Unsupervised Learning of Functional Categories in Video Scenes*

Matthew W. Turek, Anthony Hoogs, and Roderic Collins

Kitware, Inc., Clifton Park, N.Y. U.S.A.
{matt.turek,anthony.hoogs,roddy.collins}@kitware.com
http://www.kitware.com

Abstract. Existing methods for video scene analysis are primarily concerned with learning motion patterns or models for anomaly detection. We present a novel form of video scene analysis where scene element categories such as roads, parking areas, sidewalks and entrances, can be segmented and categorized based on the behaviors of moving objects in and around them. We view the problem from the perspective of categorical object recognition, and present an approach for unsupervised learning of *functional* scene element categories. Our approach identifies functional regions with similar behaviors in the same scene and/or across scenes, by clustering histograms based on a trajectory-level, behavioral codebook. Experiments are conducted on two outdoor webcam video scenes with low frame rates and poor quality. Unsupervised classification results are presented for each scene independently, and also jointly where models learned on one scene are applied to the other.

Keywords: functional modeling, unsupervised learning, video analysis.

1 Introduction

We present a new approach to video scene modeling and recognition that is based on unsupervised, location-independent segmentation of scene element categories. Existing work in video scene modeling has largely focused on segmenting dominant motion patterns [1,2,3,4,5] and significant regions such as track sources and sinks [1,6], given observed trajectories and detection algorithms for each scene element type. Here, we view the problem from the perspective of categorical object recognition, and present an approach for unsupervised learning and modeling of *functional* scene element categories – entities that are defined primarily by their behavior and/or surrounding activity, rather than their appearance or shape. Typical functional scene elements include vehicular traffic lanes, sidewalks, parking spaces, crosswalks, building entrances/exits, benches, bus stops, and so on. Many of these functional scene elements can not be distinguished

* This material is based upon work supported by the Defense Advanced Research Projects Agency under prime contract HR0011-06-C-0069, subcontract 070861. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

K. Daniilidis, P. Maragos, N. Paragios (Eds.): ECCV 2010, Part II, LNCS 6312, pp. 664–677, 2010.

by appearance alone. For instance, parking spaces and traffic lanes often appear nearly identical. However, for applications like normalcy modeling and abnormal event detection, it may be important to to understand the function of a region. Functional scene elements can also help identify the functional behavior of moving objects [16]. Identification of functional scene regions is important, then, as an enabling technology for event detection and normalcy modeling.

Inspired by the bag-of-words paradigm for object recognition, our approach identifies regions with similar behaviors in the same scene and/or across scenes, by clustering histograms based on a trajectory-level, behavioral codebook. The regions do not need to be spatially contiguous; rather, we seek regions that are spatially disjoint but have similar functional properties. A cluster of such regions (in feature space, not scene coordinates) corresponds to a functional category that can be assigned a conceptual label.

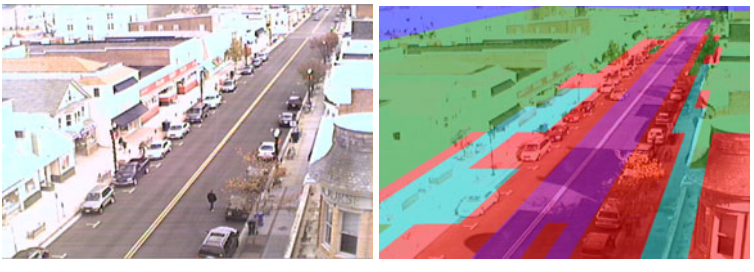


Fig. 1. Left: A typical scene from an outdoor webcam (Ocean City, NJ). **Right:** The result of unsupervised scene classification. Regions with the same color are determined to have the same functional type.

Functional object recognition was pioneered some time ago with work on recognizing chairs in static images [7]. Recently, work has appeared on integrating observed function in video with appearance for object recognition [8,9]. Maintaining distributions of track-level information at scene locations has been studied previously [3,10]. Codebooks of local kinematic and object features have been used to model motion patterns and detect unusual activities [4]. Swears and Hoogs [15] presented methods to detect specific scene elements which relied on hand-crafted detectors. Our work differs in that we do not attempt to segment the various motion patterns in a scene from each other, or to develop detection algorithms specific to any scene element category; instead we learn and classify functional regions. In addition, we learn models that are transportable across scenes.

