ATTACHMENT F

TO REQUEST FOR *EX PARTE* REEXAMINATION OF U.S. PATENT NO. 7,868,912

Moving Object Detection and Event Recognition Algorithms for Smart Cameras

Thomas J. Olson
Frank Z. Brill

Texas Instruments
Research & Development
P.O. Box 655303, MS 8374, Dallas, TX 75265
E-mail: olson@csc.ti.com, brill@ti.com
http://www.ti.com/research/docs/iuba/index.html

Abstract

Smart video cameras analyze the video stream and translate it into a description of the scene in terms of objects, object motions, and events. This paper describes a set of algorithms for the core computations needed to build smart cameras. Together these algorithms make up the Autonomous Video Surveillance (AVS) system, a general-purpose framework for moving object detection and event recognition. Moving objects are detected using change detection, and are tracked using first-order prediction and nearest neighbor matching. Events are recognized by applying predicates to the graph formed by linking corresponding objects in successive frames.The AVS algorithms have been used to create several novel video surveillance applications. These include a video surveillance shell that allows a human to monitor the outputs of multiple cameras, a system that takes a single high-quality snapshot of every person who enters its field of view, and a system that learns the structure of the monitored environment by watching humans move around in the scene.

1 Introduction

Video cameras today produce images, which must be examined by humans in order to be useful. Future 'smart' video cameras will produce information, including descriptions of the environment they are monitoring and the events taking place in it. The information they produce may include images and video clips, but these will be carefully selected to maximize their useful information content. The symbolic information and images from smart cameras will be filtered by programs that extract data relevant to particular tasks. This filtering process will enable a single human to monitor hundreds or thousands of video streams.

The research described in this report was sponsored in part by the DARPA Image Understanding Program.

In pursuit of our research objectives [Flinchbaugh, 1997], we are developing the technology needed to make smart cameras a reality. Two fundamental capabilities are needed. The first is the ability to describe scenes in terms of object motions and interactions. The second is the ability to recognize important events that occur in the scene, and to pick out those that are relevant to the current task. These capabilities make it possible to develop a variety of novel and useful video surveillance applications.

1.1 Video Surveillance and Monitoring Scenarios

Our work is motivated by a several types of video surveillance and monitoring scenarios.

Indoor Surveillance: Indoor surveillance provides information about areas such as building lobbies, hallways, and offices. Monitoring tasks in lobbies and hallways include detection of people depositing things (e.g., unattended luggage in an airport lounge), removing things (e.g., theft), or loitering. Office monitoring tasks typically require information about people's identities: in an office, for example, the office owner may do anything at any

time, but other people should not open desk drawers or operate the computer unless the owner is present. Cleaning staff may come in at night to vacuum and empty trash cans, but should not handle objects on the desk.

Outdoor Surveillance: Outdoor surveillance includes tasks such as monitoring a site perimeter for intrusion or threats from vehicles (e.g., car bombs). In military applications, video surveillance can function as a sentry or forward observer, e.g. by notifying commanders when enemy soldiers emerge from a wooded area or cross a road.

In order for smart cameras to be practical for real-world tasks, the algorithms they use must be robust. Current commercial video surveillance systems have a high false alarm rate [Ringler and Hoover, 1995], which renders them useless for most applications. For this reason, our research stresses robustness and quantification of detection and false alarm rates. Smart camera algorithms must also run effectively on low-cost platforms, so that they can be implemented in small, low-power packages and can be used in large numbers. Studying algorithms that can run in near real time makes it practical to conduct extensive evaluation and testing of systems, and may enable worthwhile near-term applications as well as contributing to long-term research goals.

1.2 Approach

The first step in processing a video stream for surveillance purposes is to identify the important objects in the scene. In this paper it is assumed that the important objects are those that move independently. Camera parameters are assumed to be fixed. This allows the use of simple change detection to identify moving objects. Where use of moving cameras is necessary, stabilization hardware and stabilized moving object detection algorithms can be used (e.g. [Burt et al, 1989, Nelson, 1991]. The use of criteria other than motion (e.g., salience based on shape or color, or more general object recognition) is compatible with our approach, but these criteria are not used in our current applications.

Our event recognition algorithms are based on graph matching. Moving objects in the image are tracked over time. Observations of an object in successive video frames are linked to form a directed graph (the *motion graph*). Events are defined in terms of predicates on the motion graph. For instance, the beginning of a chain of successive observations of an object is defined to be an ENTER event. Event detection is described in more detail below.

Our approach to video surveillance stresses 2D, image-based algorithms and simple, low-level object representations that can be extracted reliably from the video sequence. This emphasis yields a high level of robustness and low computational cost. Object recognition and other detailed analyses are used only after the system has determined that the objects in question are interesting and merit further investigation.

1.3 Research Strategy

The primary technical goal of this research is to develop general-purpose algorithms for moving object detection and event recognition. These algorithms comprise the Autonomous Video Surveillance (AVS) system, a modular framework for building video surveillance applications. AVS is designed to be updated to incorporate better core algorithms or to tune the processing to specific domains as our research progresses.

In order to evaluate the AVS core algorithms and event recognition and tracking framework, we use them to develop applications motivated by the surveillance scenarios described above. The applications are small-scale implementations of future smart camera systems. They are designed for long-term operation, and are evaluated by allowing them to run for long periods (hours or days) and analyzing their output.

The remainder of this paper is organized as follows. The next section discusses related work. Section 3 presents the core moving object detection and event recognition algorithms, and the mechanism used to establish the 3D positions of objects. Section 4 presents applications that have been built using the AVS framework. The final section discusses the current state of the system and our future plans.

2 Related Work

Our overall approach to video surveillance has been influenced by interest in selective attention and task-oriented processing [Swain and Stricker, 1991, Rimey and Brown, 1993, Camus et al., 1993]. The fundamental problem with current video surveillance technology is that the useful information density of the images delivered to a human is very low; the vast majority of surveillance video frames contain no useful information at all. The fundamental role of the smart camera described above is to reduce the volume of data produced by the camera, and increase the value of that data. It does this by discarding irrelevant frames, and by expressing the information in the relevant frames primarily in symbolic form.

2.1 Moving Object Detection

Most algorithms for moving object detection using fixed cameras work by comparing incoming video frames to a reference image, and attributing significant differences either to motion or to noise. The algorithms differ in the form of the comparison operator they use, and in the way in which the reference image is maintained. Simple intensity differencing followed by thresholding is widely used [Jain et al., 1979, Yalamanchili et al., 1982, Kelly et al., 1995, Bobick and Davis, 1996, Courtney, 1997] because it is computationally inexpensive and works quite well in many indoor environments. Some algorithms provide a means of adapting the reference image over time, in order to track slow changes in lighting conditions and/or changes in the environment [Karmann and von Brandt, 1990, Makarov, 1996a]. Some also filter the image to reduce or remove low spatial frequency content, which again makes the detector less sensitive to lighting changes [Makarov et al., 1996b, Koller et al., 1994].

Recent work [Pentland, 1996, Kahn et al., 1996] has extended the basic change detection paradigm by replacing the reference image with a statistical model of the background. The comparison operator becomes a statistical test that estimates the probability that the observed pixel value belongs to the background.

Our baseline change detection algorithm uses thresholded absolute differencing, since this works well for our indoor surveillance scenarios. For applications where lighting change is a problem, we use the adaptive reference frame algorithm of Karmann and von Brandt [1990]. We are also experimenting with a probabilistic change detector similar to Pfinder [Pentland, 1996.

Our work assumes fixed cameras. When the camera is not fixed, simple change detection cannot be used because of background motion. One approach to this problem is to treat the scene as a collection of independently moving objects, and to detect and ignore the visual motion due to camera motion [e.g. Burt et al., 1989] Other researchers have proposed ways of detecting features of the optical flow that are inconsistent with a hypothesis of self motion [Nelson, 1991].

