Collaborative Surveillance Using Both Fixed and Mobile Unattended Ground Sensor Platforms

Christopher P. Diehl[a], Mahesh Saptharishi[a],
John B. Hampshire II[a,b] and Pradeep K. Khosla[a,b]

[a]Department of Electrical and Computer Engineering
[b]Institute for Complex Engineered Systems (ICES)
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

We begin by considering current shortfalls with conventional surveillance systems and discuss the potential advantages of distributed, collaborative surveillance systems. Distributed surveillance systems offer the capability to monitor activity from multiple locations over time thereby increasing the likelihood of obtaining discriminating data necessary for interpretation of the activity. Yet the multiplicity of sensors magnifies the volumes of data that must be processed. We present our vision of a system which generates timely interpretations of activities in the scene automatically through the use of mechanisms for collaboration among sensing systems and efficient perception methods which complement the sensing paradigm. Then we review our recent efforts toward achieving this goal and present initial results.

Keywords: distributed surveillance, collaboration, moving object detection, moving object classification

1. INTRODUCTION

The role of a surveillance system is to provide a timely, concise view of relevant activities within an environment. Modern surveillance systems often employ a limited number of sophisticated sensors that can provide a wealth of sensor data about a region of interest. Unfortunately with certain sensor modalities, it is difficult to produce a timely interpretation from the volumes of data using current approaches for automated interpretation. This has two significant implications. The user derives little information from the system which can impact decisionmaking in time-critical scenarios, and is often unable to retask the system to resolve ambiguity in the original data as activity is occurring.

Given that these sensing systems are often quite expensive, one may ask whether another system architecture may offer improved performance. We believe that a distributed surveillance system, which utilizes a larger number of limited capability sensing systems, can offer significant advantages over conventional surveillance systems if one reconsiders how the sensing and perception processes interact. For example consider the task of object classification. Often algorithm designers assume that a classifier is provided with only one sample of data to assess the nature of a particular object. Therefore they may attempt to obtain robustness to a variety of possible signal distortions by utilizing elaborate processes which are computationally expensive. Yet such classification processes are not well matched to the sensing process.

A surveillance system is generally sensing the environment on a continuous basis. Therefore a classification decision need not be made after each sensor collection. If a particular collection yields a sample of data that is ambiguous, there is no need to spend a significant amount of computation attempting to compensate for distortions which may have caused the ambiguity. Using a simple classifier with an associated measure of classification confidence, the system can delay making a classification decision until enough samples have been collected to yield a classification with an appropriate level of confidence. Such a classification process is ideal for a distributed surveillance system since multiple sensors can focus on an object in the environment from different locations, thereby increasing the likelihood of obtaining discriminating data during a particular collection. Using such a paradigm, our general goal is to demonstrate that timely interpretations of the environment can be generated using feedback from perception processes, and robust performance can be achieved over a wide range of operating conditions.

Send correspondence to C.P. Diehl at diehl@ece.cmu.edu

In order to realize such a system, we must first define a framework for coordinated sensing, processing and communication among the sensing systems that compose the distributed surveillance system. The path that we have chosen to follow toward this objective involves defining an agent-based software architecture for collaboration among individual sensor systems. Ideally this architecture will allow the operator to provide the distributed surveillance system with high level surveillance objectives which in turn are decomposed automatically into a collection of taskings for individual sensor systems. As the sensor systems collect data about the environment, they will collaborate with one another to assemble a common interpretation of the environment. This may involve collaborative processing for detection, classification and geolocation or collaborative control for geolocation and tracking. Once an interpretation is formed, the taskings for individual sensor systems are then automatically updated in order to resolve remaining ambiguity or improve the system's ability to assess general activity in the environment. This way the user achieves maximum information gain with minimal input.

In this paper, we present our initial steps toward these long-term objectives. Our recent efforts have been focused on two fronts: definition of the agent-based architecture and development of efficient perception algorithms for detection and classification. We begin by providing an overview of CyberARIES, the agent-based architecture running on the CMU/ICES CyberScout distributed surveillance system. Then we focus on perception capabilities that have been instantiated to date within the system and review initial results from experiments demonstrating these capabilities.

2. CYBERARIES

2.1. Motivation

Before we review the fundamentals of the CyberARIES system, we need to begin by specifying our notion of collaboration in further detail in order to avoid ambiguity. Collaboration among agents in a distributed system can be achieved using a centralized or decentralized architecture.[1] A *centralized* architecture involves using a single control agent to coordinate the actions of the remaining agents to accomplish a particular goal. A *decentralized* architecture is either hierarchical or distributed in nature. A *hierarchical* architecture uses local control agents to coordinate the actions of subsets of agents within the group. A *distributed* architecture on the other hand has no control agents. Collaborative interactions between agents emerge from constraints placed on agent interactions with the environment and other agents.