Another recent technique is [11] where optical flow based features are combined with multi-scale analysis to learn motion patterns. The reliance on optical flow patterns adds robustness to tracking. Our motivating datasets are webcams whose low-frame rates (approx. 1-2 Hz.) provide large displacements with very sparse temporal sampling for the computation of optical flow features. Instead,

we incorporate tracking robustness by using a hierarchy of features, including detection, track, track fragment, and multi-track information.

At a high level, our method consists of: 1) developing a common, hierarchical feature-space representation for all behavioral scene element categories; 2) unsupervised learning of behavioral category models which are independent of scene location and transportable across scenes; 3) segmenting video scenes into the functional categories. To our knowledge, this has not been done previously. Our method can represent a variety of behavior-based scene elements in urban, outdoor scenes. The techniques can be used to learn a generic set of functional element categories across a variety of scenes, or to learn the specific element types present in one scene.

As mentioned previously, we operate on webcams, which are highly challenging due to poor quality and very low frame rates. An example result is shown in Figure 1, computed from 8 hours of poor quality, 1Hz video from a webcam. The camera was (manually) calibrated, and the scene was partitioned into a regular grid in the ground plane. Because of the low frame rate, banded noise and compression artifacts, detection and tracking were particularly noisy and error-prone. Nevertheless, grid cells with similar behaviors were successfully clustered to yield functional categories such as vehicular traffic lane, pedestrian traffic lane (sidewalk), parking spot, and building entrance.

The method is applicable to generic surveillance cameras as well as webcams. The latter introduces significant challenges because of poor tracking, but these are addressed through statistical methods and the hierarchical feature set as described below. Calibration to a ground plane is useful if not required; automatic calibration in video has been studied and is beyond the scope of this work, but will be utilized in future work.

The method has a few key limitations. Currently, it is assumed that a single grid cell or region is indicative of its functional type. This effectively bounds the minimum size of functional regions (spatial resolution of the grid), since a small portion of a scene element may have insufficient information. Similarly, cell neighborhoods are not directly considered during clustering, although some of the features weakly associate nearby cells. Another drawback is that temporal state transitions are not modeled explicitly. Scene elements with multiple behaviors at different times, such as intersections, are represented multiple modes in the codebook histogram. These shortcomings will be addressed in future work.

Our approach is described in Section 2. Section 3 presents our results and we conclude in Section 4.

2 Modeling Approach

Our approach adapts the bag-of-words concept to trajectory-level behavioral analysis. An overview of the method is shown in Figure 2. First, on each video scene, tracks are computed for a period of time that is sufficient to capture the range of activity in the scene (typically a few hours or a day depending on event density and scene complexity). We assume that the cameras are roughly

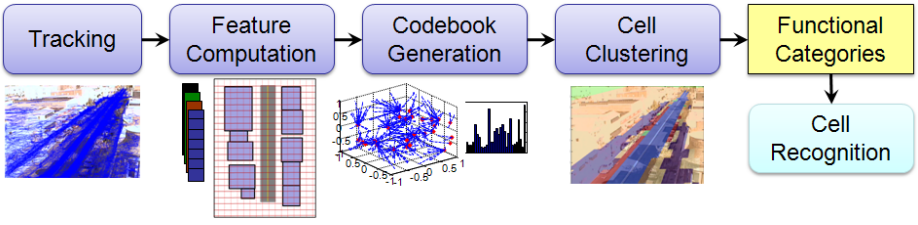


Fig. 2. The overall approach. Features describing local behaviors are computed for each track interval through each cell, and used to generate a codebook. Each cell is described by a codebook histogram. The histograms are clustered to form functional element categories, which are used for recognition.

calibrated to the ground plane, so that ground-plane tracking and normalization may be performed. Each video scene is partitioned into a set of regions, such as a regular spatial grid in the ground plane.

