Using adaptive tracking to classify and monitor activities in a site*

W.E.L. Grimson C. Stauffer R. Romano L. Lee
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{welg, stauffer, romano, llee}@ai.mit.edu

Abstract

We describe a vision system that monitors activity in a site over extended periods of time. The system uses a distributed set of sensors to cover the site, and an adaptive tracker detects multiple moving objects in the sensors. Our hypothesis is that motion tracking is sufficient to support a range of computations about site activities. We demonstrate using the tracked motion data: to calibrate the distributed sensors, to construct rough site models, to classify detected objects, to learn common patterns of activity for different object classes, and to detect unusual activities.

1 A motivating scenario

Our goal is a vision system that monitors activity in a site over extended periods of time, i.e., patterns of motion and interaction demonstrated by objects in the site. The system should provide statistical descriptions of typical activity patterns, e.g., normal vehicular volume or normal pedestrian traffic paths for a given time of day; it should detect unusual events, by spotting activities that are very different from normal patterns, e.g., unusual volumes of traffic, or a specific movement very different from normal observation; and it should detect unusual interactions between objects, e.g., a person parking a car in front of a building, exiting the car, but not entering the building.

Because a site may be larger than can be observed by a single camera, our system observes activities with a "forest of sensors" distributed around the site. Each sensor unit is a compact packaging of camera, onboard computational power, local memory, communication capability and possibly locational instrumentation (e.g., GPS). Example systems exist [4, 5, 8], and more powerful systems will emerge as technology in sensor design, DSP processing, and communications evolves. We arbitrarily distribute many sensor units around the site, by attaching them to poles, trees, and buildings for outdoor sites[1], and to walls and furniture for indoor sites, such as the Intelligent Room[2].

The forest should learn patterns of activities in a site, then monitor and classify activities based on these learned patterns. For simplicity, we assume the existence of some basic sensor units, and focus on the processing needed to learn and monitor activities. A coordinated forest of sensors needs: *self-calibration* – determine the positions of all the cameras relative to one another; *construction of rough site models* – determine the ground plane, and mark occupied areas; robust *detection of objects* in the site and *classification of detected objects*; *learning from extended observation* (e.g. over a period of weeks) the common activity patterns; and *detection of unusual events* in the site.

Our governing hypothesis is that these tasks can be accomplished simply by observing moving objects. To verify this hypothesis, we need: a robust tracker that can reliably detect moving objects and return an accurate description of the observed object, both its motion parameters and its intrinsic parameters such as size and shape; and methods that can use such tracking data to accomplish the tasks listed above.

2 A robust adaptive tracker

In this section, we describe a novel tracking system[9], based on the standard notion of background subtraction. Simple implementations of backgrounding just subtract consecutive images and threshold the resulting difference image to determine pixels that may correspond to motion. More robust methods use time averages of images [2], adaptive Gaussian estimation[12], or Kalman filtering [7] to derive the background image to be subtracted.

While such methods often run in real time, they are generally not robust. They often only detect the leading and trailing edges of large objects, they are subject

*This report describes research supported in part by DARPA under ONR grant N00014-97-0363.

[1](see `http://www.ai.mit.edu/projects/darpa/vsam/`)
[2](see `http://www.ai.mit.edu/projects/hci/hci.html`)

to noise effects, and they are susceptible to small motion effects, e.g., branches rustling in the wind.

We propose a more robust detector that adapts to the observed scene. We consider each pixel as an independent statistical process, and record the observed intensity at each pixel over the previous n frames. This set of observed samples is then optimally fit with a mixture of K Gaussians. This reflects the expectation that samples of the same scene point are likely to display normal noise distributions, and the expectation that more than one process may be observed over time. This is in contrast to Pfinder[12] which fits a single Gaussian to a background pixel's history and requires a model of the moving object. Examples of observing multiple processes at a single pixel include: swaying tree branches, where a pixel sometimes observes the branch, sometimes the scene behind the branch; and rippling water which may reflect the sky, the horizon or simply appear the color of the water.

The probability that an observed pixel has intensity value $\mathbf{x}_t$ at time t is modeled as

$$Pr(\mathbf{x}_t) = \sum_{j=1}^{K} \frac{\omega_j}{(2\pi)^{\frac{d}{2}} |\mathbf{\Sigma}_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}_t - \mu_j)^T \mathbf{\Sigma}_j^{-1} (\mathbf{x}_t - \mu_j)} \quad (1)$$

where ω_j is the weight assigned to the jth distribution of that pixel's mixture model, μ_j is its mean and $\mathbf{\Sigma}_j$ is its covariance matrix. For simplicity we use $\mathbf{\Sigma}_j = \sigma_j^2 I$. We will now show how Equation 1 models the distribution of recent observations at a particular pixel as a mixture of Gaussians which is characterized entirely by the parameters $\omega_j, \mu_j, \mathbf{\Sigma}_j$.