In many of our applications moving object detection is a prelude to person detection. There has been significant recent progress in the development of algorithms to locate and track humans. Pfinder (cited above) uses a coarse statistical model of human body geometry and motion to estimate the likelihood that a given pixel is part of a human. Several researchers have described methods of tracking human body and limb movements [Gavrila and Davis, 1996, Kakadiaris and Metaxas, 1996] and locating faces in images [Sung and Poggio, 1994, Rowley et al., 1996]. Intille and Bobick [1995] describe methods of tracking humans through episodes of mutual occlusion in a highly structured environment. We do not currently make use of these techniques in live experiments because of their computational cost. However, we expect that this type of analysis will eventually be an important part of smart camera processing.

2.2 Event Recognition

Most work on event recognition has focussed on events that consist of a well-defined sequence of primitive motions. This class of events can be converted into spatiotemporal patterns and recognized using statistical pattern matching techniques. A number of researchers have demonstrated algorithms for recognizing gestures and sign language [e.g., Starner and Pentland, 1995]. Bobick and Davis [1996] describe a method of recognizing ste-

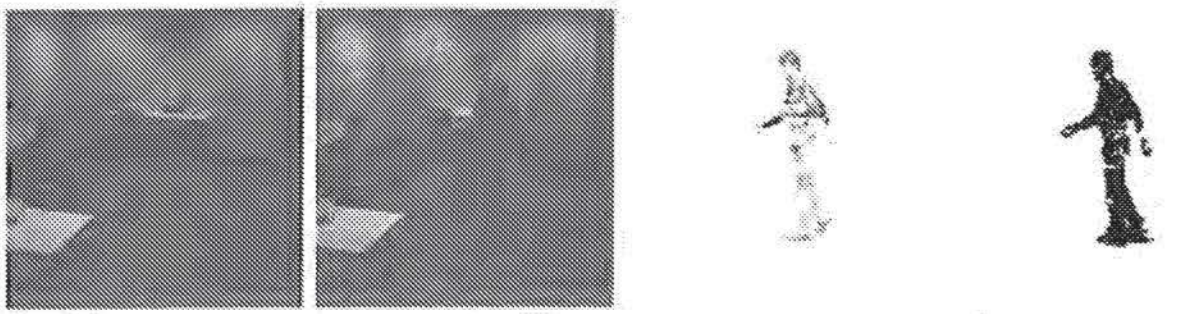

Figure 1: Image processing steps for moving object detection.

reotypical motion patterns corresponding to actions such as sitting down, walking, or waving.

Our approach to event recognition is based on the video database indexing work of Courtney [1997], which introduced the use of predicates on the motion graph to represent events. Motion graphs are well suited to representing abstract, generic events such as 'depositing an object' or 'coming to rest', which are difficult to capture using the pattern-based approaches referred to above. On the other hand, pattern-based approaches can represent complex motions such as 'throwing an object' or 'waving', which would be difficult to express using motion graphs. It is likely that both pattern-based and abstract event recognition techniques will be needed to handle the full range of events that are of interest in surveillance applications.

3 AVS Tracking and Event Recognition Algorithms

This section describes the core technologies that provide the video surveillance and monitoring capabilities of the AVS system. There are three key technologies: moving object detection, visual tracking, and event recognition. The moving object detection routines determine when one or more objects enter a monitored scene, decide which pixels in a given video frame correspond to the moving objects versus which pixels correspond to the background, and form a simple representation of the object's image in the video frame. This representation is referred to as a *motion region*, and it exists in a single video frame, as distinguished from the *world objects* which exist in the world and give rise to the motion regions.

Visual tracking consists of determining correspondences between the motion regions over a sequence of video frames, and maintaining a single representation, or *track*, for the world object which gave rise to the sequence of motion regions in the sequence of frames. Finally, event recognition is a means of analyzing the collection of tracks in order to identify events of interest involving the world objects represented by the tracks.

3.1 Moving Object Detection

The moving object detection technology we employ is a 2D change detection technique similar to that described in Jain et al. [1979] and Yalamanchili et al [1982]. Prior to activation of the monitoring system, an image of the background, i.e., an image of the scene which contains no moving or otherwise interesting objects, is captured to serve as the *reference image*. When the system is in operation, the absolute difference of the current video frame from the reference image is computed to produce a *difference image*. The difference image is then thresholded at an appropriate value to obtain a binary image in which the "off" pixels represent background pixels, and the "on" pixels represent "moving object" pixels. The four-connected components of moving object pixels in the thresholded image are the motion regions (see Figure 1).

Simple application of the object detection procedure outlined above results in a number of errors, largely due to the limitations of thresholding. If the threshold used is too low, camera noise and shadows will produce spurious objects; whereas if the threshold is too high, some portions of the objects in the scene will fail to be separated from the back-

ground, resulting in *breakup*, in which a single world object gives rise to several motion regions within a single frame. Our general approach is to allow breakup, but use grouping heuristics to merge multiple connected components into a single motion region and maintain a one-to-one correspondence between motion regions and world objects within each frame.

One grouping technique we employ is 2D morphological dilation of the motion regions. This enables the system to merge connected components separated by a few pixels, but using this technique to span large gaps results in a severe performance degradation. Moreover, dilation in the image space may result in incorrectly merging distant objects which are nearby in the image (a few pixels), but are in fact separated by a large distance in the world (a few feet).

If 3D information is available, the connected component grouping algorithm makes use of an estimate of the size (in world coordinates) of the objects in the image. The bounding boxes of the connected components are expanded vertically and horizontally by a distance measured in feet (rather than pixels), and connected components with overlapping bounding boxes are merged into a single motion region. The technique for estimating the size of the objects in the image is described in section 3.4 below.

3.2 Tracking

The function of the AVS tracking routine is to establish correspondences between the motion regions in the current frame and those in the previous frame. We use the technique of Courtney [1997], which proceeds as follows. First assume that we have computed 2D velocity estimates for the motion regions in the previous frame. These velocity estimates, together with the locations of the centroids in the previous frame, are used to project the locations of the centroids of the motion regions into the current frame. Then, a *mutual nearest-neighbor* criterion is used to establish correspondences.

Let P be the set of motion region centroid locations in the previous frame, with p_i one such location. Let p'_i be the projected location of p_i in the current frame, and let be the set of all such projected locations in the current frame. Let C be the set of motion region centroid locations in the current frame. If the distance between p'_i and $c_j \in C$ is the smallest for all elements of C, *and* this distance is also the smallest of the distances between c_j and all elements of P (i.e., p'_i and c_j are mutual nearest neighbors), then establish a correspondence between p_i and c_j by creating a bidirectional *strong link* between them. Use the difference in time and space between p_i and c_j to determine a velocity estimate for c_j, expressed in pixels per second. If there is an existing track containing p_i, add c_j to it. Otherwise, establish a new track, and add both p_i and c_j to it.

The strong links form the basis of the tracks with a high-confidence of their correctness. Video objects which do not have mutual nearest neighbors in the adjacent frame may fail to form correspondences because the underlying world object is involved in an event (e.g., enter, exit, deposit, remove). In order to assist in the identification of these events, objects without strong links are given unidirectional *weak links* to the their (non-mutual) nearest neighbors. The weak links represent potential ambiguity in the tracking process.The motion regions in all of the frames, together with their strong and weak links, form a *motion graph*.

Figure 2 depicts a sample motion graph. In the figure, each frame is one-dimensional, and is represented by a vertical line (F0 - F18). Circles represent objects in the scene, the dark arrows represent strong links, and the gray arrows represent weak links. An object enters the scene in frame F1, and then moves through the scene until frame F4, where it deposits a second object. The first object continues to move through the scene, and exits at frame F6. The deposited object remains stationary. At frame F8 another object enters the scene, temporarily occludes the stationary object at frame F10 (or is occluded by it), and then proceeds to move past the stationary object. This second moving object reverses directions around frames F13 and F14, returns to remove the stationary object in frame F16, and finally exits in frame F17. An additional object enters in frame F5 and exits in frame F8 without interacting with any other object.

