Recent Advances in Distributed Collaborative Surveillance

M. Saptharishi[a], K. Bhat[c], C. Diehl[a], C. Oliver[ab], M. Savvides[c], A. Soto[c], J. Dolan[bc], P. Khosla[abc]

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

ABSTRACT

In Carnegie Mellon University's CyberScout project, we are developing mobile and stationary sentries capable of autonomous reconnaissance and surveillance. In this paper, we describe recent advances in the areas of efficient perception algorithms (detection, classification, and correspondence) and mission planning. In detection, we have achieved improved rejection of camera jitter and environmental variations (e.g., lighting, moving foliage) through multi-modal filtering, and we have implemented panoramic backgrounding through pseudo-real-time mosaicing. In classification, we present methods for discriminating between individuals, groups of individuals, and vehicles, and between individuals with and without backpacks. In correspondence, we describe an accurate multi-hypothesis approach based on both motion and appearance. Finally, in mission planning, we describe mapbuilding using multiple sensory cues and a computationally efficient decentralized planner for multiple platforms.

Keywords: Distributed surveillance, collaboration, moving object detection, moving object classification and correspondence, mosaicing, mission planning

1. INTRODUCTION

Reconnaissance, surveillance, and security operations are time-consuming, tedious, and potentially dangerous, but critical to the success of military and other organizations. The goal of Carnegie Mellon University's CyberScout project is to create mobile and stationary sentries that extend the sphere of awareness of human operators. By increasing sensory "reach", giving the robots a significant degree of autonomy, and providing the ability to task multiple platforms as a single logical entity, CyberScout seeks to augment human capabilities, reduce exposure to risk, and present timely, relevant information to the user.

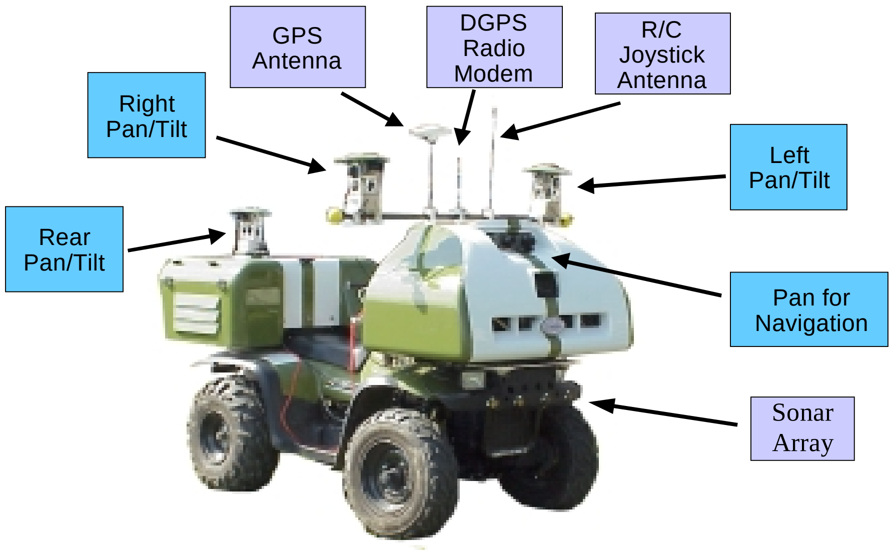

Figure 1. One of two All-Terrain Vehicles (Lewis and Clark) retrofitted for perception and navigation

Send correspondence to Mahesh Sapatharishi at mahesh@andrew.cmu.edu

The role of a surveillance system is to provide a timely, concise view of relevant activities within an environment. Modern surveillance systems often use a small number of sophisticated sensors that provide a huge volume of sensor data without the means for timely automatic interpretation. We take the alternative approach of using larger numbers of lower-cost sensors and relatively low-complexity perception algorithms that achieve high performance by cooperation between different algorithms (e.g., detection and classification) and through physical relocation of mobile sentries to adapt coverage.

In pursuing these goals, we have created a group of two mobile and four stationary sentries. The mobile sentries are retrofitted Polaris All-Terrain Vehicles (ATVs) (Figure 1) with automated throttle, steering, gearing, and braking and computation for control, navigation, perception, and communication[1]. Each ATV has multiple cameras for both navigation and surveillance. Each stationary sentry is a PC with a camera on a tripod. All sentries are able to communicate with one another via wireless Ethernet, and all run the same perception algorithms for performing surveillance[2]. Cooperation between algorithms and sentries is accommodated by a distributed, agent-based software framework called CyberARIES (Autonomous Reconnaissance and Intelligent Exploration System).

