Learning Scene Entries and Exits using Coherent Motion Regions*

Matthew Nedrich and James W. Davis

Dept. of Computer Science and Engineering
Ohio State University, Columbus, OH, USA
{nedrich,jwdavis}@cse.ohio-state.edu

Abstract. We present a novel framework to reliably learn scene entry and exit locations using coherent motion regions formed by weak tracking data. We construct "entities" from weak tracking data at a frame level and then track the entities through time, producing a set of consistent spatio-temporal paths. Resultant entity entry and exit observations of the paths are then clustered and a reliability metric is used to score the behavior of each entry and exit zone. We present experimental results from various scenes and compare against other approaches.

1 Introduction

Scene modeling is an active area of research in video surveillance. An important task of scene modeling is learning scene entry and exit locations (also referred to as sources and sinks [1]). Understanding where objects enter/exit a scene can be useful for many video surveillance tasks, such as tracker initialization, tracking failure recovery (if an object disappears but is not near an exit, it is likely due to tracker failure), camera coverage optimization, anomaly detection, etc.

In this paper we offer a novel approach for learning scene entries and exits using only weak tracking data (multiple short/broken tracks per object). Most existing approaches for learning scene entries and exits rely on strong tracking data. Such data consists of a set of reliable long-duration single-object trajectories. When an object enters the scene, it needs to be detected and tracked until it leaves the scene resulting in a single trajectory capturing the path of the object. Here, the beginning and end of the trajectory correspond to a scene entry and exit observation, respectively. Given enough of these trajectories, the set of corresponding entry and exit observations could be clustered into a set of entry and exit locations. However, collecting such a set of reliable trajectories can be cumbersome, slow, and expensive, especially for complex and crowded urban environments where tracking failures are common. Further, for many strong trackers, real-time multi-object tracking in highly populated scenes is not feasible. Alternatively, randomly selecting one of many objects in the scene to track and accumulating these trajectories over time could be unreliable as the sampled trajectories may not be representative of the true scene action (missing the more

* Appears in International Symposium on Visual Computing, November 2010.

infrequent entries/exits). Such an approach also requires a long duration of time to accumulate enough trajectories.

To compensate for these issues, we instead employ data produced from a weak tracker, which provides multiple and frequently broken "tracklets". We use a modified version of the Kanade-Lucas-Tomasi (KLT) tracker [2] for our work in this paper. Such trackers are capable of locally tracking multiple targets in real-time, and are thus well suited for busy urban environments. Using a weak tracker provides a simple way to detect and track all motion in the scene, though the produced tracklets are more challenging to analyze than reliable trajectories produced by a strong tracker. As an object moves through the scene it may produce multiple tracklets that start and stop along its path. Thus, instead of one reliable trajectory representing the motion of an object, there is a set of multiple and frequently broken tracklets. The goal of our approach is to build a mid-level representation of action occurring in a scene from multiple low-level tracklets. We attempt to cohere the tracklets and construct "coherent motion regions" and use these regions to more reliably reason about the scene entry/exit locations. A coherent motion region is a spatio-temporal pathway of motion produced by multiple tracklets of one underlying entity. Here, we define an entity as a stable object or group of objects. Thus, an entity may be a person, group of people, bicycle, car, etc.

Our approach first detects entities at a frame level. Next, we track the detected entities across time via a frame-to-frame matching technique that leverages the underlying weak tracking data from which the entities are constructed. Thus, entry and exit observations are derived from entities, not individual tracks, entering and exiting the scene. We then cluster the resulting entry/exit observations and employ a convex hull area-reduction technique to obtain more spatially accurate clusters. Finally, we present a novel scoring metric to capture the reliability of each entry/exit region. We evaluate our method on four scenes of varying complexity, and compare our results with alternative approaches.