To determine the background model, we consider each pixel individually. We order the K distributions on the basis of ω_j / σ_j^2 as distributions with larger such values are more likely to be stable background distributions. We then select the first B distributions that account for a predefined fraction of the evidence

$$B = \arg\min_b \left(\frac{\sum_{j=1}^{b} \omega_j}{\sum_{j=1}^{K} \omega_j} > T \right) \quad (2)$$

where T is an estimate of the fraction of the evidence which is produced by the background process. Thus a multi-modal distribution caused by repetitive background motion (leaves on a tree, a flag in the wind, a construction flasher) could result in several colors being included in the background model. The background model thus handles transparency effects by accepting multiple colors per pixel.

To update the model, every new pixel value, $\mathbf{x}_t$, is checked against the existing distributions, until a match is found. A match is defined as a pixel within 2 standard deviations of a distribution. If none of the model's distributions match the incoming value, the least probable distribution is removed and replaced with a distribution with that mean value, an initially high variance, and a low prior weight.

The distribution weights for each distribution at time t are adjusted by:

$$\omega_{j,t} = (1 - \alpha)\omega_{j,t-1} + \alpha(M_{j,t}) \quad (3)$$

where α is the learning rate and $M_{j,t}$ is 1 for the matched distribution and 0 for the remaining models. $1/\alpha$ defines the speed at which the distribution's parameters change. $\omega_{j,t}$ is a causal low-pass filtered (thresholded) posterior probability that the pixel values have matched model j given observations from time 1 through t. This is equivalent to the expectation of this value with an exponential window on past evidence.

The μ and σ parameters for unmatched distributions remain the same. The parameters of the matched distribution, j, are updated as follows

$$\mu_{j,t} = (1 - \rho)\mu_{j,t-1} + \rho \mathbf{x}_t \quad (4)$$

$$\sigma_{j,t}^2 = (1 - \rho)\sigma_{j,t-1}^2 + \rho(\mathbf{x}_t - \mu_{j,t})^T(\mathbf{x}_t - \mu_{j,t}) \quad (5)$$

where

$$\rho = \alpha * Pr(\mathbf{x}_t | \mu_{j,t-1}, \sigma_{j,t-1}) \quad (6)$$

which is the same type of causal low-pass filter as mentioned above, except that only the data which matches the model is included in the estimation. This allows both the mean and the variance to track slow illumination changes but leaves the background distributions uncorrupted when large changes occur.

Any pixel which is more than 2 standard deviations away from all of the background distributions is defined as part of a foreground moving object. These pixels are then clustered into connected components, and a multiple hypothesis tracker using linear predictive Kalman filters is used to determine moving component correspondence from frame to frame. It includes linear prediction in x, y and size parameters.

An example of one frame of the tracker is shown in Figure 1. Note that this process does not explicitly remove shadows, and a method such as that used in [2] could be included to handle this case.

For each tracked object, we can record a range of information for use in subsequent processes, including the size and shape of the object (as measured in image units), the location and speed of the object (as measured in image units) and the direction of motion of the object. Example traces of tracked objects[3] over a one hour time period are shown in Figure 2.

[3] Because color is not reproduced in these proceedings, we

Figure 1: *Tracker in action. Top left shows a current image. Top right shows the most probable background image (i.e. dominant Gaussian mean for each pixel's mixture model). Bottom left shows the connected components of the tracked objects. Bottom right shows the tracked objects, with velocity vectors overlaid.*

The following links provide a log of the tracking system running continuously over a period of several weeks. Hourly dumps of a sample image, and the track information for the past hour are provided: *http://www.ai.mit.edu/projects/vsam/* An example portion of this log is shown in Figure 8.

3 Using the tracker to calibrate

Since our goal is to build and maintain a global representation of the activity in a large, extended scene, it is essential to coordinate the individual video streams arriving from multiple sources, transforming multiple observations to a common coordinate frame. Expressing all local data in a common global frame will lay the groundwork for global activity understanding.

