

# Detecting behavioral zones in local and global camera views

Matthew Nedrich · James W. Davis

Received: 9 June 2011 / Revised: 13 February 2012 / Accepted: 13 February 2012  
© Springer-Verlag 2012

**Abstract** We present a complete end-to-end framework to detect and exploit entry and exit regions in video using behavioral models of object trajectories. Using easily collected “weak” tracking data (short and frequently broken tracks) as input, we construct a set of entity tracks to provide more reliable entry and exit observations. These observations are then clustered to produce a set of potential entry and exit regions within the scene, and a behavior-based reliability metric is used to score each region and select the final zones. We also present an extension of our fixed-view approach to detect entry and exit regions within the entire viewspace of a pan-tilt-zoom camera. We additionally provide methods employing the regions to learn scene occlusions and causal relationships from entry–exit pairs along with exploitation algorithms (e.g., anomaly detection). Qualitative and quantitative experiments are presented using multiple outdoor surveillance cameras and demonstrate the reliability and usefulness of our approach.

**Keywords** Scene understanding · Behavioral modeling · Computer vision · Weak tracking · Tracking · Entry and exit detection

## 1 Introduction

An important step when seeking to understand a scene is to identify regions where activity enters and exits. Such regions may correspond to a doorway, garage opening, or a walkway

that intersects the edge of the camera view (image border), and can be useful in many visual surveillance applications. For tasks, such as long-term object tracking, entry regions allow for more knowledgeable tracker initialization. In addition, if tracking is terminated at some location, but not near a known exit (or occlusion), it is likely due to tracker failure. Understanding entry and exit locations may also help in attaching semantic meaning to tracking events. If an object enters through a commonly used entry region and leaves through a popular exit, the object is likely behaving normally. However, if an uncommon region pairing is found, or an object enters or exits through an area that does not correspond to an entry/exit location, then such an event may indicate anomalous activity.

Entry and exit regions may also be useful for higher level scene analysis. It may be desirable to learn the scene “pulse” of pedestrians, cars, and cyclists as they come and go throughout the day. Knowing where objects typically enter and exit provides insight as to where the scene’s pulse can be taken. In addition, if semantic meaning (labeling) can be attached to each entry or exit region (e.g., a particular building doorway), monitoring traffic at these regions can help indicate how populated these semantically meaningful areas are (e.g., buildings). As we will demonstrate, depending on the direction of the region relationships (e.g., entry  $\rightarrow$  exit versus exit  $\rightarrow$  entry behavior patterns), region connections can help in understanding the dynamics of the scene activity as well as reasoning about occluded pathways in the scene.

### 1.1 Proposed approach

We present a novel approach to discover and exploit entry and exit regions in video from tracking data. Most scene modeling techniques require some form of object tracking as input, and many existing methods [15, 16, 29, 32] rely on

---

M. Nedrich (✉) · J. W. Davis  
Department of Computer Science and Engineering,  
Ohio State University, Columbus, OH, 43210, USA  
e-mail: nedrich.1@osu.edu

J. W. Davis  
e-mail: jwdavis@cse.ohio-state.edu

*strong* tracking data (a single persistent trajectory for each target). While such tracks are very useful, many approaches to collect them tend to be computationally expensive, able to track only a subset of objects in real-time, and are unreliable in busy outdoor urban environments. To compensate for these shortcomings we designed our approach to handle *weak* tracking data (multiple and frequently fragmented tracks per target). Weak trackers are capable of tracking many objects simultaneously in real-time and can function well in busy scenes. For our work in this paper, we use a modified version of the well-known Kanade Lucas Tomasi (KLT) tracker [27], as presented in [30], which tracks features on moving objects in the scene.

Given weak tracking data, we learn “entities” in each frame by clustering similar weak tracking observations that move together in a coherent manner (different than track stitching). Thus, an entity may correspond to a person, group of people, bicycle, car, etc. The entities are tracked loosely over time by associating them from frame-to-frame. As they move, we allow them to split and merge with other entities in the scene. This process allows us to cohere the weak tracking data into a more reliable set of tracks.

We cluster the entry and exit observations from the entities to produce a set of potential scene entry and exit zones. Each zone is then scored using a behavior-based reliability metric that analyzes the manner with which the entity trajectories form and interact with each region. We define a reliable entry region as one with tracks *emanating out* of it in a mostly directional manner, and similarly a reliable exit region as one where tracks *flow into* it in a mostly directional manner. Furthermore, other tracks in the scene should not intersect an entry/exit region in the same movement direction as the tracks that formed the region. (e.g., another entity track intersecting in the same flow direction would indicate that another entry region is close behind). After scoring each region, unreliable zones are removed by thresholding, leaving only the desired entry and exit regions.

We also present an extension of our fixed-view approach for the entire view space of a pan–tilt–zoom (PTZ) camera. We explain how each process used in the local view entry/exit approach may be extended to work in the spherical camera space. Finally, we show how relationships between entry and exit regions can be learned, and provide methods to uncover occlusions and other behavioral relationships in the scene.

In general, the main contributions of our paper can be summarized as follows:

- *Compatible with weak tracking data.* Most previous scene modeling methods require strong tracking data (long reliable object trajectories). Our approach allows for weak tracking data which makes collecting input much easier, especially for very busy and crowded scenes.
- *Inclusion of scene behavior to detect entry/exit regions.* Previous approaches have ignored the underlying behavioral component, and we show that utilizing behavior can produce better results.
- *Construction of new global method to detect entry/exit regions with respect to the entire view space of a PTZ camera.* Existing work has focused on single camera views. Although other work may utilize multiple views from different cameras, they do not concentrate on the entire camera view space as we do in our work.
- *Exploitation of the detected entry/exit regions to automatically discover occlusions in the scene.* Unlike previous work, we do not model the occluding structure directly, but rather the entry and exit region patterns encountered when objects move behind occlusions.
- *Presentation of quantitative results.* Most scene modeling work evades quantitative analysis, relying solely on subjective qualitative evaluation. Our work provides both quantitative and qualitative results.

We begin with a review of related work in Sect. 2, and provide a general system overview in Sect. 3. The entity learning process is described in Sect. 4, and the region detection and scoring method is described in Sect. 5. Our extension from the local camera view to a global camera view space is presented in Sect. 6. Using the detected regions, we demonstrate (Sect. 7) how the regions may be exploited to discover scene occlusions and learn causal relationships. A thorough evaluation is presented in Sect. 8, where we provide experimental results for our local and PTZ view space region detection approaches and region exploitation methods.

## 2 Related work

In this section, we provide an overview of related work with approaches using weak tracking as input and methods for entry/exit detection and exploitation.

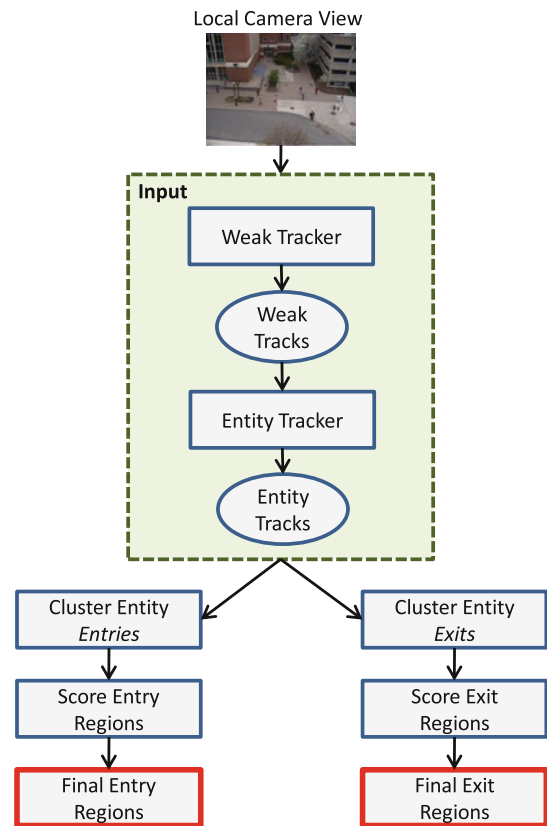
The idea of using weak tracks when reasoning about a scene has been employed for different tasks. In [22], weak tracks are clustered and used to estimate pedestrian counts in busy scenes. Their approach requires the tracks be extended and conditioned before clustering. In [4], they attempt to leverage the idea of a “coherent motion region” to learn distinct objects using weak tracks (as we similarly do with our entity approach). However, their motion regions are constructed with a user-defined bounding box representing the size of a person. Coherent motion regions are also used in [5] via clustering of weak tracks, but ignore the time stamps of the trajectories. Our approach constructs entities from weak tracking data to provide a more reliable set of entry and exit observations that we use to estimate entry and exit regions.

There has been a relatively small body of work in relation to entry/exit zone identification. In [16] and [29], trajectory endpoints are assumed to be observations of actual entry and exit locations, which are then clustered to learn entry and exit regions. A density threshold is used in [16] to remove suspected noise clusters. However, such an approach requires stronger tracking data and will not work well using weak tracks. Further, both approaches do not leverage the scene behavior to learn accurate regions (as we do), which can be especially useful if tracking data is noisy. In [32], a framework is described to model various semantic regions via trajectory clustering. When learning entry and exit locations, only the trajectory endpoints that exist near the borders of semantic regions are considered. In [30], a grid-based approach is presented where the ratio of tracks through each state (grid cell and direction) to tracks that originate in (entries) or terminate in (exits) each state is leveraged to learn entry/exit states. The entry/exit states are constrained to be near the border of an activity mask for the scene. However, they do not evaluate their method for the task of entry/exit state identification, and instead use it to detect “pathlets” (common pathway segments) in the scene. Of these approaches, only [30] is able to function using weak tracking data.

None of the aforementioned entry/exit methods attempt to identify regions that are due to occlusions in the scene (as we demonstrate). Handling occlusions typically involves detecting and modeling the occluding structures directly. Existing occlusion detection methods (such as [8,9]) model occlusions directly in 3D or in image space. In [12], they attempt to model occlusions in 3D by observing tracking data from multiple cameras. In [11], trajectory matching is used to track objects through occlusions, and they attempt to learn plausible occluding structures through image segmentation (requiring manual labeling). Our approach is unique in that we detect regions where objects enter and exit occluded areas by studying correlated scene activity between the regions. In [18], they propose a method to learn relationships between exit and entry regions between disjoint camera views to automatically learn the topology of a camera network, however, they do not attempt to learn occlusion connections. Although their approach is able to detect (entry  $\rightarrow$  exit) connections using time-lagged cross-correlation, our approach is different in that we attempt to estimate the path distance traveled between tied regions. As a result, we not only learn the connections between regions, but the actual path distance traveled between them.

### 3 System overview

In this section, we provide an overview of our approaches for local and PTZ view space entry and exit region detection.



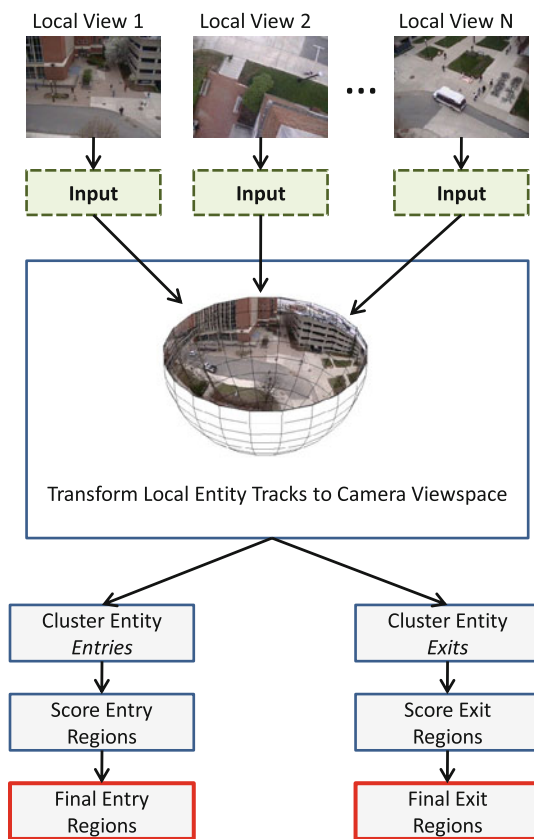
**Fig. 1** Local camera view entry/exit detection system overview

#### 3.1 Local view region detection

Our static camera view (local) approach is applicable to scenarios in which a fixed camera view is employed. Our process for detecting local entry and exit regions is summarized in Fig. 1. Given a local camera view, we track objects using a weak tracker. From the collected weak tracks, we learn a set of entity tracks (described in Sect. 4). The entry and exit observations for the entity tracks are then used to hypothesize a set of potential entry and exit regions. This is accomplished by clustering the entry and exit observations independently, removing outlier observations from each cluster (if any exist), and obtaining a region shape (Sect. 5.1). Each potential entry and exit region is then scored using a behavioral-based reliability metric (Sect. 5.2) which captures the directional and interaction consistency of each region. Regions with a low reliability score are removed, producing a final set of entry and exit regions.