Next, a descriptive set of *behavioral features* is computed. For each track, and for each grid cell that the track intersects, detection and track-level features capture single-object, local, behavioral and object characteristics. Within each cell, the these features are accumulated to form feature distributions for the cell. In addition, cell-level features capture the relationship between cells traversed by the same track, and localized relationships between tracks over time. In total, the feature set characterizes local behaviors in the same way that patch descriptors characterize local appearance for object recognition.

For the scene shown in Figure 1, tens of thousands of feature vectors are computed from 8 hours of video. These feature vectors are clustered using a method such as mean-shift or K-means to form a codebook of a size that is comparable to the number of cells. For each cell, a codebook histogram is formed by finding the closest centroid for each feature vector in the cell.

For each scene, the cell histograms are clustered using mean-shift [12] on the L_2 histogram distance. Each cluster corresponds to a set of cells with a common local behavior pattern, i.e. the same functional category.

At this point each scene is segmented into functional types by cluster index, but the types are not textually labeled. This labeling is simply the assignment of a string name to each cluster index, which is done manually with little effort.

On a new video scene, the clusters can be used to perform recognition of functional elements. As in learning, the new scene is spatially partitioned and tracked, and codebook histograms are computed for each cell. Each cell is then recognized by initializing mean-shift with the cell histogram, and outputting the cluster found by mean-shift.

The following subsections describe these steps in more detail.

2.1 Moving Object Detection and Tracking

We used a standard background subtraction algorithm [13] to detect moving objects and then used two simple data association trackers (one tuned for pedestrians

and the other for vehicles) to link these objects into tracks. The data association-based tracking is similar to [14]. These algorithms are completely automatic. A rough projective camera is computed by hand, once per scene. This projective camera is used in the tracking algorithms to help estimate object ground-plane position, size and velocity.

2.2 Behavioral Feature Set Computation

We have developed a hierarchical feature set to provide robustness to tracking difficulties and to capture multiple levels of behavior detail. Our features include detection-based features, track-level features, and cell-level features.

Table 1. Detection level features, $\boldsymbol{d}_{j,k}$, for track j in cell k; track level features, $\boldsymbol{f}_{j,k}$ for for track j in cell k; and cell level features for cell k

	Element	Feature
$\boldsymbol{d}_{j,k}$	0	Median speed
	1	Median change in speed
	2	Median change in angle
	3	Median size
	4	Median detection aspect ratio
$\boldsymbol{f}_{j,k}$	0	Track length
$\boldsymbol{c}_k$	0	Number of track starts in cell
	1	Number of track stops in cell
	2	Number of tracks through cell
	3	Entropy of outgoing heading distribution
	4	Entropy of incoming heading distribution

Our detection, track-level, and cell-level features are listed in Table 1. The first block in the table contains the detection-based features, the second block contains the track-level features, and the third block contains cell-level features. Detection-based features incorporate information based solely on moving object detections that are within a particular cell. All the detections for a particular track are combined into one feature vector, using summary statistics. Typically, we use the median of the feature for the track samples (corresponding to a particular track) within a cell. Track level features $\boldsymbol{f}_{j,k}$ are computed for each track passing through a cell. Finally, cell level features $\boldsymbol{c}_k$ are computed across all tracks that pass through a cell.

The entropy of the heading distribution for incoming (outgoing) tracks is computed as in Equation 1, with the distribution taken over the heading of all start (stop) detections in a cell. Note that the entropy measure is independent of particular orientations, thus allowing this feature to be built into a codebook on one scene and then applied on a different scene.

$$E(\tau) = -\sum_i p(\theta_i)\ log\ p(\theta_i) \tag{1}$$

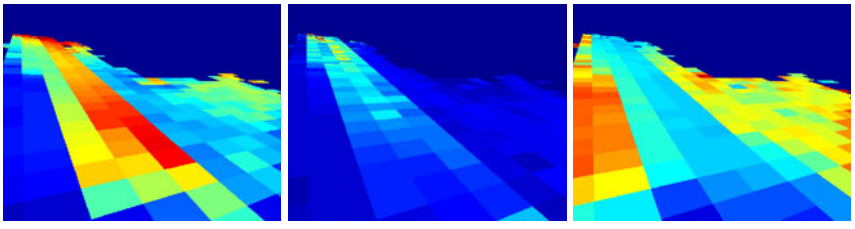


Fig. 3. Example features for the Ware scene. Size (left), speed (middle), aspect ratio (right).

where $p(\theta_i)$ is the probability of the heading of the i^{th} detection on track τ. We use a histogram of the headings along a track to model the heading probability.

