Scalable Surveillance Software Architecture

Henry Detmold, Anthony Dick, Katrina Falkner,
David S. Munro and Anton van den Hengel
School of Computer Science
The University of Adelaide
{henry,ard,katrina,dave,anton}@cs.adelaide.edu.au

Ron Morrison
School of Computer Science
The University of St. Andrews
ron@cs.st-andrews.ac.uk

Abstract

Video surveillance is a key technology for enhanced protection of facilities such as airports and power stations from various types of threat. Networks of thousands of IP-based cameras are now possible, but current surveillance methodologies become increasingly ineffective as the number of cameras grows. Constructing software that efficiently and reliably deals with networks of this size is a distributed information processing problem as much as it is a video interpretation challenge. This paper demonstrates a software architecture approach to the construction of large scale surveillance network software and explores the implications for instantiating surveillance algorithms at such a scale. A novel architecture for video surveillance is presented, and its efficacy demonstrated through application to an important class of surveillance algorithms.

1 Introduction

Video surveillance hardware has developed to the point where the construction of networks of thousands of cameras is now feasible. This is largely due to the availability of IP cameras incorporating web servers, allowing footage to be delivered across an IP network. Using IP permits cameras to be connected to the closest point on a pre-existing IP network, which offers significant reductions in deployment costs when compared to analogue networks built specifically for video surveillance. At the hardware level, the scale of video surveillance networks is limited only by the capacity of the underlying network to transmit video footage.

The volume of video generated by large camera networks exceeds the capacity of current video surveillance software for processing in a coordinated manner. Research into surveillance algorithms is focused on methods capable of utilising tens, rather than hundreds, of cameras. These techniques, having been developed at small scale, typically do not achieve scalability. For example, the complexity of determining the relationships between the fields of view of a group of cameras increases exponentially with the number of cameras considered. Effective exploitation of the video generated by a large surveillance camera network requires new approaches designed with scalability as a primary concern. Our ambition is to achieve the scalability required by thousand camera networks.

The principal contributions of this paper are: i) a model software architecture for the construction of large scale video surveillance network software, ii) insight into the implications for video surveillance algorithms to be deployed on large scale networks and iii) demonstration of the efficacy of our architecture through its application to an important class of surveillance algorithms.

2 Architectural Requirements

With current camera technology, a thousand camera networks generates about 26 TB of data per day. As a result:

- The volume of data exceeds the processing capability of even high-end single server systems. Consequently, centralised processing as employed in current current surveillance networks is inadequate and *scalable distributed processing* must be employed.
- Archiving the interesting subset (obtained through techniques like decimation) of 26 terabytes of new data per day is a storage challenge. With such data volumes, using *scalable distributed storage* becomes essential.
- We do not expect thousand camera networks to exceed the bandwidth of individual network devices. However, network capacity limits will eventually be reached, so *scalable network capacity* is required.
- Any large surveillance network has a large number of components, which fail independently. Surveillance network software must be sufficiently fault tolerant to maintain an acceptable level of *availability*, using redundant components to work around failures.

Proceedings of the IEEE International Conference
on Video and Signal Based Surveillance (AVSS'06)
0-7695-2688-8/06 $20.00 © 2006 IEEE

Scalability and availability are architectural properties of the surveillance network software. If this architecture is implicit, (as in most current surveillance systems), it is difficult for software developers to reason about such properties. The need for high-level reasoning tools has led to the discipline of *software architecture* [8, 12], which centres around making the architecture explicit, through formalisation in textual and graphical languages.

We adopt the controversial view that, at least in the short and medium term, surveillance network software is not time critical. This view derives from the whole-of-system level observation that initiation of response to threats involves human intervention, and that typical times of such response are of the order of five minutes. With this in mind, imposing deadlines of the order of a few seconds on threat detection is of dubious worth. On the other had, we would admit that our focus is on threat detection, rather than surveillance network support for management of ongoing threats, which does exhibit genuine time-critical aspects.

The prototype architecture we present in this paper is based around a distributed instantiation of the *blackboard* concept, which has been widely studied both in software architectures [8] and in AI [11]. We demonstrate the efficacy of our architecture, both in architectural terms and in terms of applicability to a representative sample of current and emerging video surveillance techniques.

