Algorithms for Cooperative Multisensor Surveillance

ROBERT T. COLLINS, ALAN J. LIPTON, HIRONOBU FUJIYOSHI, AND TAKEO KANADE, FELLOW, IEEE

Invited Paper

The Video Surveillance and Monitoring (VSAM) team at Carnegie Mellon University (CMU) has developed an end-to-end, multicamera surveillance system that allows a single human operator to monitor activities in a cluttered environment using a distributed network of active video sensors. Video understanding algorithms have been developed to automatically detect people and vehicles, seamlessly track them using a network of cooperating active sensors, determine their three-dimensional locations with respect to a geospatial site model, and present this information to a human operator who controls the system through a graphical user interface. The goal is to automatically collect and disseminate real-time information to improve the situational awareness of security providers and decision makers. The feasibility of real-time video surveillance has been demonstrated within a multicamera testbed system developed on the campus of CMU. This paper presents an overview of the issues and algorithms involved in creating this semiautonomous, multicamera surveillance system.

Keywords*—*Active vision, cooperative systems, geolocation, multisensor systems, site security monitoring, user interfaces, video surveillance.

I. INTRODUCTION

There are immediate needs for automated surveillance systems in commercial, law enforcement, and military applications. Mounting video cameras is cheap, but finding available human resources to observe the output is expensive. Although surveillance cameras are already prevalent in banks, stores, and parking lots, video data currently is used only "after the fact" as a forensic tool, thus losing its primary benefit as an active, real-time medium. What is needed is continuous 24-hour monitoring of surveillance video to alert security officers to a burglary in progress, or a suspicious individual loitering in the parking lot, while there is still time to prevent the crime.

There is growing interest in developing automated video understanding algorithms to provide this constant vigilance. Automated video surveillance addresses real-time observation of people and vehicles within a busy environment, leading to a description of their actions and interactions (e.g., [1]–[12]). Large research projects devoted to video surveillance research have been conducted in the United States (e.g., DARPA's Video Surveillance and Monitoring (VSAM) project [13]), Europe (the ESPRIT PASSWORDS [14], AVS-PV [15] and VIEWS [16], [17] projects) and Japan (the Cooperative Distributed Vision project [18]). Automated surveillance has also been the topic of recent international workshops [19]–[23] and special sections in journals [24], [25]. In addition to the obvious security and traffic monitoring applications, other diverse uses are possible, including compiling consumer demographics in shopping malls, logging routine maintenance tasks at nuclear facilities, monitoring livestock, and segmenting moving objects from commercials to provide hooks for user interaction.

A. Multisensor Surveillance

In realistic surveillance scenarios, it is impossible for a single sensor to see all areas at once, or to visually track a moving object for a long period of time. Objects become occluded by trees and buildings and sensors themselves have limited fields of view. A promising solution to this problem is to use a network of video sensors to cooperatively monitor all objects within an extended area and seamlessly track individual objects that cannot be viewed continuously by a single sensor alone. Some of the technical challenges within this approach are to: 1) actively control sensors to cooperatively track multiple moving objects; 2) fuse information from multiple sensors into scene-level object representations; 3) mon-

Manuscript received October 3, 2000; revised February 25, 2001. This work was supported by the Defense Advanced Research Projects Agency (DARPA) Image Understanding under Contract DAAB07–97-C-J031 and by the Office of Naval Research (ONR) under Grant N00014–99-1–0646.

R. T. Collins and T. Kanade are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

A. J. Lipton is with Diamondback Vision, Inc., Reston, VA 20191 USA.

H. Fujiyoshi is with Chubu University, Aichi 487-8501, Japan.

Publisher Item Identifier S 0018-9219(01)08432-8.

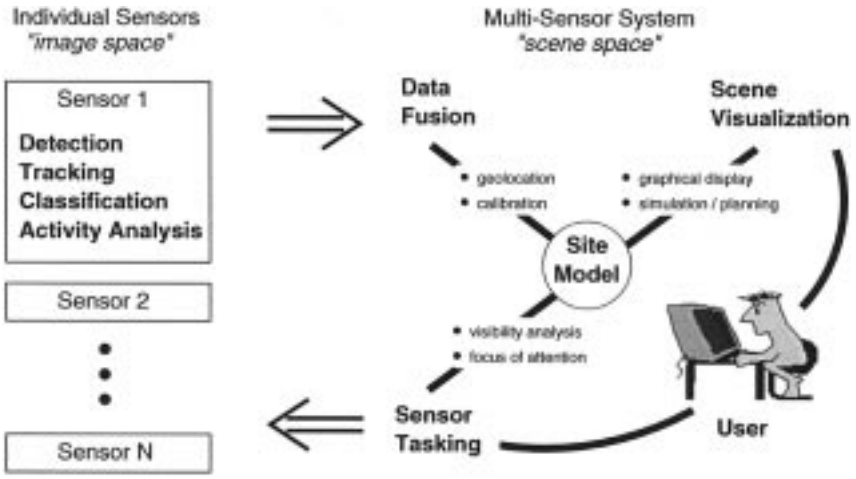


Fig. 1. Logical layout of tasks in a multisensor surveillance system.

itor the scene for events and activities that should "trigger" further processing or operator involvement; and 4) provide human users with a high-level interface for dynamic scene visualization and system tasking.

Within the context of the DARPA VSAM project, we have developed an end-to-end, multicamera surveillance system that allows a single human operator to monitor activities in a cluttered environment using a distributed network of active video sensors. Central to our design philosophy is the notion of "smart" sensors that are independently capable of performing real-time, autonomous detection of objects and events (see Fig. 1). Sensors are modular units that can be added or removed without affecting the other sensors in the network. Each sensor performs real-time video processing to digest incoming video streams into symbolic descriptions of objects and events. The current suite of video understanding algorithms running on each sensor includes moving object detection, object tracking (including active tracking by sensors having active pan/tilt/zoom control), classification of detected moving blobs into semantic categories such as human and vehicle, and identifying simple human motions such as walking and running (see Section II).

Surveillance research typically employs a single sensor to locate and track objects in the field of view, with detections indicated to the observer through graphical annotations (e.g., bounding boxes) on the video stream. However, when multiple, active sensors are used in a cooperative mode, more sophisticated surveillance capabilities and user interaction are both possible and necessary. One of the goals of the VSAM project is to alter user interaction with a surveillance system from image-space to scene-space and from sensor-specific commands such as "pan sensor A to position B" toward higher level task commands such as "alert me when a delivery vehicle enters the East parking lot." To achieve interactivity at this level requires system-level algorithms that fuse sensor data, task sensors to perform autonomous cooperative behaviors, and display results to the operator in a comprehensible form (Fig. 1).

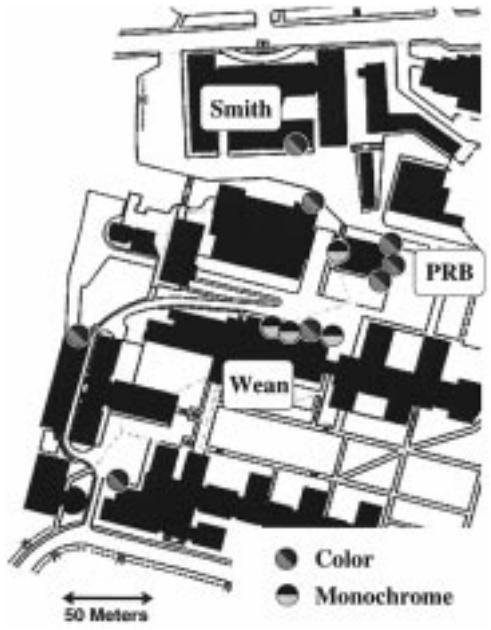
Data Fusion: Data from disparate sensors is integrated within a central three-dimensional (3-D) scene coordinate system to provide a complete representation of the union of all objects seen from all cameras. Every object observation from each sensor is mapped from the camera-centric image-space of the sensor into 3-D geodetic coordinates (latitude, longitude and elevation) through a process called geolocation. Geolocated object observations are compared to current 3-D object hypotheses maintained by the system using viewpoint-independent features and objects that match are conjoined to form an updated hypothesis (see Section III).

Sensor Tasking: An outdoor surveillance system must optimize the use of its limited sensor assets. Sensors must be allocated to perform all user-specified tasks and, if enough sensors are present, to gather redundant observations. An arbitration function determines the cost of assigning each sensor to each of the tasks, based on task priority, the load on each sensor, and visibility of the objects from each sensor. The system performs a greedy optimization of the cost to determine which pairing of sensors and tasks maximizes overall system performance requirements. Through this mechanism, objects can be tracked long distances by *handing off* between cameras situated along the object's trajectory (see Section IV).

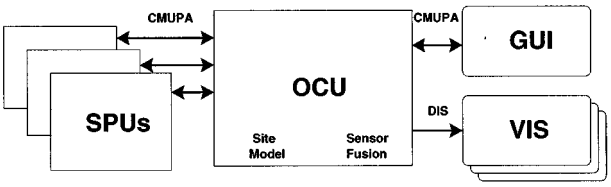
Scene Visualization: A single human operator cannot effectively monitor a large area by looking at dozens of monitors showing raw video output. That amount of sensory overload virtually guarantees that information will be ignored and requires a prohibitive amount of transmission bandwidth. Our approach is to provide an interactive, graphical user interface (GUI) showing a synthetic view of the environment, upon which the system displays dynamic agents representing people and vehicles. This approach has the benefit that visualization of scene events is no longer tied to the original resolution and viewpoint of a single video sensor and the operator can therefore infer proper spatial relationships between multiple objects and scene features (see Section V).

B. Surveillance Testbed

We have built a testbed system to demonstrate how multiple sensors using automated video understanding

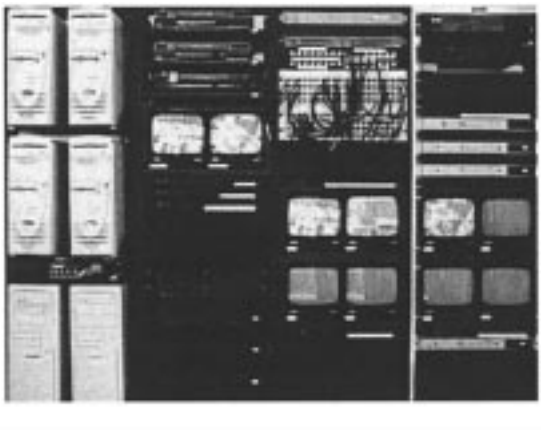


(a)



(b)

(c)



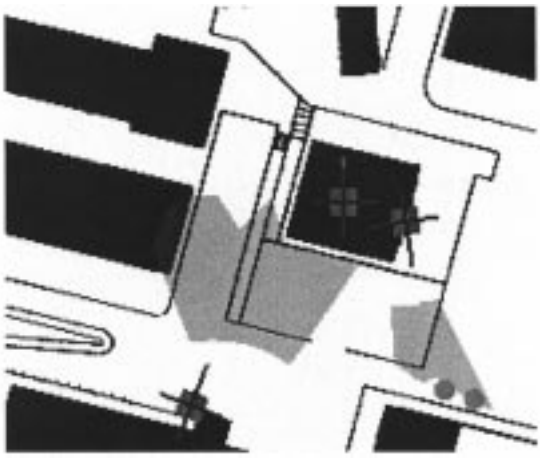
(d)

Fig. 2. (a) Placement of cameras in the current VSAM testbed system. (b) Schematic overview of the testbed system architecture. (c) Three sensors used for cooperative surveillance. (d) Central operator control station for integrating information from all sensors.

technology can be combined into a coherent surveillance system. The testbed consists of multiple cameras distributed over an area of roughly 0.4 km^2 on the campus of CMU [Fig. 2(a)]. The system architecture consists of a central operator control unit (OCU) which receives video and Ethernet data from multiple remote sensor processing units (SPUs) [see Fig. 2(b)]. The OCU uses a 3-D geometric site model to integrate symbolic object trajectory information accumulated by each of the SPUs and presents the results to the user on a map-based GUI. The data is also accessible through a set of distributed visualization nodes (VIS). Each component of the testbed system architecture is described briefly below.

Each SPU consists of a camera [Fig. 2(c)] paired with a processor to form a smart sensor that acts as an intelligent filter between a video signal and the VSAM network. The SPU analyzes raw video imagery to extract objects and events and transmits that detected information symbolically

(e)



(f)

Fig. 2. *(Continued.)* (e) Operator console where a user interacts with the system through a graphical user interface. (f) Screen dump of a map-based visualization tool that displays sensor resources and all detected objects and events.

to the OCU. Performing as much video processing as possible on the SPU greatly reduces the bandwidth requirements of the network. This arrangement also allows for many different sensor modalities to be seamlessly integrated into the system. For example, we have integrated color sensors with active pan/tilt/zoom control, fixed field of view monochrome sensors, a Cyclovision omni-directional sensor [26], thermal sensors, a van-mounted relocatable sensor system, an indoor video event detection system developed by Texas Instruments [27] and an airborne sensor platform [28], all using the same communication protocol. We have even prototyped man-portable SPUs that can be placed, calibrated and connected to the system in only a few minutes.