For surveillance applications, a distributed architecture appears to offer two distinct advantages over centralized and hierarchical architectures. When fault-tolerance is a primary concern, a distributed architecture is ideal since the system offers a graceful degradation in capability as sensor systems fail. Given this capability to adapt to changes in the architecture, a distributed architecture should in principle scale seamlessly to situations with potentially hundreds or thousands of sensor systems as well. *We believe that a distributed surveillance system with emergent collaboration can allocate system resources in a more optimal fashion over a wide range of scenarios than a centralized or hierarchical system.* Therefore we plan to pursue a distributed architecture in order to hopefully demonstrate the robustness of such a system.

2.2. Architecture Fundamentals

2.2.1. The agent

Let us begin our overview of the CyberARIES architecture by first defining the basic building block: the agent. We define an agent as software with the following properties:

- Accepts stimuli from other agents
- Has steady state behavior in the absence of stimuli
- Can provide stimuli to other agents

The general agent structure within CyberARIES is shown in figure 1. The behavior of the agent is defined by the *agent run loop* which is executed until either the operator terminates the agent or the agent terminates itself. Communication between agents occurs through connections between stimulus sources and sinks. A *stimulus sink* receives all incoming messages (stimuli) from other agents and stores them until the agent run loop requests a stimulus. A *stimulus source* receives stimuli from the agent run loop and attempts to transmit them to the stimulus sink of the specified agent(s).

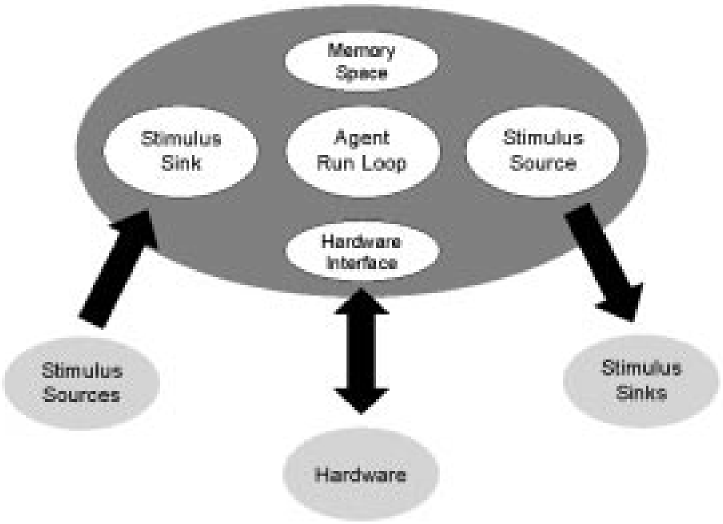

Figure 1. General agent structure within CyberARIES.

2.2.2. The distribution layer

Given that there will be no centralized or hierarchical control of the agents in the system, communication among the agents will form the basis for allocation of processing and sensing resources across the distributed surveillance system. Within CyberARIES, the *distribution layer* is the communications infrastructure that is responsible for routing stimuli between agents and regulating the flow of stimuli within the system. [*] When an agent wishes to send a stimulus to another agent, the distribution layer handles the details of establishing the necessary connections to deliver the stimulus. During transmission, it also monitors the arrival rate of stimuli relative to the processing rate of receiving agents to ensure that agents are not being overloaded. If the distribution layer detects a problem, it will ask transmitting agents to reduce their rate of transmission.

Upon receiving such a request, an agent pursues one of two options. First it communicates with other agents to determine if another agent has excess processing capacity to handle additional stimuli. If an agent accepts the processing task, some stimuli bound for the original receiving agent are transmitted to the volunteer. Otherwise if no agent can accept additional stimuli, the transmitting agent simply sleeps for a certain amount of time during each cycle of the agent run loop in order to reduce its transmission rate. Using such simple interactions among agents, we obtain a means for *dynamic load balancing* which utilizes emergent collaboration among the agents to achieve these ends.

Similar constraint-based processes that utilize the distribution layer may also lead to *emergent collaboration for perception* as well. In such tasks, one challenge is to automatically determine which agents should receive information derived from sensory data by a given agent. By utilizing some utility measure, agents receiving such information can provide feedback to the distribution layer which in turn can be used to allocate communication capacity to those connections which provide the most significant gains in perception performance. In this way, the distributed surveillance system can learn to exchange information among the agents in a manner that most effectively reduces uncertainty in the interpretation of activities in the environment.