We require the system's self-calibration to be robust, unsupervised, and recover in real-time from changes in the camera configuration. The system must also operate in sites with potentially repetitive backgrounds, such as urban or outdoor environments. In such scenes, static feature detectors will find many false correspondences that must be pruned, making traditional calibration methods very expensive.

Our system uses the image locations of scene objects tracked simultaneously in overlapping images to build point correspondences between views in real time. Each camera independently tracks the motion of an object in its field of view and stores the image coordinates of its centroid. Each new correspondence between a pair of cameras provides an additional constraint for estimating the cameras' relative geometry.

encourage interested readers to visit our web site for examples and details *http://www.ai.mit.edu/projects/darpa/vsam/*

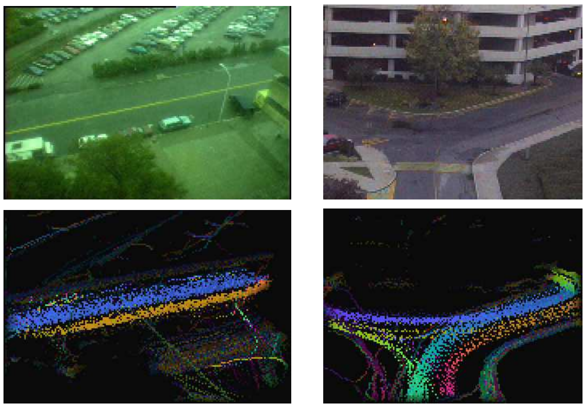

Figure 2: *Examples of tracking patterns. The top image shows the observed area, the bottom image shows the track patterns, (color encodes direction and intensity encodes speed). In each case, lanes of vehicle traffic and standard pedestrian paths are easily identified.*

We use a model of relative camera geometry appropriate for situations encountered in outdoor, urban monitoring applications. In such scenes, moving objects are typically cars or people moving on the ground plane, and therefore the objects' motion is often planar. Corresponding image points of tracked objects in a camera pair are then related by a projective linear transformation or homography [1].

We have tested the use of dynamic point correspondences for estimating the homographies between images of the scene's ground plane in a laboratory setting in which three cameras view a scene containing a single moving object. For each camera pair, a small but sufficient set of points is randomly sampled from a larger buffer of recent point correspondences and a least-squares solution for the homography is fitted to these sample correspondences. The newly computed estimate is compared to the current homography estimate for that camera pair by finding the mean squared error of both estimates on the full set of most recently buffered point correspondences. The model with the best fit is retained as the current best estimate.

For every pair of cameras with overlapping views, the ground plane homography is continuously updated, and the collection of homographies is used to warp incoming video streams from all camera views to a single reference point of view. The result is a composite video stream that displays in real time the global motion of the scene objects throughout the extended scene. This composite video displays the

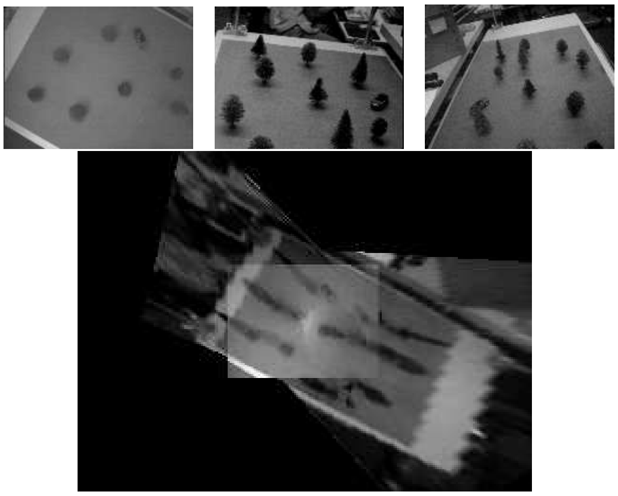

Figure 3: *Snapshot of a multiple camera video sequence. Above: A single frame from each of three raw input video sequences. Below: A single frame from the composite video sequence generated by warping each frame to a common coordinate system.*

tracked object's location from a single view even when the object is either occluded or not within the field view of the reference camera. In addition, areas of the extended scene not visible by any cameras in the system are clearly outlined. (See Figure 3.)

Enhancements to the system for handling observations of real scenes are complementary to enhancements to both the tracker of the previous section and the classifiers described in Section 5. The tracking system's adaptive background model makes it possible to continuously update the estimated camera geometry and recover from changes to the camera configuration. In addition, the multiple hypothesis tracker in conjunction with the object classifier may be used to eliminate false correspondences in camera pairs when there are multiple sources of motion. In turn, the mapping of the ground plane to a common coordinate system allows the activity patterns observed in each camera's image plane to be transformed to a common plane for the purpose of analyzing continuous global activity patterns throughout the extended scene.