It is important to choose features judiciously, as additional, non-informative, features can clutter the feature space and effectively add a noise term to feature distances which leads to misclassifications.

We create an ensemble feature vector $\boldsymbol{x}_{j,k}$ for each track j and cell k as:

$$\boldsymbol{x}_{j,k} = \begin{pmatrix} \boldsymbol{d}_{j,k} \\ \boldsymbol{f}_{j,k} \\ w_c \boldsymbol{c}_k \end{pmatrix} \tag{2}$$

where w_c is a weighting for cell features $\boldsymbol{c}_k$. The collection of all feature vectors $\mathcal{F}$ in a video sequence is:

$$\mathcal{F} = \{\boldsymbol{x}_{0,0}, \ldots, \boldsymbol{x}_{j,k}, \ldots, \boldsymbol{x}_{n,m}\} \tag{3}$$

for n tracks and m cells. One issue arises in simultaneously handling the detection, track, and cell-level features: there are more track level feature instances (one per track crossing a cell) than cell level features (one per cell). We handle this discrepancy by downweighting the influence of the cell features (through w_c). This allows us to build a single code book on a feature space including both track and cell feature dimensions. Currently we set $w_c = 1/n_{celltracks}$ where $n_{celltracks}$ is the number of tracks in a cell.

There are a few alternatives to the approach we have taken for combining the hierarchy of features. One alternative is to summarize all the track features into one feature vector per cell. This would inevitably lead to a reduction in information and cells with a multi-modal distribution of features would end up with a blended feature vector. On the detection level, we have summarized all the detections for one track in one cell into a single feature vector. Since the cells are relatively small and there are typically few detections for one track within a cell (< 3 is typical), this summarization is less problematic. However, across perhaps hundreds of tracks that pass through a cell, summarization of track level features would discard significant information. Another alternative solution would be to maintain two feature spaces, and create a codebook for each. One significant disadvantage of this approach, however, is that the centroids (Sec. 2.3) cannot capture joint information between the spaces.

2.3 Feature Codebook Generation

The use of codebooks or "visual words" has become immensely popular in visual recognition tasks, because they are an effective way of compactly representing high-dimensional, complex feature spaces. We apply them here because the behavioral feature space can be complex, but should also contain high-density areas corresponding to common activities. Stauffer and Grimson previously applied a codebook to activity analysis [4], where position was included and tracks were represented by sets of centroid labels. Here, we explicitly avoid dependence on absolute features such as position and heading, as our goal is to learn behavioral categories independent of locations or location-specific behaviors. Also, our codebook histogram is defined on cells, not tracks, which enables our hierarchical feature set.

All ensemble feature vectors in the video scenes, denoted $\mathcal{F}$, are used to create the codebook. Before clustering, each feature is normalized by its standard deviation across the observed data. We generate the codebook by clustering the ensemble features. We have experimented with both K-means and mean-shift and have found mean-shift to be considerably more stable than K-means with random initialization. Mean shift also has the advantage that K does not need to be specified, although the bandwidth parameter does impact performance. The mean shift for feature $\boldsymbol{x}_i$ with bandwidth parameter h and kernel $g()$ is defined as [12]:

$$\boldsymbol{m}_{h,G}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} \boldsymbol{x}_i g\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right\|^2\right)}. \tag{4}$$

We typically use two iterations of mean shift on the set of features $\mathcal{F}$ to generate representative features. An Epanechnikov kernel is used to represent the underlying distribution, yielding a mean-shift implementation with a uniform kernel.