2.3. Example Architecture: CyberScout Agent-Based Framework

In the CyberScout program, we will be utilizing two retrofitted Polaris all-terrain vehicles (ATVs) along with stationary sensors to perform tactical surveillance. On each ATV, processors will host a set of agents that perform the following functions shown in figure 2. To date our research group has focused primarily on defining agents for vehicular control and perception. Vehicular control agents are responsible for low-level control activities such as braking, steering, and throttle control. Perception agents are responsible for such tasks as moving object detection,

[*] In keeping with the spirit of our distributed architecture, the distribution layer is composed of a set of distribution agents with one running on every processor in the distributed surveillance system.

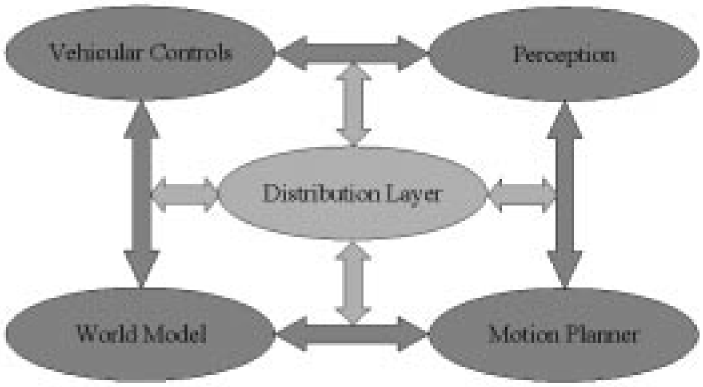

Figure 2. ATV functions defined within CyberARIES.

classification, geolocation and tracking, obstacle avoidance and landmark detection. In the next section, we will focus on our efforts to develop simple, efficient and robust methods for moving object detection and classification in video.

3. PERCEPTION FOR SURVEILLANCE

3.1. Motion Detection

Traditional approaches to motion detection compute optical flow which is computationally expensive and requires special purpose hardware to run in real-time. Only a few real-time procedures presented in the literature are suitable for standard PCs.[2–5] Such techniques generally focus on modelling the background and detecting significant changes between the background model and the current frame. Lipton et al.[4] and Haritaoglu et al.[3] employ temporal differencing strategies which involve differencing video frames with a background model, thresholding the result and extracting connected components. Such techniques are sensitive to noise and illumination changes but can quickly acquire a background model. Wren et al. and Grimson et al. avoid the sensitivities of the temporal differencing approaches by utilizing Gaussian[5] and Gaussian mixture models[2] of the background. Yet the added model complexity requires significant additional data to estimate the model parameters.

Given that the electro-optical cameras in the CyberScout system will be panning and tilting to track targets in a scene, it is important for our motion detection procedure to have the capability to quickly acquire a background model. Therefore we need to pursue a strategy similar to temporal differencing. At the same time, the motion detection algorithm should be insensitive to mild camera jitter caused by vibration on the ATV. Obviously improving the robustness to jitter negatively impacts background acquisition time. Therefore we must identify a proper balance of capabilities.

Our motion detection algorithm is very similar to that of Lipton et al. We construct a continuously adapting background model and subtract it from each frame to obtain the foreground image. Targets are then segmented using a thresholded foreground image. The background model is constructed using an autoregressive (AR) filter applied to the current frame and a time-lagged frame. This results in two foreground image hypotheses which are linearly combined to form a raw foreground image. The threshold for the subsequent frame is simply the background model obtained by applying the same AR filter to the raw foreground image. Applying systematic erosion scaled by the threshold at each pixel results in a binary image that is used for region growing. Using such an approach, we have managed to effectively eliminate most jitter and noise without a significant impact on detection performance. We anticipate further improvements by utilizing feedback from the classifier to adjust the sensitivity of the detector in order to control the false alarm rate.

3.2. Classification of Moving Objects

Once the motion detector has nominated a region of the image as a moving object, our next objective is to attempt to classify the selected image chip. We will label nominated chips as either people, vehicles or unknown objects. The people class includes individuals as well as multiple people. We will attempt to classify moving objects based on the difference image chips. The difference image was chosen because it provides a stable representation of the shape of