#### 3.2 Camera view space region detection

When a PTZ camera is employed, our process for detecting entry and exit regions within the camera view space is summarized in Fig. 2. We collect weak tracking data from a set of overlapping local camera views that collectively span the



**Fig. 2** Camera viewspace entry/exit detection system overview

entire camera viewspace. For the set of weak tracks in each local view, we obtain a set of local entity tracks. The entity tracks from each view are then projected to a common global camera viewspace (Sect. 6). Using the same processes for our local region approach, but adapted for the spherical camera viewspace, we cluster entity track entry and exit observations, remove outliers, and generate a set of potential entry and exit regions (Sect. 6.3). Lastly, each region is scored using the same behavior reliability metric as in the local approach, and regions with low reliability are removed.

## 4 Entities

Weak tracking data can be readily collected (shown in Fig. 3), but is far too noisy to be used directly for detecting entry and exit regions in a scene. However, weak tracks can be aggregated (into entities) to produce more reliable information. The resultant entity tracks differ from tracks produced using a strong tracker as we only loosely associate entities over time, allowing them to merge and split with other entities as they move through the scene. In this section, we explain how to learn entities from weak tracks by first detecting clusters

in each frame and then associating them across temporally adjacent frames.

### 4.1 Entity clustering

We cluster the weak tracking observations in each frame into sets of spatially close observations moving in similar directions. Thus, a cluster (entity) may correspond to a person, group of people, vehicle, etc. For a given image frame  $f_i \in F$ , our weak tracker produces a set of trajectory observations  $P_i$  (assuming there is object motion in the frame). Our goal is to learn a set of entities,  $T_i$ , for each frame  $f_i$ . To do so, we employ a modified version of mean-shift clustering [3] to cluster the trajectory observations ( $P_i$ ), assigning each weak track observation  $p = (x, y) \in P_i$  to an entity  $t \in T_i$ . We modify the standard mean-shift clustering formulation to introduce a velocity weight to ensure that observations that cluster together are spatially close and travel in the same direction. For a weak tracking observation  $p = (x, y)$ , it is shifted to location  $p_{\text{new}}$  until convergence, where  $p_{\text{new}}$  is computed as

$$p_{\text{new}} = \frac{\sum_{i=1}^n p_i \cdot w_{\text{vel}} \cdot K\left(\frac{|p_i - p|}{h}\right)}{\sum_{i=1}^n w_{\text{vel}} \cdot K\left(\frac{|p_i - p|}{h}\right)} \quad (1)$$

where,  $K$  is the mean-shift kernel, and  $h$  is the kernel bandwidth, chosen to reflect the size of the entities moving in the scene (we use the Gaussian kernel with  $h = 15$  or  $h = 30$  depending on the view or camera).

The velocity weight,  $w_{\text{vel}}$ , is a function of the angle ( $\phi$ ) between the velocity of points  $p_i$  and  $p$ , and is computed as

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

where  $\sigma$  defines how sharply the similarity transitions are (from 0 to 1) (we use  $\sigma = 0.07$  for our experiments). The velocity at a point is computed using observations from the previous two frames of the track.

We also introduce a blend parameter to the velocity weight computation in Eq. 2. Rather than using the initial velocity of  $p$  for each iteration of mean shift (as it moves to the closest mode), we start using the initial velocity of  $p$  and then slowly blend it with the velocities from surrounding points (spatially nearby points moving in a similar direction). Doing so ensures tighter convergence. To compute the velocity ( $dx, dy$ ) of  $p$  at iteration  $k$ , we use  $dx_p^k = (1 - \alpha) \cdot dx_p + \alpha \cdot dx_{\text{avg}}$  and  $dy_p^k = (1 - \alpha) \cdot dy_p + \alpha \cdot dy_{\text{avg}}$ . Here,  $dx_{\text{avg}}$  and  $dy_{\text{avg}}$  are weighted averages of the velocities of nearby points (computed using the same Gaussian kernel from above). The blend parameter  $\alpha$  is a linear function of the mean-shift iteration number (increasing from 0 to 1, using



**Fig. 3** KLT (weak) tracks

a constant increment and capped at 1). After clustering, we obtain a set of entities  $T_i$  in each frame (Fig. 4).

#### 4.2 Entity tracking

We next associate the entities across frames using a graph-based method. Let  $T_i$  be the set of entities in frame  $i$ , and  $T_j$  be the set of entities in the subsequent frame  $j$ . We construct a bipartite graph  $G_e(V_e, E_e)$  where  $V_e$  is the vertex set ( $V_e = T_i \cup T_j$ ) and  $E_e$  is the edge set. We connect  $t_a \in T_i$  to  $t_b \in T_j$  if the vertices (entities) are connected by at least one underlying shared weak track. An entity from frame  $i$  that is not connected to any other entity in the subsequent frame  $j$  corresponds to an entity *exit* event. Likewise, an entity in frame  $j$  not connected to any other entity in the previous frame  $i$  corresponds to an entity *entry* event. If an entity shares a trajectory with (is connected to) multiple entities from a temporally adjacent frame, we consider this to be an entity *interaction* (split, merge). This process is repeated for each pair of frames to extend the entities through time. Thus,

each entity will begin with an entry or interaction event and end with an exit or interaction event. Figure 4 displays a set of weak tracks on a target, and the corresponding entity track.

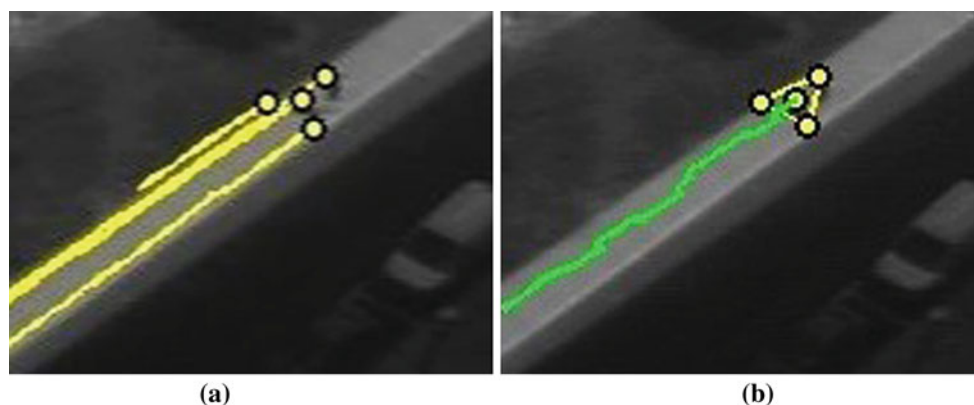
The entry and exit observations from the entity tracks are then used to hypothesize a set of potential entry and exit regions. As a comparison to use weak tracking observations, Fig. 5 displays a set of weak track start observations versus entity entry observations. As shown, the entities produce a much more reliable (and reduced) set of entry and exit observations (compared with using the weak tracking start and stop observations), although they still containing some noisy observations (which we address in the following section). It is worth noting that while our application of modified mean-shift clustering to group the weak trajectories into entities is unique, it is not the main contribution of our paper. This approach allows for the use of weak tracking data as input, which makes collecting tracking data for busy and crowded scenes easier.

### 5 Entry and exit detection

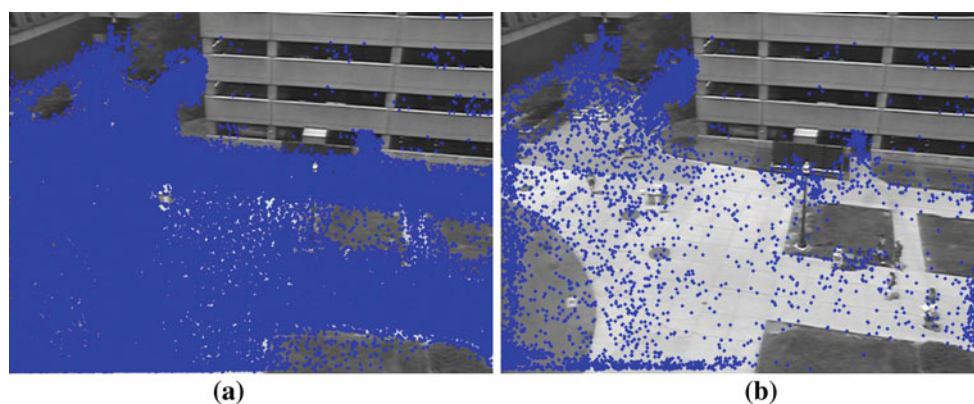
From our entity detection and tracking framework described above, we accumulate a collection of entity entry and exit location observations. We now explain how we detect entry/exit “regions” from these observations and how we score each region as to remove noise. This is accomplished by first clustering each set (separately) of entity entry and entity exit observations. We then remove outlier observations and extract a region shape from the points within each cluster. In contrast to previous density-based approaches [16], we score each potential entry and exit region using a behavior-based reliability metric that captures the consistency of the behavior for each region (rather than only density).

#### 5.1 Region shape

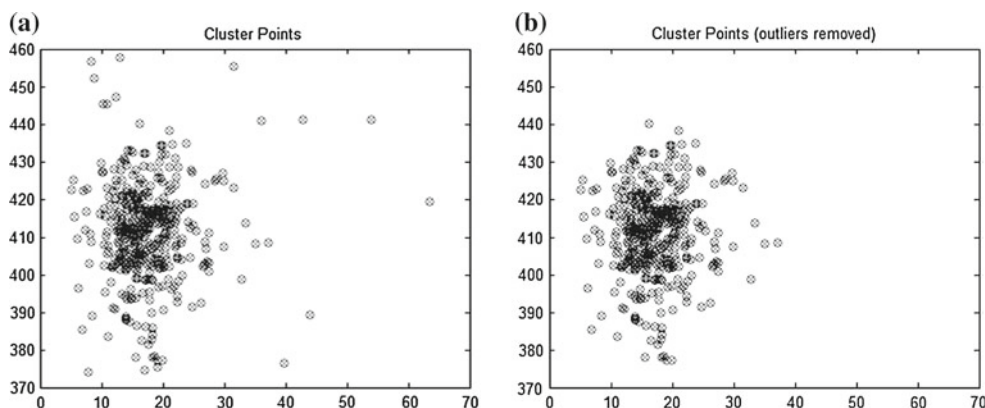
We first perform standard mean-shift clustering on our sets of entry/exit locations to produce entry and exit clusters (we



**Fig. 4** **a** Weak tracks and **b** corresponding entity track



**Fig. 5** **a** Weak tracking start observations and **b** corresponding entity entry observations



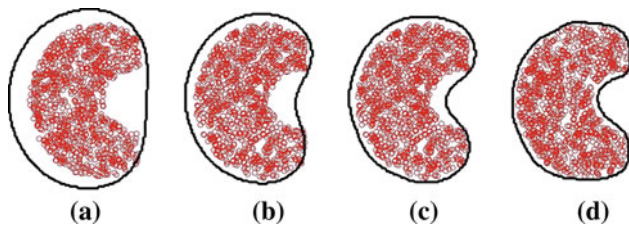
**Fig. 6** Outlier removal example. **a** Original cluster points and **b** final cluster points after removing outliers

cluster entries and exits independently). We choose mean-shift clustering at this stage over a mixture of Gaussians (MoG) approach (as in [16]) 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 a MoG approach. Techniques, such as computing the Bayesian information criterion (BIC) [26] to automatically determine the number of clusters could be employed, but may sometimes suffer from over fitting (as explained in [6]). Further, the mean-shift clusters better represent non-Gaussian shaped regions.