2 Related Work

Most existing work on learning entry and exit locations relies on strong tracking data. In [3], trajectory start and end points are assumed to be entry and exit observations, respectively. They cluster these points via Expectation-Maximization (EM) to obtain the entry and exit locations. Using a cluster density metric, they label clusters with low density as noise and discard them, leaving a final set of entry and exit locations. In [1] an EM approach is also used to cluster trajectory start and stop points, and they also attempt to perform trajectory stitching to alleviate tracking failures. Such approaches, however, are not applicable to weak tracking data. In [4] a scene modeling framework is described in which they cluster trajectories to learn semantic regions. Only entry and exit observations that exist near the borders of semantic regions are considered when learning entry and exit locations. A similar constraint is also employed in [5] where they only consider states (which they define as a grid area of the scene and motion

direction) near the borders of an activity mask to be eligible to be entry or exit states, though their state space may be constructed from weak tracking data.

Weak tracking data has been used for many applications in computer vision. Some applications employ a low-level per-frame feature clustering as we do to detect entities. In [6] weak tracking is used to perform person counting. They first condition the short and broken trajectories to smooth and extend them. Then, for each frame they perform trajectory clustering on the conditioned trajectories to associate trajectory observations existing through that frame. Using clustering constraints such as a defined person size, they leverage the number of clusters to obtain a final person count. A similar approach is used in [7], though the clustering problem is formulated in a Bayesian framework.

Both [8] and [9] attempt to leverage the idea of a "coherent motion region" from trajectories, though in [8] their regions are constructed using a user-defined bounding box that represents the size of a person. In [9], they define a coherent motion region as a consistent flow of motion which they learn via trajectory clustering, however the trajectories they cluster are collected over a long time window and may be the result of motion occurring at very different times.

3 Framework

Our framework first involves entity detection in each frame using weak tracking data, followed by entity tracking, where we associate entities frame-to-frame, producing a set of hypothesized entity entry and exit observations.

3.1 Entity Detection

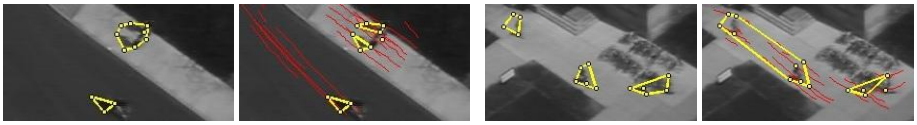
For a given frame, there exists a set of track observations (assuming there is object motion in the frame). The first step in our approach is to cluster these track observations, and thus learn a set of entities for each frame. Trajectory clustering approaches are used in both [6] and [7], but given the complexity of such trajectory clustering techniques, and their sensitivity to parameter selection, we instead use a modified version of mean-shift clustering [10].

Given a set of tracklet observations in a frame of interest, we perform a modified mean-shift clustering as follows. For an observation point $p^* = (x, y)$, we wish to find the nearest density mode from other observation points p_i. Each iteration of mean-shift clustering will move p^* closer to the nearest mode, and is computed as

$$p^*_{new} = \frac{\sum_{i=1}^{n} p_i \cdot w_{vel} \cdot K(\frac{|p_i - p^*|}{h})}{\sum_{i=1}^{n} w_{vel} \cdot K(\frac{|p_i - p^*|}{h})} \tag{1}$$

where K is a kernel (we use the Gaussian kernel), h is the kernel bandwidth, and w_{vel} is a velocity weight (to ensure that only points moving in similar directions as p^* are clustered together) computed as

$$w_{vel} = \begin{cases} \frac{1}{1+exp(-\frac{\cos(\phi)}{\sigma})} & \text{if } |\phi| < \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$



(a) (b) (c) (d)

Fig. 1. Our mean-shift clustering approach (a) and (c) compared to the trajectory clustering approach in [4] (b) and (d).

Here, ϕ is the velocity angle between p^* and p_i (velocity is computed using observations in the previous two frames), and σ defines the rate of weight transition (we use $\sigma = 0.07$ for our experiments). Thus, only points that are spatially close and traveling in similar directions will seek to the same mode.