3 Automated Video Surveillance

The most important property of the architecture in this paper is its support for scalability. From this perspective, the automated surveillance techniques in current use [3, 13, 16], can be divided into two classes:

- *Signal processing approaches* – inherently scalable, typically because they apply to cameras in isolation.
- *Distributed information processing approaches* – which require transformation to achieve scalability.

The signal processing class includes: i) *object detection* – isolation of objects of interest within frames of footage, ii) *object classification* – labelling objects according to type (*e.g.* discriminating between people and cars), iii) *object tracking* – tracking objects across time in video footage, and iv) *command, control and inspection* – enabling human operators to interact with cameras over the network [4]. Whilst none of these are solved problems, current progress on them is rapid, largely unimpeded by architectural issues.

The distributed information processing class is more challenging from the architectural perspective. It includes: i) determination and maintenance of the *activity topology* of the network – the set of routes through which activity of interest flows between cameras, ii) *network-wide object tracking* – the extension of object tracking across multiple cameras, iii) *detection of anomalous behaviour* – typically through the application of rules describing behaviour of interest, iv) *archiving* of footage, and v) *forensic queries* – the efficient extraction of footage of interest in response to instructions from human operators.

Determination of activity topology supports all the other approaches in the distributed information processing class. It determines camera adjacency for network-wide object tracking. It enables meaningful association of footage from multiple cameras for detection of anomalous behaviour. The need to maintain accurate topological information influences the approach taken to archiving. Finally, topology information is a pre-requisite for meaningful processing of queries across time and space. Determination and maintenance of activity topology is not inherently scalable; and achieving scalability is thus a key test of the architecture.

Activity topology can be determined and maintained manually, but for large networks this is laborious and error-prone. Instead, activity topology can be gradually inferred by monitoring movement through the network over a period of time. For example, Ellis et al. [6] observe motion over a long period of time and accumulate occupancy information in a histogram. This approach has been extended by Stauffer [14] and Tieu [15] to use a more rigorous definition of a transition between cameras based on statistical significance. These methods have only been demonstrated on networks of less than 10 cameras, and it is not clear how they would scale up by an order of magnitude. On a larger scale, Brand et al. [1] consider the problem of locating hundreds of cameras distributed about an urban landscape. Their method relies on having accurate internal calibration and orientation information for each camera, and enough cameras viewing common features to constrain the solution.

Section 5 describes a method for automated topology acquisition that has been adapted to the distributed architecture, and thereby scales to camera networks that are orders of magnitude larger than those to which it has been previously applied. The development of this automated topology acquisition method leads on to a distributed tracking algorithm, supported by the topology information.

4 A Surveillance Software Architecture

Our approach is based around the use of the *blackboard* architectural style and specification of three architectural views of a system: functional, physical and interaction. The functional view defines the functional requirements (including the vision algorithms used for automated surveillance), major system component types and the mapping between them. The physical view defines the hardware on which the system operates. The interaction view defines communication and coordination between the components identified in

Requirement	*Mapping*
Object detection, classification and single-camera tracking	Computation within *surveillance zones* based on background subtraction and tracking with a Kalman/particle filter, providing inputs into the *blackboard*,
Activity topology determination and maintenance	Distributed computation within the *blackboard* to correlate disappearances and appearances.
Tracking activity across the network	Distributed computation within the *blackboard*, using appearance and activity topology information and inputs injected by the *surveillance zones*.
Rules based behaviour detection	Distributed computation within the *blackboard*, based on rules/goals injected by *command centre(s)*.
Command, control and inspection	Direction of *surveillance zones* through goals injected into the *blackboard* by *control centres*; virtual circuits established to stream desired video.
Archiving	*Data hoarders* copy video footage and derived results from the *blackboard*. *Data hoarders* manage storage, through decimation of old footage, for example.
Forensic queries	*Control centres* query *data hoarders* via injection of goals into the *blackboard*; virtual circuits established to stream results.

Table 1. Functional View of the Architecture

the functional view. Here we define the three views, then discuss the key component in our architecture, the *blackboard* and how that component is distributed.