2.4 Cell Histogram Representation and Clustering

Once the codebook is defined, a codebook histogram is created for each cell. For each integrated feature vector in a cell, the closest codebook centroid is identified, and the corresponding bin in the histogram is incremented. To assure statistical significance, we ignore cells with too few feature vectors (< 20).

The final segmentation step is clustering cells that have similar histograms. We use mean-shift [12] for clustering, as we do not want to specify the number of clusters a priori, and we expect the clusters to be non-Gaussian and generally noisy. Spatial position of the cells is not considered, so that we can group cells with similar behaviors located in different parts of the same scene or in different scenes altogether. As mentioned above, we currently do not use cell neighborhood information in the clustering process, although some features are computed across neighboring cells.

Ideally, each of the resulting clusters would correspond to a single functional category such as road or parking spot, and each functional category would have

exactly one cluster. In practice, any of the categories may have a multi-modal feature distribution, in which case mean-shift should create multiple modes for one category. For recognition, all significant modes should be assigned the same label.

Generally, we expect that many scene elements will have non-trivial distributions of behavioral features. For simple categories, such as vehicular or pedestrian traffic lanes, it is strightforward to compute low-level features on individual tracks, and perform clustering on the raw features. For these cases, our use of cell-level features and codebook histogram clustering is perhaps more powerful (and more complex) than required.

Many other scene categories, however, have more complex and variable behaviors. Building entrances and exits, parking spots and crosswalks, for example, may exhibit a variety of behaviors even at the same locations. Our representation can support this, as long as the mode patterns are similar across the class. For one activity pattern in a multi-modal cell, the tracks in that pattern will give rise to feature vectors in histogram bin h_i. For a second activity pattern in the same cell, the tracks in that pattern will give rise to feature vectors in histogram bin h_j. The cell histogram will have two modes, and should be clustered with other cells that have the same two modes. This situation should arise for a cell outside a doorway, for example, if some people walk straight past the doorway (bin h_i) and others enter (bin h_j).

2.5 Scene Element Recognition

Once the clusters are formed, scene element recognition can be performed on a new video scene, or new parts of an existing scene. The spatial scale and partitioning of the new scene should be comparable to those used in training, and the scene should have comparable behaviors for the same scene categories (this may not be true across different regions of the world).

Each cell is recognized independently. Cell features are computed from its tracks as in training, and the cell histogram is created using the prior codebook. Each cluster is a collection of codebook histograms, and we label cells using mean-shift on the cell histogram, and outputting the cluster found by mean-shift.

3 Results

Experiments were conducted on many hours of video data, from two public webcams, with over 2500 tracks in each scene. Scene element learning, segmentation and recognition results are shown on each scene independently, and between scenes by learning on one and testing on the other.

3.1 Data

One webcam is in Ocean City, NJ, shown in Figure 1. Approximately 8 hours of data from one day was used. The frame rate is about 1-2 Hz, and the image

Fig. 4. The size of people in the Ocean City video. In the near-field, people are about 30 pixels in height; in the mid-field, about 16 pixels, and in the far-field, about 10 pixels. Compression artifacts are noticeable, particularly in the far-field.

size is 704×480. To provide a sense of scale and image quality, Figure 4 shows crops of a person in the scene.

The second scene is in Ware, UK. The frame rate is also ≈ 2 Hz, and the image quality is somewhat better than Ocean City (OC). The near-field has higher resolution, as the camera is mounted closer to the ground. The far-field resolution is comparable to OC. Shown in Figure 5, the scenes have a number of functional entities in common.

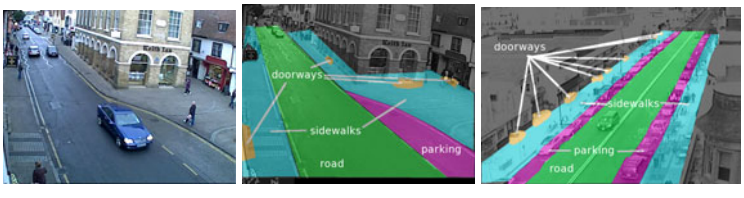


Fig. 5. The video scenes used in the experiments. Left is the Ware scene. Manually-generated ground truth labeling for Ware (resp. Ocean City) is the middle (resp. right). Ground-truth labeling is used for evaluation of results only.