The OCU [Fig. 2(d)] integrates symbolic object information from the SPUs with a 3-D site model to determine object locations. The OCU supports one GUI [Fig. 2(e)] through which all user-related command and control information is passed. The GUI contains a map of the site, overlaid with all object locations, sensor platform locations and sensor fields of view. In addition, a low-bandwidth, compressed video stream from one of the sensors can be selected for real-time display. The OCU schedules sensor tasks to perform cooperative multicamera surveillance. The operator can task individual sensor units through the GUI, as well as instructing the entire testbed sensor suite to perform surveillance operations such as generating a quick summary of all object activities in the area.

VIS nodes [Fig. 2(f)] are designed to distribute surveillance results to remote users by providing graphical representations of detected activities overlaid on maps or 3-D synthetic views. We have developed a Java-based visualization client that can be played on any laptop connected to the VSAM system network. This two-dimensional (2-D) map display maintains much of the character of the operator GUI, but without the ability to control the system. We have also interfaced to ModSAF and ModStealth, which are 2-D and 3-D scene viewers developed within the context of Synthetic Training Environments [29], [30]. See Section V.

Prior knowledge of the terrain and important scene features is represented within a 3-D site model. Some of the surveillance tasks supported by scene-specific knowledge provided by the site model are: 1) computation of object location by intersecting viewing rays with the terrain [31]; 2) landmark-based calibration of camera exterior orientation [32]; 3) visibility analysis to predict what portions of the scene are visible from which cameras, thereby improving tracking [33] and allowing more effective sensor tasking; 4) geometric focus of attention, for example to task a sensor to monitor the door of a building, or specify that vehicles should appear on roads; 5) visualization of the scene to enable quick comprehension of geometric relationships between sensors, objects and scene features and 6) simulation for planning best sensor placement and for debugging algorithms.

II. Video Understanding Technologies

A multicamera surveillance system is built upon the basic capabilities provided by each sensor. At a minimum, each sensor must be able to detect moving objects from raw video at nearly frame-rate. This section provides an overview of the video understanding algorithms implemented within the VSAM testbed to detect moving objects, track them through a video sequence, classify them into semantic categories (e.g., human and vehicle) and analyze human motions such as walking and running. These descriptions are very brief in order to devote more space to multisensor aspects of the system.

A. Moving Object Detection

Detection of moving objects in video streams is known to be a significant and difficult research problem [34]. Conventional approaches to moving object detection include temporal differencing [35], [36]; background subtraction [6], [34], [37], [38] and optical flow [39]–[41]. One of the most successful approaches to date is adaptive background subtraction [37]. The basic idea is to maintain a running

statistical average of the intensity at each pixel. When the value of a pixel in a new image differs significantly from average, the pixel is flagged as potentially containing a moving object. One problem of this approach, along with the other conventional approaches to motion detection, is that objects that cease moving within the image simply disappear from the representation. A robust detection system should continue to "see" objects that have stopped and disambiguate between overlapping objects in the image. For example, a car that comes into the scene and parks should not be considered as part of the scene background, however its stationary pixels should play the role of background for detecting motion of a person getting out of the car.

We have developed a novel approach to object detection based on layered adaptive background subtraction. Layered detection is based on two processes: pixel analysis and region analysis. Pixel analysis determines whether a pixel is stationary or transient by observing its intensity value over time. The technique is derived from the observation that legitimately moving objects in a scene cause much faster intensity transitions than changes due to lighting or weather. Fig. 3(a) graphically depicts the process. To capture the nature of changes in pixel intensity profiles, two factors are important: the existence of a significant step change in intensity and the intensity value to which the profile stabilizes after passing through a period of instability. An object moving through the pixel displays a profile that exhibits a step change in intensity, followed by a period of instability, then another step back to the original background intensity. An object moving to the pixel and stopping displays a profile that exhibits a step change in intensity, followed by a period of instability, then a step to a new intensity as the object stops. Lighting and weather effects tend to cause smooth changes with no large steps. Therefore, by observing the intensity transitions at each pixel, different intensity layers connected by transient periods can be postulated.

Region analysis collects groups of labeled pixels into moving regions and stopped regions and assigns them to spatio-temporal layers [Fig. 3(b)]. Regions that consist of stationary pixels are added as a layer over the background, or over a previously determined layer. Regions consisting of moving pixels are represented as transient objects that occlude all layers. A layer management process that operates much like the window manager on a modern workstation is responsible for creating and deleting new regions, updating intensity information and keeping track of depth ordering between overlapping regions. Pixel values in stationary layered regions and the scene background are updated by an Infinite Impulse Response (running average) filter to accommodate slow lighting changes and noise in the imagery, as well as to compute statistically significant step-change thresholds.

An additional mechanism is built into this algorithm to detect sharp changes in the overall scene, caused by motion of the camera (which is mounted on a pan/tilt head) or occasional sharp lighting changes (e.g., the sun comes out from behind cloud cover). If a majority of the image pixels are found to be changing, the detection algorithm temporarily shuts down until the view stabilizes, as determined by a simple two-frame differencing algorithm. At this point, all pixel statistics are reinitialized and the detection algorithm resumes.

This detection algorithm has been evaluated on 4 h of video tape for which ground-truth labeling of moving objects (people and vehicles) was manually determined. 2 h of data were taken on a sunny day and 2 h on a cloudy day. Probability of detection was determined as the percentage of human-detected moving objects that were also detected by the system. The detection rate was 89.6% for sunny day data and 94.5% for cloudy day data. The main reason for failure to detect was low image contrast between the moving object and the background. Sunny day detection rates are lower because of the additional loss of image contrast in areas of deep shadow. False positive detection rates were not recorded.

B. Object Tracking

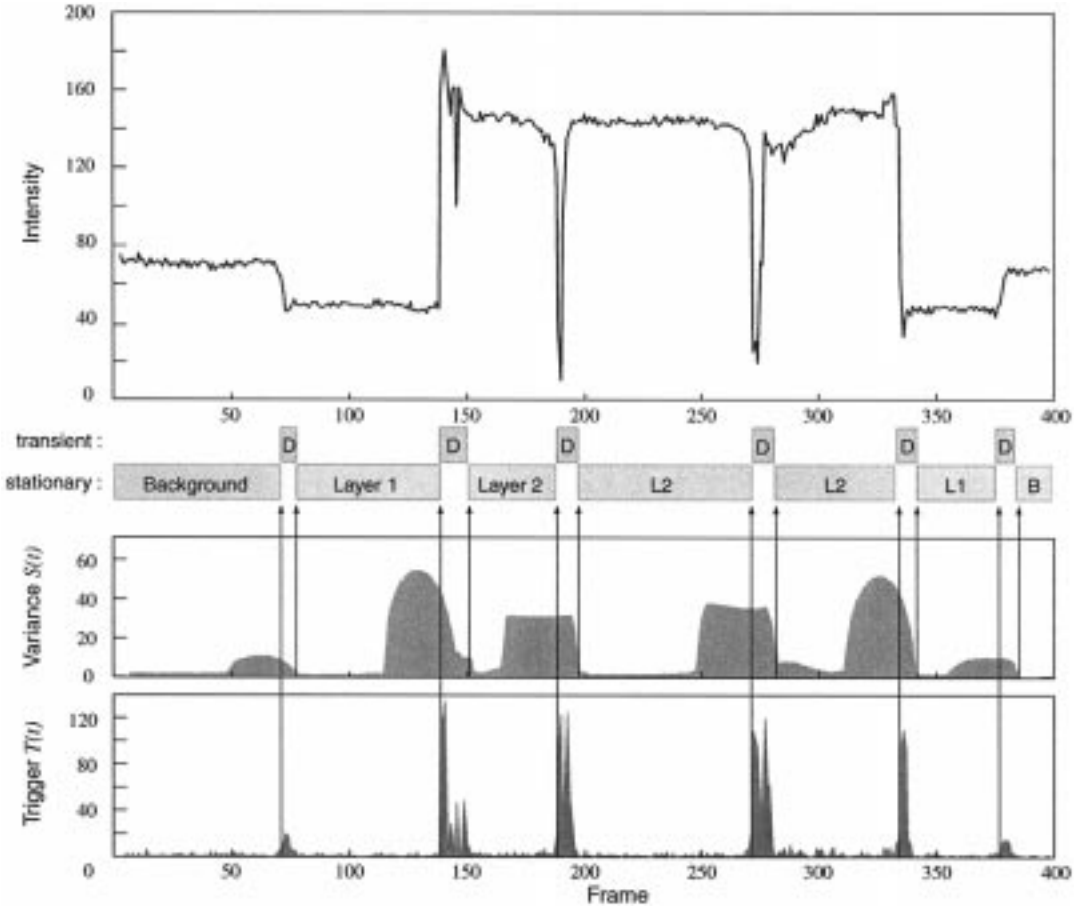
To begin building a temporal model of activity, individual object blobs generated by motion detection are tracked over time by matching them between frames of the video sequence. Among the many approaches to tracking are model-based matching [9], [42], [43], image contour matching [44] and image region matching [45], [46]. Multiple potential matches typically arise, which can be disambiguated using statistical data association techniques [47], [48] or by imposing smooth trajectory motion models using Kalman filters [49].

Our approach lies squarely in the image region matching camp. Given a moving object region in a current frame, we determine the best match in the next frame by performing image correlation matching, computed as the normalized cross correlation of the object's intensity template over candidate regions in the new image [50]. Due to real-time processing constraints in the VSAM testbed system, this basic correlation matching algorithm is only computed for "moving" pixels, regions are culled that are inconsistent with current estimates of object position and velocity and imagery is dynamically subsampled to ensure a constant computation time per match.

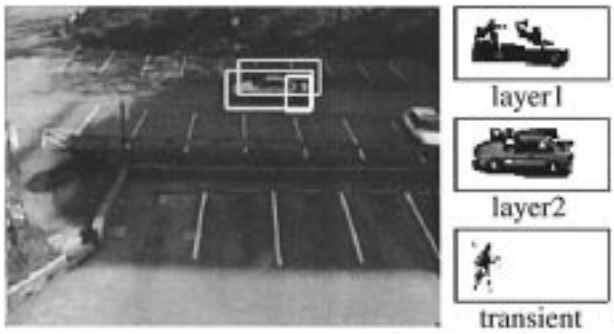
In the spirit of [44] the tracker maintains multiple match hypotheses with varying degrees of matching confidence and can merge and split hypotheses as appropriate to describe objects that temporarily occlude each other as they pass. Any object that has not been matched maintains its position/velocity estimates and current image template, but its confidence is reduced. If the confidence of any object drops below a given threshold, it is considered lost and is dropped from the list. High confidence objects (ones that have been tracked for a reasonable period of time) will persist for several frames, so if an object is momentarily occluded but then reappears, the tracker will reacquire it. More details can be found in [51], [52]. Some sample trajectories resulting from this approach are shown in Fig. 4.

C. Object Type Classification

Bottom-up motion detection and tracking algorithms (which do not try to fit *a priori* models to image data)



(a)



(b)

Fig. 3. (a) Example analysis of intensity changes over time at a single pixel as a car enters the scene and stops, a second car enters and stops in front of the first, a person gets out and walks to the first car, the person returns to the second car, the second car drives away, and finally the first car drives away. Each of these steps is visible in the pixel's intensity profile. (b) Detection results for one timestep during the described events. The algorithm has correctly detected and represented that there are three overlapping objects, namely, the first stopped car, the second stopped car, and the person walking in front of them.

view objects in the scene as moving blobs of pixels. Object classification routines begin to add semantics to these observations by providing class labels for each blob [4], [7], [53], [54]. We have developed two algorithms for view-dependent visual object classification. The first is a neural network classifier, trained for each sensor view [55] (see also [56], [57] for additional neural network approaches to object classification). The neural network is a standard three-layer network, trained using the backpropagation algorithm. Input features to the network are measured directly from the image blob and camera settings: blob dispersedness (perimeter^2/area); blob area; blob aspect ratio and camera zoom value. There are four output classes: single human; human group; vehicle and clutter. This neural network classification approach is fairly effective for single image frames; however, one of the advantages of video is its temporal component. To exploit this, classification is performed on each blob as it is tracked through the sequence

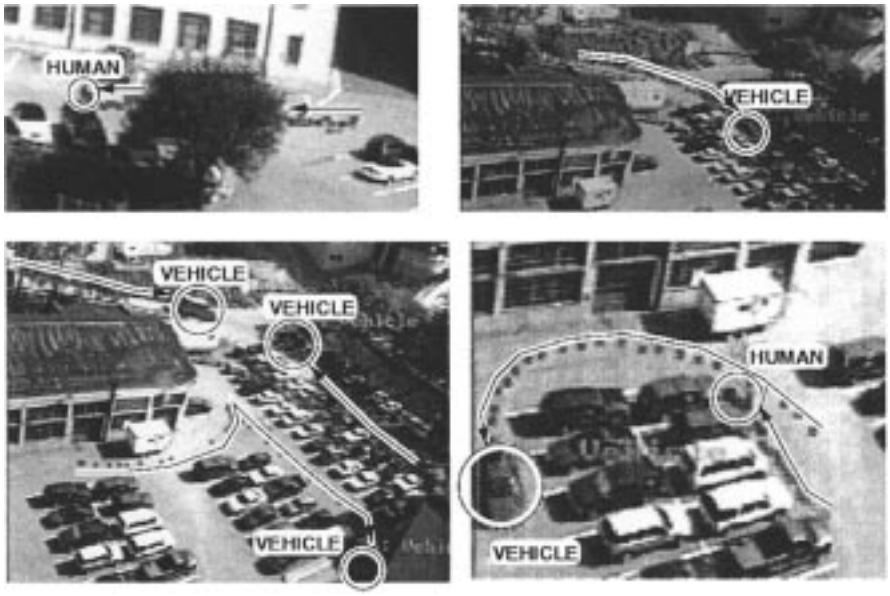


Fig. 4. Sample object trajectories and class labels.

of frames. The classification results for each frame are kept in a histogram and at each time step, the most likely class label for the blob is chosen based on all classifications that have been made for it. Sample results are shown in Fig. 4.