As indicated by the striped fill patterns in Figure 2, the correct correspondences for the tracks are am-

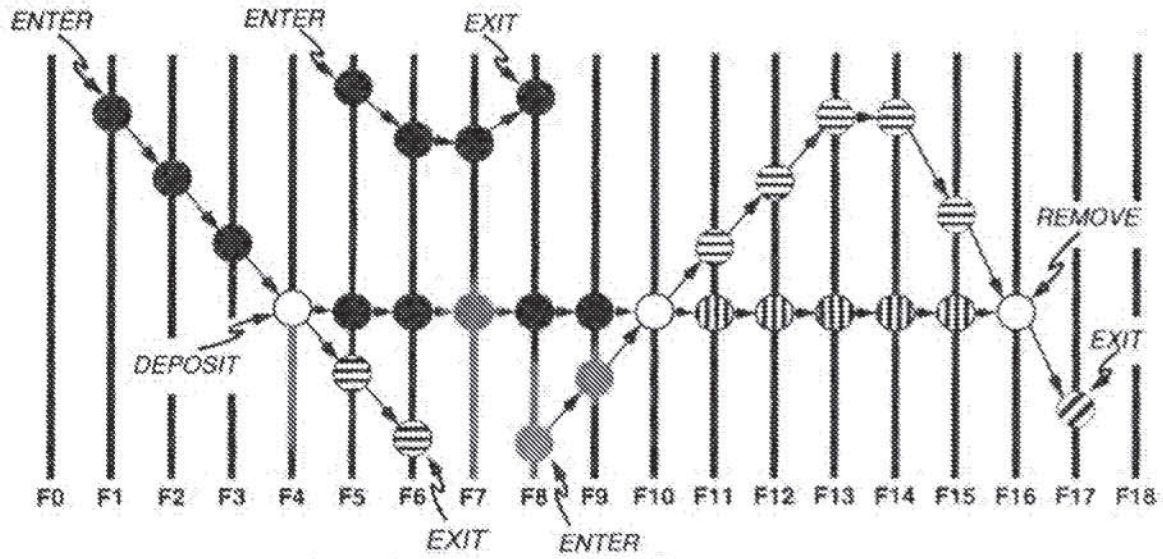

Figure 2: Event detection in the motion graph.

biguous after object interactions such as the occlusion in frame F10. The AVS system resolves this ambiguity where possible by preferring to match moving objects with moving objects, and stationary objects with stationary objects. The distinction between moving and stationary tracks is computed using thresholds on the velocity estimates, and hysteresis for stabilizing transitions between moving and stationary.

Following an occlusion (which may last for several frames) the frames immediately before and after the occlusion are compared (e.g., frames F9 and F11 in Figure 2). The AVS system examines each stationary object in the pre-occlusion frame, and searches for its correspondent in the post-occlusion frame (which should be exactly where it was before, since the object is stationary). This procedure resolves a large portion of the tracking ambiguities. General resolution of ambiguities resulting from multiple moving objects in the scene is a topic for further research. The AVS system may benefit from inclusion of a "closed world tracking" facility such as that described by Intille and Bobick [1995a, 1995b].

3.3 Event Recognition

Certain features of tracks and pairs of tracks correspond to events. For example, the beginning of a track corresponds to an ENTER event, and the end corresponds to an EXIT event. In an on-line event detection system, it is preferable to detect the event as near in time as possible to the actual occurrence of the event. The previous system which used motion graphs for event detection [Courtney, 1997] operated in a batch mode, and required multiple passes over the motion graph, precluding on-line operation. The AVS system detects events in a single pass over the motion graph, as the graph is created. However, in order to reduce errors due to noise, the AVS system introduces a slight delay of n frame times (n=3 in the current implementation) before reporting certain events. For example, in Figure 2, an enter event occurs on frame F1. The AVS system requires the track to be maintained for n frames before reporting the enter event. If the track not maintained for the required number of frames, it is ignored, and the enter event is not reported, e.g., if $n > 4$, the object in Figure 2 which enters in frame F5 and exits in frame F8 will not generate any events.

A track that splits into two tracks, one of which is moving, and the other of which is stationary, corresponds to a DEPOSIT event. If a moving track intersects a stationary track, and then continues to move, but the stationary track ends at the intersection, this corresponds to a REMOVE event. The remove event can be generated as soon as the remover disoccludes the location of the stationary object which was removed, and the system can determine that the stationary object is no longer at that location.

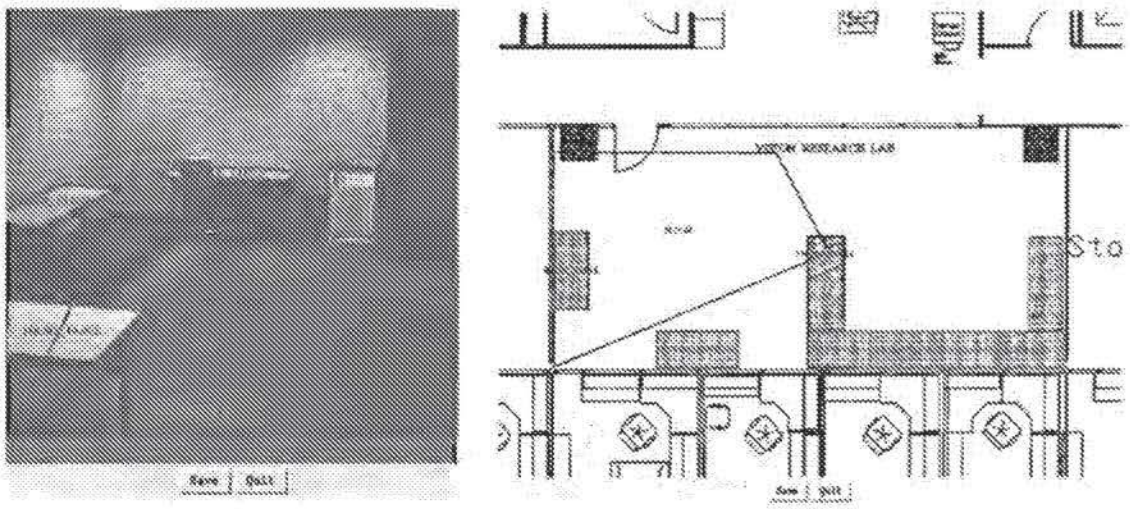

Figure 3: Establishing the image to map coordinate transformation

In a manner similar to the occlusion situation described above in section 3.2, the deposit event also gives rise to ambiguity as to which object is the depositor, and which is the depositee. For example, it may have been that the object which entered at frame F1 of Figure 2 *stopped* at frame F4 and deposited a *moving* object, and it is the deposited object which then proceeded to exit the scene at F6. Again, the AVS system relies on a moving vs. stationary distinction to resolve the ambiguity, and insists that the depositee remain stationary after a deposit event. The AVS system requires both the depositor and the depositee tracks to extend for *n* frames past the point at which the tracks separate (e.g., past frame F5 in Figure 2), and that the deposited object remain stationary; otherwise no deposit event is generated.

Also detected (but not illustrated in Figure 2), are REST events (when a moving object comes to a stop), and MOVE events (when a RESTing object begins to move again). Finally, one further event that is detected is the LIGHTSOUT event, which occurs whenever a large change occurs over the entire image. The motion graph need not be consulted to detect this event.

3.4 Image to World Mapping

In order to locate objects seen in the image with respect to a map, it is necessary to establish a mapping between image and map coordinates. This mapping is established in the AVS system by having a user draw quadrilaterals on the horizontal surfaces visible in an image, and the corresponding quadrilaterals on a map, as shown in Figure 3. A warp transformation from image to map coordinates is constructed using the quadrilateral coordinates.