We also introduce a blend parameter to the velocity weight computation in Eqn. (2). Rather than using the initial velocity of p^* for each iteration (as it seeks to the nearest mode), we start with the initial velocity of p^* and slowly blend it with the velocity of valid surrounding points as it converges toward a mode. Doing so ensures tighter convergence among points, as computing velocities over such a short window is subject to noise. To compute the velocity of p^* at iteration k, we use $dx^k_{p^*} = (1-\alpha) \cdot dx_{p^*} + \alpha \cdot dx_{avg}$ and $dy^k_{p^*} = (1-\alpha) \cdot dy_{p^*} + \alpha \cdot dy_{avg}$. Here, dx_{avg} and dy_{avg} are weighted averages of the velocities of nearby points computed as

$$dx_{avg} = \frac{\sum_{i=1}^{n} dx_{p_i} \cdot w_{vel} \cdot K(\frac{|p_i - p^*|}{\frac{h}{2}})}{\sum_{i=1}^{n} w_{vel} \cdot K(\frac{|p_i - p^*|}{\frac{h}{2}})} \tag{3}$$

where dy_{avg} is computed in the same manner. The blend parameter α is a linear function of the mean-shift iteration number (increasing from 0 to 1). Points that converge to the same mode are defined as belonging to the same entity. The outcome is a set of point clusters (entities), which we then track. Two examples are shown in Fig. 1 (a) and (c).

We compared our modified frame-based mean-shift clustering technique to detect entities to the trajectory clustering technique used in [4], which uses Spectral Clustering on a trajectory similarity matrix built using a modified Hausdorff distance metric to obtain a set of trajectory clusters. The trajectory clustering approach occasionally produced unexpected results, as seen in Fig. 1 (b) and (d). In Fig. 1 (b), it over-clusters the object on the sidewalk into three entities, while our mean-shift approach clusters it as one entity (see Fig. 1 (a)). In Fig. 1 (d), [4] under-clusters the two objects on the left, while over clustering the object on the right. Each approach would sometimes over-cluster and under-cluster, though our approach performed in a more reliable manner overall. Further, our mean-shift clustering approach only has one main parameter - the kernel bandwidth. The trajectory clustering approach is very sensitive to its parameter settings.

3.2 Entity Tracking

After detecting entities, we next associate them frame to frame. This problem can be formulated as an ad-hoc blob tracking problem, though it can leverage the underlying weak tracking data that formed the entities. We use the following graph-based assignment to associate entities. Let V_a be the set of entities that exist in frame a, and V_b be the set of entities that exist in the subsequent frame b. We construct a bipartite graph $G(V, E)$ where V is the vertex set, E is the edge set, and $V = V_a \cup V_b$. We connect v_{a_i} to v_{b_j} with an edge e_{ij}, if the vertices (entities) are connected by at least one shared trajectory from the underlying weak tracking data. Unlike [11], we do not perform any adjustments to these entity interactions. Thus, entities are free to interact with splits, merges, or any combination of the two from frame to frame. We define an entity exit event as an entity from V_a that has no edge connecting it to V_b. Likewise, we define an entity entry event as an entity from V_b that is not connected to an entity in V_a. If an entity from one set (V_a or V_b) shares a tracklet with multiple entities from the other set, we consider this to be an "interaction" (e.g. split, merge). Entity tracks between entries, exits, and interactions correspond to the coherent motion regions. The set of coherent motion regions that begin due to an entity entering the scene correspond to likely entry observations. Likewise, the set of coherent motion regions that end due to an entity exiting the scene correspond to likely exit observations. The set of coherent motion regions that begin with an entry observation and end with an exit observation are especially useful, as they correspond to entities that were able to be tracked reliably from the time they entered the scene until they exited. For such cases, we are able to strongly associate their entry and exit observations, however, we do not require such entity tracks. Figure 2 (a) shows the original weak tracking entry observations, and (b) shows the resultant entity entry observations using our framework.

4 Entries and Exits

From our entity detection and tracking framework described above, we accumulate a set of entity entry/exit location observations. We now explain how we learn entry/exit "regions" from these observations, and how we score each region.

We first perform standard mean-shift clustering on our set of entry locations (and then exits) using the same kernel bandwidth as was used in Sect. 3.1. The result is a set of entry (and exit) clusters. We choose mean-shift clustering over an EM approach (as in [3]) for a few reasons. Mean-shift clustering is able to localize on cluster modes automatically, without knowledge of the number of clusters, as would be required with an EM approach. Model selection techniques such as Bayesian Information Criterion (BIC) [12], for an EM approach, may still sometimes suffer from over fitting (as explained in [13]). Further, the mean-shift clusters better represent the shape of non-Gaussian regions.