4 Using the tracker for site modeling

4.1 Extracting a world coordinate frame

Given that we can calibrate a forest of sensors to a common image coordinate frame, we can coordinate observations from the forest. Now we need to relate this common camera coordinate frame to a world coordinate frame. We thus need to determine the pose of the ground plane relative to a camera.

Our hypothesis is that we can obtain the needed information by observing motion in the site. We assume that the site can be roughly modeled by a single ground plane. If we track an object moving through the site, we can use observed changes in its height to determine the ground plane parameters.

Assume that the origin of the coordinate frame is placed at the focal point of the camera, which we model as a pinhole device, with focal length f. Let the $\mathbf{z}$ axis lie along the optic axis, and let the base and top of the tracked object be represented by the image points $\mathbf{p}$ and $\mathbf{r}$ respectively at one time instant, and by $\mathbf{q}$ and $\mathbf{s}$ at a second time instant. If we assume that the observed object is oriented perpendicular to the ground plane (e.g., that people walk in an upright position), then the unit normal $\mathbf{n}$ of the ground plane must lie along the vector:

$$(\mathbf{r} \times \mathbf{p}) \times (\mathbf{q} \times \mathbf{s}).$$

Any point $\mathbf{P}$ lying on the ground plane must satisfy the constraint $\mathbf{P} \cdot \mathbf{n} = d$ where:

$$d = \frac{1}{fh}(\mathbf{p} \cdot \mathbf{n})\,[f(\mathbf{n} \cdot \mathbf{v}) + h + (\mathbf{n} \cdot \mathbf{z})(\mathbf{p} \cdot \mathbf{v})]\,H$$

where h is the height of the observed object in the image, H is the actual height of the object in the world, and $\mathbf{v}$ is a unit vector in the image between $\mathbf{p}$ and $\mathbf{r}$. Note that H is unknown, although we can provide an estimate for it, e.g., if the observed object is a person, we can approximate his/her height.

We can now normalize the observed height of any tracked object to account for projective foreshortening. If $\mathbf{t}$ is the image vector of the base of a new observed object, with image axis along the unit vector $\mathbf{u}$ and image height ℓ, then the corresponding world height of the object is

$$\frac{\ell(\mathbf{p} \cdot \mathbf{n})\,[f(\mathbf{n} \cdot \mathbf{v}) + h + (\mathbf{n} \cdot \mathbf{z})(\mathbf{p} \cdot \mathbf{v})]}{h(\mathbf{t} \cdot \mathbf{n})\,[f(\mathbf{n} \cdot \mathbf{u}) + \ell + (\mathbf{n} \cdot \mathbf{z})(\mathbf{t} \cdot \mathbf{u})]}H.$$

This provides us with normalized world measurements about an object that are useful in classifying objects.

4.2 Mapping out the site

Once we have an estimate of the ground plane, and a means for estimating the height of an object, we can use this information together with our tracked objects to determine a rough site model. In our approach [10], we initially consider the site, as viewed from the camera, as completely filled. Now suppose that we observe

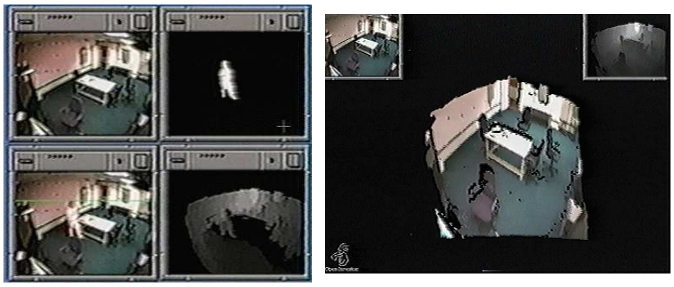

Figure 4: *Using a tracked object to create a rough site model. The left image shows a background image, an image containing a moving object, the extracted moving object and the current rough depth map, with intensity encoding distance. The right image shows the final result, with the background image, the depth map and the image texture mapped onto the depth map.*

a moving object, and that we compute the height of that object using the method above. By using our estimate of the ground plane and the computed height, we can estimate the distance to the object. This allows us to deduce that the portion of the site between the camera center and the observed object must be unoccluded, hence we can carve out that portion of the site model as being free space. As we continue to track this object, other portions of free space are swept out. Furthermore, when the object becomes occluded, this places a lower bound on the distance to the occluding portion of the site, and thus allows us to roughly block out portions of the space. Since we know the ground plane, we can place these obstacles in world coordinates on that plane. Figure 4 illustrates an example for indoor monitoring.