A second method of view-dependent object classification uses linear discriminant analysis to provide a finer distinction between vehicle types (e.g., van, truck, sedan). This method has also been successfully trained to recognize specific types of vehicles, such as UPS trucks and campus police cars. The method has two submodules: one for classifying object shape and the other for determining color. The latter is needed because the color of an object is difficult to determine under varying outdoor lighting. Each submodule computes an independent discriminant classification space using linear discriminant analysis (LDA) and calculates the most likely class in that space using a weighted k-class nearest-neighbor (k-NN) method [58]. In LDA, feature vectors computed on training examples of different object classes are considered to be labeled points in a high-dimensional feature space. Training examples are mapped into shape space as an 11-dimensional feature vector computed from the motion blob: area; center of gravity; width; height and first, second, and third moments taken along the row and column pixel axes. Color space is three dimensional, with features $I1 = 10^*(\mathrm{R}+\mathrm{G}+\mathrm{B})/3$; $I2 = 100^*(\mathrm{R}-\mathrm{B})/2$ and $I3 = 100^*(2^*\mathrm{G}-\mathrm{R}-\mathrm{B})/4$ computed from RGB values of pixels within the motion blob. Given training points in these feature spaces, LDA computes a set of discriminant functions, formed as linear combinations of feature values, that best separate the clusters of points that correspond to different object labels and color labels. See [59] for more details. Some sample results are shown in Fig. 5.

Table 1 shows cross validation between targets (columns) and classified results (rows) gathered from 4 h of hand-labeled video data. The recognition rate was roughly 90% across both sunny and cloudy weather conditions. (Note: a "Mule" is a golf-cart-like vehicle used by campus maintenance workers). Currently, the system does not work well when it is raining or snowing, because the raindrops and snowflakes interfere with the measured RGB values in the images. For the same reason, the system does not work well in early mornings and late evenings, due to the nonrepresentativeness of the lighting conditions. The algorithm is also foiled by backlighting and specular reflection from vehicle bodies and windows.

D. Human Motion Analysis

Classifying moving objects enables a surveillance system to subsequently invoke object-specific motion analysis methods to generate more detailed descriptions of object behavior. There has been considerable interest in the area of human motion tracking in recent years [2], [6], [38], [43], [60]–[67]. More references can be found in [68], [69]. Many human motion tracking algorithms assume that the size of the person in the image is large enough to track individual limbs. On the other hand, many surveillance applications involve more distant observations and a subsequent smaller number of "pixels on target."

We have developed a "star" skeletonization procedure for analyzing human gaits in these situations [70]. The key idea is that simple, fast extraction of the broad internal motion features of an object can be employed to analyze its motion. The star skeleton consists of the centroid of a motion blob and all of the local extremal points that are recovered when traversing the boundary [see Fig. 6(A)]. Fig. 6(B) shows how two properties extracted from the skeleton provide cues to the person's gait. Assume the uppermost skeleton segment represents the torso and measure the angle ϕ between this segment and vertical. Assume the lower left segment represents one of legs and measure angle θ between this segment and vertical. Fig. 6(C) shows two star skeleton motion sequences for a walking and running human and plots the values θ_n and ϕ_n over time. Examining the average values of $\overline{\phi}_n$ shows that

Fig. 5. Sample results for LDA classification of object type and color.

Table 1
Experimental Evaluation of LDA Classification

	Human	Sedan	Van	Truck	Mule	Others	Total	Errors	%
Human	67	0	0	0	0	7	74	7	91%
Sedan	0	33	2	0	0	0	35	2	94%
Van	0	1	24	0	0	0	25	1	96%
Truck	0	2	1	12	0	0	15	3	80%
Mule	0	0	0	0	15	1	16	1	94%
Others	0	2	0	0	0	13	15	2	87%
								Avg.	*90%*

the posture of the running person can easily be distinguished from that of the walking person (people lean forward when they run). Also, the frequency of cyclic motion of the leg segments provides cues to whether this is a walking or running gait.

Gait classification using star skeleton features has been tested on a set of video sequences of adults and children walking and running. There are approximately 20 video sequences in each category, with pixels on target ranging from 50 to 400. Star skeletons and values for θ and ϕ were extracted from the video at a frame rate of 8 Hz. The average walking frequency was found to be 1.75 (Hz) and for running 2.875 (Hz). A threshold frequency of 2.0 (Hz) correctly classifies 97.5% of the gaits. Note that these frequencies are twice the actual footstep frequency because only the leftmost leg segment is considered. For each video sequence, the average inclination $\overline{\phi}$ of the torso showed that the forward leaning torso of a running figure can be clearly distinguished from the more vertical torso of a walking one. A threshold value of 0.15 rad correctly classifies 90% of the gaits. More details can be found in [70].

III. Multisensor Data Fusion

A multisensor surveillance system is more than a collection of sensors acting independently to detect and track objects. At some point, all observations must be brought into a common frame of reference to form a coherent, dynamic scene representation. This scene representation should be complete, in that it contains the union of all observations made by all sensors. Multiple observations of the same object from different cameras should be identified and merged into a single, more accurate object description. The representation should make explicit the spatial relationships between sensors, objects, and scene features, to aid sensor tasking and visualization of the scene by the human operator.

We believe that bringing all observations into a common 3-D coordinate system is the key to coherently representing a large number of object hypotheses from multiple, widely spaced sensors. We choose geodetic coordinates as this common coordinate system. In contrast to all other surveillance systems that we know of, which work in an arbitrary local scene coordinate system, we compute the latitude, longitude, and elevation with respect to the WGS84 datum

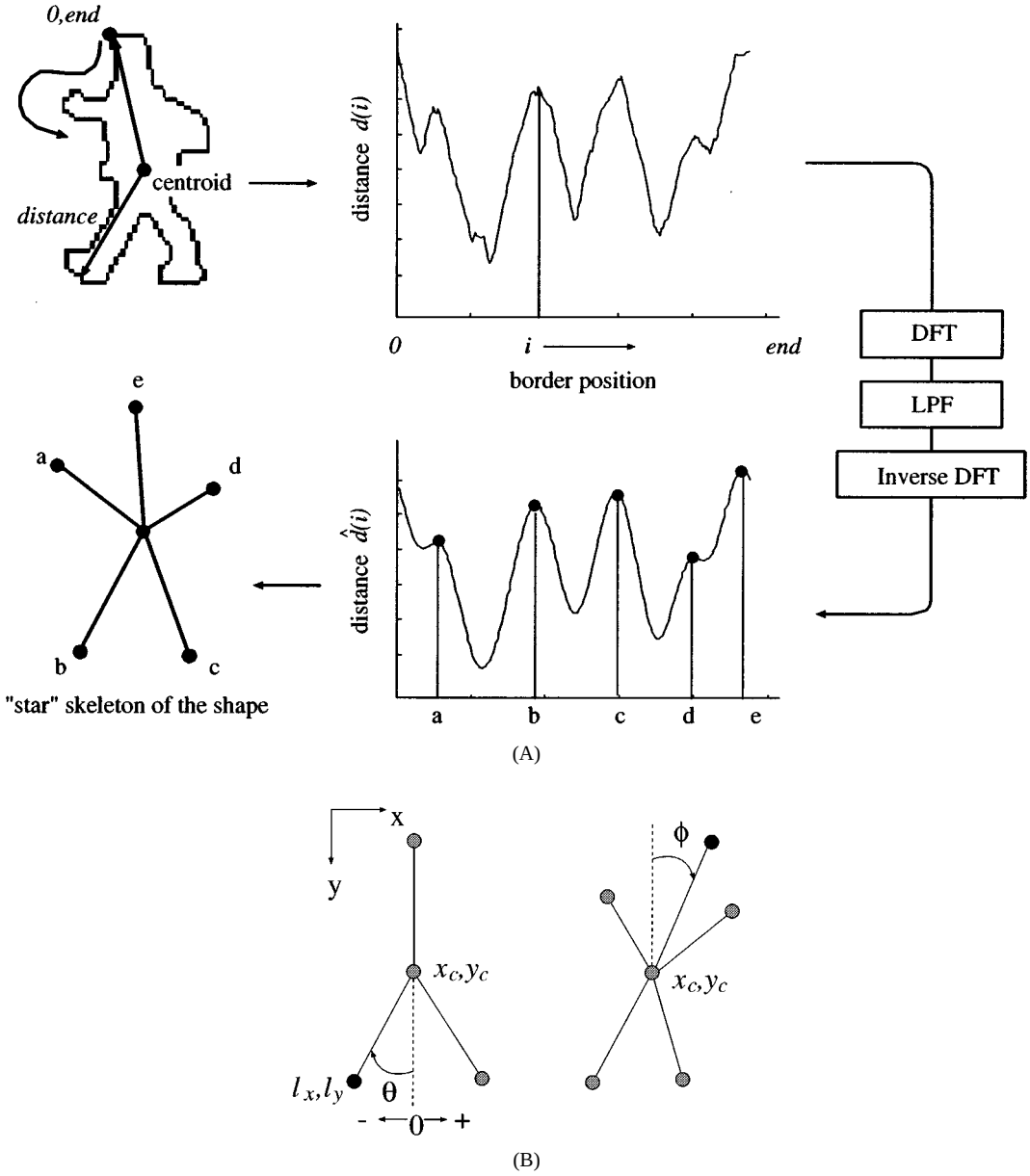
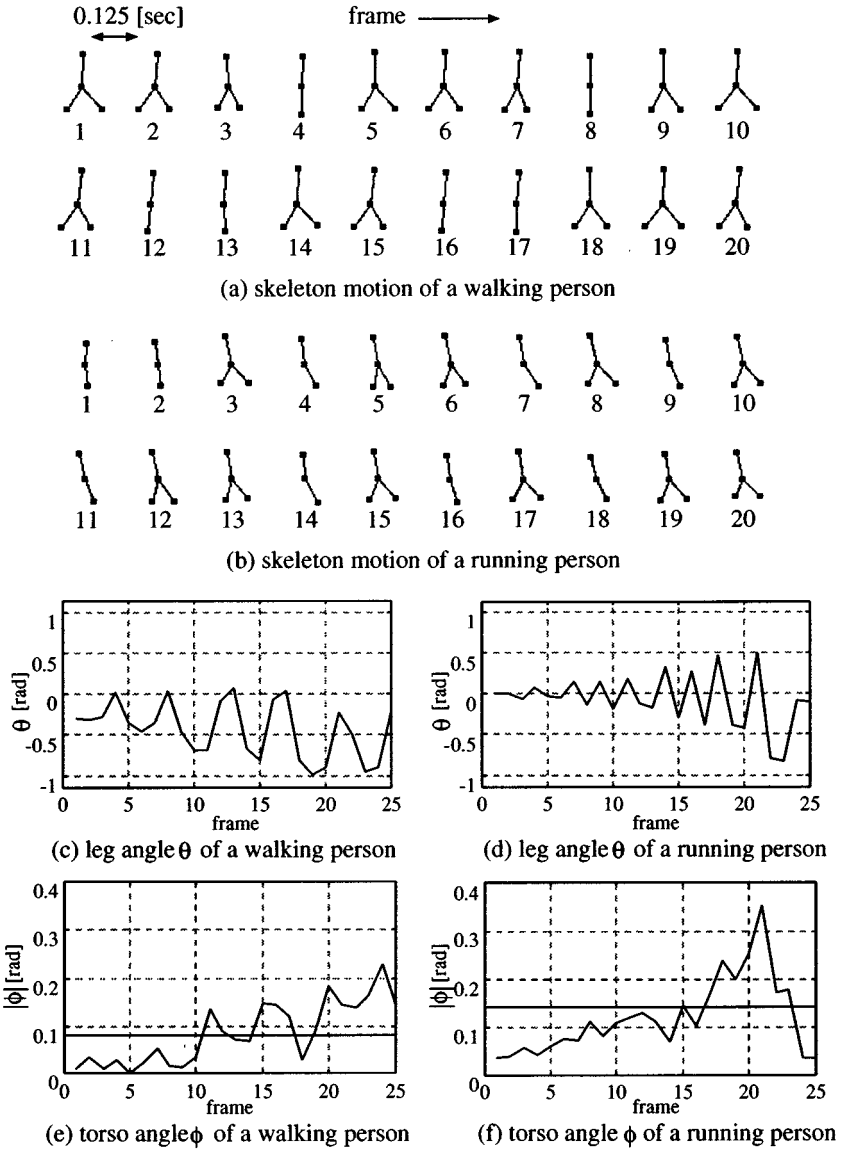


Fig. 6. (A) The star skeleton is created by "unwrapping" the silhouette boundary as a distance function from the centroid, and extracting extremal points. (B) Determination of posture features from the skeleton: θ is the angle the left cyclic point (leg) makes with the vertical, and ϕ is the angle the torso makes with the vertical.