After clustering the data we attempt to remove outliers in each cluster, and localize on the area of highest density within each cluster. To accomplish this we first determine the convex hull of each cluster. We then area-reduce the convex

hull by removing unreliable observations. We compute a density score for the points in each cluster via kernel density estimation [14] using only the points in the cluster. We then remove points one at a time in ascending order of density score, and compute the change in area of the new convex hull after each point is removed (the removal of outlier points will cause a large reduction of convex hull area). Thus, we have a distribution of convex hull area changes (from removing each point). We compute the variance of this distribution (assuming a zero mean), and select observations greater than $\sigma = 1.5$ standard deviations away. Of the cluster points that produced these outlier convex hull area changes, we choose the point with the highest density score, and discard all cluster points with lower density scores. Thus, we have a new set of points which better represent the true mass of the cluster. To generalize the shape of this new region, we compute a kernel density surface using the new set of remaining points, determine the point who is lowest on the density surface, and slice the surface at that density value. The perimeter of the slice outlines the final entry or exit region. Thus, unlike [3], our entry and exit clusters reflect the true spatial density and distribution of their underlying observations (which may not be Gaussian).

4.1 Entry/Exit Zone Reliability

We now describe how we validate our entry and exit regions to distinguish reliable entries and exits from those that are the result of noise or partial scene occlusions. In [3], they compute an entry/exit region density and then label regions with density below an arbitrary threshold as noise. Such an approach will not work well if scene traffic is imbalanced, as entries/exits with low popularity (and thus low density), may be regarded as noise. Also, if a scene is very noisy, this method may also classify noise as being a good entry/exit region. We define a good entry region as one with entity tracks emanating out of it, and a good exit region as one with entity tracks flowing into it. Entry regions whose entry-only tracks (or exit regions whose exit-only tracks) exhibit bidirectional activity are unreliable regions, and may be the result of areas with a high rate of tracking failure, partial scene occlusions, or scene noise (trees or other such movement that the tracker may pick up). Further, for entry regions, other tracks in the scene should not intersect the region in the same emanating direction that defines the region (i.e., the entry region should not be a "through" state). Such a scenario would indicate that another entry region exists behind the current one. The same idea holds for exit regions. Thus, we attempt to capture both the consistency of the entry/exit entity tracks that define each region, as well as the consistency of the interaction between other entity tracks and each entry/exit region.

Using the entry/exit entity tracks that define each region, we learn the distribution of directions that these tracks leave (for entries), and enter (for exits), the region by quantizing the angle that each track intersects the region into one of b bins (we use $b = 8$ in our experiments). This histogram is normalized to provide a probabilistic measure for the directions that entry/exit tracks leave/enter each region. From this distribution r, we compute a directional consistency function $\hat{r}$, which accounts for any symmetry of the entity track distribution for each

entry and exit region in the following manner. For a bin i with probability $r(\theta_i)$, every other bin probability $r(\theta_j)$ is subtracted from $r(\theta_i)$, in a weighted manner such that bin angles that are directly opposite of i receive high weight (as they correspond to bi-directional behavior), and bin angles close to i receive lower weight. For a region k,

$$\hat{r}_k(\theta_i) = \frac{\max\left[0, \sum_{j=1}^{b} w_{ij} \cdot (r_k(\theta_i) - r_k(\theta_j))\right]}{\sum_{j=1}^{b} w_{ij} \cdot r_k(\theta_i)} \cdot r_k(\theta_i) \tag{4}$$

where w_{ij} is an angle similarity weight that give more emphasis to angles corresponding to bidirectional behavior with respect to θ_i, and is computed as