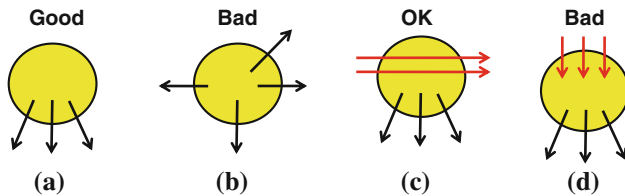
After clustering the data, we attempt to remove outliers in each cluster and localize on the area of highest density within each cluster. To accomplish this, we employ a convex hull area-reduction technique. We first compute a convex hull around each cluster of observations. Then, for each cluster, points on the perimeter are iteratively removed in order of ascending density (the density of each point is computed using kernel density estimation (KDE) [20]). After each point is removed, we compute a new convex hull and record the area change from the previous convex hull. The intuition is that removal of outlier points will result in large convex hull area changes.

We compute the variance of this distribution of area changes, and select observations greater than  $\sigma_r$  standard deviations. Of the cluster points that produced these outliers, we choose the point that was most recently removed (most dense), and delete all previous (lower density) points. This results in a new reduced set of points which better represents the true mass of the cluster. An example noisy cluster is shown in Fig. 6a, and the resultant cluster after removing outliers using our approach is shown in Fig. 6b.

We next fit a shape to each cluster to represent the region. We first compute a density surface for the points in each cluster using KDE. We then select the point from the cluster sitting lowest on the surface (lowest density), and slice the surface at that density. The perimeter of the slice becomes the final region shape. Depending on the kernel bandwidth used to generate the KDE surface, the shape will vary. Larger kernel bandwidths will generalize the shape around the points, and smaller kernel bandwidths will force the shape to wrap more tightly to the point mass. Example region shapes obtained using different KDE bandwidths are shown in Fig. 7 (the black outline represents the region shape learned). Thus, unlike [16], our entry and exit clusters better reflect the true spatial density and distribution of their underlying observations (which may not be Gaussian).



**Fig. 7** Region shapes learned via KDE using a kernel bandwidth of **a** 25, **b** 15, **c** 10, and **d** 5



**Fig. 8** Example entry regions displaying **a** good directional consistency, **b** bad directional consistency, **c** good interaction consistency, and **d** bad interaction consistency

## 5.2 Region reliability

We now describe how we validate the regions to distinguish reliable entries and exits from those that are the result of noise or areas of difficult tracking. In [16], they compute an entry/exit region density value and remove regions with a density below an arbitrary threshold and such an approach will not work well if the amounts of scene traffic are imbalanced, as entries/exits with low popularity (and thus low density), may be regarded as noise. Further, if tracking is very noisy, this method could potentially classify noisy areas as being good entry/exit regions.

We define a good entry region as one with entity tracks emanating out of it in a mostly directional manner, and a good exit region as one with entity tracks flowing into it in a mostly directional manner. Entry regions whose entry-only entity tracks (or exit regions whose exit-only tracks) exhibit random or bi-directional activity are deemed unreliable regions, and may be the result of areas with a high rate of tracking failure or scene noise (e.g., tracking swaying trees). An illustrative example of entry regions exhibiting good and poor directional consistency are provided in Fig. 8a, b.

Furthermore, for entry regions, we desire that other tracks in the scene not intersect the region in the same emanating direction that defines the region (i.e., the entry region should not be a “through” state) and such a scenario would indicate that another entry region exists close behind. The same concept extends to exit regions. Figure 8c shows an example of an allowable interaction, and Fig. 8d displays an example of an undesirable interaction. Thus, we attempt to capture both the directional consistency of tracks that define each region, as well as the nature of the interaction with other entity tracks.

To measure the directional consistency using the entry/exit entity tracks that define each region, we learn the distribution of directions that these tracks leave (for entries), or enter (for exits), the region. This is accomplished by quantizing the velocity angle at the location that each track intersects the region into one of  $b$  directional bins (we use  $b = 8$ ,  $45^\circ$  bins, in our experiments). To determine the angle of intersection for an entity track and entry/exit region we first locate the observations from the entity track on either side of the intersection location (one observation inside of the region, and one outside), and average the velocity angles from the two track observations. Locating the observations inside the entry/exit regions requires a polygon intersection test. The velocity angle for an entity track observation is computed as an average of the velocity angles of the underlying weak tracking observations (from which the entity was constructed). This angular histogram is normalized to provide the distribution  $q$ . From  $q$ , we compute a directional consistency function  $\hat{q}$ , which accounts for any symmetry (undesired) in this distribution, as computed in the following manner. For bin  $i$  with probability  $q(\theta_i)$ , every other bin probability  $q(\theta_j)$  is subtracted from  $q(\theta_i)$  in a weighted manner such that bin angles that are directly opposite of  $i$  receive high emphasis (as they correspond to bi-directional behavior), and bin angles close to  $i$  receive lower emphasis. For a region  $k$ ,

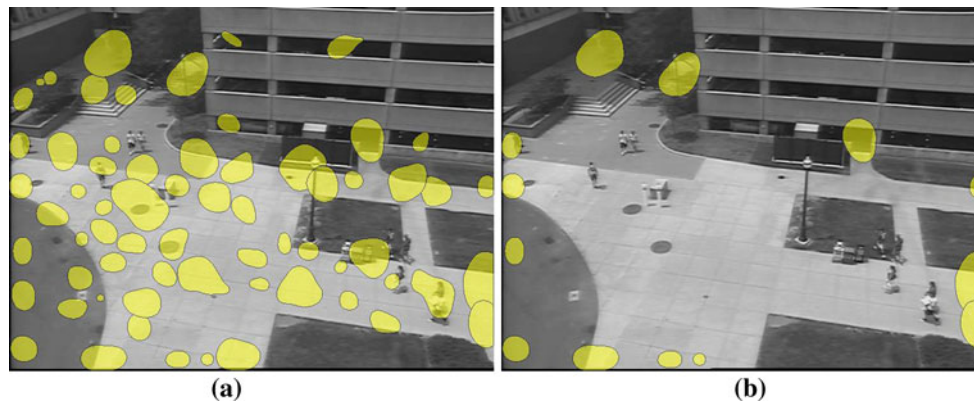
$$\hat{q}_k(\theta_i) = \frac{\max\left[0, \sum_{j=1}^b w_{ij} \cdot (q_k(\theta_i) - q_k(\theta_j))\right]}{\sum_{j=1}^b w_{ij} \cdot q_k(\theta_i)} \cdot q_k(\theta_i) \quad (3)$$

where  $w_{ij}$  is an angle similarity weight that gives more emphasis to angles corresponding to bi-directional behavior with respect to  $\theta_i$ , and is computed as

$$w_{ij} = \begin{cases} \exp(-|1 + \cos(\theta_i - \theta_j)|) & \text{if } \cos(\theta_i - \theta_j) < 0 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Here,  $\theta_j$  is ignored if it is within  $90^\circ$  of  $\theta_i$ , and most heavily weighted when it is exactly opposite of  $\theta_i$ . Thus, when a region exhibits completely directional consistent behavior,  $\sum_{i=1}^b \hat{q}_k(\theta_i) = 1$ . As the activity becomes more directionally inconsistent,  $\sum_{i=1}^b \hat{q}_k(\theta_i)$  approaches 0.

In addition to modeling the directional consistency, we also incorporate the interaction consistency of regions with other activity in the scene. As illustrated in Fig. 8d, an entry region that emanates activity in a particular direction should not be intersected by other entity tracks (not used to form the region) traveling in the same emanating direction (the same logic extends to exits). For an entry or exit region  $k$ , let  $D_k$  be the total number of entity tracks that define the region. Let  $D_k(\theta_i)$  be the number of entity tracks that leave the region (for entries) or enter the region (for exits) at angle  $\theta_i$ . Further, let  $M_k$  be the set of outside entity tracks that intersect



**Fig. 9** **a** Plausible entry regions and **b** reliable entry regions with  $\Psi > 0.75$

region  $k$ , and  $M_k(\theta_i)$  be the number that intersect region  $k$  at angle  $\theta_i$ . If there are many tracks that intersect region  $k$  at the same angle as the tracks that define the region, the region should be regarded as unreliable. This concept can be formulated by the ratio

$$\frac{\sum_{i=1}^b \hat{q}_k(\theta_i) \cdot M_k(\theta_i)}{\sum_{i=1}^b \hat{q}_k(\theta_i) \cdot D_k(\theta_i)} \quad (5)$$

Because the number of intersecting tracks that could discredit a region ( $M_k$ ) approaches the number of tracks that define region  $k$  ( $D_k$ ), the value  $\frac{\sum_{i=1}^b \hat{q}_k(\theta_i) \cdot M_k(\theta_i)}{\sum_{i=1}^b \hat{q}_k(\theta_i) \cdot D_k(\theta_i)}$  will approach and surpass 1.

The interaction consistency ratio can be combined with the previously defined directional consistency measure to create a single reliability score  $\psi_k$  for region  $k$ , computed as

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

Here,  $\sum_{i=1}^b \hat{q}_k(\theta_i)$  is the directional consistency term (Eq. 3) across all angles. This term will be low (approach 0) if the defining tracks leaving an entry, or entering an exit, are not directionally consistent. The interaction consistency (second term) reflects the manner in which other tracks in the scene intersect an entry or exit region. As the number of discrediting tracks grows, this term will approach 0.

The resultant behavioral consistency score  $\psi_k$  is then passed through a sigmoid function, allowing the model to be smoothly adaptive to various noise levels. The final region score,  $\Psi_k$  is computed as

$$\Psi_k = \frac{1}{1 + \exp\left(-\frac{\psi_k - \mu_\Psi}{\sigma_\Psi}\right)} \quad (7)$$

where  $\mu_\Psi$  and  $\sigma_\Psi$  control the centering and sharpness of the sigmoid, and can be set based on the scene noise (e.g., tracking inaccuracies, partial occlusions), which may cause some

of the regions to not adhere to our behavioral modal as strictly. Figure 9a displays a set of potential entry regions, and Fig. 9b shows entry regions with a reliability score  $\Psi > 0.75$ .

## 6 Extension to PTZ camera viewspace

Although many commercial surveillance cameras provide only a single fixed view, most infrastructures employ active PTZ cameras. Such cameras have a pan-tilt motor which gives two degrees of freedom (pan and tilt) to re-orient and view different parts of the scene. Although learning the entry/exit regions for a specific camera view is useful, when the camera is re-oriented, the zones for the new view would have to be learned. In this section, we extend our single-view approach to the entire viewspace of a PTZ camera. To achieve this, we work in the camera's global PTZ space, and incorporate data from many overlapping camera views. There are many other works (e.g., [10, 13]) that associate data between different camera views, often by registering data to a common ground plane. In our work, we choose to work in the camera's PTZ space over a common world coordinate plane for a few reasons. While we do associate data from different camera views, all of the views are from the same PTZ camera. Detecting entry/exit regions with respect to each camera's general 3D *viewspace*, rather than a common world coordinate *plane*, allows the scene behavior to be understood relative to each camera without projection (e.g., perspective) distortions. In a ground plane approach, such distortions may occur if object tracks do not corresponded to the bottom (ground portion) of the object (e.g., feet of people). If required, the learned PTZ scene information could later be transformed to a ground plane using the approach presented in [24], however, it makes the most sense to analyze the scene in the PTZ space as we can directly translate the local image view to the PTZ space without losing information (this would not be the case if a common ground plane were used).



In our approach, we track objects in multiple overlapping local camera views (from the same PTZ camera), learn entities in the local views, and then project the local data into a global camera viewspace where the data can be unified. This requires a model to map each local image track coordinate  $(x, y)$  to the corresponding global pan-tilt orientation  $(\theta, \phi)$  of the camera (a mapping model that provides a means to re-center the camera on a particular pixel coordinate). We next cluster the entity entry and exit observations in the global viewspace and analyze the region behaviors to learn the set of global entry and exits. Clustering these observations in the camera viewspace and analyzing each cluster requires our previous local (Euclidean) approaches to be extended to perform in the global (Spherical) camera viewspace. Once the regions are found in the global viewspace, the entry/exit regions for *any* local camera view can be determined.