(so-called "GPS coordinates" [71]), of each person and vehicle we detect and track. Since geometric computations can be difficult in the spherical geodetic coordinate system, our internal computations are carried out within a set of local Cartesian frames defined throughout the site; however, all of these Cartesian frames have known transforms to and from geodetic coordinates. We believe that having precise knowledge of where objects are in the world, represented in a commonly agreed upon global (in the literal sense of the word) coordinate system, is a significant advantage. For example, this choice has allowed us to easily merge our ground-base surveillance results with hypotheses generated independently by an airborne sensor operated by The Sarnoff Corporation and the U.S. Army's Night Vision and Electronic Sensors Directorate [28], [72]. It also has allowed us to use third-party cartographic software and datasets such as United States Geological Survey maps, orthophotos, DEMS, and road network graphs, in the development of our site model [72].

A. Camera Calibration

A mapping between sensor coordinates and scene coordinates is determined by calibrating each sensor with respect to the geodetic coordinate system. We have developed methods for fitting a projection model consisting of intrinsic (lens) and extrinsic (pose) parameters of a camera with active pan, tilt and zoom control. Intrinsic parameters are calibrated by fitting parametric models to the optic flow induced by rotating and zooming the camera. This procedure is fully automatic



(C)

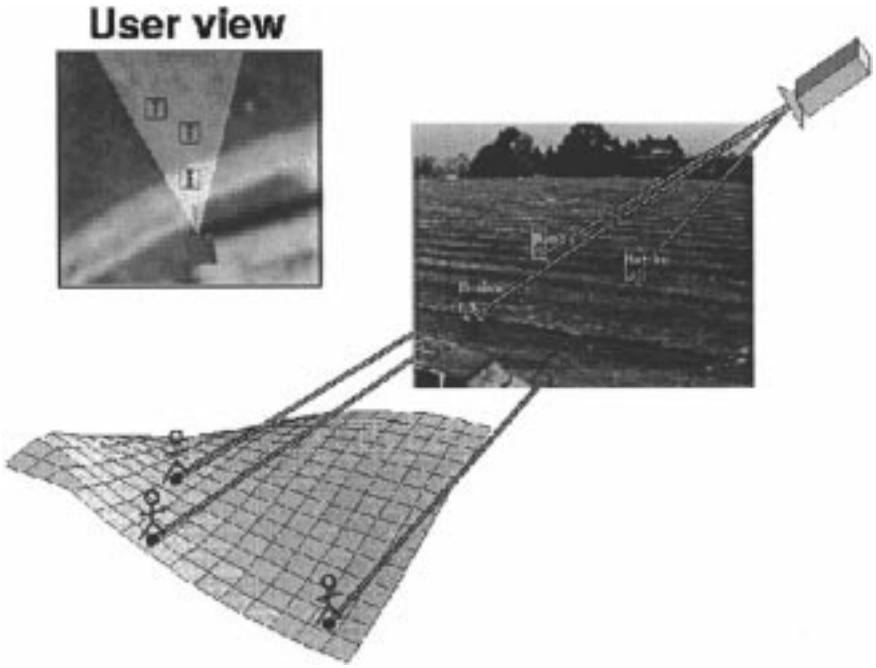
Fig. 6. *(Continued.)* (C) Skeleton motion sequences. The periodic motion of θ_n provides cues to the person's gait, as does the mean value of $\overline{\phi}_n$.

and does not require precise knowledge of 3-D scene structure. Extrinsic parameters are calculated by sighting a sparse set of scene landmarks that have been surveyed using differential GPS [71]. Actively rotating the camera to measure landmarks over a virtual hemispherical field of view leads to a well-conditioned exterior orientation estimation problem. Details of the calibration procedures are presented in [32].

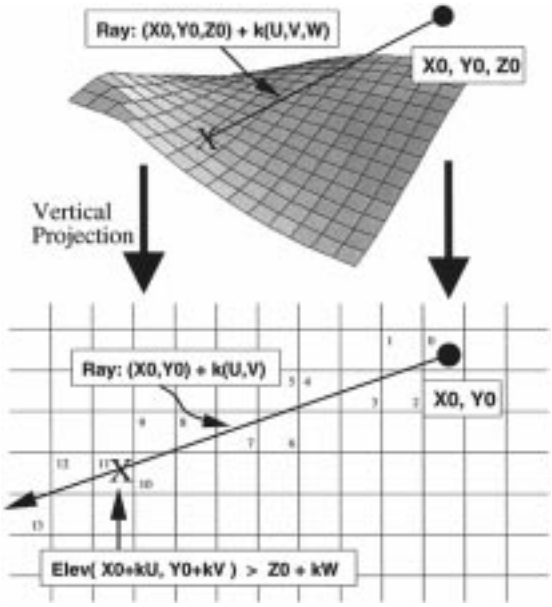
B. Geolocation

In regions where multiple sensor viewpoints overlap, object locations can be determined by wide-baseline triangulation [73]. However, regions of the scene that can be simultaneously viewed by multiple sensors are likely to be a small percentage of the total area of regard in real outdoor surveillance applications. Determining object locations from a single sensor requires domain constraints, in this case the assumption that the object is in contact with the terrain. This contact location is estimated by passing a viewing ray through the bottom of the object in the image and intersecting it with a model representing the terrain [see Fig. 7(a)].

Previous uses of the ray intersection technique for object localization in surveillance research have been restricted to small areas of planar terrain, where the relation between image pixels and terrain locations is a simple 2-D homography [9], [49], [74], [75]. This has the benefit that no camera calibration is required to determine the backprojection of an image point onto the scene plane, provided the mappings of at least four coplanar scene points are known beforehand. However, large outdoor scene areas may contain significantly varied terrain. To handle this situation, we perform geolocation using ray intersection with a full terrain model provided by a georeferenced digital elevation map (DEM). A simple geometric traversal technique based on the well-known Bresenham algorithm [76] for drawing rasterized line segments is used. Consider the vertical projection of the viewing ray



(a)



(b)

Fig. 7. (a) Estimating object geolocations by intersecting backprojected viewing rays with a terrain model. (b) A Bresenham-like traversal algorithm determines which DEM cell contains the first intersection of a viewing ray and the terrain.

onto the DEM grid [see Fig. 7(b)]. Starting at the grid cell (x_0, y_0) containing the sensor, each cell (x, y) that the ray passes through is examined in turn, progressing outward, until the elevation stored in that DEM cell exceeds the z component of the 3-D viewing ray at that location. See [31] for more details.

Since geolocation estimates are computed by backprojecting the center of the lowest side of the bounding box enclosing a moving blob, the surveillance system maintains a running estimate of the variance of this point. An internal estimate of horizontal variance of the geolocated point is formed by propagating this variance from the image, through the inverse projection equations, onto a horizontal plane with an elevation corresponding to the Z value of the 3-D geolocation estimate. This in general yields a covariance matrix with elliptical contours. A simplified uncertainty representation consisting of a single variance value is formed by taking the trace of this covariance matrix and dividing by 2. The horizontal (X, Y) location of a point, along with its approximate variance as computed above, is called the "map-plane" coordinate of a point in Section III-C.

We have evaluated geolocation accuracy for two cameras on the CMU campus using a Leica laser-tracking theodolite to generate ground truth (see Fig. 8). The experiment was run by having a person carry the theodolite prism for two loops around the parking lot, while the system logged time-stamped horizontal (X, Y) locations measured by the theodolite. The system also simultaneously tracked the person using each camera, while logging time-stamped geolocation estimates. Standard deviations of errors between ground truth locations and geolocation estimates from each camera are roughly on the order of .6 m along the axis of maximum spread and roughly .25 m at minimum, over an average camera-object distance of 65 m. The axis of maximum error for each camera is oriented along the direction vector from the camera to the object being observed. For more details, see [59].

C. Data Fusion

Sensor data fusion is a classic subject in engineering [77], [78]. In our case, the central idea of the data fusion process is to make associations between image-based sensor observations and scene-based object hypotheses. The central operator control unit (OCU) receives a continual stream of time-stamped symbolic object observations from each remote sensor processing unit (SPU). Each observation is geolocated, as described previously and compared to a list of known object hypotheses to see if it is an observation of an object already being tracked by the system. However, because the SPU sensors are scattered widely throughout the scene, one may be viewing the front of an object, one the side, and another the top. Therefore, comparison of SPU observations to 3-D hypotheses needs to use features that are insensitive to viewpoint [8], [79]. We use three features: 1) object geolocation (Section III-B); 2) object class (Section II-C); and 3) color. Color is represented by three coarse color histograms (red, green, and blue), concatenated into a single vector and normalized so that the sum over all counts is 1. This has the effect of normalizing the resulting color histogram, so that the representation is less sensitive to color variations due to viewpoint, illumination and sensor color response [80].

Observations received by the OCU are processed in time-stamped order. Features of a new observation are compared against the features stored for each existing object hypothesis using a match score function. Two situations may arise.

- The observation does not match any known object (match score is below a threshold for all hypotheses). In this case, a new object is hypothesized and its location, class and color information are initialized to those provided by the observation. A confidence value stored with the hypothesis is set to a nominal low value.
- The observation matches at least one object hypothesis (match score exceeds a threshold for one or more hypotheses). The object hypothesis with the highest match score value is chosen as the best match. The feature values of the SPU observation are used to update the features of this object hypothesis and the confidence of the object is increased.

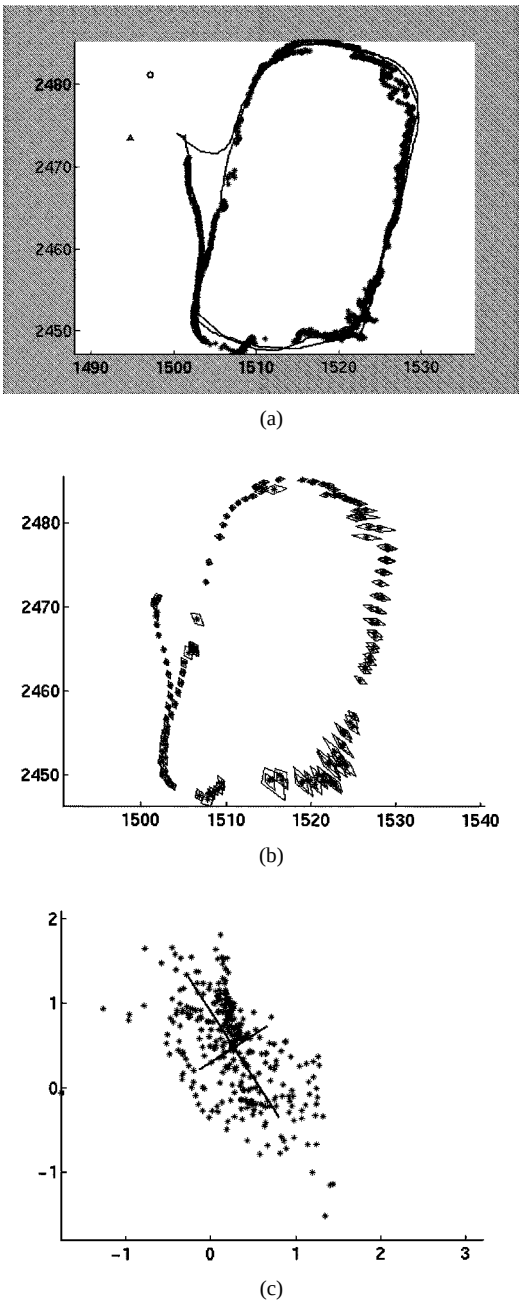


Fig. 8. (a) Ground truth trajectory determined by a theodolite, overlaid with geolocations estimated by the system while automatically tracking the same object from one camera. (b) Geolocation error boxes computed by the system. (c) Plotted covariances of the horizontal displacement errors between estimate geolocations and ground truth locations for corresponding time stamps. All scales are in meters.

A separate mechanism culls objects that leave the field of regard of the entire system. Any 3-D object hypothesis that is not matched and updated for 2 s is flagged as inactive. After ten more seconds pass with no activity, the hypothesis is declared dead and removed from the list of known hypotheses.

Computing the Match Score: Match score between an incoming observation and a known object hypothesis is computed by comparing location, class, and color information

$$M_{\text{match}} = M_{\text{location}} + M_{\text{class}} + M_{\text{color}}.$$

To compare location information, the geolocation technique described previously is used to transform the location measurement of the sensor observation into a "map plane" 2-D location measurement P and an associated variance σ_P^2, which corresponds to a circular Gaussian covariance matrix of the form $\sigma_P^2 I$. In the following equations, all covariances are approximated as circular Gaussians of the form $\sigma^2 I$ and the uncertainty propagation equations are simplified by using a single variance weight σ^2. The underlying mechanism, however, is propagation of 2-D Gaussian covariances, as in [81], with any resulting 2-D covariance matrix Σ being approximated as $\sigma^2 = \text{Trace}(\Sigma)/2$. For a given object hypothesis with a sequence of previous map plane trajectory points $(H_i, H_{i-1}, \ldots)$ and variances $(\sigma_i^2, \sigma_{i-1}^2, \ldots)$, the OCU predicts a new 2-D location H and variance σ_H^2 for the hypothesis assuming a constant velocity linear trajectory

$$H = H_i + (H_i - H_{i-1})\Delta$$

and

$$\sigma_H^2 = \frac{(\sigma_i^2 + \Delta^2\sigma_i^2 + \Delta^2\sigma_{i-1}^2)}{2}$$

where $\Delta = (t_{i+1} - t_i)/(t_i - t_{i-1})$ is a function of the timestamps $(t_{i+1}, t_i, t_{i-1}, \ldots)$ on the sequence of processed video frames from which the hypotheses were generated.