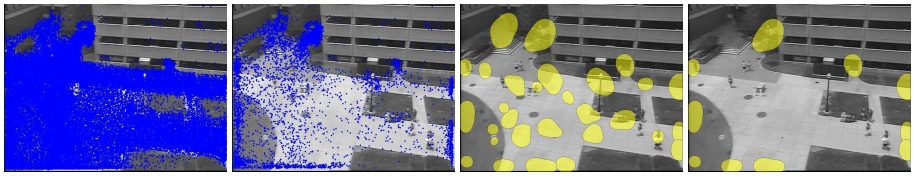
$$w_{ij} = \begin{cases} \exp(1 + \cos(|\theta_i - \theta_j|)) & \text{if } \cos(|\theta_i - \theta_j|) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Thus, θ_j is ignored if it is within 90 degrees of θ_i, and most heavily weighted when it is exactly opposite of θ_i. For a region k, $\sum_{i=1}^{b} \hat{r}_k(\theta_i)$ is 0 when the tracks leaving an entry (or entering an exit) are completely symmetric (and thus strongly bi-directional), reflecting that the region is unreliable, and $\sum_{i=1}^{b} \hat{r}_k(\theta_i)$ is 1 when a region is completely non-symmetric, and thus reliable.

In addition to this directional consistency measure, we also need to capture the way in which other tracks interact with each entry or exit region. We combine the previous directional consistency measure (Eqn. (4)) with an interaction component to compute a total reliability score, ψ, for each region k

$$\psi_k = \left(\sum_{i=1}^{b} \hat{r}_k(\theta_i)\right) \cdot \left(1 - \min\left[1, \frac{\sum_{i=1}^{b} \hat{r}_k(\theta_i) \cdot M_k(\theta_i)}{\sum_{i=1}^{b} \hat{r}_k(\theta_i) \cdot N_k(\theta_i)}\right]\right) \tag{6}$$

Here, N_k is the number of entity tracks that define an entry/exit region k, and $N_k(\theta_i)$ is the number of entity tracks that leave an entry (or enter an exit) region k at angle i. Similarly, M_k is the number of other entity tracks in the scene that intersect an entry/exit region, and $M_k(\theta_i)$ is the number of those entity tracks that intersect and entry/exit region k at angle i. As described earlier, for a reliable region these tracks should not intersect an entry region in the same direction as the entry tracks that leave it (or intersect an exit region in the same direction as its tracks exit). The first term $\sum_{i=1}^{b} \hat{r}_k(\theta_i)$ is the directional consistency term for the entry/exit region and acts as a prior (Eqn. (4)). This score will dominate the total score if $M_k = 0$ (no tracks intersect the region k). If $M_k > 0$, and the intersecting tracks support the region as being reliable, then $\sum_{i=1}^{b} \hat{r}_k(\theta_i) \cdot M_k(\theta_i) \approx 0$. If however there are tracks that intersect the region and contribute evidence that the region may be unreliable (due to tracking failures, the region being partially occluded, etc.), $\frac{\sum_{i=1}^{b} \hat{r}_k(\theta_i) \cdot M_k(\theta_i)}{\sum_{i=1}^{b} \hat{r}_k(\theta_i) \cdot N_k(\theta_i)}$ will approach 1 as the number of discrediting intersecting tracks approaches N_k, and grow large if the number of tracks $> N_k$, penalizing the region score. If N_k is 0, then we define the region as unreliable. Lastly, we compute a final region score as



(a) (b) (c) (d)

Fig. 2. (a) Weak tracking entries, (b) entity entries, (c) potential entry regions, and (d) entries with reliability score $\Psi > 0.75$. [Best viewed in color]

$$\Psi_k = \frac{1}{1 + exp(-\frac{\psi_k - \mu}{\sigma})} \tag{7}$$

where μ and σ should be determined based on the scene noise, tracking reliability, etc. For our experiments we used $\mu = 0.5$ and $\sigma = 0.15$. Formulating the final score as such allows for noise tolerance, and makes the model able to adapt to scenes with different noise levels. Figure 2 (c) shows entry regions detected for a scene, and (d) shows the resultant reliable entry regions ($\Psi > 0.75$).

In addition, we also compute a popularity score, $Popularity = N_k$, for an entry/exit region k (number of entry/exit tracks that define the region).

5 Experiments

We performed experiments to compare our learned entry/exit regions to those using other approaches. We employ data from four scenes of varying complexity captured from cameras mounted on four and eight story buildings, at a resolution of 640×480. The duration of the datasets used were 1 hour (Scene 4), 2 hours (Scene 1), and 3 hours (Scenes 2 and 3). We also show how our framework may be used to learn relationships between entries and exits in the scene.