### 6.1 Pixel to pan-tilt

As stated earlier, we require a camera model to map local image track coordinates  $(x, y)$  to a global camera pan-tilt  $(\theta, \phi)$  space where the data from different local camera views may be combined. To achieve this, we employ the method presented in [25]. The only camera parameter this model requires is the camera focal length, and [25] provides a method to automatically estimate it, which we leverage in our work. An evaluation of the camera model accuracy is also presented in their work, and while there may be potential for “drift” in the camera motor positioning, we have observed that this model is sufficient for our needs.

For an  $(x, y)$  coordinate in a local image view of a PTZ camera, the change in pan and tilt  $(\delta\theta, \delta\phi)$  required to re-orient the camera from  $(\theta, \phi)$  to focus at that  $(x, y)$  location is computed using the following two equations:

$$\delta\theta = \tan^{-1} \left( \frac{x}{y \cdot \sin \phi + f \cdot \cos \phi} \right) \tag{8}$$

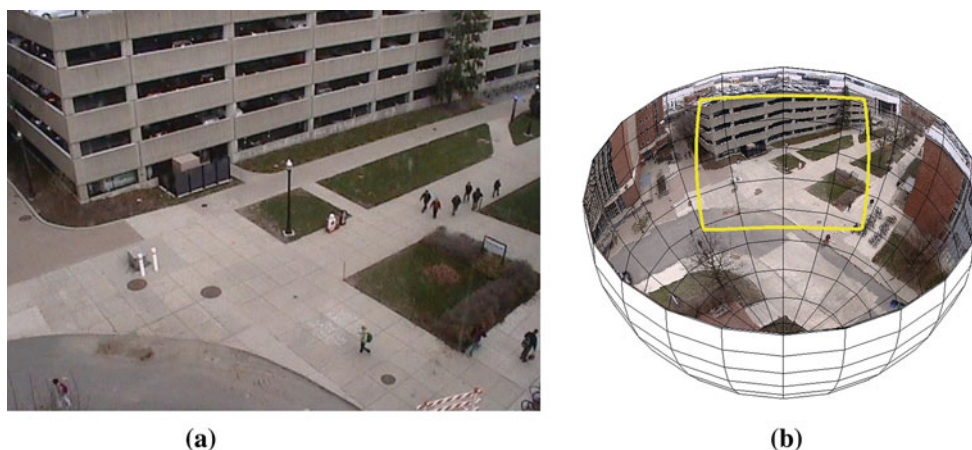
$$\delta\phi = \tan^{-1} \left( \frac{y + a}{f \cdot \cos \left( \tan^{-1} \left( \frac{a}{b} \cdot \frac{x}{y+a} \right) \right)} - \frac{a}{f} \right) \tag{9}$$

Here,  $f$  is the focal length of the camera (learned automatically [25]),  $a = \frac{f}{\tan \phi}$ , and  $b = \frac{a}{\sin \phi}$ . This approach is based on the elliptical intersection of an infinitely extended image plane with the cone carved out by the pan rotation of the camera, and is not affected by the misalignment of the geometric center of the camera pan-tilt motor and the optical center of the camera. Thus, the resulting global orientation in (pan, tilt) for a local image coordinate  $(x, y)$  is computed as

$$\theta_{(x,y)} = \theta_{\text{current}} + \delta\theta, \quad \phi_{(x,y)} = \phi_{\text{current}} + \delta\phi \tag{10}$$

This direct mapping allows us to combine data collected from multiple camera orientations.

The global pan-tilt space of a PTZ camera can be naturally represented as a (lower) hemisphere (with the camera at the sphere center), where each pan-tilt  $(\theta, \phi)$  camera orientation corresponds to a location on the hemisphere. An example is illustrated in Fig. 10, where a local camera view (Fig. 10a) is highlighted on the camera viewspace hemisphere (Fig. 10b, constructed by stitching a set of overlapping local camera views together using [25]). We will use this hemispheric representation for all of the global computations to follow. To visualize and show results calculated with the spherical approach, we employ a projection of the 3D hemispheric data onto a 2D linear spherical panorama (using [25]). The panorama is a projection of the viewspace hemisphere onto the equator plane such that the camera tilt varies linearly from the center of the panorama image. The panorama provides a single view representation of the global 3D camera viewspace for presentation of our results. Thus, we employ the hemisphere for computation and the panorama for visualization. A panorama for the hemisphere shown in Fig. 10b is shown in Fig. 11.



**Fig. 10** **a** Local camera view and **b** the global camera viewspace with the local view highlighted



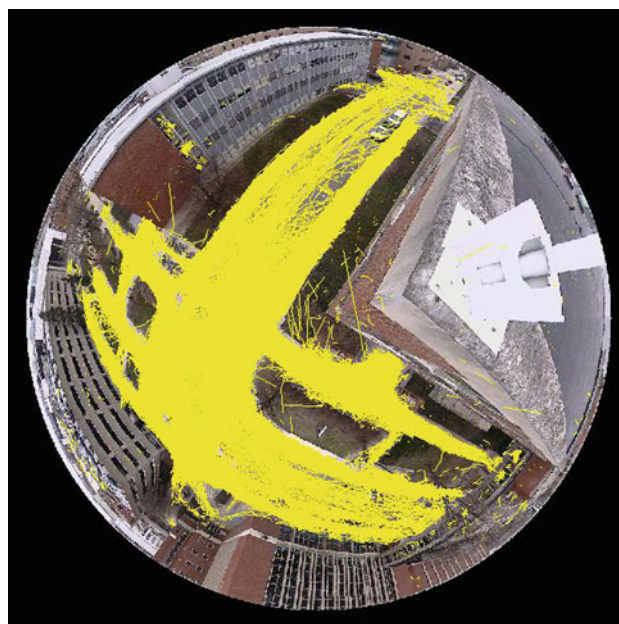
**Fig. 11** Spherical panorama projection

## 6.2 Viewspace data collection

To collect data across the camera viewspace, we first generate a list of PTZ orientations that yield overlapping views and cover the entire viewspace (the camera field-of-view is known). We then randomly sample a view from the orientation set, move the camera to that orientation, and track objects via the weak tracker (same as for the local view approach) for a short duration of time (e.g., 300 frames, roughly 40 s, in our experiments). A different view is then sampled from the collection and the process repeats until all the views are used. We choose a random sampling of the space (rather than sequential) to prevent any bias when observing the scene. We regard this chunk of data as a single “pass” (as it is a pass of the entire viewspace) and employ multiple passes to collect the tracking dataset. For our experiments, we used two different lists of PTZ orientations—one slightly shifted from the first—to provide different overlap among the camera views.

## 6.3 Viewspace region detection

After the tracks are collected from multiple passes, the tracks in each local view are separately formed into entity tracks (using the local approach from Sect. 4). The entity tracks from each local view are then transferred to the global viewspace using the camera model described in Sect. 6.1 (each track observation is mapped to a particular pan–tilt orientation, as shown in the panorama of Fig. 12). This provides a set of entity entry and entity exit observations in the full camera viewspace (on the hemisphere). As before, we wish to cluster



**Fig. 12** Entity tracks in the camera viewspace

these entry and exit observations and then score each cluster by its behavior. To accomplish this, we adapt our previous local approach to function in the spherical pan–tilt space of the camera.

Previously, we clustered entity entries and exits for a local view using mean-shift clustering, where the distance between any two points was the 2D Euclidean distance (on the image plane). To extend this approach to the hemisphere surface, we employ a geodesic approach and define the distance between two points as the length of the great circle arc that connects the points (on the sphere surface). The great circle distance between two points  $p$  and  $q$  on the surface of a sphere is computed as

$$d_{\text{sphere}} = R \cdot \Delta\hat{\sigma} \quad (11)$$

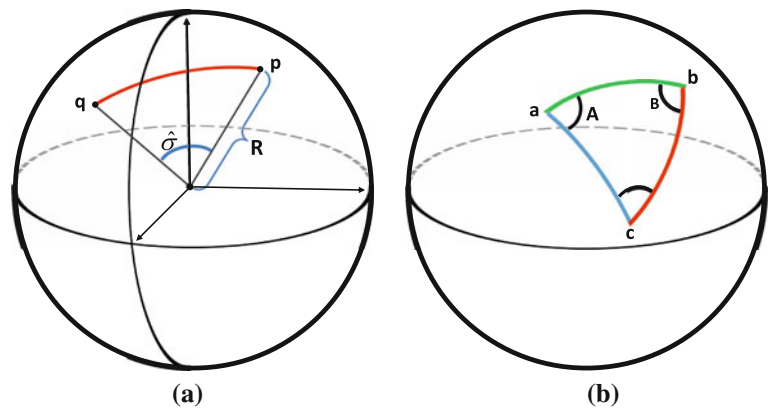
where  $R$  is the radius of the sphere and  $\Delta\hat{\sigma}$  is the central angle between the two points (see Fig. 13a). The central angle may be computed using the Haversine formula [28] as

$$\Delta\hat{\sigma} = 2 \cdot \sin^{-1} \left( \sqrt{\sin^2 \left( \frac{\Delta\phi}{2} \right) + \cos \phi_p \cos \phi_q \sin^2 \left( \frac{\Delta\theta}{2} \right)} \right) \quad (12)$$

The  $\theta$  and  $\phi$  in the Haversine formula typically specify longitude and latitude coordinates, respectively, though they naturally correspond to the pan–tilt coordinates of our camera viewspace. Using this spherical distance, we cluster the entry and exit observations directly in the camera viewspace.

After we obtain the entry and exit clusters, we then remove outliers in each cluster using the convex hull area-reduction technique described in Sect. 5. The convex hull for a cluster

**Fig. 13** **a** Sphere geometry for central angle and **b** spherical polygon



of points on a spherical surface is similar to the convex hull for points on a 2D plane, though the perimeter is made up of great circle arcs rather than line segments. For each entry and exit cluster, we compute a 3D convex hull using the points in the cluster (on the sphere), and the sphere center point. Points in this 3D convex hull that are connected to the sphere center produce the set of points for the convex hull on the sphere surface. The area of the convex hull (spherical polygon) is computed [2] as

$$S = (\Theta - (n - 2) \cdot \pi) \cdot R^2 \quad (13)$$

where  $\Theta$  is the sum of interior angles in the spherical polygon,  $n$  is the number of vertices, and  $R$  is the sphere radius. Each interior angle is computed using the law of spherical cosines. For three points on a spherical surface ( $a$ ,  $b$ , and  $c$ ), let  $A$ ,  $B$ , and  $C$  be the angle of the spherical triangle at each of the three vertices (shown in Fig. 13b). Then, any angle in the triangle ( $A$  for example) can be computed as

$$A = \cos^{-1} \left( \frac{\cos(d_{bc}) - \cos(d_{ab}) \cos(d_{ac})}{\sin(d_{ab}) \sin(d_{ac})} \right) \quad (14)$$

where  $d_{xy}$ , is the great circle distance between points  $x$  and  $y$  on the sphere.