5 Using the tracker to classify

We can collect these pieces into a more complete system. We use the tracker to observe moving objects, recording in each frame a set of relevant parameters for each detected object, e.g., the position, direction of motion, velocity, size, height, aspect ratio of each connected region. We are currently running the system in real time on an SGI O2, processing 7 quarter frames a second. We regularly run our system nonstop for periods of several weeks, recording track patterns in each observed camera. Figure 8 illustrates hourly readouts of the track patterns for one camera.

5.1 Classifying objects

We can use these track patterns to classify common activity patterns. First, individual tracked objects can be classified into general classes, based on observed

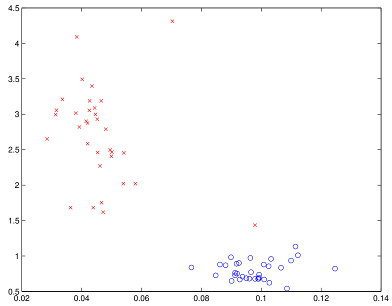

Figure 5: *Scatter plot of tracked objects, plotting mean aspect ratio (x axis) versus size (y axis). Circles are vehicles, crosses are pedestrians. A standard clustering algorithm can easily separate these two classes.*

data. For example, the aspect ratio of the tracked object can be used to identify cars, trucks, and people. This identification is further enhanced by the use of world size information, based on the calibration of the system. This allows us to label individual tracked objects, and to collect statistics about activity in the site, e.g., count the number of pedestrians or vehicles observed at different times of day.

In figure 5, a 10 minute segment was analyzed. Every object which entered this scene – in total, 33 cars and 34 people – was tracked. The system correctly classified every car except in one case, where it classified two cars as the same object because the two cars simultaneously entered and left the scene at the same point. It found only one person in two cases where two people where walking in physical contact. It also double counted 2 objects because they had ambiguous interactions for too long to maintain their identity.

5.2 Classifying actions

The tracks of moving objects can also be used to classify activities. By clustering the tracks on the basis of common attributes, we can automatically deduce lanes of vehicular traffic, and pedestrian paths, and we can automatically correlate volumes of such traffic with time of day. Our approach of using motion information to categorize activities is similar in spirit to [3], although we differ in several key details.

Once we have extracted clusters representing common patterns of activity, we can cue our system to look for unusual events. These are outliers in the clustered distributions. An example of such an event is illustrated in Figure 2, where a truck has recently crossed a pedestrian lane to reach a loading dock. It's track is easily identified in the lower right corner as an outlier compared to normal activity patterns in this area, and thus is marked for investigation. Other examples include a vehicle moving at an unusual speed given normal patterns in the area, or volumes of pedestrian or vehicle traffic that are unusual compared to normal rates for the corresponding time of day.

We have considered two approaches for classifying actions. In the first method, we cluster the tracker output using an entropy minimization algorithm by Wallace[11], the numeric iterative hierarchical cluster (NIHC) algorithm. The NIHC algorithm starts with randomly assigning data to clusters in a B-tree structure (a binary tree in which all internal nodes have two children) and iteratively reduces the total Gaussian entropy of the tree. At each iteration, the algorithm greedily moves subtrees that will result in the largest decrease in the sum of the entropy of the subtrees. The iteration stops when there is no single move that can further reduce the entropy (although a sequence of two or more moves may still reduce the entropy, those are not considered). The resulting tree is a partially optimal hierarchical clustering of the data. An MDL (minimum description length) cut is then taken on the tree to find the level of clusters that best describe the data. The description length is a function of the entropies of all clusters at the cut plus the cost of representing that cut. Thus description length is rewarded by taking a cut with smaller clusters because smaller clusters have smaller entropy, but penalized for taking such a cut because the representation of the cut is longer.

We apply a modified version of Wallace's algorithm to the output of the adaptive tracker. Figure 6 is an example of the clustering result of the scene in that figure. We collected 3 minutes of track data from video. The data set used has 6 dimensions: x and y location, log of the size of moving object, speed of the object, and unit vector of the direction of motion. An (x, y) projection of all the clusters is shown in Figure 6.

Given the cluster descriptions, we can detect particular activities by specifying parameters that are characteristic of those activities. For example, to find a group of people in queue, we can look for clusters that contain objects that are slow moving and have weak directionality in the motion. The lower right image of Figure 6 shows all the clusters with the aforementioned characteristics, which correspond to the line of people.