5.1 Entry/Exit Region Evaluation

We compared our results to the methods described in [3] and [5] (as [5] also uses weak tracking data to determine entries/exits). In [5], they partition the scene into a grid of states, where each state is defined by a grid cell location and motion direction. They then map each tracklet to a set of states. They define an entry and exit weight metric to score each state s_i as $W_E = C_{start} \cdot max(0, 1 - \frac{C_{in}}{C_{start}})$ (entry weight), and $W_X = C_{stop} \cdot max(0, 1 - \frac{C_{out}}{C_{stop}})$ (exit weight). Here, C_{start} is the number of tracklets that start in state s_i, C_{in} is the number of tracklets that transition into s_i, C_{stop} is the number of tracklets that stop in state s_i, and C_{out} is the number of tracklets that transition out of state s_i. Low weight entry and exit states can then be removed to obtain a final set of entry and

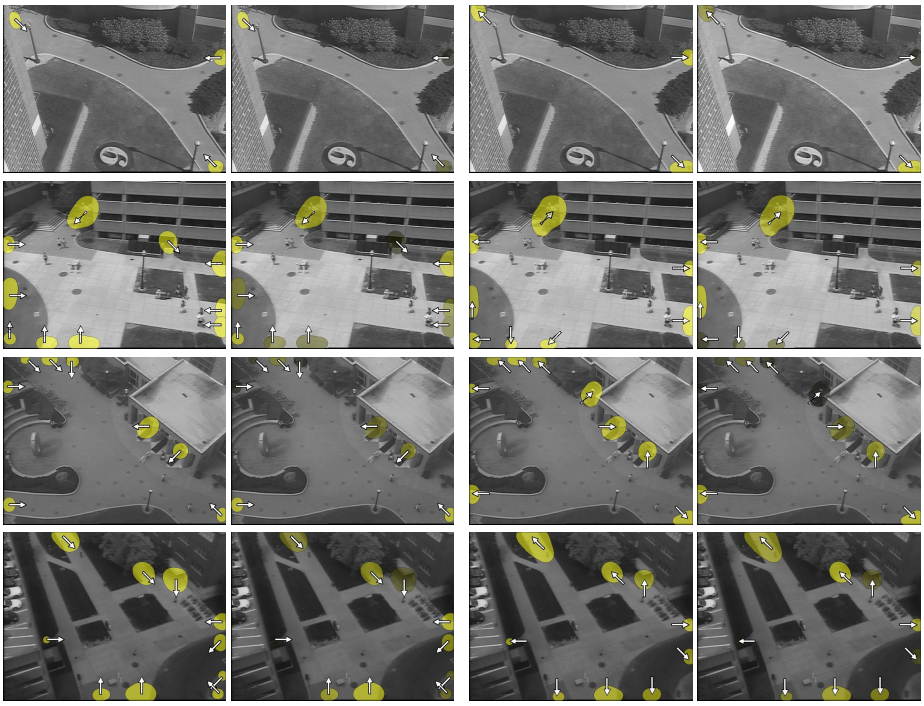


Fig. 3. Entry and exit regions (using our proposed method) with a reliability score $\Psi > 0.75$ and popularity > 10 tracks. [Best viewed in color]

exit states. They also use a binary activity mask to force entry/exit states to be on the border of the scene activity, though we do not employ this technique to allow for a fair comparison (neither our method, nor the method in [3] have such constraints). In [3], they use EM to cluster entry and exit track points. They then use a density metric to determine which clusters to keep, computed as W/E where W is the ratio of points belonging to a cluster, and E is the area of the Gaussian ellipse, calculated as $E = \pi \cdot I_1 \cdot I_2$ where I_1 and I_2 are the eigenvalues of the covariance matrix. We used our *entity* track entry/exit observations as input for this method, as it is intractable and noisy (see Fig. 2 (a)) to learn entries/exits using weak tracking (tracklet) start/stop observations.