To evaluate the algorithm, we manually created ground-truth labels for the roads, sidewalks, parking areas, and building entrances/exits as shown in the figure. These labels were not used by the algorithm in any way; they were used only for evaluation of the results.

3.2 Tracking

We used a background subtraction algorithm [13] to detect moving objects and a simple data association tracker to link these objects into tracks. The algorithms are completely automatic. We did not use any scene specific information to improve the moving object detection or tracking, except to use an approximate projective camera to compensate for the change in object size from the far-field to the near field.

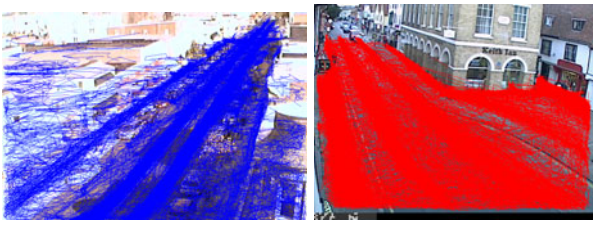


Fig. 6. Computed tracks on the Ocean City (left) and Ware (right) scenes

Figure 6 shows the computed tracks used in the experiments for both scenes. There are more than 2500 tracks in each scene, and the dominant motion patterns are clearly evident. Recall, however, that we do not attempt to segment these motion patterns, but rather to group all of the cells into a set of functional categories. Hence our desired result would match the ground truth in Figure 5, which does not separate the two lanes of the road.

While the tracking results are good overall, there is significant track fragmentation as pedestrians disappear under overhangs, signs and trees, or seem to disappear because of saturation effects. There are also a number of false tracks and track switches caused by false alarms in moving object detections. These are particularly evident near times of global lighting changes, because the false alarm rate rises until the background model catches up.

Our approach is quite robust against tracking errors because of its statistical nature. Many tracks are considered at each cell (≥ 20), and more video can be added as necessary.

3.3 Unsupervised Scene Segmentation and Classification

We conducted an initial experiment to evaluate the need for hierarchical features. We used the detection level features, denoted $\boldsymbol{d}_{i,j}$ in Table 1 as the only feature set, and proceeded to run the remainder of the algorithm, including code-book generation and cell clustering. Two representative results for the Ware scene are provided in Figure 7. (The ground truth for Ware is the middle image in Figure 5.) The detection level features are able to discriminate pedestrian/vehicle areas from the background. However, there is little ability to discriminate between the pedestrian areas and the roadway. In addition, the segmentation results are quite noisy.

To characterize the performance of the full system, including the full, hierarchical feature set, we conducted further experiments. We began by running experiments on the two web-cam datasets. In each case, a codebook was built on the scene and then that codebook was used to classify the cells in the scene.

On each scene, the system produces cluster IDs corresponding to scene element types, which are then scored against the ground-truth. In order to do this scoring, semantic labels must be assigned to the clusters. There is no clear-cut method

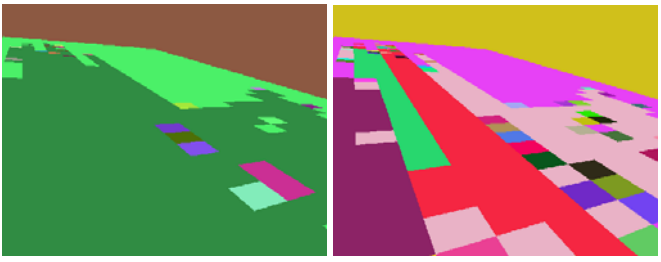
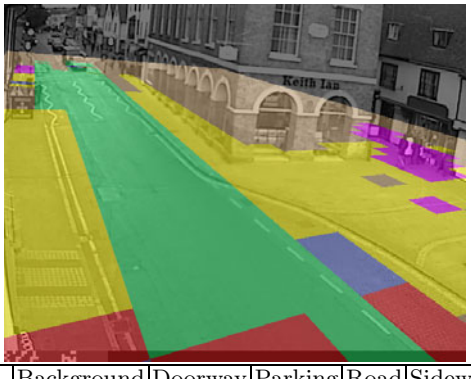