Once the transformations are established, the system can estimate the location of an object (as in Flinchbaugh and Bannon [1994]) by assuming that all objects rest on a horizontal surface. When an object is detected in the scene, the midpoint of the lowest side of the bounding box is used as the image point to project into the map window using the quadrilateral warp transformation [Wolberg, 1990].

4 Applications

The AVS core algorithms described in section 3 have been used as the basis for several video surveillance applications. Section 4 describes three applications that we have implemented: situational awareness, best-view selection for activity logging, and environment learning.

4.1 Situational Awareness

The goal of the situational awareness application is to produce a real-time map-based display of the locations of people, objects and events in a monitored region, and to allow a user to specify alarm conditions interactively. Alarm conditions may be based on the locations of people and objects in the scene, the types of objects in the scene, the events in which the people and objects are in-

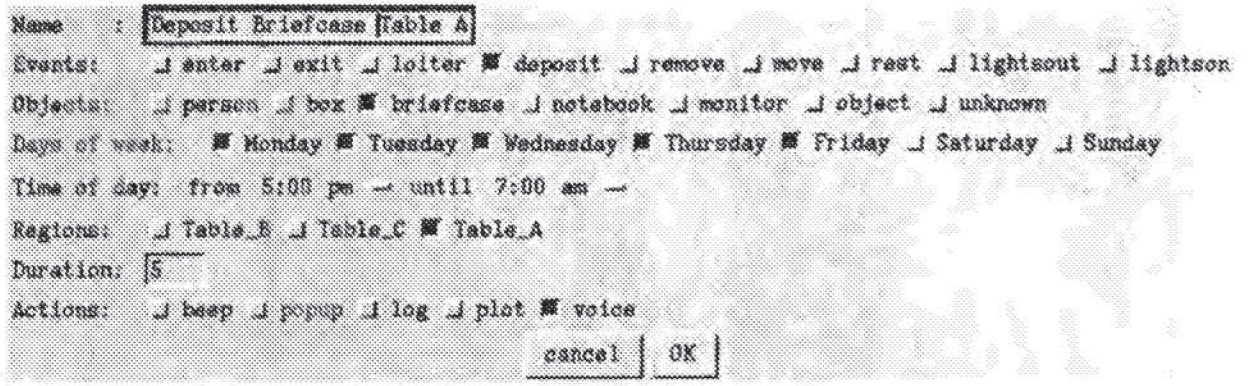

Figure 5: User interface for specifying a monitor in AVS

volved, and the times at which the events occur. Furthermore, the user can specify the action to take when an alarm is triggered, e.g., to generate an audio alarm or write a log file. For example, the user should be able to specify that an audio alarm should be triggered if a person deposits a briefcase on a given table between 5:00pm and 7:00 am on a weeknight.

The architecture of the AVS situational awareness system is depicted in Figure 4. The system consists of one or more smart cameras communicating with a Video Surveillance Shell (VSS). Each camera has associated with it an independent AVS core engine that performs the processing described in section 3. That is, the engine finds and tracks moving objects in the scene, maps their image locations to world coordinates, and recognizes events involving the objects. Each core engine emits a stream of location and event reports to the VSS, which filters the incoming event streams for user-specified alarm conditions and takes the appropriate actions.

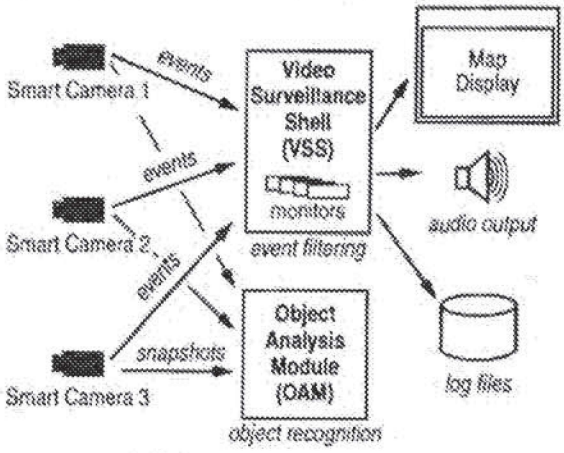

Figure 4: The situational awareness system

In order to determine the identities of objects (e.g., briefcase, notebook), the situational awareness system communicates with one or more object analysis modules (OAMs). The core engines capture snapshots of interesting objects in the scenes, and forward the snapshots to the OAM, along with the IDs of the tracks containing the objects. The OAM then processes the snapshot in order to determine the type of object. The OAM processing and the AVS core engine computations are asynchronous, so the core engine may have processed several more frames by time the OAM completes its analysis. Once the analysis is complete, the OAM sends the results (an object type label) and the track ID back to the core engine. The core engine uses the track ID to associate the label with the correct object in the current frame (assuming the object has remained in the scene and been successfully tracked).

The VSS provides a map display of the monitored area, with the locations of the objects in the scene reported as icons on the map. The VSS also allows the user to specify alarm regions and conditions. Alarm regions are specified by drawing them on the map using a mouse, and naming them as desired. The user can then specify the conditions and actions for alarms by creating one or more *monitors*. Figure 5 depicts the monitor creation dialog box. The user names the monitor and uses the mouse to select check boxes associated with the conditions that will trigger the monitor. The user selects the type of event, the type of object involved in the event, the day of week and time of day of the event, where the event occurs, and what to do when the alarm condition occurs. The monitor specified in Figure 5 specifies that a voice alarm

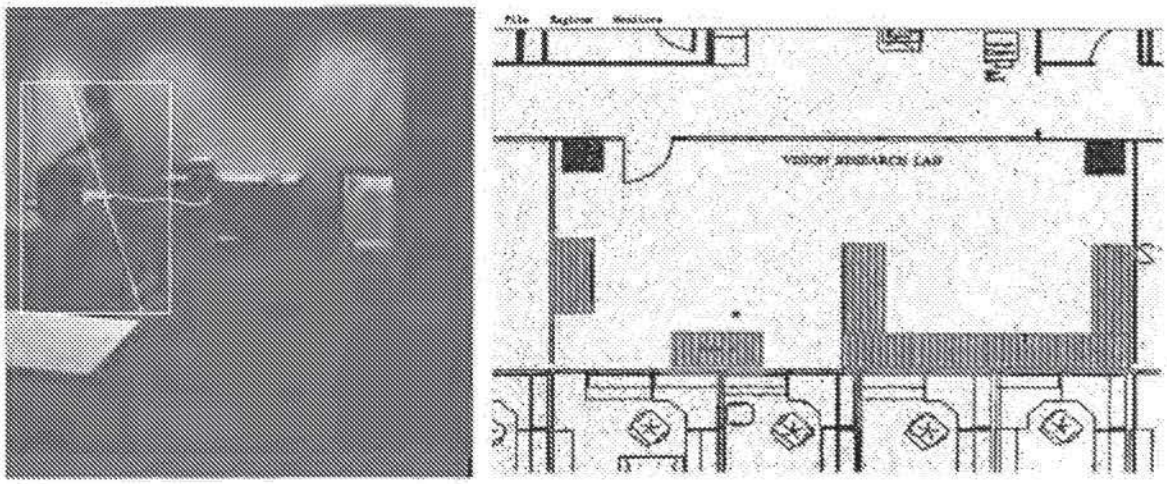

Figure 6: Tracking an object in the scene on the map

will be sounded when a briefcase is deposited on Table_A between 5:00pm and 7:00am on a weeknight. The voice alarms are customized to the event and object type, so that when this alarms is triggered, the system will announce "deposit box" via its audio output. Figure 6 shows a person about to trigger this alarm.

5 Best-View Selection for Activity Logging

In many video surveillance applications the goal of surveillance is not to detect events in real time and generate alarms, but rather to construct a log or audit trail of all of the activity that takes place in the camera's field of view. This log is examined by investigators after a security incident (e.g., a theft or terrorist attack), and is used to identify possible suspects or witnesses.