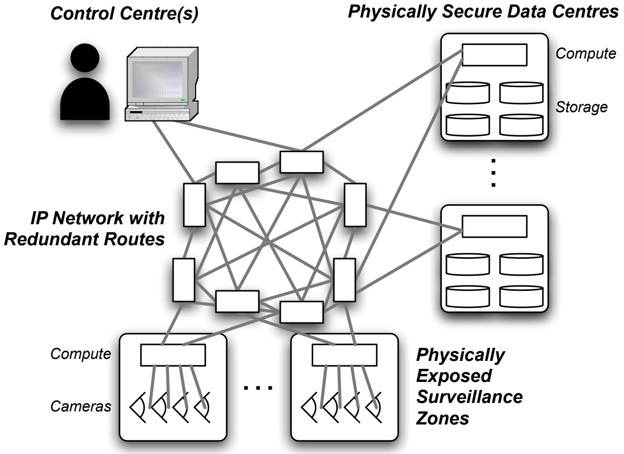
Control Centre(s)
Physically Secure Data Centres
Compute
Storage
IP Network with Redundant Routes
Compute
Cameras
Physically Exposed Surveillance Zones

Figure 1. Physical View of the Architecture.

4.1 Functional View

The components in the architecture are *surveillance zones*, *data hoarders*, *control centres* and the *blackboard*. The blackboard supports both distributed processing of surveillance algorithms and interaction between the other components. Surveillance zones are groups of cameras together with computational capacity; these feed video data into the blackboard. Data hoarders are responsible for obtaining and preserving both historical footage and derived results, and for serving queries on this data. Finally, control centre components provide a human interface for inspection and control of the network. Table 1 shows the mapping of major functional requirements to these components.

4.2 Physical View

Figure 1 shows the physical view of our surveillance network architecture. Surveillance networks typically protect high value facilities, and as such have access both to high-capacity TCP/IP network infrastructure and to computation and storage within dedicated data centres.

4.3 Interaction View

Interaction between components in the architecture is principally via the *blackboard*; Figure 2 shows this interaction. As shown in Figure 3, the blackboard is organised as a number of interacting levels, with lower levels providing results to be used by higher levels and higher levels providing goals guiding the operation of lower levels. The detection level applies processes like background subtraction to raw video and derives hypotheses about activity. The single scene analysis level derives hypotheses like “left luggage” from activity and detection of particular objects via signatures. The multi scene analysis level derives hypotheses concerning network topologies and tracking people on paths. The reasoning level derives hypotheses related to “suspicious behaviour” etc. The blackboard does not meet the strict timeliness constraints needed for video streaming, so virtual circuits are exposed from the underlying network to allow direct communication in such cases.

Each level of the blackboard has essentially the same internal structure. One or more *input drivers* processes the inputs and adds them to the level’s *information base*. In the the lowest level, the inputs come from cameras, whereas for higher levels, the inputs are from lower levels. Each level has a *controller*, which is responsible for monitoring the information base and selecting and activating *rules* from the *rule base* when data in the information base satisfies constraints for the rules. The rules encapsulate the surveillance techniques in the architecture, and can encode arbitrarily sophisticated algorithms. The application of these rules leads to the derivation of new results (*hypotheses*) by forward chaining; these are then are added to the information base. At higher layers, rules can also issue directions (*goals*) to the controller of the layer below, causing the controller to perform backward chaining to attempt to satisfy the goals. Finally, each level has one or more *significance filters* which determine which of the data in the information base is propagated as inputs to the layer above.

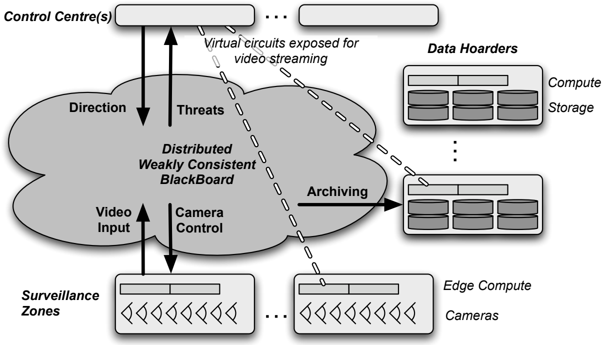

Figure 2. Interaction View of the Architecture.