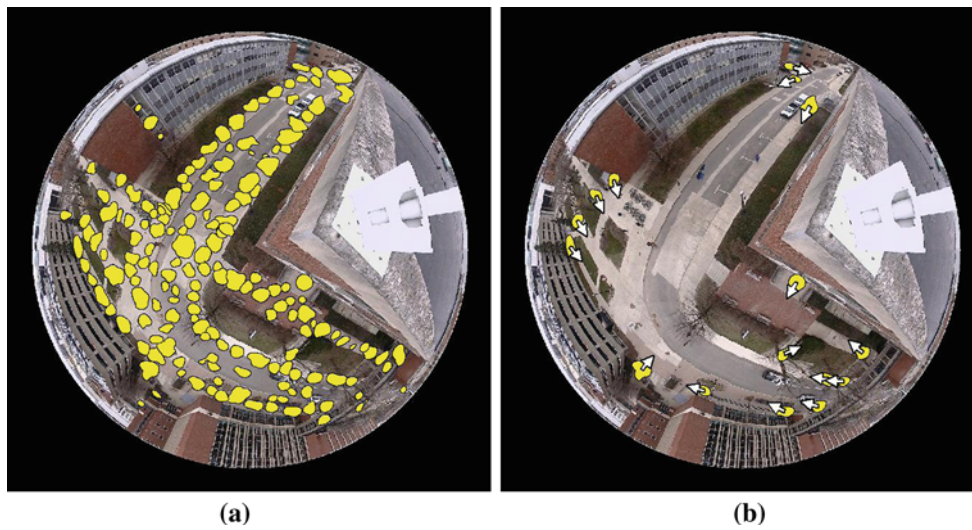
The last extension needed is to determine if a given orientation ( $\theta$ ,  $\phi$ ) on the hemisphere is contained within an entry or exit region (represented as a spherical polygon). This is used to determine the intersection angle between an entity track and entry/exit region, and is achieved via a polygon intersection test (as in the local-view approach). To test whether a point in pan-tilt space is within a spherical polygon, intersection with each polygon edge (arc) is tested. Using the approach described in [1], the arc between the point in question and the North Pole (not in the viewport) is tested for intersection with each spherical polygon edge. An even number of edge intersections indicates that the point is outside the spherical polygon, and an odd number indicates it is inside.

Using the above spherical transformations and calculations, we learn the set of hypothesized entry and exit regions.

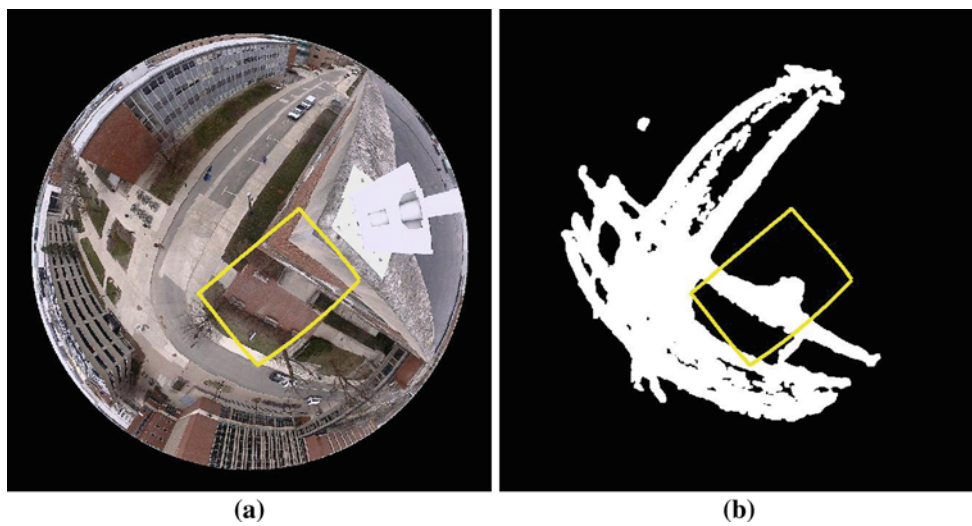
Due to multiple overlapping views, many potential entry/exit regions may be learned (activity at the edge of each local camera view will typically create an artificial entry/exit region). However, our approach is able to eliminate these regions, as tracks from other (overlapping) camera views will pass through these regions, reducing their interaction consistency score (i.e., these regions will resemble through states). By scoring the *behavior* of each region, we can learn the true environmental entry/exit regions (e.g., building doorways) within the camera viewport as well as regions at the border of the viewport. Figure 14a displays a set of potential entry regions in the camera viewport, and b displays the corresponding set of reliable regions (after scoring and thresholding each region). An advantage of this global modeling approach is that regardless of where the camera is oriented within its viewport, a test can be performed to see if any global entries or exits exist within the local camera view (for any view).

In addition, we can use the camera viewport to provide areas of activity that will appear at the borders of a local view. We achieve this by constructing a global “activity mask” that captures common areas of motion in the viewport, and then intersecting the particular local camera view with the viewport activity mask. The camera viewport is first gridded (using a pan  $\times$  tilt grid), and the viewport entity tracks are mapped to the grid. This creates a histogram of activity at each pan-tilt ( $\theta$ ,  $\phi$ ) grid cell. A threshold is then applied to remove bins with low activity. The histogram is then mapped to a panorama image and image morphology techniques are applied (median filter, dilation, connected components). An example panoramic activity mask is shown in Fig. 15.

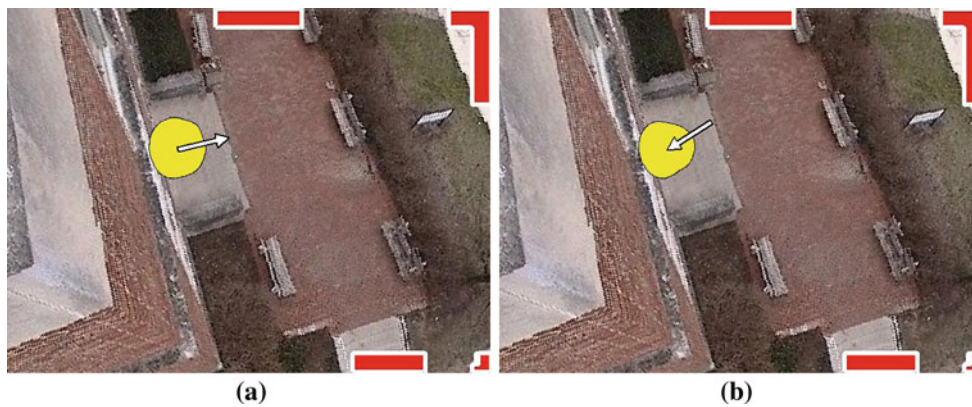
For any local camera view, the intersection of the local view border with the viewport activity mask is used to determine areas of image border activity. These local view border regions are combined with any global entry or exit regions that exist in the local view to create a more extensive summarization of activity for in the local view (an example is shown in Fig. 16 for the view highlighted in Fig. 15). Here,



**Fig. 14** **a** Camera viewspace potential entry regions and **b** reliable entry regions ( $\Psi > 0.75$ )



**Fig. 15** **a** Local camera view highlighted on panorama and **b** viewspace activity mask



**Fig. 16** **a** Viewspace camera entry region and **b** viewspace exit region for a local view with image border activity zones shown

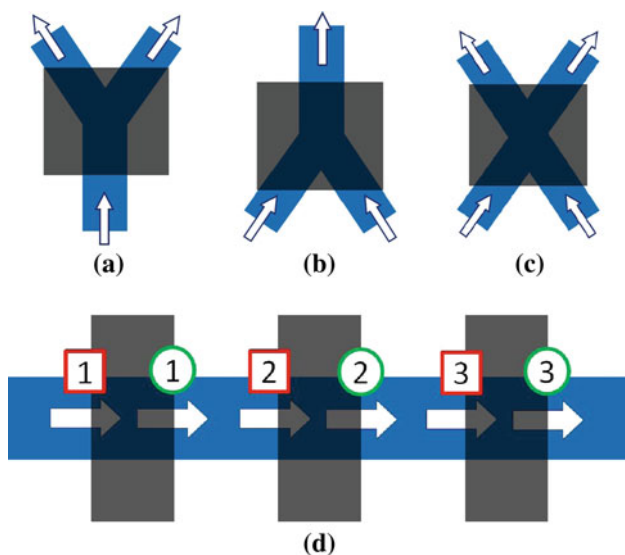
the global entry/exit regions are shown in yellow and the image border activity is shown in red. Again, an advantage of this approach is that global entry/exit zones and border activity can be easily extracted and exploited for *any* local view.

## 7 Exploiting entry and exit regions

Given our methods to detect entry and exit regions (local and global), we now discuss applications of the detected regions. We show that relationships between regions can be used to detect and model activity through occlusions and also to discover common entry  $\rightarrow$  exit pairings within a scene (useful for anomaly detection). The methods to be described are applicable to local camera views, however, the camera view-space could be used if the region pairs exist within a local camera view (temporal correspondence of region activity is needed).

### 7.1 Exit $\rightarrow$ Entry occlusion relationships

Given a set of entry and exit regions, we introduce a method to detect the location of static scene occlusions. An occlusion area may be characterized as an exit region (or set of exit regions) where activity disappears into the occlusion, and a corresponding re-entry region (or set of re-entry regions) where activity reappears in the scene. In the simplest case, an occlusion is captured by one exit region followed by a single corresponding re-entry region. However, more difficult scenarios can occur (Fig. 17a–c). We generalize our model to allow for any number of scene exit regions where activity



**Fig. 17** a–c Various entry/exit occlusion scenarios and **d** occlusion series

disappears behind an occlusion and any number of corresponding scene entry regions where activity reappears from the occlusion.

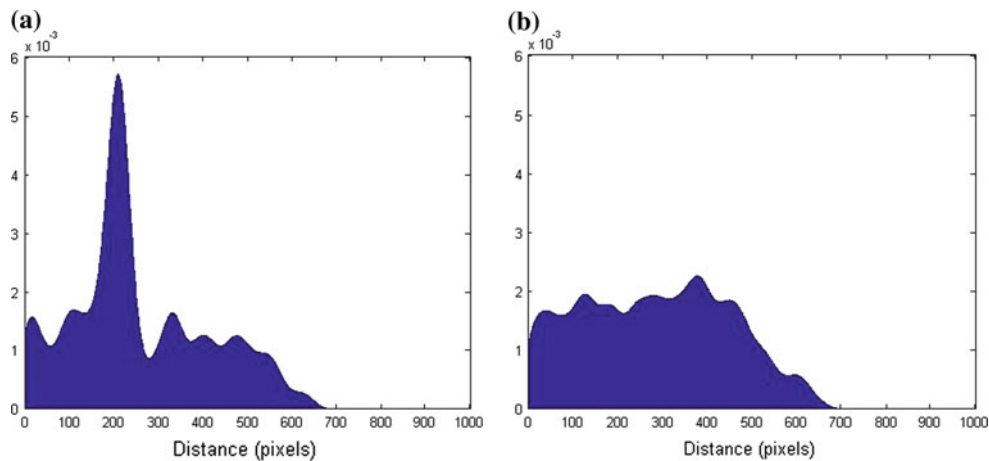
We detect such connections by constructing a graph,  $G_o$ , capturing causal relationships between each possible pair of exit–entry regions in the scene. Let  $G_o(V_o, E_o)$  be a graph where  $V_o = R \cup X$  (the set of entry  $R$ , and exit  $X$  regions), and  $E_o$  is the set of causal relationships in the scene. For each possible exit–entry pair  $(x, r)$ , where  $x \in X$  and  $r \in R$  we do the following. For each event at exit region  $x \in X$ , we look forward in time over a window of  $w$  frames. If an entry event occurs at time lag  $w_i \leq w$  for entry region  $r$ , we estimate the expected pixel distance  $\hat{d}(x, r)$  the entity would have traveled given its velocity and the number of frames between the exit  $x$  and the entry  $r$  observation, computed as  $\hat{d}(x, r) = t_v \cdot w_i$ , where  $t_v$  is the average of the exit and re-entry speeds for the entity. We do this for all entries in region  $r$  that occur within  $w$  frames of the exit event. This distance-based formulation is more robust than using simple time-lagged event correlation, as entities may travel at different speeds (e.g., consider a pedestrian vs. cyclist).

Given a set of distance estimates  $\hat{d}(x, r)$  between an exit and entry region  $(x, r)$  from multiple events across time, we determine if there exists a commonly traveled distance between the two regions. To accomplish this, we employ an entropy approach based on the idea that if there is no occlusion region between  $(x, r)$ , the distribution of distance estimates should be random (no connection between leaving and re-entering). However, if there is an occlusion region between  $(x, r)$ , the distribution of distance estimates should contain a strong mode, represented by a large peak in the distribution (the true distance traveled through the occluded region). Here, we assume that the pathway through an occlusion is structured and that targets pass through at a relatively constant velocity. Example distributions from our experiments are shown in Fig. 18, where the distribution for a true occlusion shows a strong peak (corresponding to the distance traveled by entities through the occluded region).