In this paper, we summarily describe the following recent advances in distributed collaborative surveillance under the CyberScout project: multi-modal filters and mosaicing for detection, finer classification discrimination, multi-hypothesis correspondence, map-building, and multiple-sentry mission planning.

2. A DISTRIBUTED AGENT-BASED FRAMEWORK FOR SURVEILLANCE

2.1 Motivation

Consider the task of performing surveillance using a network of stationary sentries. The objective is to detect, classify, track and interpret all movements in the environment. The task can be tackled using a set of concurrently executing algorithms or modules. A detector module uses the input from the camera to parse moving objects from the scene. A classifier module classifies each of the moving objects. A correspondence module temporally associates the moving objects. Moving objects can be "handed-off" from one surveillance platform to another by the correspondence agent. Other modules interpret the actions of the moving objects using the output of the correspondence module. Each module needs to work with the output of some or all of the other modules. Inherent within this modular construction is the need for robust communication. Communication between modules can either occur within a computation platform or between a networked set of platforms. A distributed collaborative surveillance system requires a software infrastructure that can abstract the functionality of low-level resources such as the network, the computation platform and other hardware or software modules. Such a software infrastructure promotes increased performance and easier development of surveillance systems.

2.2 CyberARIES

CyberARIES provides the necessary software infrastructure that abstracts low-level resources and takes full advantage of the modularity of the system. The fundamental building block within CyberARIES is an "agent". The agent can be viewed as a module contained within a shell that provides access to necessary resources through a simple abstract interface. The agent, an enhanced module, can now be viewed as an entity, which requests and/or provides services to other agents in the system. Figure 2 shows the connectivity of the agent shell. The *agent runloop* holds the algorithm specific to a module. For example, in the case of the classifier, the classification algorithms would be contained within the agent runloop. The input to the classification algorithm is accepted from other agents via the *sink*. The output of the algorithms is then distributed to other agents via the *source*. The *resource management* block manages the computational, hardware and software resources for the agent. The algorithms contained within the runloop can manually control the resource management for the agent or let the *resource management* block do it automatically. The *distribution agent* abstracts network communication between agents from the algorithms contained within the runloop. The *distribution agent* is also a CyberARIES agent and one instance of this agent exist on each surveillance platform in the network.

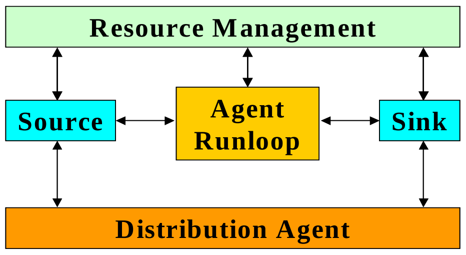

Figure 2. An agent consists of an algorithm (module) contained within the agent *runloop* and wrapped with a *source, sink, distribution agent* and a *resource management* shell.

Using the agent as a fundamental building block, CyberARIES provides for a powerful software architecture for surveillance. Using this software architecture, CyberARIES provides seamless inter-agent communication, balances the computational load of the system over the entire surveillance network, provides for mobile software agents, and allows automatic software versioning. The resource management block of an agent communicates with the resource management blocks of other agents and performs a simple optimization procedure to maximize the performance of the entire system. A surveillance system also needs to be easily reconfigurable. Reconfiguration requires that the agent be easily movable within a platform and between platforms. A CyberARIES agent has to possess the capability to move itself from one surveillance node to another. The network of stationary sentries could potentially be quite large. CyberARIES provides automatic versioning of the agents. If a newer version of an agent is installed on one node in the surveillance network, the newer version is propagated through the entire network. Tasks such as reconfiguration and versioning can be performed while the system is executing, thus, the entire network need not be stopped for a simple software upgrade.

3. DETECTION

3.1 Multi-modal detection

Most surveillance systems described in literature[2,3,4,5] have found background subtraction to be an efficient means of motion detection with a stationary camera. Unfortunately background subtraction techniques are not always robust under camera jitter, varying lighting conditions and moving foliage. Problems such as jitter and moving foliage that are periodic in nature induce multi-modal distributions of pixel intensity values. We propose a simple technique that generates a multi-modal background model of the scene. The multi-modal background can then be used for motion detection and segmentation via background subtraction.