For our method, we chose a reliability score threshold of $\Psi > 0.75$ to select a set of final entry/exit states. We also remove any remaining regions with a popularity score less than 10. Results from our method can be seen in Fig. 3. We display those regions along with the strongest track entry/exit angle for each region. We also display entry/exit region popularity by the strength of the entry/exit region color. It is worth noting that we experimented with various kernel bandwidth (h) values (from 10 to 20), and found the results did not vary

significantly. For our experiments we used $h = 13$. For the method described in [5], we display states with an entry/exit weight at least 0.75 of the max entry/exit weight. The states are displayed by arrows over a scene grid denoting the state location and direction. For the method described in [3], we clustered the entity entry and exit observations using EM, with a relatively large number of clusters (as they do). We display entry/exit regions with density scores $>$ 0.01 of the of the maximum density score (produced the best results). Results for these two alternative methods can be seen in Fig. 4.

For Scene 1, our method was able to learn all expected entry and exit regions. The methods in [5] and [3] both learn incorrect entries/exits due the streetlight occlusion in this scene. Our method is able to recognize this as a partial scene occlusion due to the motion symmetry and entity tracks intersecting the region. Further, [5] fails to learn the entry/exit on the top right of the scene, and [3] learns various noise clusters which result from the KLT tracker picking up bushes in the scene that move due to wind. The motion of such action is inconsistent, and thus is flagged by our method as being an unreliable region.

For Scene 2 our method was able to learn the desired entry/exit regions with the exception of the region in the top left of the scene. This is due to the region being difficult to track, and as such this region exhibits symmetric behavior. The method in [5] also does not learn this region. Both [5] and [3] again learn entry/exit regions that result from the streetlight occlusion in this scene. Also, [3] also tends to over-cluster many valid regions, and both methods learn noisy regions in the middle of the scene.

In Scene 3, all of the regions that we learn are valid entry/exit regions, though we also learn one exit region corresponding to activity at a picnic table. For this scene, [3] fails to learn the ramp entry and exit on the top left of the scene (due to low traffic) and learns a few noisy entry regions near a bush. The method in [5] also fails to learn the ramp entry and exit on the top left, as well as the exit on the top of the scene and one of the building exits. Further, [5] learns a noisy entry and exit region near the streetlight occlusion.

For Scene 4 our method learns all expected regions with the exception of the top left sidewalk. Again, this is due to this region being difficult to track. The method from [5] also fails to learn this region as an entry, and [3] only learns it as an exit. Further, [3] fails to learn the parking garage entry/exit regions as well as the building entry region, and it also over-clusters the entry near the top of the scene. In addition, [5] fails to learn the building and parking garage entries and exits, as well as the entry near the top of the scene. It also learns various noisy regions in the middle of the scene.

Choosing a robust density threshold proved to be a difficult challenge for [3]. A threshold that works well for one scene may not generalize for other scenes, as seen in the results. Also, neither method enforces entry/exit directional consistency, and thus may learn very noisy entries/exits (method [3] in Scene 1), or regions that result from partial scene occlusion (e.g., streetlight examples in Scenes 1, 2, and 3). Overall, the proposed approach produced the best results.