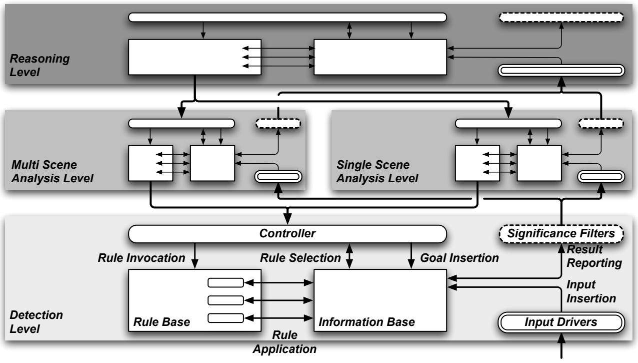

Figure 3. The Blackboard Component.

4.4 Distribution and Scalability

Figure 4 shows how the blackboard is distributed. In the vertical dimension, different levels of the blackboard are placed on different processing nodes (*i.e.* the system is partitioned according to level). At the detection and analysis levels, the blackboard is horizontally partitioned by surveillance zones (groups of cameras): each zone has associated processing partitions containing blackboards for detection and both types of analysis. Finally, the blackboard at the reasoning level is partitioned so that each partition is associated with a group of neighbouring zones.

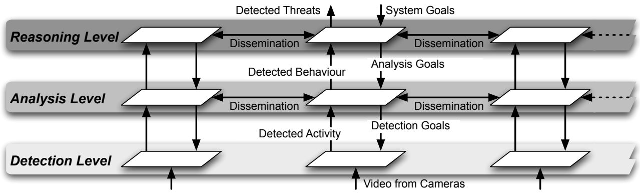

Figure 4. Distribution of the Blackboard.

Communication between the nodes has three forms:

- *Reporting* – Each node has a small set of *superior* nodes within an hierarchical structure. Each node reports results it has derived from its input data to its superiors, with reporting prioritised by significance. For example: detection nodes report objects identified by background subtraction to analysis nodes. In the case of reasoning level nodes, the superiors are human operators in a control centre. The use of multiple superior nodes provides fault tolerance in respect of the loss of any particular node, and thus availability.

- *Direction* – Each node at the reasoning and analysis levels has several *subordinates* to which it can issue directions. For example, when a reasoning node has detected a person moving along a path, it may send its subordinates a signature for that person and instructions to track him or her.

- *Dissemination* – Nodes within a level share their significant results with a small set of *peers* at the same level. As nodes derive results, they pro-actively share results that exceed a significance threshold with their peers. Also, nodes may send goals to query their peers for results in which they have an interest. Dissemination replicates the significant information in the blackboard, enabling redundancy of processing, thereby providing fault tolerance and improving availability.

There are no guarantees about consistency of shared information. Instead, dissemination provides a weakened consistency model where highly significant information propagates quickly (and will thus be reasonably consistent). In contrast, information deemed to have little significance may never leave its originating node.

The scalability of the architecture is founded on two hypotheses about surveillance network data, derived from observations made in our previous (six camera scale) work [5].

Proceedings of the IEEE International Conference
on Video and Signal Based Surveillance (AVSS'06)
0-7695-2688-8/06 $20.00 © 2006 IEEE

First, significant information within a local group of cameras (*i.e.* a surveillance zone) grows at a slower rate than the number of cameras. Second, most data produced in a zone is only relevant within that zone.

These hypotheses inform both the engineering and the science in our work. The engineering assumes the hypotheses as a basis, and the distributed blackboard has thus been formulated to exploit locality whilst supporting distribution of significant data. Part of the science in our work is to test the hypotheses on thousand camera networks; detailed empirical study of network performance will be needed for such testing, and we expect this study also to reveal unanticipated emergent effects. A future test of the long-term effectiveness of the architectures will be how well it supports architectural evolution [9] to respond to emergent effects.

5 Applying the Architecture

We now show how the architecture enables large scale distribution of a technique for automated topology acquisition and tracking described in [5]. This technique uses a hidden Markov model (HMM) to learn the activity topology of the camera network and to inform subsequent tracking. The HMM operates over N discrete states corresponding to regions in cameras' fields of view. The current state of the HMM is determined by the position in space of the person being tracked. The HMM comprises an $N \times N$ transition matrix $\mathbf{A}$ and an N dimensional initial state vector $\mathbf{b}$. The initial state vector contains one entry for each state, reflecting the probability that a person appears for the first time in that region in space. The transition matrix has a row and a column for each state, with the matrix entry $\mathbf{A}_{ij}$ giving the probability that a person moves from state i to state j. A separate HMM tracks each person under surveillance.