The prediction and the current observation are then tentatively merged into a joint sample point M with variance σ_M^2

$$M = \frac{P\,\sigma_H^2 + H\,\sigma_P^2}{\sigma_P^2 + \sigma_H^2}$$
$$\sigma_M^2 = \frac{1}{\frac{1}{\sigma_P^2} + \frac{1}{\sigma_H^2}}.$$

This is treated as a statistical distribution on M and the score M_{location} of matching the SPU observation P with object hypothesis H is taken to be proportional to the joint probability of two independent samples P and H drawn from the distribution for M

$$M_{\text{location}} = \exp\left\{-\frac{\|P - M\|^2 + \|H - M\|^2}{2\,\sigma_M^2}\right\}.$$

Note that this is not a proper probability since the bivariate Gaussian normalization constant has been dropped.

To compute M_{class}, a simple heuristic function is used. The current classifications supported by the match score function are: vehicle, human, human group, and unclassified (if the classification algorithm does not have enough data to make a suggestion). Given the classification of an object hypothesis C_i and the classification of the new observation C, the heuristic score M_{class} is computed as follows: If $C = C_i$, the value is 1.0; if either C or C_i is unclassified, the value is 0.75; if C is human and C_i is human group, or vice versa, the value is 0.6; otherwise, the value is 0. Color comparison between a new observation and an existing object hypothesis is based on their color histogram vectors. The value of M_{color} is taken as $1.0 - \text{med}(\Delta H)$ where ΔH is the set of differences between the color vectors of the two objects.

Feature Updating: When a new observation matches an object hypothesis already known to the system, the features of the hypothesis are updated by merging the new information from the observation with the previously stored values in the hypothesis. The new location and variance estimates for the matched hypothesis become $H_{i+1} = M$ and $\sigma_{i+1}^2 = \sigma_M^2$. The hypothesis class is updated to be the most frequent classification given to that hypothesis so far, as determined by a histogram of associated classifications (this is the same mechanism used to improve temporal classification performance at the SPU). The color vector of the object hypothesis is simply replaced with the color vector from the new observation.

IV. Multisensor Tasking and Control

An important goal of VSAM is to enable a single human operator to task a multisensor surveillance system at a relatively high level of abstraction. Traditional camera-centric commands such as "pan sensor A to position B" become cumbersome when many sensors are available, and it is nearly impossible for a person to orchestrate control strategies by commanding multiple sensors in a specific temporal order. We seek instead to issue high-level requests such as "track this car" or "report any red sedans that enter the gate," and have the system (specifically the OCU) decompose them into a sequence of low-level commands issued to the appropriate sensors at the correct times.

A. Sensor Tasking

High-Level Tasking: Through the GUI, the operator can specify objects to be actively tracked and geographic locations to monitor for events. The operator can choose to operate in either an image-centric or scene-centric coordinate system. For example, the user can choose to see video imagery from one of the sensors and can click on an image feature to task the system to control the sensor pan and tilt to bring that feature into the center of the image. If the user clicks on a moving object within the video display, the system is automatically tasked to actively track that object.

More interesting and novel, is the ability of the operator to specify operations in scene space using a map overlaid with sensor locations and moving icons representing tracked objects. For example, the user can click on a sensor location and drag the mouse to a point on the map, which tasks the sensor to look at that geographic location in the scene. The user can also click on any of the moving object icons displayed on the GUI and the system will allocate resources to continually track that object. Another high-level tasking command is to specify a region of interest (ROI) event trigger. To specify a ROI, the user traces the outline of a polygonal region on the map and the OCU then determines which sensors have the best view of that area and assigns them to observe it. Any object entering the ROI triggers an alert to the operator. The operator can also specialize a ROI trigger to particular classes of objects (e.g., human or vehicle) and the system will provide an alert only when that type of object enters. For example, it is easy to task the system to "report all pedestrians entering

this restricted area" by creating an appropriate ROI trigger. Given suitably sophisticated video understanding algorithms at each sensor, ROI triggers could also be specialized to specific events or activities occurring in a geographic location.

Sensor Arbitration: The GUI sends high-level user requests to the OCU, which keeps a list of all tasks currently being handled by the system. Each SPU has only a few basic capabilities, such as GO TO PAN TILT ZOOM(pan, tilt, zoom) and TRACK TARGET(object). The OCU must therefore "compile" high-level user requests into a sequence of sensor specifications and commands. This compilation mainly involves choosing a sequence of sensors to carry out the task and commanding them to look at the appropriate scene locations at the right times. Sensor-specific pan, tilt, and zoom commands associated with a particular object or ROI are computed by the OCU using the known calibration parameters of the sensor and the known geometry of the object and scene.

The hard part of automated sensor tasking is allocating sensors to perform all of the tasks required by the system and requested by the operator. When there are not enough resources to complete all required tasks, the system performance should degrade gracefully. When there are relatively few tasks, the system should automatically exploit redundant sensors to provide as much information as possible. Automated sensor selection for each system task is performed according to a cost matrix. At every iteration of system time, each sensor is assigned a cost of performing each task and a greedy strategy is used for assigning tasks to sensors. The factors that contribute to the cost function are as follows.

1) **Visibility V_{ij}**: This is a binary measure (0 or 1) indicating whether sensor i can view the geographic location associated with task j (1 indicates visible). Visibility is determined using the geometric site model to determine if there are any significant occluding objects between the camera and the point of interest.
2) **Distance D_{ij}**: This is the distance between sensor i and the location of task j.
3) **Tasking T_{ij}**: This is a binary measure (0 or 1) indicating whether sensor i is currently executing operations involved in task j (0 indicates already tasked).
4) **Priority P_j**: This is the priority of the task assigned by the user (a low value indicates a high priority). Tasks with low priority, such as scanning the area for moving objects, become default tasks that are performed whenever a camera would otherwise be idle. These default tasks are preempted whenever a higher priority task, such as tracking a particular object, are scheduled.

A matrix of cost values is created where each value C_{ij} represents the cost of sensor i performing task j. The cost function is

$$C_{ij} = V_{ij}\,(K_D\,D_{ij} + K_T\,T_i + K_P\,P_j)$$

where the Ks are tuning constants. If C_{ij} is zero, it indicates that the task cannot be performed by the sensor. Otherwise, a small nonzero value indicates a desirable tasking. The assignment of tasks to sensors at each time step is scheduled as follows.

- If a user has taken direct control of a sensor, remove it from consideration.
- For each task, select the sensor that has minimum nonzero cost to perform it and remove that sensor from the list of untasked sensors.
- After all tasks are assigned, any sensors left untasked are assigned to perform their minimum cost task. These sensors thus provide redundancy in carrying out that task.

This arbitration scheme automatically allocates sensors such that: 1) high priority tasks are performed at the expense of less important ones; 2) no sensors are ever idle; and 3) sensors with better viewpoints of a particular area are favored over those farther away. Note that arbitration does not simply choose the closest sensor—a more distant sensor can be selected if the closest sensor is occluded or busy with a higher priority task.

B. Multisensor Cooperative Control

If a sensor is actively tracking an object that is moving out of its field of view and into the field of view of another sensor, the cost associated with having the first sensor track it will increase, while the cost associated with the second sensor will decrease, until the point where the second sensor will automatically be tasked to take over the surveillance. Thus, cost-based sensor arbitration allows the OCU to automatically coordinate multiple sensors to seamlessly track moving objects over an extended area. Just before the hand-off between sensors occurs, the second sensor is commanded by the OCU to point in the right direction at the right zoom factor. The OCU then issues a TRACK-TARGET command, passing the estimated object image location to the SPU along with a target description that uses the same view-independent classification and color features used for data fusion matching. The SPU compares all moving objects in its field of view with the same matching function used in data fusion comparison (Section III-C)

$$M_{\text{match}} = M_{\text{location}} + M_{\text{class}} + M_{\text{color}}$$

except that the location comparison now uses sensor-specific 2-D image coordinates rather than horizontal map-plane coordinates.

An example of using cost-based tasking to achieve autonomous multisensor hand-off to track a vehicle as it travels through campus is shown in Fig. 9. This diagram shows continuous tracking of a single object for a distance of approximately 400 m and a time of approximately 3 min. In Fig. 9(a), two sensors cooperatively track the object. At the time shown in Fig. 9(b) the object is occluded from sensor 2, but is still visible from sensor 1, which continues to track it. When the object moves out of the occlusion area, sensor 2 is automatically commanded to

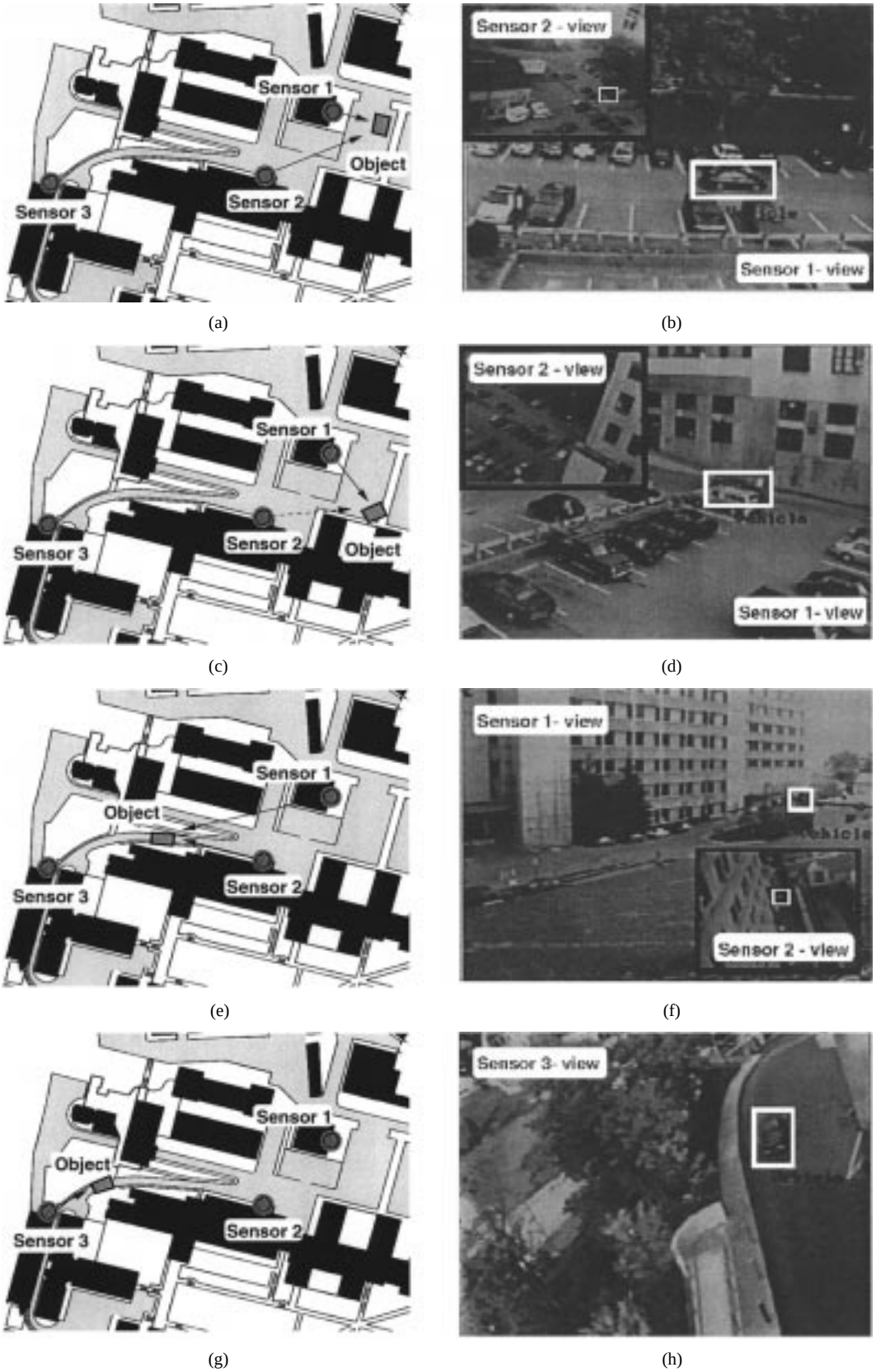


Fig. 9. Cooperative, multisensor tracking (see text for description).

track it again [Fig. 9(c)]. Finally, when the object moves out of the field of regard of both sensors, a third sensor is automatically commanded to continue surveillance, as shown in Fig. 9(d). By automatically managing multiple, redundant camera resources, the vehicle is continuously tracked through a complex urban environment.

(a) (b)

Fig. 10. Example of camera slaving. (a) Wide-angle view in which a person is detected. (b) A better view from a second camera, which has been tasked to intercept the person's estimated 3-D path.