To compute the distribution entropy, we first construct a histogram of estimated distances (pixels), where each bin corresponds to a particular pixel distance (from 1 to 700 in our experiments). This distribution is then smoothed to generalize the data (we use a Gaussian with  $\sigma = 25$ ). The entropy score  $H(x, r)$  is then computed as

$$H(x, r) = - \sum_{i=1}^n \hat{d}_i(x, r) \cdot \log(\hat{d}_i(x, r)) \quad (15)$$

where  $\hat{d}_i(x, r)$  is the normalized count of observations that have distance  $i$  in the distribution. If  $H(x, r)$  is sufficiently small (peaked, e.g.,  $H(x, r) < H_{\text{thresh}}$ ), we add a directed edge  $x \rightarrow r$  in  $G_o$ . We do this for all possible exit–entry pairings.



**Fig. 18** **a** Estimated pixel distance distribution between an occlusion exit region and corresponding entry region and **b** estimated distance between and exit region and entry region not corresponding to an occlusion

Consider a scenario where occlusions exist in series, one after the other, as shown in Fig. 17d. In such a scenario, exit 1 would link to entries 1, 2, and 3, as they are causally related by traffic that passes through the series of occlusions. Similarly, exit 2 would link to entries 2 and 3. However, we observe that traffic going through exit 1 must go through entries 1 and 2 before it reaches entry 2. Thus, if an exit  $x$  is connected to an entry region  $r$ , but there also exists a path connecting  $x$  to  $r$  going through other regions,  $x \rightarrow r$  must not be a “first-order” relationship, and thus must not correspond to a direct occlusion. We define a first-order relationship as a connection that exists between two regions, A and B, such that there do not exist any alternative paths to get to B from A other than going straight from A to B.

To construct the final graph  $G_o$ , we examine all entry–exit, exit–entry, exit–exit, and entry–entry regions to fully model the possible region relationships in the scene using the same approach to match exit regions to entry regions. For each case, if a forward-in-time causal relationship (peaked distribution) exists between any pair of regions, we add a directed edge in  $G_o$  representing the direction of the relation. Detecting the final set of occlusions is then reduced to searching  $G_o$  for a unique set of first-order paths between exit–entry pairs. We employ a depth-first search [23] (although other graph search algorithms could be used) to find these relationships. The complexity of this is  $\mathcal{O}(n^2)$  for constructing the graph where  $n$  is the greater of the number of entries and exits, and  $\mathcal{O}(v^2 + v \cdot e)$  for searching the graph where  $v$  is the number of vertices (entries plus exits) and  $e$  is the number of edges (region relationships) in the graph. We feel that this is reasonable considering that it is only required when detecting the region relationships for the first time, or updating them.

## 7.2 Entry $\rightarrow$ Exit non-pathway relationships

As we used the connected behavior between regions to locate occlusions, we can examine the formed entry  $\rightarrow$  exit relationships to determine the likelihood of a connection between each entry–exit region pair. This may be used to discover common entry  $\rightarrow$  exit connections in the scene, and to learn a distribution of exit likelihoods for a given entry region. Once these connections are captured, we can also use them to score the likelihood of an object trajectory through the scene.

Given each  $(r, x)$  pairing, we estimate the probability of an object leaving an exit region  $x \in X$  given that the object entered the scene via entry region  $r \in R$  as

$$p(x|r) = \frac{\hat{d}_{\text{mode}}(r, x)}{\sum_{x_c \in X} \hat{d}_{\text{mode}}(r, x_c)} \cdot \left[ 1 - \exp\left(-\frac{H_{\text{max}} - H(r, x)}{\sigma_p}\right) \right] \quad (16)$$

where  $\hat{d}_{\text{mode}}$  is the raw histogram count for the distance voted for most strongly in the distance estimate distribution. Here, the first term compares the connection strength of a particular entry  $\rightarrow$  exit pair  $(r, x)$  to the strength of the same entry  $r$  to every other exit  $x_c \in X$ . The second term,  $1 - \exp\left(-\frac{H_{\text{max}} - H(r, x)}{\sigma_p}\right)$ , captures the reliability of the mode in the distribution. Here,  $H_{\text{max}}$  is the maximum possible entropy score given the number of bins used to model the distribution, and  $\sigma_p$  is a normalization parameter (we use  $\sigma_p = 0.05$ ). Under this formulation, distributions with stronger modes (more observations) receive more weight. Thus, for each entry  $r$ , we have a likelihood (Eq. 16) that it will leave any scene exit  $x \in X$ .

Such a formulation may be leveraged to compute a likelihood score for an observed object trajectory (from a strong tracker). When an object enters a scene entry  $r$ , Eq. (16) gives the likelihood of the object leaving through each scene exit  $x \in X$ . From the estimated distance distribution constructed for  $(r, x)$ , we also have an expected distance that the object will travel between  $r$  and  $x$ . With these, we can score the likelihood of the trajectory that entered entry region  $r$  and exited through exit region  $x$  as

$$p(\text{traj}|x, r) = p(x|r) \cdot \exp\left(-\frac{(\hat{d}_{\text{mag}}(r, x) - |\text{traj}|)^2}{\sigma_t^2}\right) \quad (17)$$

Here,  $|\text{traj}|$  is the length of the object trajectory (pixels),  $\hat{d}_{\text{mag}}(r, x)$  is the expected distance traveled between  $r$  and  $x$  (distance value at  $\hat{d}_{\text{mode}}(r, x)$ ), and  $\sigma_t$  allows for some variance between the expected distance and the actual trajectory distance. For our experiments and image size ( $640 \times 480$ ) we use  $\sigma_t = 50$ , though it could be learned from examples.

This formulation also allows for anomalous tracks to be detected. Such tracks may correspond to an object taking an abnormally long path (meandering) through the scene, an object taking an abnormally short path (such as cutting through a restricted area), an object leaving at an unexpected exit, or an object not leaving any known exit (resulting from a tracker failure or from the object leaving at an unknown exit). We will explore such cases in the experiment section.

## 8 Experiments

In this section, we provide experimental results for our local and viewspace entry/exit region discovery methods and exploitation approaches. First, we provide results for detecting entry and exit regions for different local scenes of varying difficulty, and compare to two existing approaches. We then provide a quantitative analysis of our region shape detection method. Next, we evaluate our global viewspace region detection method using multiple PTZ cameras. Lastly, we present exploitation results for detecting occlusions and entry  $\rightarrow$  exit non-pathway relationships (along with anomaly detection).

### 8.1 Region detection

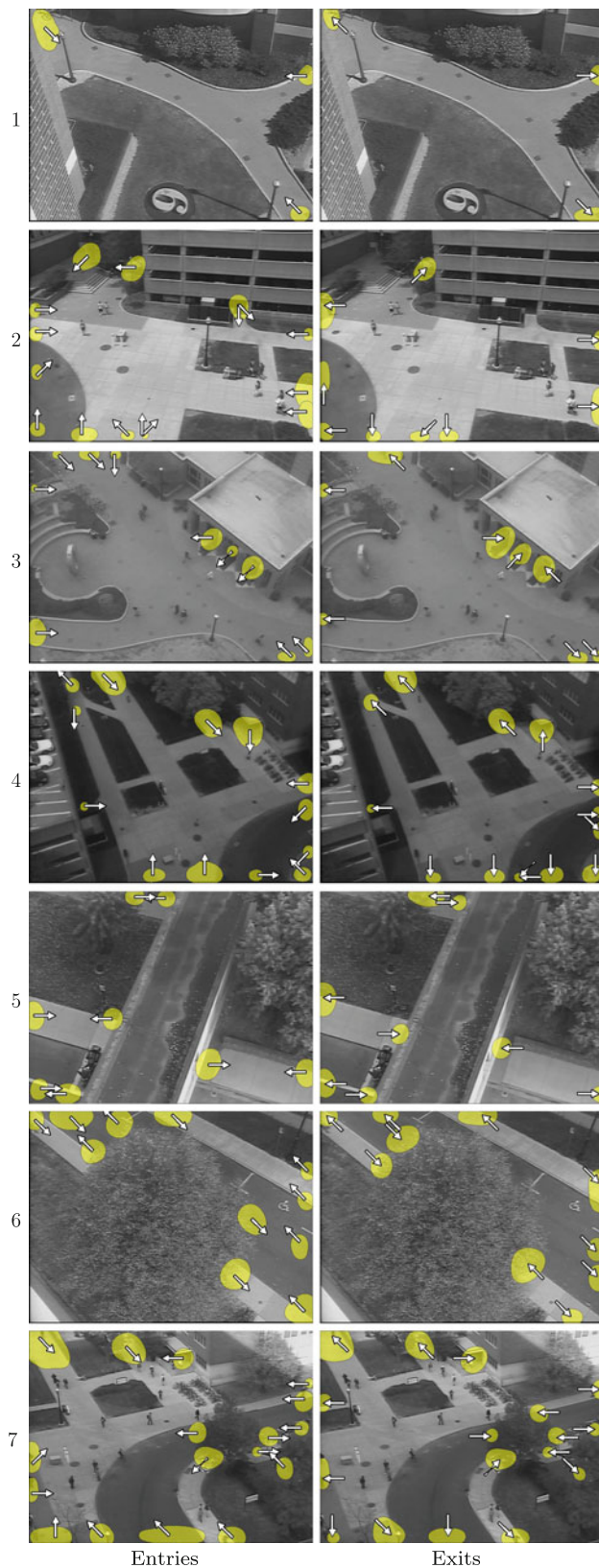
We employed our local single-view entry and exit region detection method on seven scenes of varying difficulty using outdoor surveillance cameras located on four and eight story buildings. All seven scenes were captured at a resolution of  $640 \times 480$  (at 10fps) and were recorded for the durations listed in Table 1. We extracted potential entry and exit

**Table 1** Data collection durations (min) for Scenes 1–7

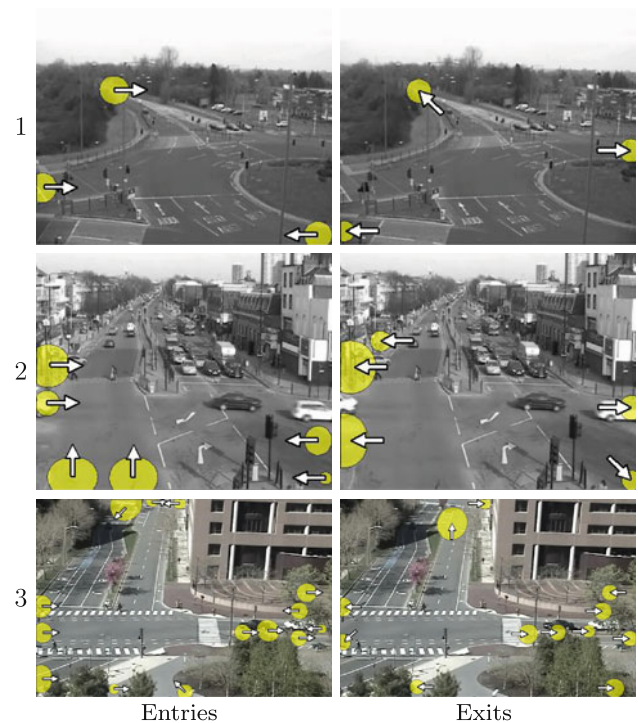
Scene	1	2	3	4	5	6	7
Duration	120	180	180	60	60	120	120

regions for each scene, and kept regions with a reliability score  $\Psi > 0.75$  and having at least 10 tracks leaving/entering each region. The results shown used a kernel bandwidth of  $h = 10$  to cluster entry and exit observations for Scenes 2–4 and 7, and a kernel bandwidth of  $h = 15$  for Scenes 1, 5, and 6 (as they have a more zoomed-in view). In addition, when performing our convex hull area-reduction technique to remove outliers, we chose a threshold of  $\sigma_r = 1.5$  (see Sect. 5.1). When computing the final region score  $\Psi$  for each entry and exit region, we used parameter values  $\mu_\Psi = 0.04$  and  $\sigma_\Psi = 0.10$  (see Eq. 7). The results for Scenes 1–7 are shown in Fig. 19. For each scene, we display the results with the extracted region shapes and an arrow corresponding to the most prominent direction of motion out of (for entries) or into (for exits) each detected region.