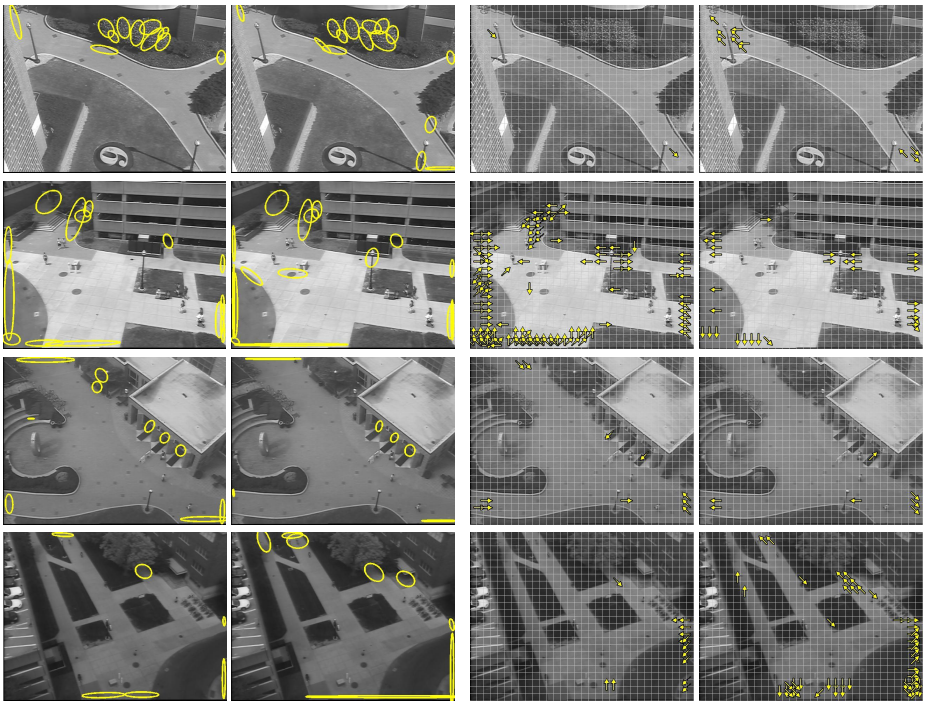


Fig. 4. Entry and exit regions using the method in [3] (cols 1 and 2). Entry and exit states using the method in [5] (cols 3 and 4). [Best viewed in color]

5.2 Semantic scene actions

In addition to using our method to learn scene entries and exits, we can also learn relationships between entries and exits. This is possible using the set of coherent motion regions that begin with an entry and end with an exit observation. This information could be used to see where people go without caring about the actual path they take. The top entry/exit relationships for Scenes 2-4 can be seen in Fig. 5. We were able to learn the relationships via entities constructed from weak tracking data using our framework. Such analysis usually requires large amounts of strong tracking data.

6 Conclusion

We proposed a novel framework to learn scene entries and exits using weak tracking data. We described a method that forms weak tracking data into coherent motion regions by detecting entities at a frame level and tracking the entities

Scene 2 Scene 3 Scene 4

Fig. 5. Strongest activities from various scenes displayed as arrows from entry regions to exit regions. [Best viewed in color]

through time, resulting in a set of reliable spatio-temporal paths. We also introduced a novel scoring metric that uses the activity in the scene to reason about the reliability of each entry/exit region. Results from multiple scenes showed that our proposed method was able to learn a more reliable set of entry/exit regions as compared to existing approaches. This research was supported in part by the US AFRL HE Directorate (WPAFB) under contract No. FA8650-07-D-1220.

References

1. Stauffer, C.: Estimating tracking sources and sinks. In: In Proc. Second IEEE Event Mining Workshop. (2003)
2. Shi, J., Tomasi, C.: Good features to track. In: CVPR. (1994)
3. Makris, D., Ellis, T.: Automatic learning of an activity-based semantic scene model. In: AVSS. (2003)
4. Wang, X., Tieu, K., Grimson, E.: Learning semantic scene models by trajectory analysis. In: ECCV. (2006)
5. Streib, K., Davis, J.: Extracting pathlets from weak tracking data. In: AVSS. (2010)
6. Rabaud, V., Belongie, S.: Counting crowded moving objects. In: CVPR. (2006)
7. Browstow, G., Cipolla, R.: Unsupervised Bayesian detection of independent motion in crowds. In: CVPR. (2006)
8. Cheriyadat, A., Bhaduri, B., Radke, R.: Detecting multiple moving objects in crowded environments with coherent motion regions. In: POCV. (2008)
9. Cheriyadat, A., Radke, R.: Automatically determining dominant motions in crowded scenes by clustering partial feature trajectories. In: Proc. Int. Conf. on Distributed Smart Cameras. (2007)
10. Cheng, Y.: Mean shift, mode seeking, and clustering. PAMI **17** (1995)
11. Masoud, O., Papanikolopoulos, N.: A novel method for tracking and counting pedestrians in real-time using a single camera. IEEE Trans. on Vehicular Tech. **50** (2001)
12. Schwarz, G.: Estimating the dimension of a model. Ann. of Statist. **6** (1978)
13. Cruz-Ramirez, N., et al.: How good are the Bayesian information criterion and the minimum description length principle for model selection? A Bayesian network analysis. In: Proc. 5th Mexican Int. Conf. on Artificial Intelligence. (2006)
14. Parzen, E.: On estimation of a probability density function and mode. Ann. Math. Statist. **33** (1962)