, **29**:1, pp. 5-28, 1998.
- [3] M. Chu, S. Mitter, F. Zhao, "An information architecture for distributed inference in ad hoc sensor networks", *41st Annual Allerton Conference on Comm., Control, and Computing*, October 1-3, 2003.
- [4] R. T. Collins, A. J. Lipton, H. Fujiyoshi, T. Kanade, "Algorithms for cooperative multisensor surveillance", *Proc. of IEEE*, **89**:10, October 2001.
- [5] F. Cupillard, F. Bremond, M. Thonnat, "Group behavior recognition with multiple cameras", *Proc. 6th IEEE Wkshp on Appl. of Computer Vision*, 2002.
- [6] F. R. Kschischang, B. Frey, H.-A. Loeliger, "Factor graphs and the sum-product algorithm", *IEEE Trans. Inform. Theory*, **47**:2, pp. 498-519, 2001.
- [7] J. Liu, J. Liu, J. Reich, P. Cheung, F. Zhao, "Distributed group management for track initiation and maintenance in target localization applications", *IPSN'03*, April, 2003.
- [8] J. J. Liu, M. Chu, J. Liu, J. Reich, F. Zhao, "Distributed state representation for tracking problems in sensor networks", *IPSN'04*, April, 2004.
- [9] C. Stauffer, E. Grimson, "Learning patterns of activity using real-time tracking", *PAMI*, **22**:8, pp. 747-757, 2000.
- [10] S. Ullman, "High-level Vision", MIT Press, Cambridge, MA, 1996.