One popular method for background model generation is through the use of AR or IIR filters[3,6]. A single AR or IIR filter is used for each pixel to estimate the dominant mode of the background. Our technique extends this method by allowing for the estimation of all the modes of the background. This is accomplished by appropriately adding an AR filter for each mode. The system is initialized with a single AR filter for each pixel. The AR filter estimates the center and width of the dominant mode of the background. Associated with each filter per pixel is a value that approximates the probability that the mode represented by the filter is seen by the pixel. When an intensity value seen by the pixel falls within the mode of one of the pixel's filters and the associated probability is greater than a preset threshold, then the pixel is declared as background. When a particular intensity value is not represented by any of the filters for that pixel, a new filter is added with an associated low probability. When a filter represents an intensity value, its probability is increased and the probabilities of the rest of the filters associated with that pixel are decreased. At each detection cycle the filters corresponding to a pixel adapt to better fit the modes of the background. In practice, we have found that no more than three or four filters are required for a robust background model.

This motion detection technique is enhanced by the use of feedback from higher-level agents such as the classifier and correspondence agents. The classifier, described in section 4, has the capability to reject spurious detections. This information can be used to either instantiate a new filter for the pixel or to increase or decrease the probability threshold for detection. Information from the correspondence agent can be used to predict future locations of detections. This considerably improves the quality of segmentation.

3.2 Mosaicing

We have developed an efficient, pseudo-real-time algorithm for constructing an image mosaic from a sequence of background images. The detection algorithm described above continuously updates the background represented by the viewable subset of the image mosaic. Several techniques have been proposed to create an image mosaic from sequences of images[7,8,9]. They obtain the registration between images by minimizing the sum-squared error of image intensities at each pixel. Although these techniques produce very accurate registration results, they tend to be slow, and typically require user interaction to initialize the registration. We create an image mosaic in pseudo-real time by locating and tracking feature points in the image sequence. This technique is much faster than the techniques developed previously, and does not require any user intervention. We also propose a method to accurately index the viewable portion of the image mosaic corresponding to a particular camera rotation.

We have implemented a fully automatic algorithm to stitch together images captured by a pan-tilt camera. The algorithm uses a feature tracker to robustly identify and track feature points through the background image sequence[10]. Feature points are chosen on the basis of the degree to which they contribute to the texture of the image window. We select and track N feature points (N = 80 to 100) through the image sequence, and use the coordinates of the tracked feature points to fit an affine model between the two images I and J in the sequence. Once the affine parameters have been calculated, we can warp all the images with respect to a common coordinate system. For our experiments, we have arbitrarily chosen the first image as the reference, and warped all other images into the first image's coordinate system. We use a triangular weighting function (with maximum weight at the image center and zero weight at the edges) to blend overlapping regions between different warped images.

The affine transformation allows for representation of the motion of a planar surface under orthographic projection. A more accurate motion model is the perspective transformation, which is characterized by eight parameters. However, for our application, we have found that an affine model is sufficiently accurate. Moreover, it is extremely simple to implement and is very fast[6].

Our system initially builds a background mosaic of the entire viewable environment. As the camera rotates, we index and update the corresponding viewable subset of the background mosaic to perform motion detection and segmentation. During the mosaic construction, we store the pan and tilt angles for all the images in the sequence, in addition to their affine warp parameters. For given rotation angles we can coarsely index the mosaic using the stored affine parameters. However, due to hysteresis and other errors in the pan-tilt unit, the indexed image is offset relative to the actual camera image by around 2-3 pixels. We solve this problem by performing registration between the indexed mosaic image and the actual camera image. We approximately mask the large moving regions in both images by subtracting the images and ignoring regions with large errors. Then we solve for the translation between the two masked images using the same techniques described above. Once we find the translation parameter, we can index the correct portion of the mosaic and can perform accurate motion segmentation. An example of the mosaicing process is shown in Figure 3.