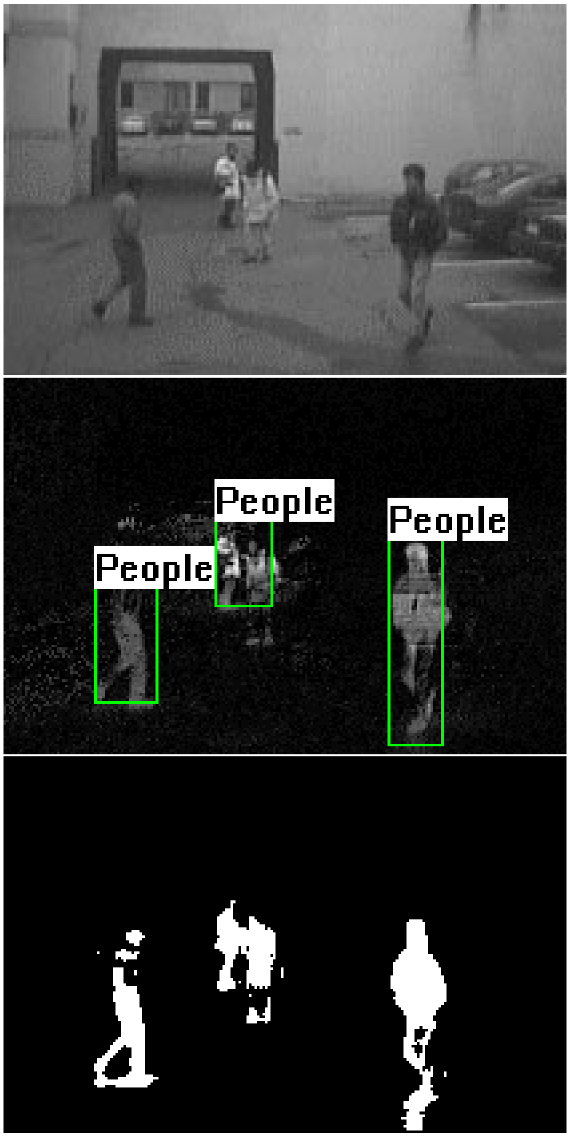

Figure 3. Motion detection and classification example: (*top*) original image, (*middle*) difference image, (*bottom*) thresholded difference image

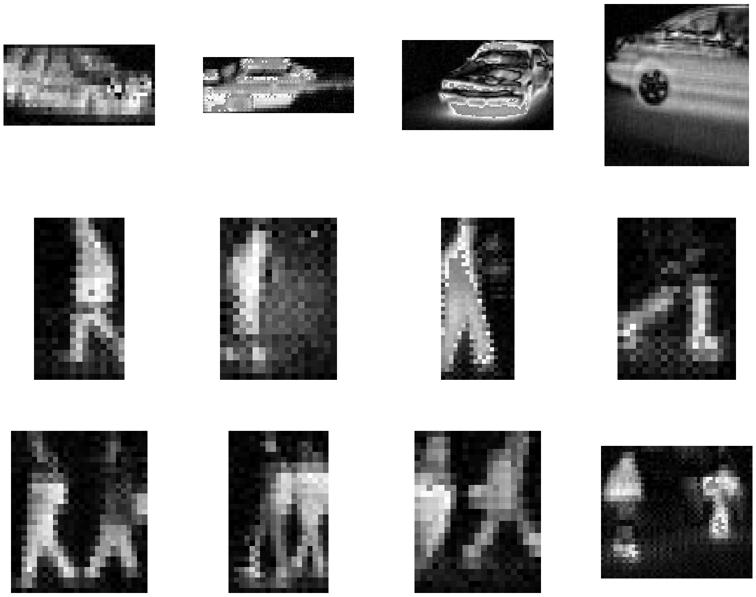

Figure 4. Difference image examples of detected people and vehicles

the object, as compared to the thresholded difference image, and minimizes the influence of the background on the classification decision.

Figure 4 displays a series of difference images of moving people and vehicles. These examples highlight various types of pattern variations that the classifier will be presented with. We will attempt to compensate for the scale variation by resizing all chips so that they fit within a 20x20 window. No other normalizations are performed prior to passing the chips to the classifier.

In keeping with our philosophy introduced earlier, our objective is to design a simple classifier with a measure of classification confidence that allows us to capitalize on the surveillance system's ability to collect multiple images of a moving object over time and space. Given that the class-conditional densities associated with people and vehicles are poorly represented by classic parametric densities, we have chosen to pursue a nonparametric approach to classifier design whereby we select the functional form of the classifier and learn the appropriate parameters from a set of examples.

The approach we employ for training the classifier is *differential learning.*[6] In contrast to *probabilistic learning* which attempts to approximate the *a posteriori* class probabilities using the available functional complexity, the objective of differential learning is simply to learn the most likely class label for each example. Using such an approach, the data requirements for certain levels of generalization performance are orders of magnitude less than those for probabilistically generated classifiers.