The second method of classifying action involves first quantizing the six-dimensional continuous observations, $\mathbf{o}_t \in \mathbb{R}^6$ (x,y,dx,dy,size,aspect-ratio), which describe the state of the objects being tracked. This reduction in the state space is accomplished by overfitting with a large number of Gaussians, each represent-

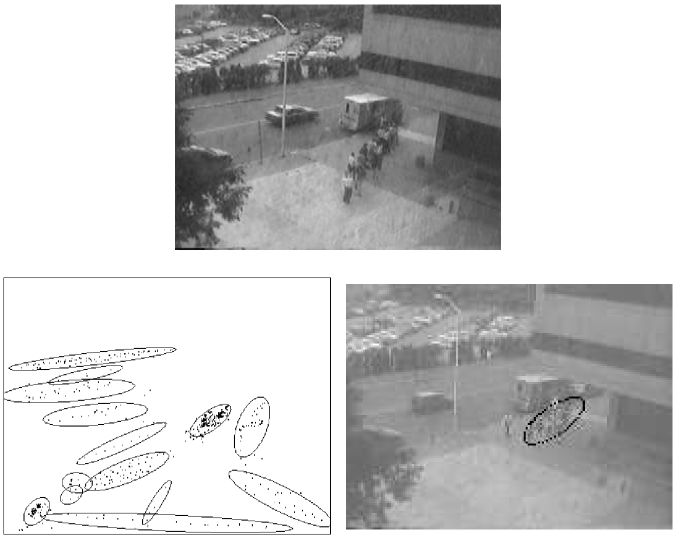

Figure 6: *Given track data for the shown scene, our method finds the best clusters based on all parameters. Lower left shows the (x, y) projection of those clusters. If we specify particular parameters, e.g, low speed, weak directionality, we can extract specific activities, e.g. people in line, as shown in the lower right image.*

ing a small region of the (x,y,dx,dy,size,aspect-ratio) state space. These representative state approximators for the data are found by a K-means approximation. Once the space is quantized, a sequence of observables resulting from an object being tracked, $\mathbf{o}_1, ..., \mathbf{o}_T$ where $\{\mathbf{o}_t \in \mathbb{R}^d\}$, can be represented by a sequence of labels corresponding to the discrete states in that path, $x_1, ..., x_T \in \{l_1, ..., l_N\}$. Vector quantization can also be used[6] to reduce the continuous state space to a discrete space.

We calculate accumulated co-occurrence statistics of the labels over all sequences using each sequence of labels as an equivalence class. This results in an N by N matrix, $\mathbf{C}$:

$$C_{i,j} = \frac{1}{S} \sum_{s=1}^{S} \left(\sum_{t=1}^{T_s} \frac{(x_t = l_i)}{T_s} * \sum_{t=1}^{T_s} \frac{(x_t = l_j)}{T_s} \right) \quad (7)$$

where S is the number of sequences available and T_s is the length of the sequence s.[4] $C_{i,j}$ is the probability that the states l_i and l_j will co-occur in the same sequence. If an object tends to present as both l_i and l_j, the corresponding probability will generally be higher. If no object in the database presents as both l_i and l_j, the corresponding probability will be zero. We use this measure of similarity to cluster the

[4]Note: $(x_t = l_i)$ can be replaced with $Pr(l_i|x_t)$.

state approximators rather than the established methods of determining their proximity in the state space.

We are currently investigating many different methods of abstracting the underlying processes using the co-occurrence statistics. Given two simplifying assumptions—that the underlying processes are IID (independent and identically distributed) across the labels and that the sequences are the same length, T—$C_{i,j}$ approaches a biased estimation of the weighted joint probability density, as the number of sequences increases

$$\lim_{S\to\infty} C_{i,j} = \sum_{k=1}^{K} \pi_k * \left((\frac{T-1}{T}) p_k(i) * p_k(j)^T + (\frac{\delta_{i,j}}{T}) p_k(i) \right) \tag{8}$$

where π_k is the prior for the kth process and $p_k(i)$ is the probability of the kth process producing the label i ($\Pr(l_i|Model_k)$). Note that as the length of the sequence increases, the bias disappears, leaving simply

$$\lim_{T\to\infty} C_{i,j} = \sum_{k=1}^{K} \pi_k * (p_k(i) * p_k(j)) \tag{9}$$

in which case, $\mathbf{C}$ is the weighted sum of the joint probabilities of all the underlying distributions. When the underlying distributions are significantly independent, the underlying processes could be solved for by direct methods or be found by using an EM least squares approximation of the co-occurrence matrix from the underlying component densities.