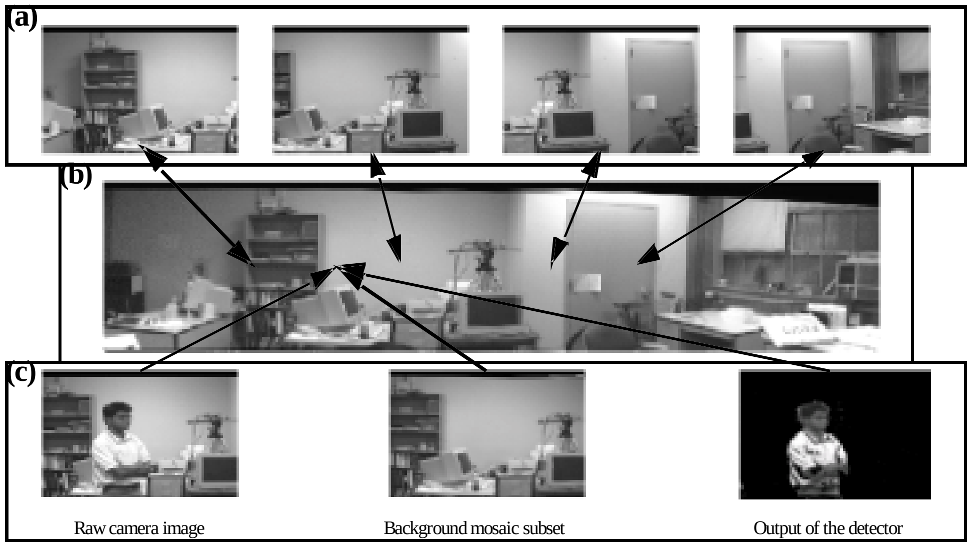

Figure 3. Results of mosaicing and detection. (b) shows the mosaic constructed using a spatial sequence of 70 images. 4 images from this sequence are shown in (a). (c) shows a moving person at a particular camera rotation, the corresponding subset indexed from the background mosaic, and the extracted foreground object. The mosaicing algorithm initially registers images at 5Hz, and the subsequent indexing and detection algorithms execute at 10 Hz on a Pentium 266MHz laptop.

4. CLASSIFICATION

4.1 General approach

Detected and segmented moving objects are presented to the classifier. The objective of the classifier is to hierarchically classify a moving object. Computational efficiency and robustness are of critical importance for classification. A surveillance system cannot be trained to recognize all types of moving objects it is ever going to see. Thus, the classifier needs to be able to determine when a novel object is present. Figure 4 shows the hierarchical classification scheme.

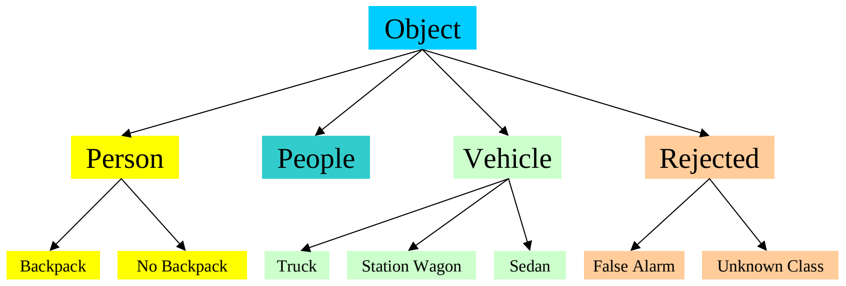

Figure 4. Given an object, the classifier attempts to hierarchically classify it. Of crucial importance is the rejection capability. The rejection process coupled with correspondence forms the basis for false alarm and novel object detection.

An object is initially classified as person/people/vehicle or is rejected. Finer distinctions are made after this initial step. As targets are temporally corresponded, the entire sequence is classified instead of just settling for a per frame classification. When an object is rejected, but is successfully corresponded (section 5), it is set aside for investigation as a potentially new class, else it is considered a false alarm. Section 4.2 describes the person/people/vehicle classification scheme. Section 4.3 describes the backpack/no backpack classification scheme.

4.2 Person/people/vehicle classification

Given that the computational resources available for classification are limited, our goal is to design a simple classification procedure that can leverage the capability to observe objects over time for robust, real-time classification. The approach we investigated involves focusing on only the spatial information contained within the image sequence. Each image within the sequence is classified independently of the other images. The sequence class label is chosen by selecting the class label that occurs most frequently. The image classifier has a rejection option to handle ambiguous examples. Those examples that are rejected will not participate in the voting for the sequence class label.

Using such a simple scheme, the sequence classification problem is transformed into a problem of classifying images from various object classes. One of the potential difficulties we must consider when designing the image classifier is the fact that the images in our dataset are not all independent. During the learning process, we operate under the assumption that the examples are independent. Yet during the evaluation of a resulting classifier's performance, we assume only that the *image sequences* are independent. Given the voting strategy we have chosen, the performance metric we use to evaluate the image classifier is the *sequence image error rate* which is defined as the expected value of the fraction of images classified incorrectly in a given sequence.

In order to learn an image classifier from the training data, we select a parameterized functional form for the classifier and employ *differential learning* to learn a set of discriminant functions that partition image space. The parameterized functional form we use is the logistic linear function. Our objective of learning a partition of image space is motivated by our interest in image rejection. In order to reject an unknown object class, we must be able to discriminate between the unknown class and the known classes. This implies that the representation must be rich enough to provide unique characterizations of examples from a variety of classes. Therefore we have selected to partition a size normalized image space instead of some low dimensional feature space[3] chosen prior to the learning, which places stronger constraints on the types of object classes that can be discriminated. Using this scheme we obtain class conditional sequence image error rates less than 5%.