Differentially generated classifiers are obtained by maximizing a *classification figure-of-merit* objective function which is a monotonic function of the difference between the classifier outputs associated with the correct class and the largest other class. This quantity is referred to as the *discriminant differential.* In the spirit of differential learning, we utilize the difference between largest and next largest classifier outputs as a measure of classification confidence.

3.3. Experimental Results

For our initial experiment, we trained a *logistic linear classifier* on a collection of difference image chips using differential learning with weight decay and early stopping. We then evaluated the generalization performance of the classifier using a disjoint test set. Rejection thresholds were chosen using the test set and the false alarm set. The false alarm set constructed consists of false alarm examples due to changing illumination. Table 1 lists the number of examples used in the training and test sets.

Table 2 displays the generalization performance of the logistic linear classifier. The results clearly indicate that these classes are for the most part linearly separable. Unfortunately a significant fraction of the false alarms are classified as vehicles. After examining the weights of the classifier, it becomes evident why this is occurring. Due to variations in object position within the image chips, no obvious details of people and vehicles appear to be used in the classification. Instead only gross dimensions of the objects appear to play a significant role in classification.

In order to compensate for variations in position, we have developed a *locally translation invariant logistic linear classifier* that classifies a series of horizontal translations of a given image chip and selects the class label corresponding to the translation which yields the largest differential above the rejection threshold. Table 3 shows the generalization performance of this classifier. Although the probability of correct classification has fallen slightly due primarily to more errors on examples of people, we have obtained a large increase in false alarm rejection. This appears to be attributable to a more structured weight layer. Comparing these results with those in table 4 for the classification scheme proposed by Lipton et al., we see that our detection and classification scheme provides a significant increase in performance with a similar detection method and a lower complexity classifier. Using this classification procedure in concert with a method for corresponding objects over time and space, we will achieve additional improvements in classification and rejection performance.

4. CONCLUSION

In this paper we have attempted to motivate an alternate approach to the problem of automated surveillance. Specifically we have advocated using a distributed surveillance system containing a large number of limited capability sensors that provide increased coverage and fault tolerance at a reduced cost. In order to interpret the volumes of data generated by this type of system in a timely fashion, we are pursuing strategies to induce collaboration among software agents responsible for processing sensor data on individual sensing systems. Using simple, constrained interactions among agents along with efficient, low complexity algorithms for perception and control, we hope to show that robust, adaptive systems for distributed surveillance can emerge. We believe the initial results we have presented here highlight the promise in such a system, and we intend to demonstrate expanded capabilities for collaborative perception in the near future.

ACKNOWLEDGMENTS

We gratefully acknowledge funding provided for this research under DARPA grant F04701-97-C-0022. We also thank Ben Pugliese, Brian Boylston and Matt Ritchey for their outstanding efforts in the development of CyberARIES.

REFERENCES

1. Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, “Cooperative mobile robotics: Antecedents and directions,” *Autonomous Robots* **4**, pp. 1–23, 1997.
2. W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee, “Using adaptive tracking to classify and monitor activities in a site,” in *Proceedings, 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 22–29, 1998.
3. I. Haritaoglu, D. Harwood, and L. S. Davis, “w^4: Who? when? where? what? a real time system for detecting and tracking people.” Submitted to FGR98.
4. A. J. Lipton, H. Fujiyoshi, and R. S. Patil, “Moving target classification and tracking from real-time video,” in *Proceedings, DARPA Image Understanding Workshop*, 1998.
5. C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, “Pfinder: Real-time tracking of the human body,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(7), pp. 780–785, 1997.
6. J. B. H. II and A. Waibel, “A novel objective function for improved phoneme recognition using time-delay neural networks,” *IEEE Transactions on Neural Networks* **1**, pp. 216–228, June 1990.

Class	*Training*	*Testing*
People	1100	1094
Vehicle	1035	1027
False Alarm		134

Table 1. Number of examples of each class used for training and testing

$P_{CorrectClassification} = 0.980$

Class	*Correctly Classified*	*Misclassified*	*Rejected*
People	0.926	0.027	0.047
Vehicle	0.941	0.011	0.048
False Alarm		0.299	0.701

Table 2. Generalization performance of the logistic linear classifier

$P_{CorrectClassification} = 0.946$

Class	*Correctly Classified*	*Misclassified*	*Rejected*
People	0.910	0.080	0.010
Vehicle	0.952	0.023	0.025
False Alarm		0.067	0.933

Table 3. Generalization performance of the locally translation invariant logistic linear classifier

$P_{CorrectClassification} = 0.952$

Class	*Correctly Classified*	*Misclassified*	*Rejected*
People	0.828	0.062	0.110
Vehicle	0.868	0.025	0.107

Table 4. Generalization performance of the classifier proposed by Lipton et al.