Unfortunately, in our case, neither of the above assumptions are completely valid, so it is not possible to determine the exact underlying processes. Rather than clustering the sequences into N different activity clusters, as in [6], we determine a compact, hierarchical representation. Our goal is to determine relatively independent sets of state approximators at each level of our representation.

This problem fits rather well into a formalism of graph bi-partitioning. Beginning with the complete set of state approximators corresponding to a universal event, we recursively partition the set minimizing the cut co-occurrence probabilities. This corresponds to dividing the parent class into two classes such that labels which often occur in the same sequence tend to remain in the same class. To find a minimal partition, we use a Hopfield network to find a minimum of the function

$$E = -\sum_{i,j} \left(C_{i,j} S_i S_j + \lambda \left(\sum_i S_i \right)^2 \right) \tag{10}$$

where $S_i = 1$ if l_i is part of the left child and $S_i = -1$ if l_i is part of the right child. This is implemented as a standard Hopfield network. Figure 7 shows the hierarchical classification using one hour of tracking from a particular scene. Each state's approximator is shown as a box with a line, where the size of the box and direction and length of the line represent the relative size, direction, and speed of the objects it represents. The top of the pyramid corresponds to the universal event. Evaluating a particular sequence on the universal event gives a measure of the typicality of the event. The left branch corresponds to eastbound traffic and its children correspond to faster/slower and larger/smaller subsets of the states. Other nodes represent pedestrian traffic, westbound traffic, eastbound pedestrians and westbound pedestrians.

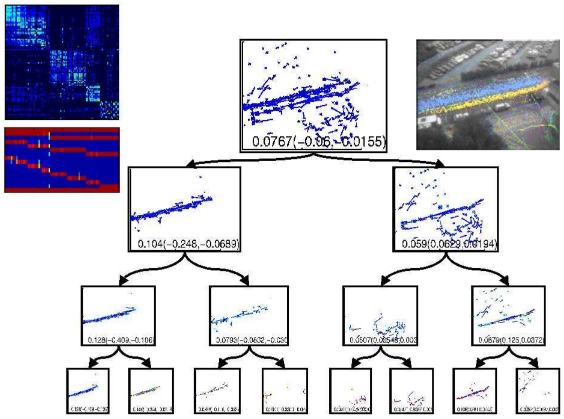

Figure 7: *Preliminary clustering results. Upper left: the co-occurrence matrix(KxK) and the membership function(15xK). Upper right: the corresponding scene. Middle: the hierarchical clustering of the state approximators (see text).*

We plan to investigate other partitioning functions (e.g. not binary, not exclusive), investigate using this mechanism to do other classification tasks(e.g. visual classification), investigate using this method to integrate multiple classification systems(e.g. behavior, shape, color classifiers), use this method to create a minimally supervised classifier, and determine how to make these classifiers generalize to novel situations.

We plan to investigate using this mechanism for performing other classification tasks (e.g., visual classification), integrating multiple classification systems (e.g., behavior, shape, and color classifiers), and creating a minimally supervised classifier. In addition, we will examine other partition functions (e.g. not binary, not exclusive) and explore how to make these classifiers generalize to novel situations.

6 Summary

Our hypothesis is that robust, adaptive, multiobject tracking can serve as the basis for systems that detect and classify activities in extended sites. Although the work reported here is still in progress, we have already demonstrated the use of such tracking information in multicamera calibration, rough site modeling, object detection, object classification and activity detection and classification.

References

[1] Faugeras, O.D., *Three-Dimensional Computer Vision: A Geometric Viewpoint,* The MIT Press, 1993.

[2] Friedman, N., and S. Russell, *Image segmentation in video sequences: A probabilistic approach* Proc. Uncertainty in Artificial Intelligence, 1997.

[3] Johnson, N., and, D. C., *Learning the distribution of object trajectories for event recognition* in: Pycock, D (editors) British Machine Vision Conference 1995, vol.2, pp.583-592. BMVA, 1995.

[4] Horswill, I. and M. Yamamoto, *A $1000 Active Stereo Vision System,* Proc., IEEE/IAP Workshop on Visual Behaviors, Seattle, August 1994.

[5] Horswill, I. *Visual routines and visual search: A real-time implementation and automata-theoretic analysis,* Proc. IJCAI, 1995.