4.3 Backpack classification

As a step towards a hierarchical classification system, in which classification proceeds from the general to the specific, we have considered the problem of determining whether or not a person identified by the person/people/vehicle agent is wearing a backpack. This is initially performed using a back-propagation neural network trained on labeled examples of profile silhouette images of persons wearing backpacks on either side of their body and images of persons not wearing backpacks. To achieve scale invariance, the target chips were re-sampled (preserving aspect ratio) and mapped on to a 40x40 image chip which is passed to the input of the neural network. To a certain degree, translation invariance is achieved by another pre-processing step: positioning the center of mass of the re-sampled objects on the center of the input retina of the neural network.

The resulting trained logistic linear classifier performs very well on profile views of people (when performing surveillance approximately normal to a pathway, where people are moving laterally with respect to the camera), successfully detecting even small backpacks with approximately 90% accuracy. However, the classifier falsely detects backpacks given head-on views of a person. To address this deficiency, a second neural network was trained using imagery of persons with and without a backpack, sampled from a wider range of camera viewpoints. The collected data set was labeled, sorted, and processed on a sample-by-sample basis, mirroring each sample image as needed so that the resulting data set is composed of all persons with backpacks wearing their backpack on the left side of the image chip. The resulting trained weights are mirrored for detecting right-hand side backpacks, as well.

The trained neural network is more robust against false backpack detections for a wider range of person viewpoints, correctly labeling persons with no backpack with over 95% accuracy. However, this viewpoint invariance is achieved at the cost of being able to detect small backpacks: preliminary results show approximately 85% detection accuracy of backpacks of various sizes. If the first neural network detects a backpack on either side, it will display a "backpack" label on top of the person. If the second, viewpoint-robust neural network detects a backpack, it will display an "L" on the left of the person or a corresponding "R" on right-hand side of the person, indicating the position of the detected backpack. Figure 5 (a) shows two people walking in the far background; the person on the left is wearing a right-hand backpack, and the other person is not wearing a backpack. Figure 5 (b) shows two people walking at different depths. The person in the distance seems to be wearing a small backpack. This is detected by the first classifier, but not confirmed by the second, due to the small size of the backpack. However, the person walking in the foreground is clearly wearing a backpack on the left hand side, and this observation is correctly detected and labeled by both classifiers as shown below.

Work thus far has aimed to maximize the reliability of classification on the basis of single frames. Further work will link this information with correspondence (see section 5) to provide temporal classification for a given tracked object over a sequence, and presumably greater classification accuracy.

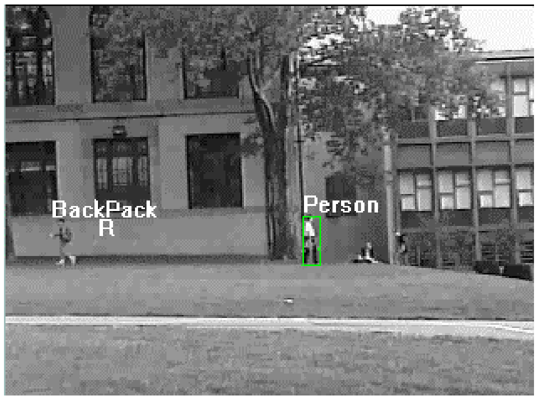

Figure 5 (a). Person on the left is wearing a normal size backpack on the right hand side and is correctly detected by both classifiers. The person on the right is not wearing a backpack.

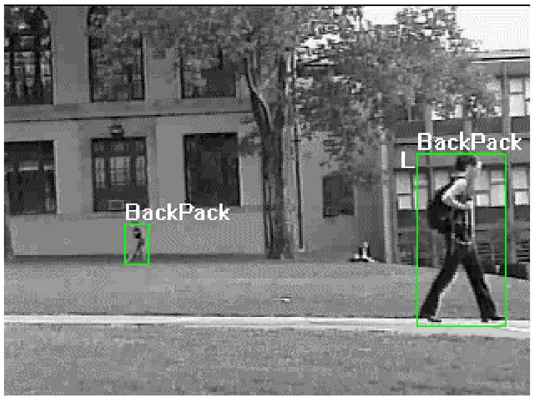

Figure 5 (b). The person on the left is wearing a small backpack, and as expected is not picked up by the second classifier. The person on the right, is clearly wearing a normal size backpack, and is detected and labeled correctly by both classifiers.