In Scene 1, we were able to detect all three expected entry and exit regions corresponding to activity moving across the sidewalk. In Scene 2, we were able to detect all of the expected entries and most of the expected exits. For the entry regions, we split a few of the regions along the bottom of the image which would likely be merged semantically. For the exits, we fail to learn the region in the top-left due to unreliable tracking in that part of the scene (due to camera perspective and shadows). We also fail to learn a region corresponding to motion going into the parking garage due to a larger formed region intersecting with outside trajectories making it a through state. In Scene 3, we were able to detect all of the expected entry and exit regions. We learn three regions entering into the building and three regions exiting (corresponding to three doors). We also detect an entry and exit region in the top-left of the scene corresponding to a small walkway even though it is infrequently traveled with respect to the other areas in the scene. We also detect most of the expected regions in Scene 4 and are able to learn the sparse activity coming and going from the parking garage on the left of the scene. The one region we do have a problem with is the top-left walkway of the scene. We learn an incorrect exit region and two incorrect entry regions. This can be attributed to a low-contrast shadow area on the sidewalk where our weak tracker did not perform well. In Scenes 5 and 6, we learn all of the expected regions. Both scenes exhibit an occlusion (bridge walkway in Scene 5, and large tree in Scene 6) and we learn entry and exit regions at locations where people walk behind and re-appear from these occluding structures. The other interesting aspect in Scene 5 is that the regions tend to form on the right-side of the walkways, capturing the manner that pedestrians typically travel.



**Fig. 19** Detected entry and exit regions for Scenes 1–7



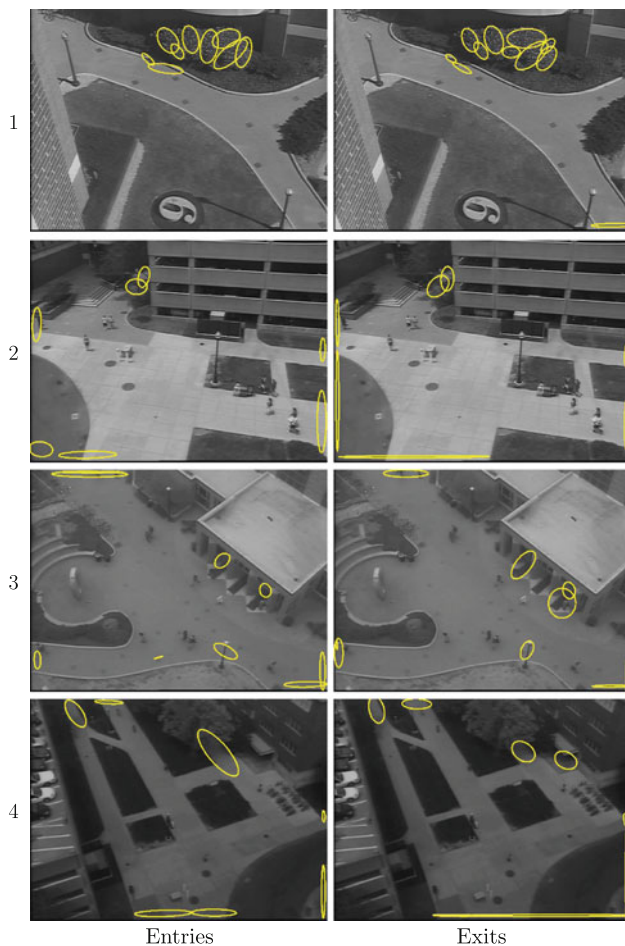
**Fig. 20** Detected entry and exit regions for public datasets 1 (roundabout), 2 (junction), and 3 (MIT)

Lastly, we detect the expected regions in Scene 7. We learn entry and exit regions corresponding to activity on sidewalks that intersects the edge of the view, as well as activity coming and going from the building in the top-right of the scene.

We also employed our method on three public datasets using the same parameters from the previous experiment. Two of the datasets (roundabout and junction) are from [14], and the third dataset (MIT) is from [31]. The results are shown in Fig. 20. In the roundabout scene (row 1), we were able to detect the expected entries for traffic moving into the roundabout (moving clockwise in the scene), and the expected exits for traffic leaving the roundabout. In the junction scene, we detect entry regions for traffic in both lanes of traffic entering from the bottom left of the view. We also detect regions for traffic entering from the left and the right of the scene, including their corresponding exit regions. We failed to detect regions in the top of the view due to a large change in perspective as cars drive away into the distance. In the MIT Scene we are able to detect entry and exit regions for traffic on the left and top of the scene. Our approach has difficulty on the right side of the scene due to several tree occlusions which cause many tracking failures in this part of the scene.

For comparison, we evaluated our results against the methods described in [16,30] on four different scenes. The

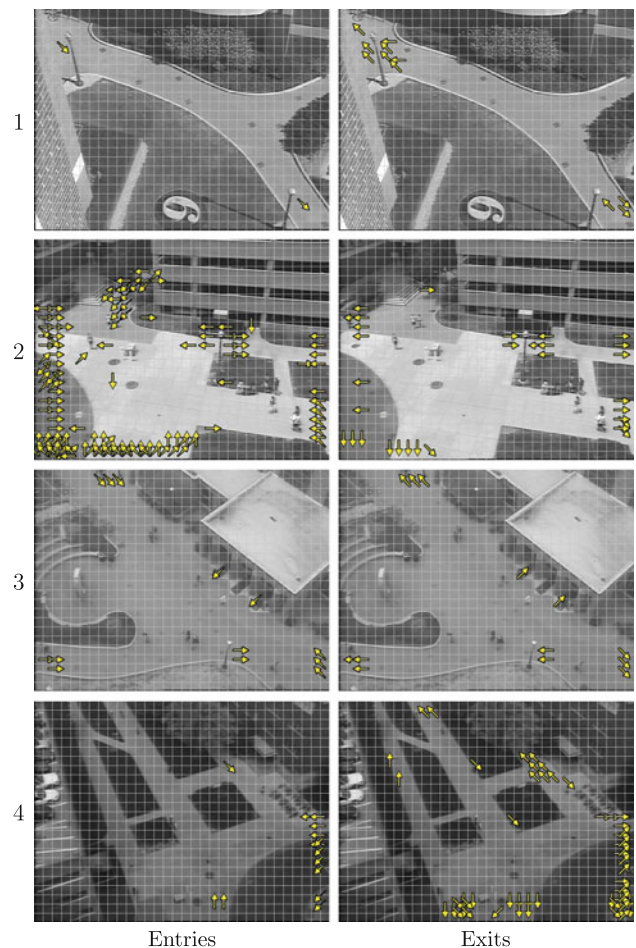




**Fig. 21** Entry and exit regions using the method in [16]

approach from [16] uses a MoG clustering approach to detect regions and a density-based scoring metric to remove unreliable clusters. The approach from [30] employs a grid-based method to detect entry and exit “states” in a scene.

In [16], an EM-based MoG approach is used to cluster trajectory start and end points to obtain a set of potential entry and exit regions (described by Gaussian ellipses). To determine which clusters to retain they use a density metric defined as  $W/E$ , where  $W$  is the percentage of points belonging to the cluster and  $E$  is the area of the Gaussian ellipse. Here, the area is computed as  $E = \pi \cdot I_1 \cdot I_2$  where  $I_1$  and  $I_2$  are the eigenvalues of the covariance matrix of region points. When employing this approach, we used our *entity* tracks to provide a fairer comparison (using weak tracking observations would be too noisy for this approach). When clustering each entry and exit set, we chose a large number of initial clusters (25) for each scene (as in [16], although one could potentially use a method such as BIC [26] to automatically estimate the number of clusters). We kept clusters with weight  $W/E > 2e^{-5}$  (produced the best results). The results for Scenes 1–4 are shown in Fig. 21, with the reliable entry/exit clusters plotted as ellipses.



**Fig. 22** Entry and exit regions using the method in [30]

The approach had difficulty distinguishing reliable entry/exit regions from ones that resulted in tracking noise in many of the scenes. We acknowledge that a later method proposed to remove some of this noise was presented in [17], which attempts to remove trajectories that start and end in the same local region. In comparison, our approach does not threshold and remove tracks, but examines general region-based behavioral qualities. In Scene 1, the approach failed to detect any of the expected entry regions and only detected one of the three expected exits. Most of the regions detected resulted from noise due to bushes swaying in the wind. Our method was able to distinguish such regions as being unreliable by analyzing the behavior consistency of each region. In Scene 2, the approach from [16] over-clustered the entry/exit region near the top-middle of the scene. Although they do detect many of the expected regions, they fail to detect activity coming and going from the parking garage. They also miss the region in the top-left (we detected it as an entry but not an exit).

In Scene 3, the method from [16] was able to detect many of the expected entry and exit regions, though it failed to learn the middle doorway of the building and the walkway in the

top-left of the scene. In addition, it detected motion around the streetlight as an entry and exit region (due to tracking gaps occurring around this location).

In Scene 4, the approach was able to detect most of the expected regions. Although the method failed to detect the region corresponding to activity coming and going from the parking garage and the entry region corresponding to the building doorway (which we detect), they are able to detect an exit region in the top-left of the scene which we miss (due to tracking problems in this part of the scene caused by perspective and shadows).

In general, choosing a robust means to distinguish reliable from unreliable regions with the density-based approach in [16] was difficult. Changing the density threshold can drastically alter the results, and it is also unclear how robust such a threshold is as it had difficulty generalizing across all four scenes (it performed very poorly for Scene 1). Although we used two sets of kernel bandwidth values in our approach, the results do not change significantly if a constant bandwidth is used. Further, the bandwidth value could conceivably be learned (as a function of the size of the objects moving through the scene) by observing scene activity.

We also compared our approach to [30]. In this method they partition the scene into a grid of states, where each state is defined by a grid cell location and motion direction (8 possible directions). They also use a binary activity mask to constrain entry/exit states to be on the border of the scene activity, although we do not employ this part of the technique to allow for a fair comparison across approaches (neither our method nor [16] have such a constraint). They map each weak track duration to a set of states, and then compute an entry and exit weight score for each state. The entry and exit weights ( $W_E$  and  $W_X$ ) for state  $s_i$  are computed as  $W_E = C_{\text{start}} \cdot \max\left(0, 1 - \left(\frac{C_{\text{in}}}{C_{\text{start}}}\right)\right)$  and  $W_X = C_{\text{stop}} \cdot \max\left(0, 1 - \left(\frac{C_{\text{out}}}{C_{\text{stop}}}\right)\right)$ . Here,  $C_{\text{start}}$  is the number of weak tracks that start in state  $s_i$ ,  $C_{\text{stop}}$  is the number of weak tracks that stop in state  $s_i$ ,  $C_{\text{in}}$  is the number of weak tracks that transition into state  $s_i$ , and  $C_{\text{out}}$  is the number of weak tracks that transition out of state  $s_i$ . Entry and exit states with a low weight score are removed. We retained states with at least 0.75 of the max entry/exit weight for the scene. The results for Scenes 1–4 can be seen in Fig. 22, where the final entry/exit states are denoted with arrows (showing the motion direction of the state).

This approach failed to detect most of the entry activity in Scene 1, although was able to detect most of the exit activity (detecting multiple states per region). In Scene 2, the approach detected states that result from the streetlight occlusion, and also learned noisy states in the middle of the scene.

The results for Scene 3 are very reasonable, though noisy states are detected around the streetlight occlusion, and the entry/exit near the top-left of the scene and the entry/exit

corresponding to the middle doorway of the building are not detected.

In Scene 4, the approach fails to detect most of the entry activity in the left-half of the scene. Although they are able to detect a fair amount of valid exit states, they miss the parking garage entry/exit and the building doorway entry/exit.