The original algorithm operates in distinct learning and tracking stages. During learning the movement of a single person is tracked through the network of cameras. The point at which this person enters the network results in an update to the $\mathbf{b}$ vector, and that person's subsequent movement is used to update the $\mathbf{A}$ matrix. During tracking, the matrix and vector are combined with appearance information to track people as they move within and between regions.

The distributed blackboard architecture enables solutions to a number of problems identified in testing of the original system. Specifically: i) the original implementation used a single PC for processing did not scale beyond six cameras due to limitations in bandwidth and processing capacity; ii) the transition matrix in the original algorithm increases as the square of the number of regions under surveillance. This is impractical for a thousand camera network, which with twelve regions per camera would require 144 million matrix entries!

In the architecture, cameras are grouped into surveillance zones, with which are associated processing elements in the form of detection and analysis blackboards. The transition matrix and vector for a given zone are maintained in that zone's multi-scene analysis blackboard. Since zones contain only a limited number of cameras (up to ten), the size of the matrix is manageable (less than 15,000 entries with twelve regions per camera). Further, tracking within the zone is localised (using the pre-existing algorithm) and thus tracking proceeds in all zones in parallel. In addition to the appearance vector and transition matrix, each zone maintains a disappearance vector, $\mathbf{d}$, giving the probability that a person is last seen within the zone in a given region.

Elements of $\mathbf{d}$ having high probabilities identify the corresponding region as a likely exit, whereas high $\mathbf{b}$ values identify likely entries into the zone. Objects arriving at entries and leaving from exits are reported to the reasoning level of the blackboard. The reasoning level matches an exit from some zone $\mathbf{Z1}$ with an entry into another zone $\mathbf{Z2}$, and concludes that the two zones must be adjacent in some fashion. It then directs zone ($\mathbf{Z2}$ to create an immigration matrix $\mathbf{I}^{(\mathbf{Z1},\mathbf{Z2})}$. The rows of $\mathbf{I}^{(\mathbf{Z1},\mathbf{Z2})}$ are exit regions of $\mathbf{Z1}$, whereas the columns are entry regions of $\mathbf{Z2}$. The value $\mathbf{I}^{(\mathbf{Z1},\mathbf{Z2})}_{\mathbf{ij}}$ gives the probability that a person leaving zone $\mathbf{Z1}$ at region $\mathbf{i}$ will enter $\mathbf{Z2}$ at region $\mathbf{j}$, assuming that the person enters $\mathbf{Z2}$ rather than some other zone. When tracking in zone $\mathbf{Z1}$ detects a person leaving (no longer seen after a period of time), it disseminates that fact to its peers. Each peer then uses the its immigration matrix for $\mathbf{Z1}$ to attempt to continue tracking the person.

6 Related Work

Caflisch *et.al.* [2] propose an *architecture-centred* approach to the development and maintenance of surveillance systems. We agree with the philosophy of their approach, but go deeper into the architecture as it applies to the system's desired properties (in particular, scalability). We also follow mainstream thinking in the software architectures discipline, in that we re-use and adapt an established architecture (the blackboard), whereas their architecture is created *de novo*. They provide a useful industrial perspective on the construction of surveillance systems, which motivates them to discuss software life cycle requirements such as configuration, portability and ease of extension. We avoid discussing these requirements in this paper, but address them in our ongoing work, and note that, as was their experience, the architecture provides key ingredients in the satisfaction of these requirements.

The principal contribution of Enficiaud *et.al.* [7] is the adoption of a service oriented architecture. Via this architecture's loose coupling of modules, multiple vision applications can be developed independently and subsequently integrated. The efficacy of this approach is demonstrated through its application to develop and integrate three vision applications. Whilst we have not investigated capacity for independent development directly, it is highly likely that the loosely coupled nature of the distributed blackboard will also promote such development processes.

Yuan *et.al.* [17] present a distributed vision-based surveillance system. Their system has a single centralised server (a potential bottleneck), but that server is able to "farm out" intensive computations to PCs on a LAN, thus providing some distributed processing. Interestingly, this work is formulated in terms of *hypothesis generation* and *hypothesis verification*, which accords well with the blackboard-based approach. However this work lacks an overarching software architecture; this would enable, for example, replacement of the central server with decentralised distributed processing.