In order to gain experience with this type of application, we have used the tracking and event detection capabilities described in section 3 to construct a program that monitors and records the movements of humans in its field of view. For every person that it sees, it creates a log file that summarizes important information about the person, including a snapshot taken when the person was close to the camera and (if possible) facing it. The log files are made available to authorized users via the World-Wide Web.

5.1 Architecture

The application makes use of the AVS core algorithms to detect and track people. Upon detection of a track corresponding to a person in the input, the tracker associates a data record with the track. The data record contains a summary of information about the person, including a snapshot extracted from the current video image. As the person is tracked through the scene, the tracker examines each image of that person that it receives. If the new image is a better view of the person than the previously saved snapshot, the snapshot is replaced with the new view. When the person leaves the scene, the data record is saved to a file.

Each log entry file records the time when the person entered the scene and a list of coordinate pairs showing their position in each video frame. Each log entry file also contains the snapshot that was stored in the track record for the person when they exited the scene. Because of the way snapshots are maintained, the final snapshot is the best view of the person that the system had during tracking. Finally, the log entry file contains a pointer to the reference image that was in effect when the snapshot was taken. This information forms an extremely concise description of the person's movements and appearance while they were in the scene.

Selecting the best view: The system uses simple heuristics to decide when the current view of a per-

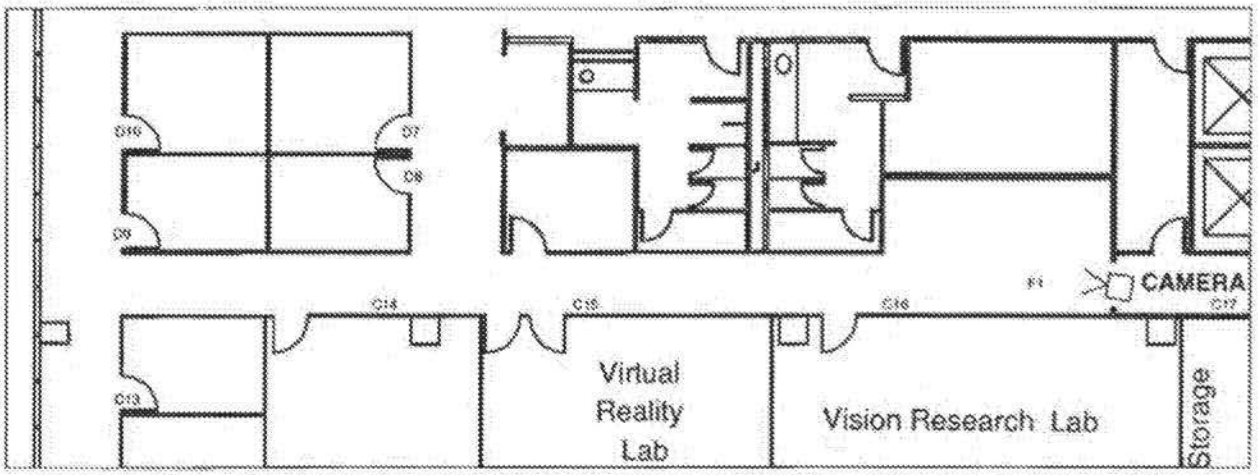

Figure 7: Floor plan of area used for hallway monitoring experiments. Camera is located at right and monitors the hallway and printer alcove.

son is better than the previously saved view. First, the new view is considered better if the subject is moving toward the camera in the current frame, and was moving away in the previously saved view. This causes the system to favor views in which the subject's face is visible. If this rule does not apply, the new view is considered better if the subject appears to be larger (subtends a larger visual angle). This causes the system to prefer views in which the subject is close to the camera. Other possible view selection heuristics are discussed in Kelly et al. [1995].

Handling background change: The test environment experiences significant lighting variation during the day due to window lighting, opening and closing doors etcetera. In addition, during the day people frequently deposit, remove, or reposition objects in the scene. This creates permanent regions of difference between the scene and the reference image. Without some mechanism for updating the reference image, the system would continue to track these difference regions as objects. Therefore, the tracker was instructed to discard the current tracks and grab a new reference image whenever it determined that all objects in the scene were stationary, and that no object had moved for several seconds.

User Interface

Log files are saved in a directory tree associated with the camera that produced the data. Along with the log files, the monitoring application creates HTML documents that allow a web browser to navigate the directory tree and access the log entries. Log entries are displayed by a Java applet that displays the best snapshot of the person in the context of the reference image, and overlays the person's path through the scene on the image. The applet runs as an independent thread that checks periodically to see if any new log entries have been created. Thus if the user is browsing the entries for the current day, new entries become available to the browser as soon as they occur.

5.2 Experiments

The system described above was tested in a hallway of our laboratory. Figure 7 shows the hallway floor plan. The camera is mounted in the hallway ceiling and looks west toward a window-lit corridor that runs around the perimeter of the building. The hallway experiences heavy traffic, because it contains a laser printer, a copier, and the office water cooler. The hallway passes under the camera and continues to the east out of the field of view.

The system was allowed to run for a total of 118 hours over a period of a week. Most laboratory personnel were unaware that a test was in progress, so the system was exposed to normal daily activity. During the test the system recorded a total of 965 log entries. Figure 8 shows the browser display for a typical log entry. In this sequence the subject entered the scene from the cross corridor at rear and came down the hallway on his way to the copier, out of view at lower right. His path is shown as a line on the floor, which appears red when viewed with a color browser.