A second form of sensor cooperation is sensor slaving. This is not a side effect of cost-based tasking, as hand-off is, but is instead a special, user-selectable task programmed into the system. A camera slaving system has one master camera and at least one slave camera. The master camera is set to have a wide-angle view of the scene so that it can track all objects in a wide area without moving. The object trajectories generated by the master camera are sent to the OCU, where they are converted into 3-D trajectories. After estimating the 3-D location of an object from the first camera's viewpoint, the OCU transforms the location into a pan-tilt command to control the slave camera. The slave camera, which is highly zoomed in, can then follow the trajectory to generate close-up imagery of the object. If more than one object is detected by the master camera at one time, the OCU will "multitask" the slave camera to cycle through the objects, visiting each one in turn for one second to generate an updated close-up view. Fig. 10 shows an example of camera slaving. A person has been detected and is being tracked in the wide-angle view shown in the left image. A second narrow field of view camera is continually tasked by the OCU to move slightly ahead of the person's estimated 3-D trajectory to generate a close-up view, as shown in the right image.

V. Dynamic Scene Visualization

Keeping track of people, vehicles, and their interactions over a large area is a difficult job for a human observer. It certainly cannot be done effectively by looking at a wall of video screens each showing a disparate sensor view. Our approach is to provide an interactive, graphical visualization by automatically placing dynamic agents representing people and vehicles into a synthetic view of the environment. This graphical approach to operator display has the benefit that visualization of an object is no longer tied to the original resolution and viewpoint of the video sensor, since a synthetic replay of the dynamic events can be constructed from any perspective. Particularly striking is the amount of data compression that can be achieved by transmitting only symbolic object information instead of raw video data. Currently, we can process NTSC color imagery with a frame size of 320×240 pixels at ten frames per second on a Pentium II computer, so that data is streaming into the system at a rate of roughly 2.3 Mb/s per sensor. After automated video processing, detected object hypotheses contain information about object type, location, and velocity, as well as measurement statistics such as a time stamp and a description of the sensor (current pan, tilt, and zoom, for example). Each object data packet takes up roughly 50 bytes. Therefore, with our current communication protocol, a sensor tracking three objects for 1 s at ten frames per second ends up transmitting 1500 bytes back to the OCU, well over a thousandfold reduction in data bandwidth.

A. Map-Based Operator GUI

The human operator interacts with the VSAM system through a single, map-based graphical user interface (GUI). The GUI is dominated by a scalable and scrollable georeferenced map of the site, overlaid with all current object locations, camera locations, and camera fields of view (see Fig. 11). Each of these graphical entities can be selected using the workstation's mouse. To the right of the map is a status pane, showing how many sensors are active, how many objects are currently being tracked, and other vital statistics of the system. The lower right pane is a low-bandwidth, compressed video stream from one of the cameras, which can be selected by the user. The video from that sensor is compressed by transmitting a spatially subsampled version of the adaptive background model used for motion detection (Section II-A), updated once every few seconds, overlaid with spatially subsampled pixels from within the bounding box of detected moving objects, updated at 10 Hz. These moving foreground objects overlaid on the background can also be selected using the mouse. The lower left pane of the GUI is the sensor-suite tasking interface, through which the operator can task individual sensor units, as well as the entire testbed, to perform surveillance operations such as generating a quick summary of all object activities in the area or creating a region of interest event trigger (Section IV-A).

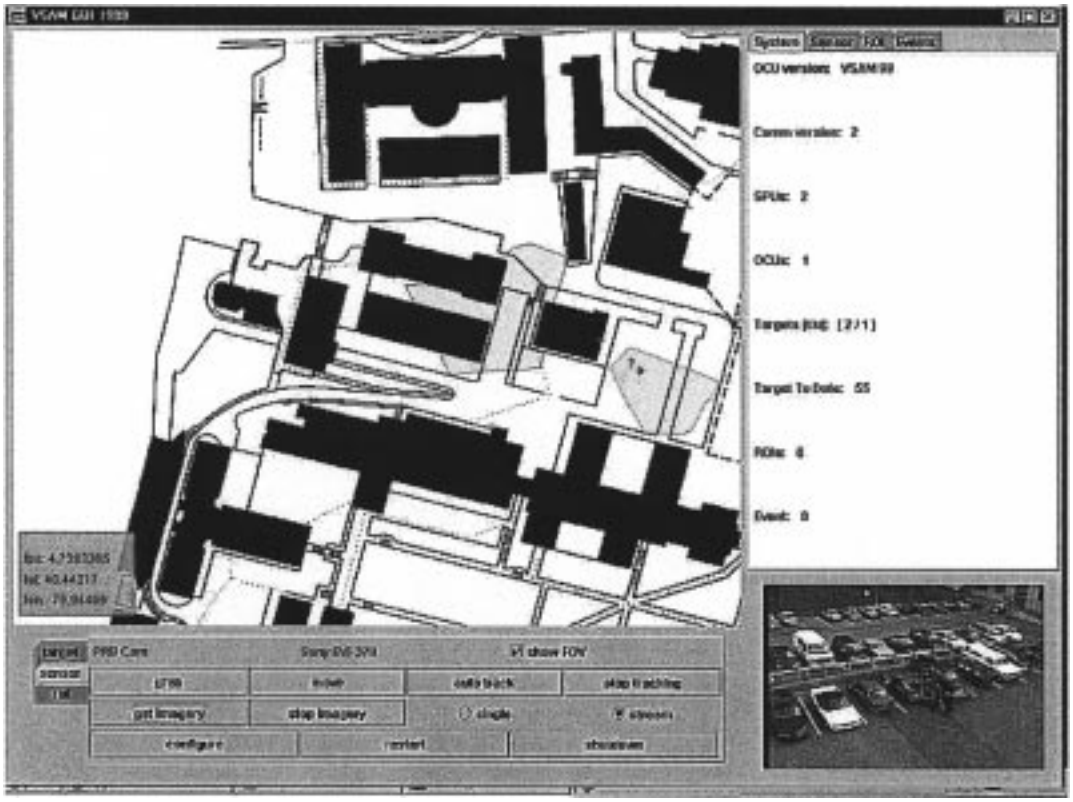


Fig. 11. Map-based graphical user interface, through which a human operator tasks the VSAM sensor suite and visualizes objects tracked by the system.

B. Immersive 3-D Visualization

Ultimately, we believe that the key to comprehending large-scale, multiagent events is a full, 3-D immersive visualization that allows the human operator to move at will through the environment to view dynamic events unfolding in real-time from any viewpoint. This goal guided our selection of the Compact Terrain Database (CTDB) format for representing our 3-D model of the surveillance site. The CTDB format was originally designed to represent large expanses of terrain for U.S. Department of Defense distributed interactive simulation (wargaming) exercises [82], [83]. In addition to terrain elevation, the CTDB format also represents relevant cartographic features on top of the terrain skin, including buildings, roads, bodies of water, and tree canopies. An important benefit to using CTDB as a site model representation is that it allows us to easily interface with third-party cartographic modeling and visualization tools developed to provide synthetic training environments [30]. The Modular Semi-Automated Forces (ModSAF) program provides a 2-D graphical interface similar to our VSAM GUI, with the ability to insert computer-generated human and vehicle avatars that provide simulated opponents for training [84], [85]. The ModStealth program generates an immersive, realistic 3-D visualization of texture-mapped scene geometry and computer generated avatars [29].

We have built an interface to the ModSAF and ModStealth programs from the VSAM testbed system. At the OCU, 3-D object hypotheses are repackaged into a data packet format specifying the type of avatar wanted (various human and vehicle types are available) and the current 3-D location and orientation of the avatar within the CTDB site model. These data packets are then broadcast (multicast) on the VSAM network, where any running ModSAF or ModStealth visualization clients pick them up and display them within their respective 2-D or 3-D synthetic environments. Fig. 12(a) and (b) shows an example of three people being tracked by the VSAM system and a screen dump of their avatars represented within a 2-D ModSAF display. Fig. 12(c) and (d) shows a person being tracked and the corresponding avatar being viewed within a 3-D ModStealth viewer. Fig. 13 shows an example of multiple objects detected automatically and inserted as avatars within a 3-D ModStealth view of the CMU campus. These experiments demonstrate that it is possible to automatically detect, track, and classify multiple people and vehicles using an automated surveillance system and then insert them as avatars in a synthetic environment for real-time visualization.

C. Preliminary Work: Web-Page Event Reporting

In addition to on-line interactive use by an operator, another useful mode of operation for an automated surveillance system is unsupervised data logging, followed by off-line operator review days or months after collection of the data. We have begun to develop a prototype web-based data logging system to explore this application. For each object detected and tracked by the system, all symbolic data (e.g., object classification; color information; time-stamped trajectory data) is

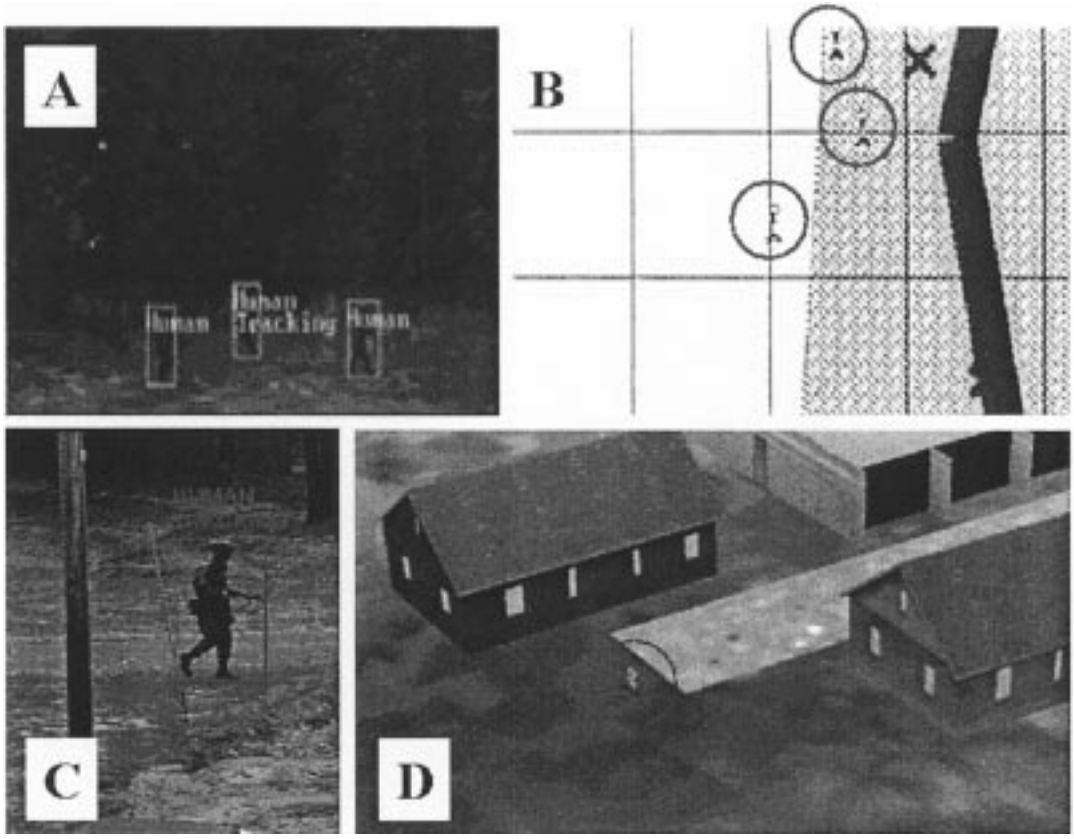


Fig. 12. Sample synthetic environment visualizations of VSAM detection and tracking data. (a) Automated tracking of three soldiers. (b) ModSAF 2-D orthographic map display of estimated geolocations. (c) Tracking of a soldier walking out of town. (d) Immersive, texture-mapped 3-D visualization of the same event, seen from a user-specified viewpoint.