[6] Johnson, N. and D. Hogg. *Learning the distribution of object trajectories for even recognition,* Image and Computing, vol. 14(8), August 1996, pp. 609-615.

[7] Koller, D., J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao and S. Russell, *Towards robust automatic traffic scene analysis in real-time* Proc. ICPR, Israel, 1994.

[8] Konolige, K., *Small vision systems: Hardware and Implementation,* Eighth International Symposium on Robotics Research, Hayama, Japan, October 1997.

[9] Stauffer, C., *Adaptive background mixture models*, in preparation.

[10] Stauffer, C., *Scene Reconstruction Using Accumluated Line-of-Sight,* SM Thesis, MIT, June 1997.

[11] Wallace, R., *Finding Natural Clusters through Entropy Minimization* PhD thesis, CMU, CMU-CS-89-183, 1989.

[12] Wren, C., A. Azarbayejani, T. Darrell, A. Pentland. *Pfinder: Real-Time Tracking of the Human Body,* IEEE Trans. PAMI, **19**(7):780–785, July 1997.

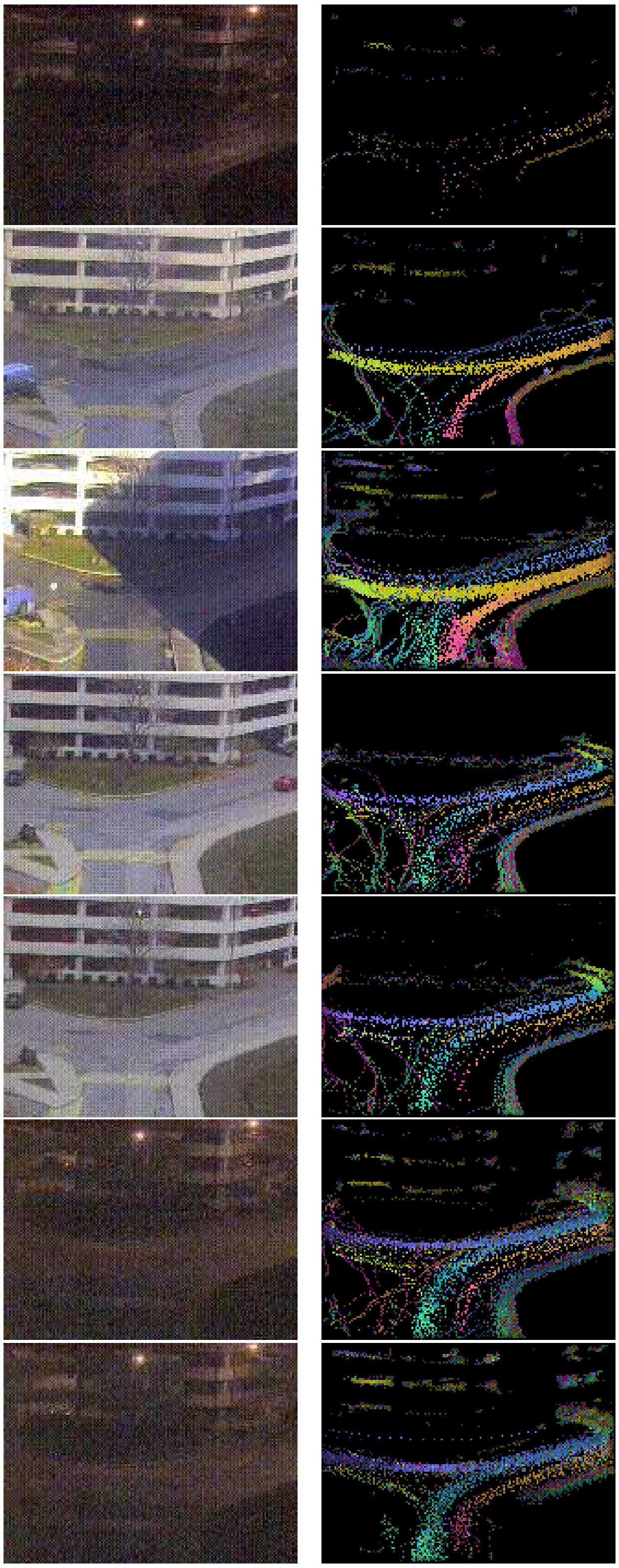

Figure 8: *Selected example of tracking patterns over extended period of time. Each row shows an example image and the track patterns for the previous hour, with times ranging from 7am to 4pm.*