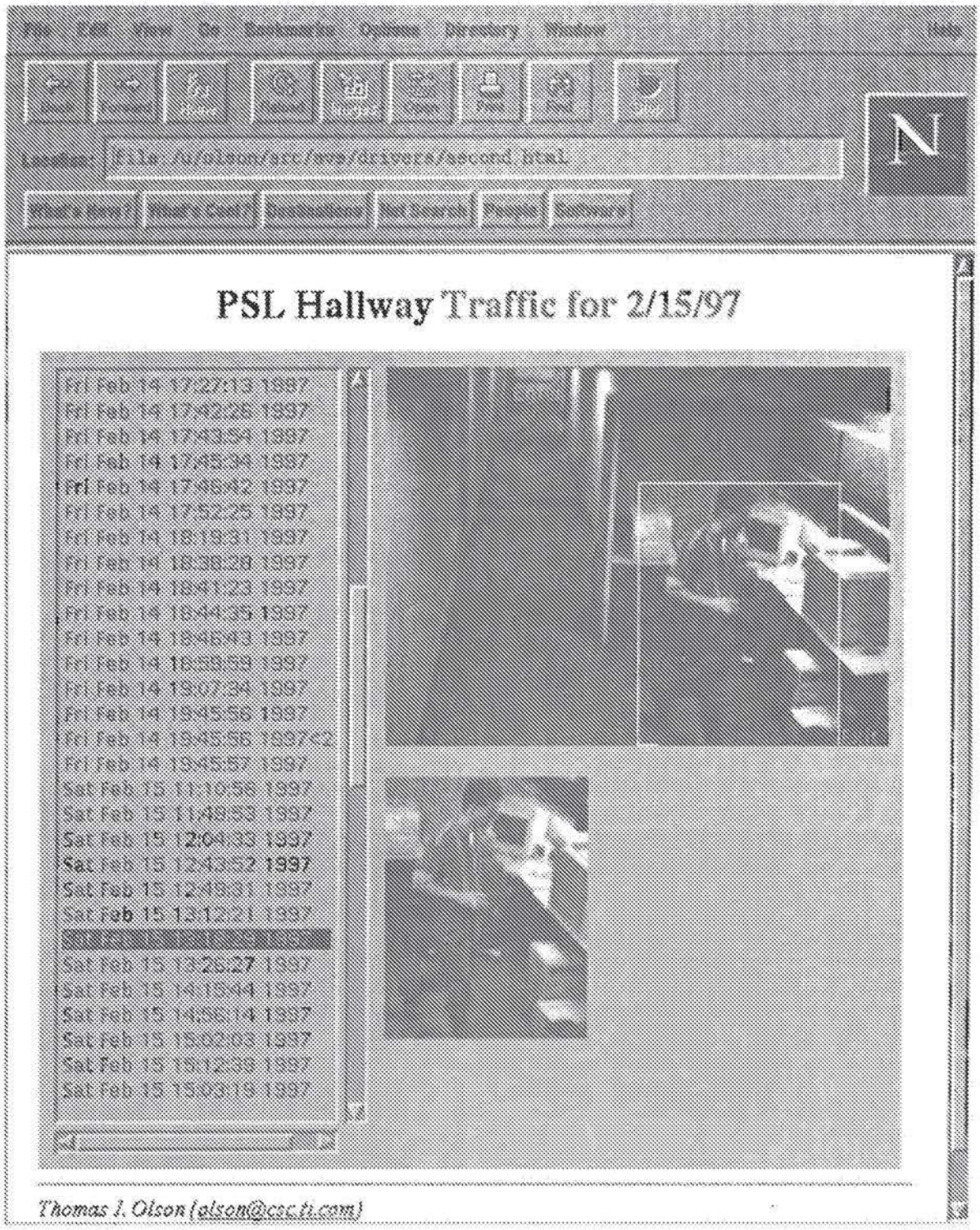

Figure 8: Log entry browser interface. The line drawn on the floor in the upper image shows the subject's path from entry to exit. The list entry selected at left is the time at which the image was taken.

Figure 9 demonstrates the effect of the system's preference for frontal views. In this sequence the subject entered at the bottom of the scene and walked away from the camera. He turned around and took a few steps back toward the camera, then turned away again and continued down the hallway, eventually exiting via the first door on the left. Although the subject's back was toward the camera

most of the time, the view preference heuristics selected a view taken while he was facing the camera.

Performance Evaluation

In order to assess the performance of the monitoring application, all of the log entries for the experiment period were examined and scored by one of the authors. Entries were classified as follows:

Face/Non-face: Entries containing a view of a subject's head were classified as FACES if the subject's face (specifically, subject's nose) was visible, otherwise they were classified as NONFACES.

False Alarm: Images which contained no human and appeared to be caused by noise were classified as FALSE ALARMS.

Bad Path: Entries in which the floor trace is clearly corrupt in some way were classified as BAD PATHs.

Bad Choice: In some cases it is clear from the floor trace that the system made a poor choice of which image of a person to save in the log entry. These entries were classified as BAD CHOICE.

False Negative: In some cases it is clear that the system failed to take a usable picture of a person who was in the scene. These were classified as FALSE NEGATIVES. About half of the false negatives occurred when the system selected a view in which the subject's head is not visible, typically because they were in the act of passing through a doorway. The others occurred when the system became confused by occlusion, and incorrectly grouped two people into a single log entry. Note that we do not have ground truth for the observation period, so there may have been other detection failures that were not detected. However, monitoring by the authors during the daytime revealed no failures of this type. We believe that the FALSE NEGATIVE count is a good estimate of the number of detection failures.

Table 1 shows the classification counts for the test period. Assuming that the false negative count is

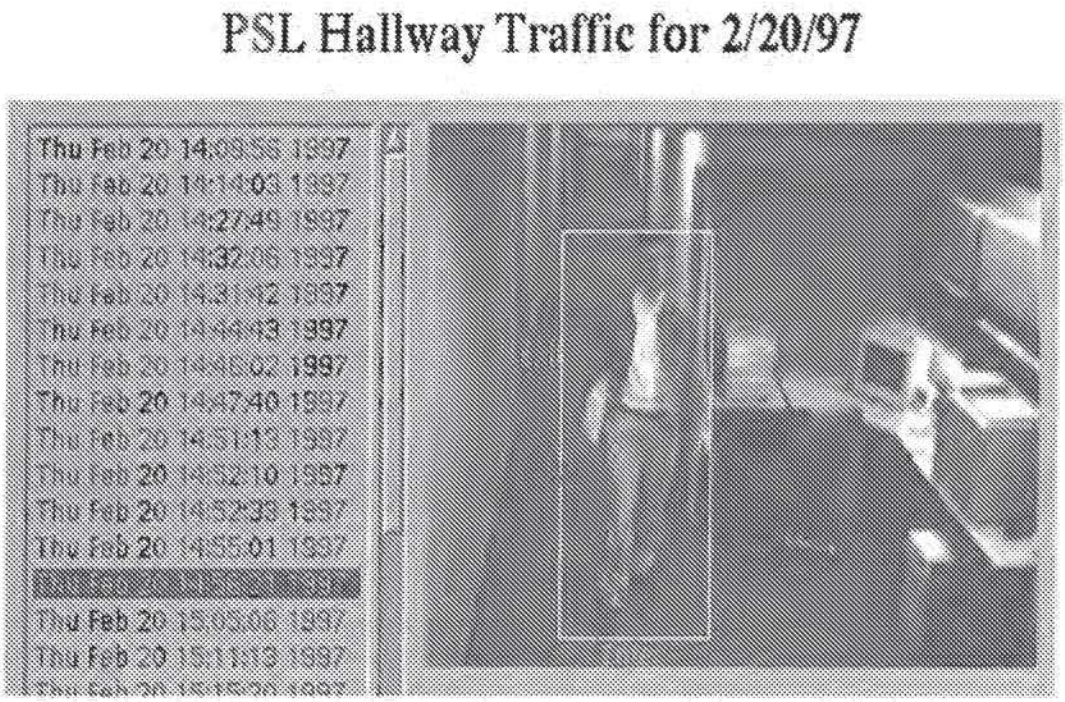

Figure 9: Log entry showing the effect of the view selection heuristic preference for frontal views. The subject was walking away from the camera for most of this sequence, but the system was able to capture a view while he was facing the camera.

Table 1: Long-term monitoring system performance

log entry type	Number of entries
FACE	493
NONFACE	380
FALSE ALARM	62
FALSE NEGATIVE	44
BAD PATH	112
BAD CHOICE	29
TOTAL ENTRIES	965

valid, the system achieved a detection rate of 95.2% with a false alarm rate of 6.4%. The recorded path of the subject was correct (or at least plausible) in 88.4% of entries, and the system made conspicuously bad choices of what image to save in only 3% of entries.

Of the valid images of humans, 56.6% showed the subject's face, vs. 43.4% that did not. Note that in most cases where the image does not show the face, the subject entered the scene from below the camera and walked away from it, so there was never an opportunity for a frontal view. Earlier experiments without the frontal view heuristic captured FACE and NONFACE images with roughly equal frequency, so the it is clear that the heuristic helps.

At the end of the experiment, the camera directory occupied 34.5 megabytes, or about seven megabytes per day of monitoring. Almost all of the storage consists of image files, so presumably compression with an image-specific algorithm would produce substantial savings. Use of an MPEG-like algorithm on the reference images would be extremely effective, since the reference images are all very nearly identical, and lossless compression would not be necessary.

6 Learning Environment Structure

The AVS tracking and event recognition software uses corresponding rectangles in image and world coordinates to compute an approximate image-to-world mapping. These rectangles are created by a human when the camera system is set up. In many situations it would be preferable to eliminate even this minimal calibration step, in order to reduce setup cost to a minimum.

We have developed a system that learns the image-to-world mapping by watching humans move around in the scene. Changes in the apparent size and position of humans in the image provide information about the existence and depth of world surfaces. Appearance and disappearance of humans provides information about occlusion boundaries and locations where humans can enter or exit the scene.

6.1 Method

The computation assumes weak perspective projection, i.e. that objects in the scene are first projected orthographically to a plane passing through a reference point on the object and parallel to the image plane, and then projected to the image plane using true perspective. It is also assumed that humans are usually in contact with a world surface that supports them, that the camera is in an upright position (has roll angle zero), and that the internal calibration parameters of the camera are known.