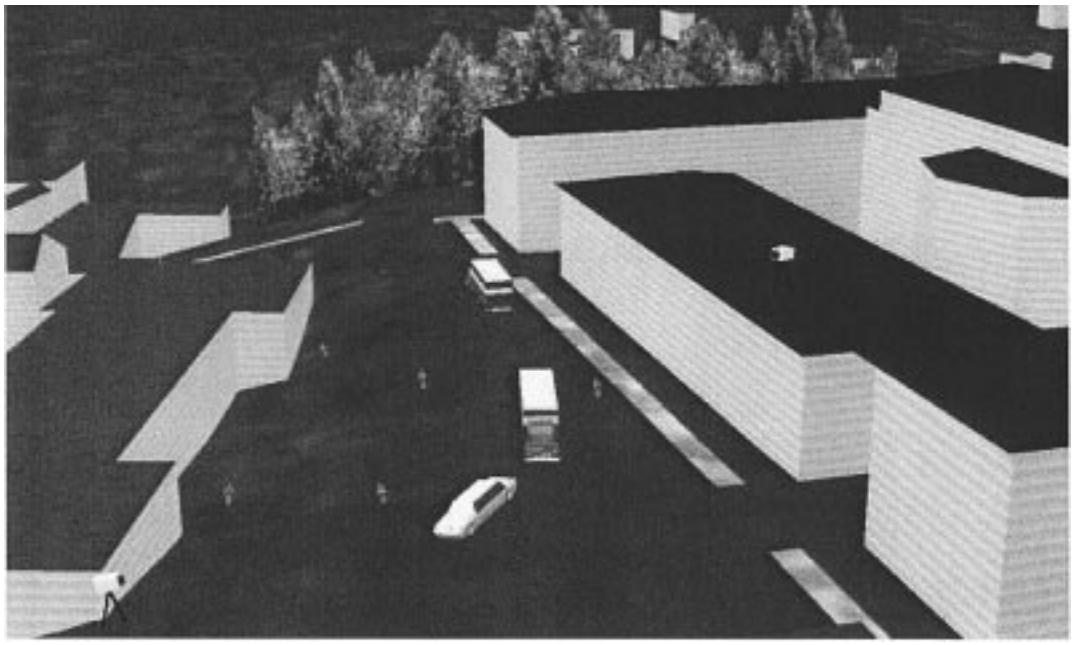


Fig. 13. Real-time, 3-D ModStealth visualization of objects detected and classified by the VSAM testbed system.

saved, along with a cropped image of the object. All observations can then be explored by web browsing via CGI through an HTTP server [86], so that a human reviewer can access the data from anywhere. Fig. 14(a) shows a sample object report web page. To cut down on information overload, the user can select specific subsets of object classes to view. When the

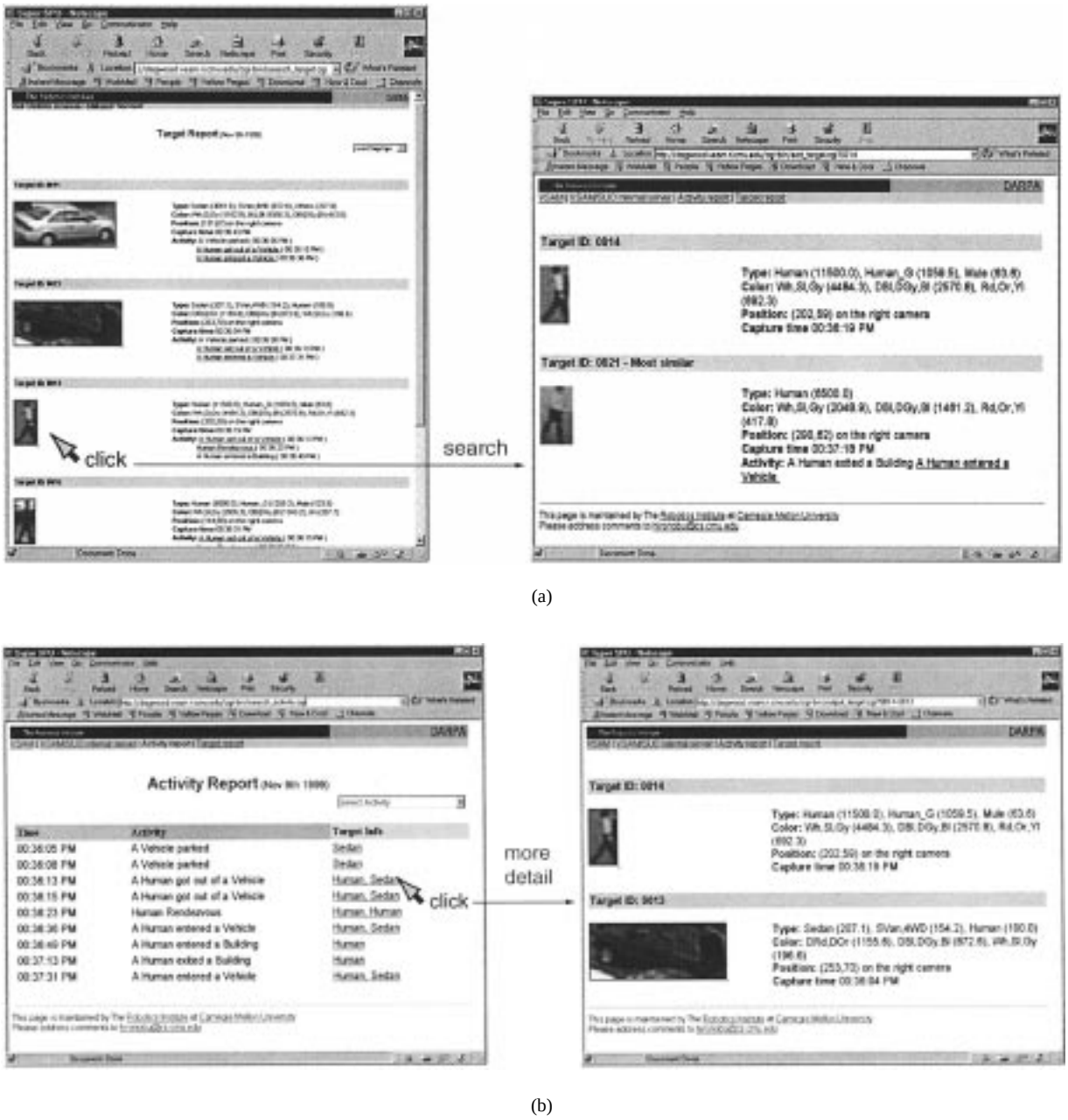


Fig. 14. (a) Object report web page. (b) Activity (event) report web page.

user selects an object, the system automatically brings up a page showing other objects of the same class having similar color features. In this way, it might be possible for a user to detect the same vehicle or person being observed at different places and times around the surveillance site.

Unfortunately, in a high-traffic area, data on dozens of people and vehicles can be collected in just a few minutes of observation. Browsing the huge volume of raw surveillance data collected over even a single day is unmanageable and some type of higher-level semantic organization of the data is desired. To this end, we have begun developing an event detection program that scans the log files for common "events" that can be given a semantic label. There has been much work on parsing sequences of low-level surveillance observations, particularly time-stamped trajectories, into events that signify object interactions [5], [10]–[12], [15], [17], [87]–[90]. Our prototype event detector is based on hidden Markov models (HMMs) [10], [87], trained to recognize simple object interactions. Briefly, output from the low-level detection and tracking surveillance algorithms is quantized into the following discrete set of attributes and values for each motion blob:

1) **object class:** Human, Vehicle, HumanGroup;
2) **object action:** Appearing, Moving, Stopped, Disappearing;
3) **interaction:** Near, MovingAwayFrom, MovingToward, NoInteraction.

The activities that can be labeled are: a) human entered a vehicle; b) human exited a vehicle; c) human entered a building; d) human exited a building; e) a vehicle parked; and f) human

rendezvous. To train the activity classifier, conditional and joint probabilities of attributes and actions are obtained by generating many synthetic activity occurrences in simulation. This simulation-based training approach is motivated by [10]. Fig. 14(b) shows a preliminary example of a generated activity report. The activity report shows labeled events such as a "car parked," or "human entered a building," sorted by time. If a user wants more detail, a hypertext link brings up a page showing a cropped image of the object, along with its class and color information.

VI. Summary

The paper has presented an overview of video understanding algorithms developed at CMU to perform cooperative, multisensor surveillance. A network of smart sensors are deployed that are independently capable of performing real-time, autonomous object detection, tracking, classification and gait analysis. Results from these single-sensor technologies are combined into a coherent overview of the dynamic scene by multisensor fusion algorithms running on a central operator control station. The key to data integration from these multiple, widely spaced sensors is computation of object location with respect to a 3-D site model, followed by object hypothesis comparison and matching using a set of viewpoint-independent descriptors.

A single user tasks the system through an intuitive graphical user interface. The system automatically allocates sensors to perform these tasks using an arbitration function that determines the cost of assigning each sensor to each task. The system performs a greedy optimization over this cost table to maximize overall system performance. Through this cost-based scheduling approach, multiple sensors are automatically tasked to cooperatively track objects over long distances and through occlusion.

Visualizing the relative locations of people and vehicles over a large area is a difficult task. We provide the user with 2-D and 3-D synthetic views of the environment, within which detected people and vehicles are displayed as dynamic agents. This approach has the benefit that visualization of scene events is no longer tied to the original resolution and viewpoint of a single video sensor and the operator can therefore infer proper spatial relationships between sets of objects and between objects and scene features such as roads and buildings, leading to a better understanding of the evolving scene.

Acknowledgment

The authors would like to thank the CMU VSAM team members: D. Duggins, Y. Tsin, R. Patil, D. Tolliver, O. Hasegawa, N. Enomoto, Y.-T. Do, and A. Lee, for their tireless efforts and good humor. They also thank the members of the Sarnoff Corporation: P. Burt, L. Wixson, J. Eledath, and D. Mishra, who developed and demonstrated automated real-time airborne surveillance techniques within the scope of this effort.

References

[1] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A real-time computer vision system for measuring traffic parameters," in *Proc. 1997 Conf. Computer Vision and Pattern Recognition* , San Juan, Puerto Rico, June 1997, pp. 495–501.

[2] T. Darrell, G. G. Gordon, M. Harville, and J. Woodfill, "Integrated person tracking using stereo, color and pattern detection," *Int. J. Comput. Vis.*, vol. 37, pp. 175–185, June 2000.

[3] J. M. Ferryman, S. J. Maybank, and A. D. Worrall, "Visual surveillance for moving vehicles," *Int. J. Comput. Vis.*, vol. 37, no. 2, pp. 187–197, June 2000.

[4] G. L. Foresti, "Real-time system for video surveillance of unattended outdoor environments," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, p. 697, Oct. 1998.

[5] M. Haag and H. H. Nagel, "Incremental recognition of traffic situations from video image sequences," *Image Vis. Comput.*, vol. 18, pp. 137–153, Jan. 2000.

[6] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 809–830, Aug. 2000.

[7] J. Heikkila and O. Silven, "A real-time system for monitoring of cyclists and pedestrians," in *Proc. 2nd IEEE Int. Workshop Visual Surveillance*, Fort Collins, CO, June 1999.

[8] T. Huang and S. Russell, "Object identification: A Bayesian analysis with application to traffic surveillance," *Artif. Intell.*, vol. 103, no. 1-2, pp. 77–93, Aug. 1998.

[9] D. Koller, K. Daniilidis, and H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *Int. J. Comput. Vis.*, vol. 10, pp. 257–281, June 1993.

[10] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian computer vision system for modeling human interactions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 831–843, Aug. 2000.

[11] P. Remagnino, T. Tan, and K. Baker, "Multiagent visual surveillance of dynamic scenes," *Image Vis. Comput.*, vol. 16, pp. 529–532, June 1998.

[12] R. Rosales and S. Sclaroff, "3D trajectory recovery for tracking multiple objects and trajectory-guided recognition of actions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Fort Collins, CO, June 1999, pp. 117–123.

[13] "Section I, video surveillance and monitoring," in *Proc. DARPA Image Understanding Workshop*, vol. 1, Monterey, CA, Nov. 1998, pp. 1–400.

[14] M. Bogaert, N. Chleq, P. Cornez, C. S. Regazzoni, A. Teschioni, and M. Thonnat, "The passwords project," in *Int. Conf. Image Processing*, 1996, pp. 675–678.

[15] M. Rota and M. Thonnat, "Video sequence interpretation for visual surveillance," in *Proc. 3rd IEEE Int. Workshop on Visual Surveillance*, Dublin, Ireland, July 2000.

[16] "The Views project and wide-area surveillance," VIEWS , PM-02-ECCV92-01, 1992.

[17] H. Buxton and S. G. Gong, "Visual surveillance in a dynamic and uncertain world," *Artif. Intell.*, vol. 78, pp. 431–459, Oct. 1995.

[18] T. Matsuyama, "Cooperative distributed vision," in *Proc. DARPA Image Understanding Workshop*, vol. 1, Nov. 1998, pp. 365–384.

[19] "CDVWS," in *Proc. 1st/2nd/3rd Int. Workshop Cooperative Distributed Vision*, Kyoto, Japan, 1997/1998/1999.

[20] "VS'2000," in *Proc. 3rd IEEE Int. Workshop Visual Surveillance*, Dublin, Ireland, July 2000.

[21] "VS'98," in *Proc. 1st IEEE Int. Workshop Visual Surveillance*, Bombay, India, Jan. 1998.

[22] "VS'99," in *Proc. 2nd IEEE Int. Workshop Visual Surveillance*, Fort Collins, CO, June 1999.

[23] "PETS'2000," in *Proc. First IEEE Int. Workshop Performance Evaluation of Tracking and Surveillance*, J. Ferryman, Ed., Grenoble, France, Mar. 2000.

[24] R. T. Collins, A. J. Lipton, and T. Kanade, "Introduction to the special section on video surveillance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 745–746, Aug. 2000.

[25] S. Maybank and T. Tan, "Introduction — surveillance," *Int. J. Comput. Vis.*, vol. 37, pp. 173–173, June 2000.

[26] T. E. Boult, R. Micheals, X. Gao, P. Lewis, C. Power, W. Yin, and A. Erkan, "Frame-rate omnidirectional surveillance and tracking of camouflaged and occluded targets," in *Proc. 2nd IEEE Int. Workshop Visual Surveillance*, Fort Collins, CO, June 1999.

[27] B. E. Flinchbaugh, "Robust video motion detection and event recognition," in *Proc. 1997 DARPA Image Understanding Workshop*, New Orleans, LA, May 1997, pp. 51–54.

[28] L. Wixson, J. Eledath, M. Hansen, R. Mandelbaum, and D. Mishra, "Image alignment for precise camera fixation and aim," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Santa Barbara, CA, June 1998, pp. 594–600.