5. CORRESPONDENCE

5.1 Motivation

Temporal correspondence of a moving object plays a very important role in robustly classifying, tracking and interpreting an object's actions. The complexity of motions in the environment precludes the use of simple positional correspondence, i.e., correspondence based purely on the positions of moving objects. Positional correspondence also fails when moving objects are relatively large with respect to the field of view of the sensors. In such situations, other features of the moving objects, such as different appearance traits, need to be put to good use for robust correspondence. How can we select appearance features so as to facilitate good correspondence? The measure of goodness of the features we choose not only depends on the object in question, but also on other objects in the scene. A globally "good" set of features can be estimated a priori, but only a subset of these features might be relevant to the correspondence of a particular object. We pose the estimation of the relevance of globally good features for corresponding a particular object as an on-line learning task. We have developed a technique called *differential discriminative diagnosis* to provide a systematic method for estimating the relevance of features and checking the temporal consistency of these features for a particular object.

5.2 Approach

The correspondence task is tackled by a two-step procedure. The first step relies on positional correspondence using linear prediction. This step nominates a set of likely candidate matches for a reference object. The second step uses appearance-based correspondence to find the best match (if one exists) among those nominated by linear prediction. The appearance based correspondence technique is described in detail in reference 11. A simple linear classifier is trained to decide whether or not two images are of the same object. The input to the classifier is the absolute value of the difference between the two images being compared. Differential discriminative diagnosis is used to analyze the sensitivity of the classifier to variations in the differences between images of the same object and of different objects. Differential discriminative diagnosis identifies those features that are most relevant to the correspondence task. Efficient correspondence is achieved by enforcing the temporal consistency of the relevances for a particular object. This technique corresponds moving objects with an accuracy of 96%. This technique is currently being extended to perform correspondence of moving object across surveillance platforms.

6. MAP-BUILDING

For the construction of environment maps we are using a probabilistic approach based on an occupancy grid representation. We use a Bayes Net as our central mechanism for probabilistic integration and fusion of perceptual information. We are using vision as our main sensor modality for obstacle detection, due both to its ability to sense a large number of different features of the environment and its passive, unobtrusive nature. Our approach to visual perception is an adaptive integration of multiple visual cues, such as color, stereo vision, motion and texture. In addition to vision, we are using a set of front sonars (see Figure 1) as a backup safety system for close obstacles (< 5m). Figure 6 shows the configuration of the Bayes Net that we are currently using. A description of our algorithms to detect obstacles using stereovision and color can be found in reference 12.

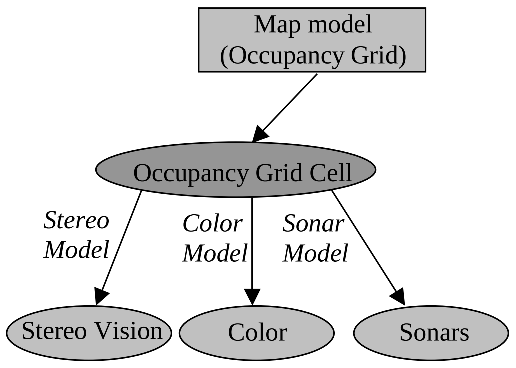

Figure 6. Bayes Net used to construct environment maps

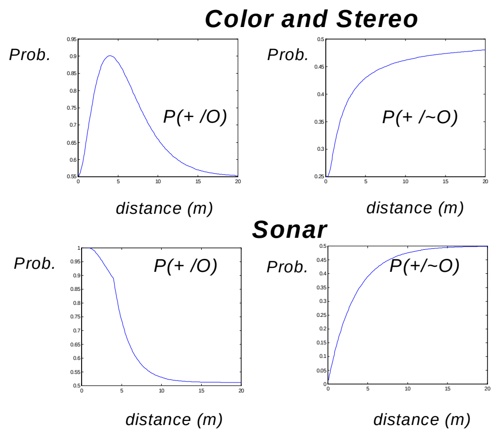

Figure 7. Conditional probability distributions for sonar, stereo vision, and color.

The use of Bayes Nets provides easy extensibility of the perceptual system to incorporate new visual cues or different sensor modalities. In order to add a new sensor, the only requirement is to provide the conditional probability distribution of the sensor in terms of its parent nodes. Figure 7 shows the empirical models that we are currently using for the color, stereo, and sonar sensors.