More precisely, assume front projection with the camera focal point at the origin and looking down the Z axis of a left-handed coordinate system. Suppose the camera observes a person in the world with head at world point $H = (X_H, Y_H, Z_H)$ and feet at world point F. Let F be the reference point for weak perspective projection. Then the apparent height of the person in the image is given by

$$y_H - y_F = \frac{f}{Z_F}(Y_H - Y_F) = \frac{f}{Z_F}|H - F|\cos\theta$$

where θ is the camera tilt angle relative to the local vertical direction. Solving for depth gives

$$Z_f = f\cos\theta\left(\frac{|H - F|}{y_h - y_f}\right)$$

The person's height $|H - F|$ has a known probability distribution, and the tilt angle term $\cos\theta$ can be

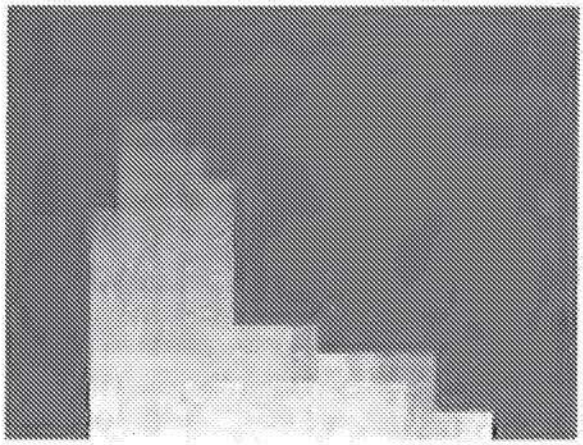

Figure 10: Apparent height data collected in the experiment. Cell intensity is the median of the image heights of observed humans when their feet were imaged in the cell. Dark grey regions contain no data.

estimated from the appearance of the person, or simply ignored for the shallow tilt angles typical of security camera installations. Given enough observations, the equation can be used to estimate the distance from the camera to points in the world where people commonly walk.

The idea of recovering structure from observed sizes of humans is conceptually related to shape-from-texture work in which the texture is made up of discrete elements that are uniform in size and shape [Aloimonos and Swain, 1988, Blostein and Ahuja, 1989]. In this case the texels (people) do not lie in the imaged surface, and their size in the world is known. This makes depth recovery substantially easier than it is in general shape-from-texture work.

6.2 Mapping the Environment

The equation derived above has been used in a program that learns the structure of its environment by watching humans move around in it. The program makes use of the AVS core algorithms to detect and track people. Over time, it builds up an image in which pixel value represents depth to the nearest world surface in the corresponding direction.

The camera image is partitioned into a grid of 16x16-pixel squares, each of which is associated with a histogram. Whenever the program detects a person in the scene, it locates the histogram associated with the place where they are standing, i.e., the one associated with the square containing the bottom center of the motion region for the person. The apparent height of the person is recorded in that histogram. Over time, the histogram for each location in the image builds up a sample distribution for the apparent (image) height of humans at that location. This can be used with the equation derived previously to estimate the depth at that point.

The program was allowed to operate for twenty-four hours during a typical working day. Input was provided by the hallway camera used in section 5. Figure 10 shows the raw output of the program. In the figure pixel intensity corresponds to the median observed height for the corresponding location. Dark grey pixels are those for which no observations were recorded. The program was instructed to discard observations in which the motion region for the person touched the upper or lower image border, since the apparent height is invalid in that condition. For this reason, there are no counts for the end of the hallway.

The height data of Figure 10 were converted to depths using the equation derived above. Vertical pixel pitch was taken from the camera technical manual, and the nominal lens focal length was used to approximate the true focal length. Histogram cells for which fewer than ten total observations were recorded were discarded.

Figure 11 shows the final depth map superimposed on the image. The range estimates cover image regions corresponding to the floor, and vary smoothly over most of the image. Anomalously large values occur in several locations at right center below the small printer and workstation. These errors occur because the office chair is frequently moved around in this region, and the system sometimes mistakes it for a person. Since it is significantly smaller than a real person, the system interprets it as evidence that the floor supporting it is further away than it actually is. A similar problem produces the anomalously high value of 8.9 meters at left center, at the base of the doorway. It frequently happens that as a person exits the hall via the doorway, their head goes out of sight while their body and feet are still visible. The system records the height of the visible portion of the person in the cell at the base of the doorway. Since this

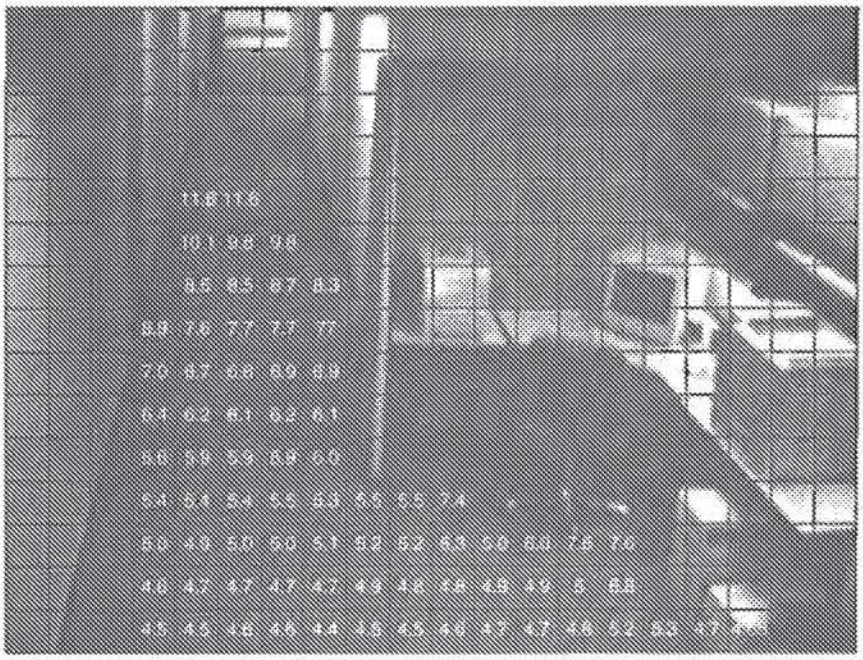

Figure 11: Depth map recovered from the height data of figure 10. Depths are in meters.

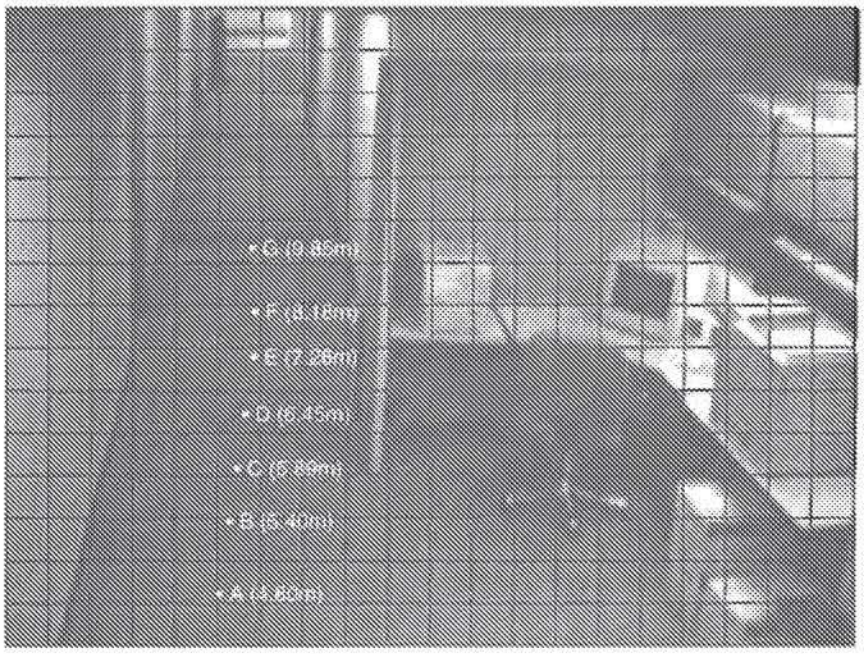

Figure 12: Ground truth range values for comparison with figure 11.

value is smaller than the true height of the person, that cell appears to be further away than it really is.