[29] K. Cauble, "The OpenScene ModStealth," in *Proc. Simulation Interoperability Workshop* , 1997, Paper 97S-SIW-008.

[30] D. B. Cavitt, C. M. Overstreet, and K. J. Maly, "Modeling and distributed simulation techniques for synthetic training environments," in *Modeling and Simulation of Advanced Computer Systems: Applications and Systems*, K. Bagchi, Ed. Philadelphia, PA: Gordon & Breach, 1996, pp. 237–260.

[31] R. Collins, Y. Tsin, J. R. Miller, and A. Lipton, "Using a DEM to determine geospatial object trajectories," in *Proc. 1998 DARPA Image Understanding Workshop*, Monterey, CA, Nov. 1998, pp. 115–122.

[32] R. T. Collins and Y. Tsin, "Calibration of an outdoor active camera system," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Fort Collins, CO, June 1999, pp. 528–534.

[33] M. Haag, T. Frank, H. Kollnig, and H. H. Nagel, "Influence of an explicitly modeled 3D scene on the tracking of partially occluded vehicles," *Comput. Vis. Image Understand.*, vol. 65, pp. 206–225, Feb. 1997.

[34] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proc. Int. Conf. Computer Vision*, Corfu, Greece, 1999, pp. 255–261.

[35] C. Anderson, P. Burt, and G. van der Wal, "Change detection and tracking using pyramid transformation techniques," in *Proc. SPIE Intelligent Robots and Computer Vision*, vol. 579, 1985, pp. 72–78.

[36] P. L. Rosin and T. Ellis, "Image difference threshold strategies and shadow detection," in *Proc. British Machine Vision Conf.*, 1995, pp. 347–356.

[37] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 747–757, Aug. 2000.

[38] C. R. Wren, A. Azarbayejani, T. J. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, pp. 780–785, July 1997.

[39] J. Barron, D. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 42–77, 1994.

[40] C. Cedras and M. Shah, "Motion-based recognition: A survey," *Image Vis. Comput.*, vol. 13, pp. 129–155, Mar. 1995.

[41] G. Halevy and D. Weinshall, "Motion of disturbances: Detection and tracking of multi-body nonrigid motion," *Mach. Vis. Applicat.*, vol. 11, no. 3, pp. 122–137, 1999.

[42] D. G. Lowe, "Robust model-based motion tracking through the integration of search and estimation," *Int. J. Comput. Vis.*, vol. 8, pp. 113–122, Aug. 1992.

[43] S. Wachter and H. H. Nagel, "Tracking persons in monocular image sequences," *Comput. Vis. Image Understand.*, vol. 74, pp. 174–192, June 1999.

[44] M. Isard and A. Blake, "C-conditional density propagation for visual tracking," *Int. J. Comput. Vis.*, vol. 29, pp. 5–28, Aug. 1998.

[45] G. D. Hager and P. N. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 1025–1039, Oct. 1998.

[46] Q. Zheng and R. Chellappa, "Automatic feature point extraction and tracking in image sequences for arbitrary camera motion," *Int. J. Comput. Vis.*, vol. 15, pp. 31–76, June 1995.

[47] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*. Boston, MA: Academic, 1988.

[48] I. J. Cox, "A review of statistical data association techniques for motion correspondence," *Int. J. Comput. Vis.*, vol. 10, pp. 53–66, Feb. 1993.

[49] K. Bradshaw, I. Reid, and D. Murray, "The active recovery of 3D motion trajectories and their use in prediction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, pp. 219–234, Mar. 1997.

[50] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. Boston, MA: Academic, 1982.

[51] A. Lipton, "Local application of optic flow to analyze rigid versus nonrigid motion," in *IEEE Int. Workshop Frame-Rate Vision*, Corfu, Greece, Sept. 1999.

[52] ——, "Virtual postman—an illustrative example of virtual video," *Int. J. Robot. Automat.*, vol. 15, no. 1, pp. 9–16, 2000.

[53] B. Bhanu, Y. Q. Lin, G. Jones III, and J. Peng, "Adaptive target recognition," *Mach. Vis. Applicat.*, vol. 11, no. 6, pp. 289–299, 2000.

[54] B. G. Mobasseri, "Automatic target scoring system using machine vision," *Machine Vis. Applicat.*, vol. 8, no. 1, pp. 20–30, 1995.

[55] A. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving target detection and classification from real-time video," in *Proc. 1998 Workshop Applications of Computer Vision*, Princeton, NJ, Oct. 1998.

[56] D. P. Casasent and L. M. Neiberg, "Classifier and shift-invariant automatic target recognition neural networks," *Neural Netw.*, vol. 8, no. 7-8, pp. 1117–1129, 1995.

[57] G. Pasquariello, G. Satalino, V. Laforgia, and F. Spilotros, "Automatic target recognition for naval traffic control using neural networks," *Image Vis. Comput.*, vol. 16, pp. 67–73, Feb. 1998.

[58] K. Fukunaga, *Statistical Pattern Recognition*: Academic, 1989.

[59] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa, "A system for video surveillance and monitoring: VSAM final report," Robotics Inst., CMU-RI-TR-00-12, 2000.

[60] A. Baumberg and D. Hogg, "Generating spatiotemporal models from examples," *Image Vis. Comput.*, vol. 14, pp. 525–532, Aug. 1996.

[61] C. Bregler, "Learning and recognizing human dynamics in video sequences," in *Proc. 1997 Conf. Computer Vision and Pattern Recognition* , San Juan, Puerto Rico, June 1997, pp. 568–574.

[62] R. Cutler and L. S. Davis, "Robust real-time periodic motion detection, analysis and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 781–796, Aug. 2000.

[63] Y. Guo, G. Xu, and S. Tsuji, "Tracking human body motion based on a stick figure model," *J. Vis. Commun. Image Represent.*, vol. 5, pp. 1–9, 1994.

[64] D. Hogg, "Model-based vision: A program to see a walking person," *Image Vis. Comput.*, vol. 1, no. 1, pp. 5–20, 1983.

[65] Y. Ricquebourg and P. Bouthemy, "Real-time tracking of moving persons by exploring spatio-temporal image slices," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 797–808, Aug. 2000.

[66] K. Rohr, "Toward model-based recognition of human movements in image sequences," *Computer Vis., Graphics Image Process.*, vol. 59, pp. 94–115, Jan. 1994.

[67] S. Ju, M. Black, and Y. Yacoob, "Cardboard people: A parameterized model of articulated image motion," in *Proc. Int. Conf. Face and Gesture Analysis*, Killington, VT, Oct. 1996.

[68] J. K. Aggarwal and Q. Cai, "Human motion analysis: A review," *Comput. Vis. Image Understand.*, vol. 73, pp. 428–440, Mar. 1999.

[69] D. M. Gavrila, "The visual analysis of human movement: A survey," *Comput. Vis. Image Understand.*, vol. 73, pp. 82–98, Jan. 1999.

[70] H. Fujiyoshi and A. Lipton, "Real-time human motion analysis by image skeletonization," in *Proc. 1998 Workshop Applications of Computer Vision*, Princeton, NJ, Oct. 1998.

[71] G. Strang and K. Borre, *Linear Algebra, Geodesy and GPS*. Cambridge, MA: Wellesley-Cambridge, 1997.

[72] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson, "Advances in cooperative multi-sensor video surveillance," in *Proc. 1998 DARPA Image Understanding Workshop*, vol. 1, Monterey, CA, Nov. 1998, pp. 3–24.

[73] *Manual of Photogrammetry*, 4th ed., American Society of Photogrammetry, Falls Church, 1980.

[74] L. Lee, R. Romano, and G. Stein, "Monitoring activities from multiple video streams: Establishing a common coordinate frame," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 758–767, Aug. 2000.

[75] T. N. Tan, G. D. Sullivan, and K. D. Baker, "Recognizing objects on the ground-plane," *Image Vis. Comput.*, vol. 12, pp. 164–172, Apr. 1994.

[76] J. E. Bresenham, "An algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, 1965.

[77] J. M. Brady, "Special issue on sensor data fusion," *Int. J. Robot. Res.*, vol. 7, no. 6, pp. 1–161, 1989.

[78] P. K. Varshney, "Special issue on data fusion," *Proc. IEEE*, vol. 85, pp. 3–5, Jan. 1997.

[79] J. Orwell, P. Remagnino, and G. A. Jones, "Multi-camera color tracking," in *Proc. 2nd IEEE Int. Workshop Visual Surveillance*, Fort Collins, CO, June 1999.

[80] G. Healey, "Segmenting images using normalized color," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 64–73, Jan. 1992.

[81] J. L. Crowley and F. Ramparany, "Mathematical tools for representing uncertainty in perception," in *Proceedings of Spatial Reasoning and Multi-Sensor Fusion Workshop*. San Mateo, CA, 1987, pp. 293–302.

[82] P. K. Davis, "Distributed interactive simulation in the evolution of DoD warfare modeling and simulation," *Proc. IEEE*, vol. 83, pp. 1138–1155, Aug. 1995.
[83] J. E. Smith, "Recent developments in ModSAF terrain representation," in *Proc. 5th Conf. Computer Generated Forces and Behavioral Representation*, Orlando, FL, May 1995, pp. 375–381.
[84] A. J. Courtemanche and A. Ceranowicz, "ModSAF development status," in *Proc. 5th Conf. Computer Generated Forces and Behavioral Representation*, Orlando, FL, May 1995, pp. 3–13.
[85] M. D. Petty, "Computer generated forces in distributed interactive simulation," *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*, pp. 251–280, Apr. 1995.
[86] T. Boutell, *CGI Programming in C and Perl.* Reading, MA: Addison-Wesley , 1996.
[87] M. Brand and V. Kettnaker, "Discovery and segmentation of activities in video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 844–851, Aug. 2000.
[88] R. J. Howarth and H. Buxton, "Conceptual descriptions from monitoring and watching image sequences," *Image Vis. Comput.*, vol. 18, pp. 105–135, Jan. 2000.
[89] Y. A. Ivanov and A. F. Bobick, "Recognition of visual activities and interactions by stochastic parsing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 852–872, Aug. 2000.
[90] T. Wada and T. Matsuyama, "Multiobject behavior recognition by event driven selective attention method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 873–867, Aug. 2000.

Robert T. Collins received the Ph.D. degree in computer science from the University of Massachusetts (UMass), Amherst, in 1993 for work on scene reconstruction using stochastic projective geometry.

He is a Member of the Research Faculty at the Robotics Institute, Carnegie Mellon University (CMU), Pittsburgh, PA. From 1992 to 1996, he was technical director of the DARPA RADIUS project at UMass, culminating in the ASCENDER system for populating 3-D site models from multiple, oblique aerial views. From 1996 to 1999, he was technical co-director of the DARPA Video Surveillance and Monitoring (VSAM) project at CMU. This project developed real-time, automated video understanding algorithms that guide a network of active video sensors to monitor the activities of people and vehicles in a complex scene. He has published for over a decade on topics in video surveillance, 3-D site modeling, multi-image stereo, projective geometry, and knowledge-based scene understanding.

Alan J. Lipton received the Ph.D. degree in electrical and computer systems engineering from Monash University, Melbourne, Australia, in 1996. For his thesis, he studied the problem of mobile robot navigation by natural landmark recognition using on-board vision sensing.

He is currently with Diamondback Vision, Inc., Reston, VA. From 1997 to 2000, he was on the faculty at the Robotics Institute of Carnegie Mellon University, Pittsburgh, PA, where he was technical co-director of the DARPA Video Surveillance and Monitoring (VSAM) effort. Under this program, he performed research in the areas of real-time object detection, tracking, and recognition from video.

Hironobu Fujiyoshi received the Ph.D. degree in electrical engineering from Chubu University, Japan, in 1997. For his thesis, he developed a fingerprint verification method using spectrum analysis, which has been incorporated into a manufactured device sold by a Japanese security company.

He is a Member of Faculty at the Department of Computer Science, Chubu University. From 1997 to 2000, he was a post-doctoral fellow at the Robotics Institute of Carnegie Mellon University, Pittsburgh, PA, working on the DARPA Video Surveillance and Monitoring (VSAM) effort and the humanoid vision project for the Honda Humanoid Robot. He performs research in the areas of real-time object detection, tracking, and recognition from video.

Takeo Kanade (Fellow, IEEE) received the B.E. degree in electrical engineering in 1968, the M.E. degree in 1970, and the Ph.D. degree in 1973 from Kyoto University, Japan.

Currently, he is Helen Whitaker Professor of Computer Science at Carnegie Mellon University, Pittsburgh, PA.

Dr. Kanade was a recipient of the Robotic Industry Association, Joseph F. Engelberger Award in 1995, the Japan Robotics Association, JARA Award in 1997, the Yokogawa Prize at the International Conference on Multi Sensor Fusion and Integration for Intelligent Systems in 1997, the Hip Society, Otto AuFranc Award in 1998, the Hoso Bunka Kikin Foundation Award in 1994, and the Marr Prize at The Third International Conference on Computer Vision in December 1990. He was also selected as the author of one of the most influential papers that appeared in the *Artificial Intelligence* journal in the last ten years in 1992. He is Founding Chief Editor of the *International Journal of Computer Vision*. He is a Member of the National Academy of Engineering and a Fellow of the American Association for Artificial Intelligence (AAAI).