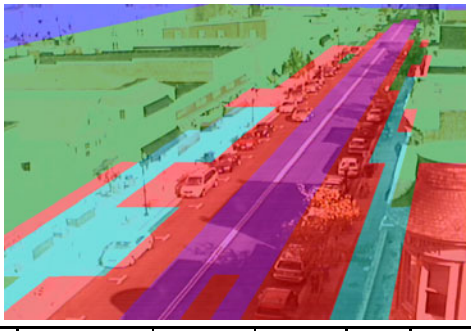
Fig. 7. Two representative clustering results on the Ware scene using detection features only. Several clustering experiments were performed using detection level features only, with the goal of separating pedestrian and vehicle areas. Results were either overly smooth (left) or quite noisy (right). While there is some discrimination of pedestrian/vehicle areas, the performance does not approach that of the proposed hierarchical features.



class	Background	Doorway	Parking	Road	Sidewalk	PCC
Background	391	11	1	1	5	0.9560
Doorway	0	4	0	0	4	0.5000
Parking	1	0	2	0	6	0.2222
Road	37	0	0	125	6	0.7440
Sidewalk	119	13	1	0	107	0.4458

Fig. 8. Top: Unsupervised classification result on Ware. Cells in the same cluster have the same color. Areas outside the ground-plane grid had too few tracks and were not considered. Other cells with fewer than the minimum number of tracks are blue and were not considered. **Bottom:** Confusion matrix. Each row represents the correct class, and each column is the computed class. The rightmost column is the per-class probabability of correct classification.

for this, as there may be multiple clusters that should correspond to one ground-truth label. Conversely, there may be clusters that encompass multiple labels. We assigned the semantic labels by hand, picking the label which visually made the most sense for the supplied clusters.



class	Background	Doorway	Parking	Road	Sidewalk	PCC
Background	621	0	20	19	7	0.9310
Doorway	2	0	0	0	2	0
Parking	1	0	31	0	1	0.9394
Road	0	0	11	39	0	0.7800
Sidewalk	7	0	20	0	15	0.3571

Fig. 9. Top: Unsupervised classification result on OC. **Bottom:** Confusion matrix.

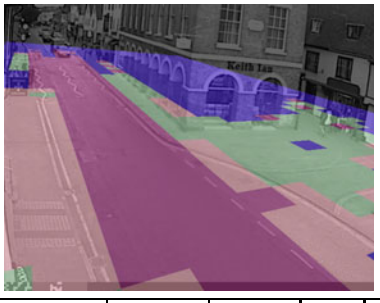
Results on the Ware and OC scenes (computed independently) are shown in Figures 8 and 9. All parameters are the same for both scenes, except that the bandwidth parameter for final clustering was adjusted to compensate for the different number of grid cells (this could be automated). The grid cell size is approximately the same as well. The codebook size is 80, computed with mean-shift. Each feature vector has dimension 11. The final clustering was also computed with mean-shift.

The results indicate that the more basic categories, road and sidewalk, are reasonably learned and classified. Doorways and parking are more difficult categories. There is some confusion between parking and sidewalk in both scenes, because people exit their cars and walk through the parking areas. The small parking area in Ware is partitioned into two clusters, because cars frequently drive through the blue cluster but rarely park there.

Doorways are detected in both scenes, but only partially. The doorway area on the far right in Ware is detected as cluster. There are three doors there, as well as a busy pedestrian thoroughfare just in front of the doors. A small patch in the upper left is clustered into the same class, because this area is a street corner where people emerge from a side street, and also pause while waiting to cross the main road.