In order to assess the accuracy of the recovered depth map, we measured the distance from the camera to seven points on the floor. The seven points and their distances from the camera are shown superimposed on the image in figure 12. Table 2 shows the estimated and actual ranges to the test points, as well as the error in meters. The average absolute error for the seven test points is 26cm, which is less than 5% of the average distance.

Table 2: Estimated vs. Actual Range (meters) to ground truth points

point	estimate (meters)	actual (meters)	error (meters)
A	4.70	4.80	-0.10
B	5.00	5.40	-0.40
C	5.90	5.89	0.01
D	6.10	6.45	-0.35
E	6.80	7.26	-0.46
F	7.70	8.18	-0.48
G	9.80	9.85	-0.05

7 Conclusion

The goal of our research is to develop algorithms and systems that can be used to describe a video sequence in terms of moving objects and events. These algorithms will enable a generation of smart cameras that deliver information about scenes rather than raw images. We have created a set of core algorithms comprising the Autonomous Video Surveillance (AVS) system, including routines for moving object detection, tracking, and abstract event recognition. The AVS system has been used to create several surveillance applications, including a video surveillance shell, a program that creates concise logs of activity in the field of view, and a program that learns scene structure by watching humans moving around in the environment.

Our future work on AVS will address weaknesses in the current system, and will add new capabilities that support more complex applications. Work is planned in three main areas:

Robust Change Detection and Tracking: Experiments have shown that errors in the moving object detection computation are the most common cause of errors in our applications. This is particularly a problem in outdoor environments. We plan to develop new change detection algorithms based on dynamic background models that capture the way the background changes over time. We will also exploit contextual information to predict the expected size and appearance of moving objects in the scene.

Improved Event Recognition: We will extend our motion-graph-based event recognition algorithms to a broader range of events, and will develop methods of specifying and recognizing compound events and event sequences.

Applications: We will extend the existing video surveillance shell to make use of authentication sensors, and to distinguish between authorized and unauthorized individuals. We will continue to use AVS technology to develop applications that address military and other government video surveillance needs.

References

[Aloimonos and Swain, 1988] Y. Aloimonos and M. Swain. Shape from texture. In *Proc Ninth International Joint Conf. on Artificial Intelligence*, Los Angeles, pp. 926-931, 1988.

[Blostein and Ahuja, 1989] D. Blostein and N. Ahuja. Shape from texture: Integrating Texture-Element Extraction and Surface Estimation. *IEEE. Trans. Pattern Analysis and Machine Intelligence*, v. 11 no. 12, pp. 1233-1251, Dec. 1989.

[Bobick and Davis, 1996] A. Bobick and J. Davis. Real-time recognition of activity using temporal templates. In *Proc. Third IEEE Workshop on Applications of Computer Vision*, pp. 39-42, 1996.

[Burt et al., 1988] P. Burt, J. Bergen, R. Hingorani, R. Kolczynski, W. Lee, A. Leung, J. Lubin, and H. Shvayster. Object tracking with a moving camera. In *Proc. IEEE Workshop on Visual Motion*, pp. 2-12, Irvine, March 1988.

[Camus et al, 1993] T. Camus, J. Monsarrat and T. Dean. Planning and Selective Perception for Target Retrieval. In *Proc. DARPA Image Understanding Workshop*, pp. 593-597, April 1993.

[Courtney, 1997] J. D. Courtney. Automatic video indexing via object motion analysis. *Pattern Recognition*, 30(4), April 1997.

[Flinchbaugh, 1997] B. Flinchbaugh. Robust Video Motion Detection and Event Recognition. in *1997 Proc. DARPA Image Understanding Workshop* (these proceedings).

[Flinchbaugh and Bannon, 1994] B. Flinchbaugh and T. Bannon. Autonomous scene monitoring system. In *Proc. 10th Joint Government-Industry Security Technology Symposium, American Defense Preparedness Association*, June 1994.

[Gavrila and Davis, 1996] D. Gavrila and L. Davis. Tracking of humans in action: A 3-D model-based approach. In *1996 Proc. DARPA Image Understanding Workshop*, pp. 737-746, 1996.

[Intille and Bobick, 1995] S. Intille and A. Bobick. Closed-world tracking. In *Proc. Fifth International Conference on Computer Vision*, pp. 672-678, 1995.

[Jain et al., 1979] R. Jain, W. Martin, and J. Aggarwal. Segmentation through the detection of changes due to motion. *Computer Graphics and Image Processing*, **11**, pp. 13-34, 1979.

[Kahn et al., 1996] R. Kahn, M. Swain, P. Prokopowicz, and J. Firby. Gesture recognition using the Perseus architecture. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 734-741, 1996.

[Kakadiaris and Metaxas, 1996] I. Kakadiaris and D. Metaxas. Model-based estimation of 3D human motion with occlusion based on active multi-viewpoint selection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 81-87, San Francisco, 1996.

[Karmann and von Brandt, 1990] K. P. Karmann and A. von Brandt. Moving object recognition using an adaptive background memory. In *Time Varying Image Processing and Moving Object Recognition*, V. Cappellini (ed.). **2**, pp. 1060-1066, Elsevier, Amsterdam, 1990.

[Kelly et al., 1995] P. Kelly, A. Katkere, D. Kuramura, S. Moezzi, S. Chatterjee and R. Jain. An Architecture for Multiple Perspective Interactive Video. Visual Computing Laboratory Technical Report VCL-95-103, UCSD, March 1995.

[Koller et al., 1994] D. Koller, J. Weber, J. Malik. Robust multiple car tracking with occlusion reasoning. In *Proc. Third European Conference on Computer Vision*, pp. 186-196, LNCS 800, Springer-Verlag, May 2-6, 1994.

[Makarov, 1996a] A. Makarov. Comparison of background extraction based intrusion detection algorithms. In *Proc. Int'l Conference on Image Processing*, Lausanne, Sept. 1996.

[Makarov et al., 1996b] A. Makarov, J. M. Vesin, and F. Reymond. Intrusion detection robust to slow and abrupt lighting changes. In *Proc. SPIE: Real-Time Imaging*, **2661**, pp. 44-54, 1996.

[Nelson, 1991] R. C. Nelson. Qualitative detection of motion by a moving observer. *International Journal of Computer Vision*, **7**(1), pp. 33-46, November 1991.

[Pentland, 1996] A. Pentland. Machine understanding of human action. In *Proc. DARPA Image Understanding Workshop*, pp. 757-764, 1996.

[Rimey and Brown, 1993] R. Rimey and C. Brown. Studying Control of Selective Perception with T-world and TEA. In *Proceedings of the DARPA Image Understanding Workshop*, Washington D.C., pp. 575-580, April 1993.

[Ringler and Hoover, 1995] C. Ringler and C. Hoover. Evaluation of commercially available VMDs. Sandia National Laboratories Technical Report SAND94-2875, June 1995.

[Rowley et al., 1996] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *Proc. DARPA Image Understanding Workshop*, pp. 725-735, 1996.

[Starner and Pentland, 1995] T. Starner and A. Pentland. Visual recognition of American Sign Language using hidden Markov models. In *Int'l Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland, 1995.

[Sung and Poggio, 1994] K. K. Sung and T. Poggio. Example-based learning for view-based human face detection. In *Proc. DARPA Image Understanding Workshop*, pp. 843-850, 1994.

[Swain and Stricker, 1991] M. J. Swain and M. Stricker, editors. Promising directions in active vision. University of Chicago Technical Report CS 91-27, November 1991.

[Wolberg, 1990] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[Yalamanchili et al., 1982] S. Yalamanchili, W. Martin, and J. Aggarwal. Extraction of moving object descriptions via differencing. *Computer Graphics and Image Processing*, **18**, pp. 188-201, 1982.