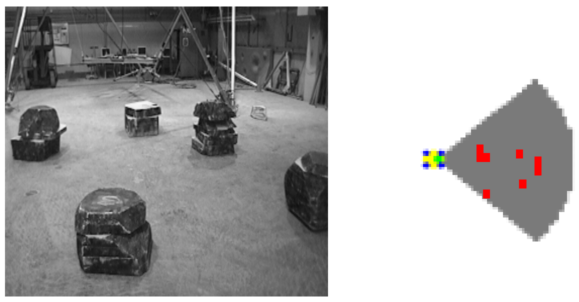
Figure 8. Set of obstacles and resultant map

For a given sensor, the probability distribution P(+/O) represents the probability that the sensor effectively detects an obstacle. P(+/~O) represents the probability of detecting a non-existent obstacle, i.e., a false positive. The figure shows that the models assign greater confidence to the ultrasonic sensors for the detection of close obstacles. In contrast, greater confidence is assigned to vision for far obstacles.

Figure 8 shows an image of a map built using the Bayes net discussed above for case of the obstacles inside the current field of view of the robot.

7. MISSION PLANNING

Examples of reconnaissance and surveillance missions include building/area stakeouts, area patrols, mapping unknown terrain and battlefield awareness. All of these scenarios require robust behavior in the face of a dynamic environment involving moving obstacles, targets, and sentries. We have developed a framework for high-level mission specification, control, and coordination of multiple sentries performing reconnaissance and surveillance. From the user's point of view, mission planning involves specifying a surveillance area and then viewing the results. However, between these two events an autonomous multi-agent mission control architecture (Figure 9) must perform the following:

1. Task Decomposition and Goal Allocation
2. Path Planning
3. Plan Merging
4. Mission Coordination and Execution
5. Surveillance

Task decomposition refers to the interpretation of the user's specifications, and goal allocation involves tasking the appropriate sentry to its most advantageous surveillance point. Currently, the user can identify a polygonal area for surveillance, and the map-watching agent will determine optimal surveillance points for the sentries. In future iterations of this architecture, sentries may also be tasked to intermediate locations in order to maximize real-time map-building efficiency.

Once the goal and waypoints have been determined, each mobile sentry plans its own path based on *a priori* or environmental map data (see section 6). This decoupled path-planning approach is used because it scales well with an increasing number of mobile sentries, but the resulting paths must still be merged to avoid collisions and gridlock conditions. The process of plan merging gives rise to the concept of checkpoints. Checkpoints are positions in a mobile sentry's plan where some specialized action must be executed. With respect to spatial considerations, prioritized checkpoints are placed near possible intersections to avoid collisions. Other checkpoint-related activities include performing intermediate surveillance and map-building, implementing the automatic convoying capabilities of a group of mobile sentries, or avoiding moving obstacles detected by other sentries.

After ATV path plans are merged, the finalized mission plans are constructed from a collection of available functions called the Checkpoint/Priority/Action Database (CPAD). CPAD routines provide the framework for specifying and executing mission plans, advanced maneuvers and statistical obstacle avoidance. Mission plans are executed by the mission controllers located on each ATV (see Figure 9). A mission controller sends commands to its respective low-level vehicular controller while stepping through the high-level mission plan. A set of mission controllers will communicate state information while coordinating vehicle motions at an intersection or while performing automatic convoying.

During execution of a mission plan, the map-watching agent may determine that a current path is no longer valid, causing the mission controllers to halt execution and re-plan. This assertion is based on the detection of new obstacles that prohibit successful completion of the current plan. While the map-watching agent assimilates sensor data from all sensors in the system to generate the current map, it also has access to past versions of the map. As the workspace evolves over time, moving obstacles can be detected, and if there is a significant probability that the obstacles will collide with an ATV, then an appropriately placed checkpoint is inserted into the mission plan for that vehicle.

By automatically handling the dynamic workspace of the CyberScout sentries, this autonomous mission control architecture is able to offer robustness without the need for continuous, direct user intervention.