3.4 Unsupervised Classification across Scenes

We also wished to understand the transferability of a codebook learned from features on one scene to a classification problem on another scene. In the next experiment, we learn scene element models (codebook and clusters) on one scene, still unsupervised, and "classify" cells in the other. Each cell is classified using



class	Background	Doorway	Parking	Road	Sidewalk	PCC
Background	391	0	0	4	14	0.9560
Doorway	0	0	0	1	7	0
Parking	1	0	0	1	7	0
Road	37	0	0	126	5	0.75
Sidewalk	119	0	0	7	114	0.4750

Fig. 10. Top: Unsupervised classification on Ware using models learned on OC. **Middle:** Classification scored against ground-truth for each cell. **Bottom:** confusion matrix.

mean-shift as described in the previous section. Semantic labels are applied with the same mapping from cluster index to semantic label used on the learning scene. The evaluation procedure is the same as for single-scene scoring. Results are shown in Figures 10 for models learned on OC and tested on Ware.

The results are quite comparable to those computed independently on Ware, indicating the that the method can generalize effectively beyond a single scene. The scenes are geometrically similar, but they are in different parts of the world, with different types of vehicles, buildings and so on. The cameras are different too, with slightly different frame rates, resolutions and quality. Traffic and pedestrian density is considerably higher in Ware.

Although not shown, we conducted the same experiment but with learning on Ware and testing on OC. The scores were slightly lower, but still comparable to the single-scene OC results.

4 Conclusion

We have developed a method that performs unsupervised classification of functional scene element categories observed in video. By abstracting away from specific locations and scenes, and by introducing a descriptive feature vector that characterizes local behavior, we learn generic behavior patterns that correspond to functional categories. Multiple, spatially-disjoint instances of the same scene element type can be identified within a scene, or between different scenes. Results are shown on two scenes and four categories – a small set, but the results are encouraging. In future work we plan to address the limitations outlined in the introduction.

References

1. Wang, X., Ma, K.T., Ng, G.W., Grimson, W.E.L.: Trajectory analysis and semantic region modeling using a nonparametric bayesian model (pdf). In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
2. Makris, D., Ellis, T.: Learning semantic scene models from observing activity in visual surveillance. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, 397–408 (2005)
3. Hu, W., Xiao, X., Fu, Z., Xie, D., Tan, T., Maybank, S.: A system for learning statistical motion patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 1450–1464 (2006)
4. Stauffer, C., Grimson, E.: Learning patterns of activity using real-Time tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 747–757 (2000)
5. Swears, E., Hoogs, A., Perera, A.G.A.: Learning motion patterns in surveillance video using hmm clustering. In: Proceedings of the IEEE Workshop on Motion and Video Computing (2008)
6. Stauffer, C.: Estimating tracking sources and sinks. In: IEEE Conference on Computer Vision and Pattern Recognition Workshop, vol. 4 (2003)
7. Stark, L., Bowyer, K.: Achieving generalized object recognition through reasoning about association of function to structure. IEEE Transactions on Pattern Analysis and Machine Intelligence 13, 1097–1104 (1991)
8. Peursum, P., West, G., Venkatesh, S.: Combining image regions and human activity for indirect object recognition in indoor wide-angle video. In: Proceedings of IEEE International Conference on Computer Vision (2005)
9. Gupta, A., Davis, L.: Objects in action: An approach for combining action understanding and object perception. In: IEEE Conference on Computer Vision and Pattern Recognition (2007)
10. Basharat, A., Gritai, A., Shah, M.: Learning object motion patterns for anomaly detection and improved object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
11. Yang, Y., Liu, J., Shah, M.: Video scene understanding using multi-scale analysis. In: Proceedings of IEEE International Conference on Computer Vision (2009)
12. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. on Patt. Analysis and Machine Intelligence 24 (2002)
13. Stauffer, C., Grimson, W.: Adaptive background mixture models for real-time tracking. In: IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2 (1999)
14. Perera, A.G.A., Srinivas, C., Hoogs, A., Brooksby, G., Hu, W.: Multi-object tracking through simultaneous long occlusions and split-merge conditions. In: IEEE Conference on Computer Vision and Pattern Recognition (2006)
15. Swears, E., Hoogs., A.: Functional scene element recognition for video scene analysis. In: IEEE Workshop on Motion and Video Computing (2009)
16. Oh, S., Hoogs, A., Turek, M., Collins, R.: Content-based Retrieval of Functional Objects in Video using Scene Context. In: 11th European Conference on Computer Vision (2010)