Work such as Lim *et.al.* [10], like some of our own prior work [5], concentrates on the development of algorithms to exploit multiple cameras (typically about six) with centralised processing (a single server). This work has informed the construction of surveillance zones within our approach, but does not in and of itself scale to the thousand camera networks we envisage.

Finally, we adopt a blackboard approach rather than the more modern multi-agent approach. Our conservatism is based on the belief that the volume of data to be managed requires purpose specific handling (i.e. the blackboard) that generic agents would struggle to replicate. Use of a multi-agent approach would be an interesting future experiment.

7 Conclusion

This paper presents a software architecture which provides video surveillance network software with the *scalability* to support sophisticated analysis on thousand camera networks, and enables it to employ sufficient fault tolerance to achieve *availability* in a system consisting of thousands of devices. The efficacy of this architecture is demonstrated through its application to provide scalability and availability for an important class of surveillance approaches. Our work is not limited to scalability and availability: current and future work uses the architecture to provide *evolvability*, the capacity of the network to change through the addition and modification of both hardware and software components, whilst its surveillance function remains operational.

References

[1] M. Brand, M. Antone, and S. Teller. Spectral solution of large-scale extrinsic camera calibration as a graph embedding problem. In *European Conference on Computer Vision*, pages 262–273, 2004.

[2] L. Caflisch, A. Savigni, R. Schettini, and F. Tisato. A software architecture for real-time embedded monitoring systems. In *Proc. IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 540–545, 2005.

[3] R. Collins, A. Lipton, and T. Kanade. Introduction to the special section on video surveillance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):745–746, 2000.

[4] A. Dick and M. J. Brooks. Issues in automated video surveillance. In *Proc. 7th International Conference on Digital Image Computing: Techniques and Applications (DICTA'03)*, pages I:195–204, Sydney, 2003.

[5] A. Dick and M. J. Brooks. A stochastic approach to tracking objects across multiple cameras. In *Proc. Australian Joint Conference on Artificial Intelligence*, pages 160–170, 2004.

[6] T. J. Ellis, D. Makris, and J. Black. Learning a multi-camera topology. In *Joint IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 165–171, 2003.

[7] R. Enficiaud, B. Lienard, N. Allezard, R. Sebbe, S. Beucher, X. Desurmont, P. Sayd, and J. Delaigl. Clovis - a generic framework for general purpose visual surveillance applications. In *IEEE Workshop on Visual Surveillance*, pages 177–184, 2006.

[8] D. Garlan and M. Shaw. An introduction to software architecture. *Advances in Software Engineering and Knowledge Engineering*, 1, 1993.

[9] R. Greenwood, K. Mayes, S. W., B. Warboys, D. Balasubramaniam, G. Kirby, R. Morrison, and A. Sage. The impact of software-architecture compliance on system evolution. In *Software Evolution and Feedback: Theory and Practice*, pages 269–280. Wiley, 2006.

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

[11] H. P. Nii. Blackboard systems. *AI Magazine(in 2 parts)*, 7(2 and 3):38–53 and 82–106, 1986.

[12] D. Perry and A. Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4):40–52, 1992.

[13] C. Regazzoni, V. Ramesh, and G. Foresti. Scanning the issue/technology: Special issue on video communications, processing and understanding for third generation surveillance systems. *Proc, of the IEEE*, 89(10):1355–1366, 2001.

[14] C. Stauffer. Learning to track objects through unobserved regions. In *IEEE Computer Society Workshop on Motion and Video Computing*, pages II: 96–102, 2005.

[15] K. Tieu, G. Dalley, and W. Grimson. Inference of non-overlapping camera network topology by measuring statistical dependence. In *Proc. IEEE International Conference on Computer Vision*, pages II: 1842–1849, 2005.

[16] M. Valera Espina and S. A. Velastin. Intelligent distributed surveillance systems: A review. *IEE Proceedings - Vision, Image and Signal Processing*, 152(2):192–204, April 2005.

[17] X. Yuan, Z. Sun, Y. L. Varol, and G. Bebis. A distributed visual surveillance system. In *Proc. IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 199–204, 2003.