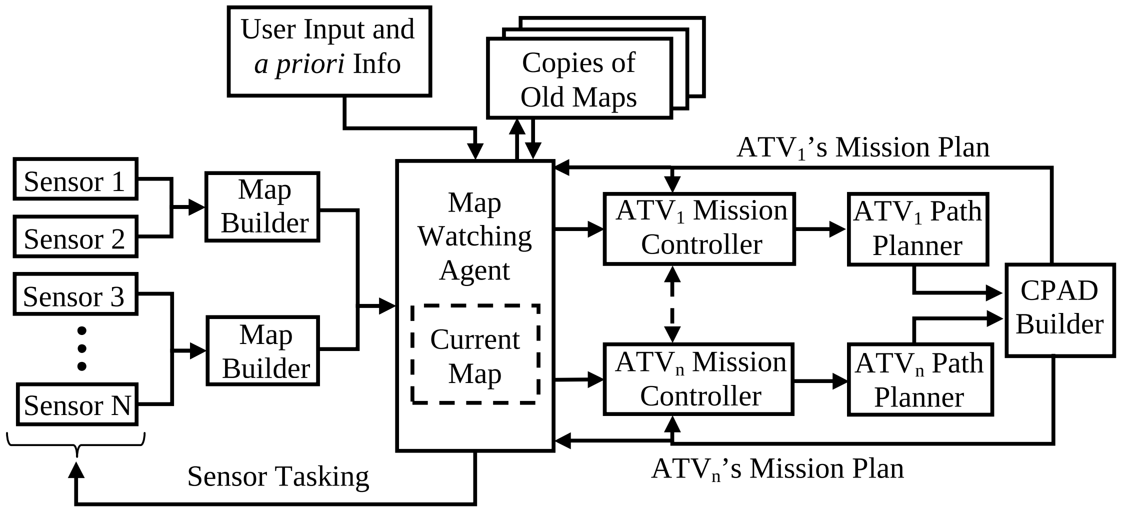

Figure 9. Autonomous Multi-Agent Mission Control Architecture

8. CONCLUSION

We have presented some of the key advances and thrusts of Carnegie Mellon University's CyberScout project. The thrust, thus far, has been in creating the software and hardware infrastructure, and cutting edge low-level and higher-level algorithms for surveillance and reconnaissance. Future work will place a larger emphasis on novel collaboration techniques, using the current infrastructure, for highly robust autonomous surveillance.

9. ACKNOWLEDGMENTS

This work was supported through DARPA's Tactical Sensors Program via a Space and Naval Warfare Systems Center Contract F04701-97-C-0022. The technical point of contact for this DARPA program is Dr. Edward Carapezza.

10. REFERENCES

1. J.M. Dolan, A. Trebi-Ollennu, A. Soto, and P. Khosla, "Distributed Tactical Surveillance with ATVs," SPIE Proceedings on Unmanned Ground Vehicle Technology, *AeroSense '99*, **vol. 3693**, pp. 192-199, April 1999.
2. C. P. Diehl, M. Saptharishi, J. B. Hampshire II, P. K. Khosla, "Collaborative Surveillance Using Both Fixed and Mobile Unattended Ground Sensors," *Proceedings of the SPIE*, **vol. 3713**, pp. 178-185, July 1999.
3. A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving target classification and tracking from real time video," *IEEE Workshop on Application of Computer Vision*, pp. 8-14, 1998.
4. W. E. L. Grimson, L. Lee, R. Romano, and C. Stauffer, "Using adaptive tracking to classify and monitor activities in a site," *CVPR98*, pp. 22-31, 1998.
5. I. Haritaoglu, D. Harwood, and L. Davis, "W4: Who? When? Where? What? A real time system for detecting and tracking people," *IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 222-227, 1998.

6. K. Bhat, M. Saptharishi and P. Khosla, “Motion Detection and Segmentation Using Image Mosaics,” to be presented at *IEEE International Conference on Multimedia and Expo 2000 (ICME)*, Aug 2000.
7. R.Szeliski and H.Shum, “Creating full view panoramic image mosaics and environment maps,” *Computer Graphics Proceedings*, *Annual Conference Series*, pp. 251-258, 1997.
8. M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu, “Mosaic representations of video sequences and their applications,” *Signal Processing: Image Communication*, *special issue on Image and Video Semantics: Processing, Analysis, and Application*, **vol. 8**, no. 4, pp.327-351, May 1996.
9. F. Dufaux and F. Moscheni, “Background Mosaicing for low bit rate video coding,” *IEEE ICIP'96*, pp. 673-676, September 1996.
10. J. Shi and C. Tomasi, “Good Features to Track,” *CVPR94*, pp. 593-600, June 1994.
11. M. Saptharishi, J. B. Hampshire II, and P. K. Khosla, “Agent-Based Moving Object Correspondence Using Differential Discriminative Diagnosis,” *to be published in CVPR2000.*
12. A. Soto, M. Saptharishi, A. Trebi-Ollennu, J. Dolan, and P. Khosla, “Cyber-ATVs: Dynamic and Distributed Reconnaissance and Surveillance Using All-Terrain UGVs,” *Proceedings of the International Conference on Field and Service Robotics*, pp. 329-334, August, 1999.