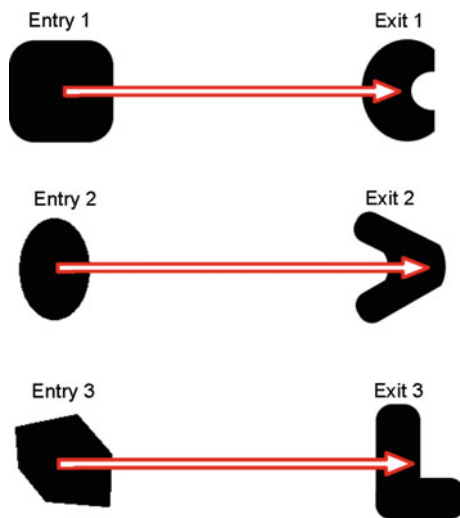
In general, the approach from [30] tended to detect some unexpected regions (resulting from noise), and it also failed to detect many desired regions. As much of the difficulty for this method involves not detecting expected states (with some exception in Scene 2), introducing their motion mask to constrain entry/exit states near the border would not greatly improve their results. It is also unclear how the states should be grouped after they are detected to represent entry/exit regions. Overall, our approach produced the best qualitative results across the scenes in comparison to the two alternative methods.

## 8.2 Quantitative evaluation

The issue of ground truth validation for entry and exit regions has not really been studied. This can be attributed to the difficulty of quantifying such work. A “ground truth via consensus” approach could be employed by asking subjects to manually mark the entry and exit zones. However, subjects would likely be concentrating solely on the scene structure and ignoring the actual behavior (otherwise they would be required to watch very long video recordings). Hence, they may mark a doorway that is never used or miss an entrance or exit that results from a shortcut traveled by pedestrians. In addition, subjects may mark the width of a sidewalk that intersects the image border as an exit when the exit does not really span the entire width of the sidewalk (due to people generally walking on the right-side of the sidewalk). Such subtleties can only be revealed when the behavior of the scene is carefully analyzed. The author from [29] also shares our view and states that while some entry/exit locations are more obvious, there are others that cannot be manually labeled without analyzing tracking data for a scene. However, they fail to comment any further on ground truth evaluation. Here, we offer a quantitative evaluation of our method for learning the entry/exit region shapes, as well as an analysis of the amount of data required to detect the regions shapes reliably.

We designed an experimental framework to examine the shape of the regions that we learn as compared to ground truth with respect to varying amounts of scene noise. To do so, we manually created three entry–exit region pairs (shown in Fig. 23) and generated synthetic weak tracks between each pair.

For each synthetic “object” moving between an entry and exit region, we first create a base track by randomly sampling a starting location within the entry region and a correspond-





**Fig. 23** Synthetic entry and exit regions

ing ending location in its exit region. Using a pre-defined circle around the base start and end locations, we randomly generate 1–3 additional tracks (within the circles and within the region). These additional tracks are generated with the same constraint as our actual KLT tracker, where weak tracks cannot be too close to each other. We repeat this process 1,000 times for each entry  $\rightarrow$  exit region pair to simulate 1,000 objects moving between the regions. We then fragment the weak tracks for eight different noise levels (1–8) by adding gaps to the synthetic tracks for each object. Here, the noise level corresponds to the average number of gaps per track for each object (e.g., for noise level 3 each track contains 3 gaps on average). When adding each gap, a random location, gap size, and track for the object are chosen. We also ensure that at least one track segment for the object is retained at the ends (i.e., gaps are not added to the beginning or ending of all the tracks for an object).

From this process, we obtained eight different realistic weak track sets (one for each noise level) for each of the three entry–exit pairs. We then ran our entry/exit detection method using these synthetic weak tracks. From this we formed entities, learned the potential entry/exit regions, scored the regions, and kept the regions with a reliability score  $\Psi > 0.75$ . We then compared the detected entry and exit regions with the actual ground truth regions used to create the weak tracks. For our method we used a kernel bandwidth of  $h = 25$  to cluster the entity entry/exit observations ( $h = 10$  when constructing the KDE surface to learn the region shape), a threshold of  $\sigma_r = 2.5$  to remove outlier observations, and  $\mu_\Psi = 0.4$  and  $\sigma_\Psi = 0.10$  when computing the region score. The results for each entry–exit pair are summarized in Tables 2, 3, and 4. For each entry–exit pair, we display the F1 score, precision, and recall, computed at the sub-pixel level by comparing the intersection between the ground truth region polygon and the learned region polygon (each averaged over five trials) capturing how well our detected regions compared to ground truth. We also provide a signal-to-noise ratio (SNR), computed as the ratio of the number of ground truth entry/exit observations to the total entry/exit observations in the scene (averaged over five trials), for each case.



In all cases, our method was able to detect the correct number of entries and exits (for each trial). As the number of gaps per track was increased from 1 to 8 (making weaker tracks), our method performed expectedly worse. Though our precision scores were still high as the noise increased, the recall scores were impacted by noise. This can be attributed to the noise starting to approximate the density of the ground truth entry/exit observations, resulting in problems for our outlier removal process. In essence, as the noise increases, the detected regions are eroded. Our scores are noticeably worse for exit 2. This can be attributed to the region having a “concavity” filled with sparse noise, since our algorithm relies on a “convex” hull approach (exit 1 has a concavity on

**Table 2** Quantitative results for Entry and Exit 1

Entry	Noise	F1/SD	$P$	$R$	SNR	Exit	Noise	F1/SD	$P$	$R$	SNR
	1	0.975/0.010	0.983	0.968	39.683		1	0.948/0.008	0.911	0.987	39.683
	2	0.975/0.010	0.976	0.975	6.188		2	0.926/0.029	0.882	0.978	6.188
	3	0.947/0.030	0.934	0.966	2.357		3	0.893/0.068	0.827	0.981	2.357
	4	0.900/0.046	0.990	0.829	1.253		4	0.894/0.026	0.861	0.935	1.252
	5	0.841/0.038	0.985	0.737	0.755		5	0.856/0.057	0.932	0.805	0.755
	6	0.621/0.309	0.965	0.514	0.501		6	0.784/0.076	0.960	0.669	0.501
	7	0.402/0.174	0.978	0.269	0.370		7	0.697/0.068	0.977	0.546	0.370
	8	0.300/0.183	0.981	0.192	0.304		8	0.714/0.028	0.942	0.578	0.304



Scores for F1, precision ( $P$ ) and recall ( $R$ ) are averaged over five trials

**Table 3** Quantitative results for Entry and Exit 2

Entry	Noise	F1/SD	<i>P</i>	<i>R</i>	SNR	Exit	Noise	F1/SD	<i>P</i>	<i>R</i>	SNR
	1	0.962/0.036	0.948	0.980	43.478		1	0.759/0.157	0.637	0.995	43.478
	2	0.945/0.042	0.902	0.996	6.702		2	0.750/0.106	0.617	0.992	6.702
	3	0.960/0.023	0.936	0.987	2.550		3	0.666/0.065	0.619	0.881	2.550
	4	0.941/0.049	0.905	0.984	1.286		4	0.570/0.104	0.966	0.413	1.286
	5	0.939/0.019	0.960	0.924	0.795		5	0.543/0.091	0.949	0.386	0.795
	6	0.911/0.022	0.925	0.905	0.534		6	0.438/0.092	0.940	0.291	0.534
	7	0.815/0.045	0.957	0.714	0.393		7	0.390/0.121	0.927	0.255	0.393
	8	0.777/0.076	0.919	0.696	0.316		8	0.306/0.067	0.890	0.187	0.316

Scores for F1, precision (*P*) and recall (*R*) are averaged over five trials

**Table 4** Quantitative results for Entry and Exit 3

Entry	Noise	F1/SD	<i>P</i>	<i>R</i>	SNR	Exit	Noise	F1/SD	<i>P</i>	<i>R</i>	SNR
	1	0.955/0.011	0.931	0.980	37.594		1	0.951/0.005	0.908	0.998	37.594
	2	0.957/0.011	0.949	0.965	6.676		2	0.947/0.008	0.907	0.991	6.676
	3	0.945/0.014	0.954	0.940	2.447		3	0.921/0.018	0.934	0.910	2.447
	4	0.923/0.028	0.913	0.942	1.298		4	0.868/0.019	0.879	0.874	1.298
	5	0.850/0.055	0.930	0.810	0.771		5	0.828/0.033	0.938	0.747	0.771
	6	0.845/0.070	0.948	0.774	0.518		6	0.706/0.064	0.961	0.563	0.518
	7	0.790/0.071	0.908	0.722	0.386		7	0.688/0.064	0.932	0.555	0.386
	8	0.567/0.114	0.957	0.416	0.309		8	0.603/0.017	0.932	0.446	0.309

Scores for F1, precision (*P*) and recall (*R*) are averaged over five trials

the opposite side where no tracks enter the region). This concave-related problem could be corrected by replacing our convex hull area-reduction approach with a representation that could accommodate concave shapes (e.g., such as alpha shapes [7]). However, we do not typically encounter such concave entry/exit regions in real scenes.

We next examined the effect of varying the number of underlying objects. We chose a fixed noise level (3) and a set of synthetic tracks representing the motion of 1,000 objects for that noise level (from the previous experiment). We then defined five object sets (containing 25, 50, 100, 250, and 500 objects) and randomly sampled the tracks from the full collection for each object set ten times (5 sets  $\times$  10 trials). We ran our entry/exit region detection method for each trial set of objects. The results for entry  $\rightarrow$  exit 3 are presented in Table 5. The results with the complete set of 1,000 objects (representing the complete set of tracks from which each other set was sampled) are also provided as a reference.

As the number of objects was increased (from 25 to 1,000), we were able to recover the underlying entry and exit shape with greater accuracy (F1 score) and reliability (less vari-

ance between trials). The exit shape in this experiment (the ‘L’) was a more difficult shape for our method to learn than the corresponding hexagonal-shaped entry region. The recall score was generally high for all object counts. This can mostly be attributed to the objects being uniformly sampled from within each region. Without many samples, however, it is more difficult to determine the object border. This is reflected by the (generally) lower precision score for smaller object sets. Overall, our algorithm was able to accurately recover the shape of the underlying entry/exit regions with increased performance as the quantity of entry/exit observations increased.

### 8.3 PTZ viewport

Next, we ran various experiments to examine the spherical viewport extension of our region detection method. We present entry and exit detection results for three different PTZ cameras and a comparative result of the regions learned from a local scene with the corresponding area on the camera viewport.

**Table 5** Object count variation results for Entry and Exit 3

Entry	Objects	F1/SD	<i>P</i>	<i>R</i>	Exit	Objects	F1/SD	<i>P</i>	<i>R</i>
	25	0.798/0.179	0.838	0.839		25	0.692/0.200	0.612	0.852
	50	0.816/0.142	0.780	0.920		50	0.736/0.105	0.620	0.942
	100	0.844/0.077	0.760	0.969		100	0.755/0.120	0.632	0.974
	250	0.896/0.034	0.830	0.978		250	0.791/0.088	0.681	0.968
	500	0.910/0.027	0.868	0.962		500	0.821/0.012	0.757	0.902
	1000	0.940	0.99	0.89		1000	0.90	0.940	0.866

The F1, precision (*P*), and recall (*R*) scores are averaged over ten trials for object quantities 25–500. The values for 1,000 objects represent the full set of data

### 8.3.1 Viewspace region detection

We tested our viewspace entry/exit region detection algorithm on three outdoor PTZ cameras located on 4 and 8 story buildings. We first collected data for each camera as outlined in Sect. 6.2. Each camera’s viewspace was sampled multiple times (several passes). The number of passes for each camera along with an estimated total collection duration (at 10 fps) is provided in Table 6. We used a kernel bandwidth of  $h = 0.02$  to cluster entry and exit observations on a unit sphere for Camera’s 1 and 2 (on a 4-story building) and a kernel bandwidth of  $h = 0.01$  for Camera 3 (on an 8-story building). We used a threshold of  $\sigma_r = 2.0$  to remove outliers from each cluster, and set  $\mu_\psi = 0.4$  and  $\sigma_\psi = 0.05$  when computing reliability scores. We kept regions with a reliability score  $\Psi > 0.75$  and having at least 10 tracks leaving/entering each entry/exit region. The results are shown in Figs. 24 and 25.

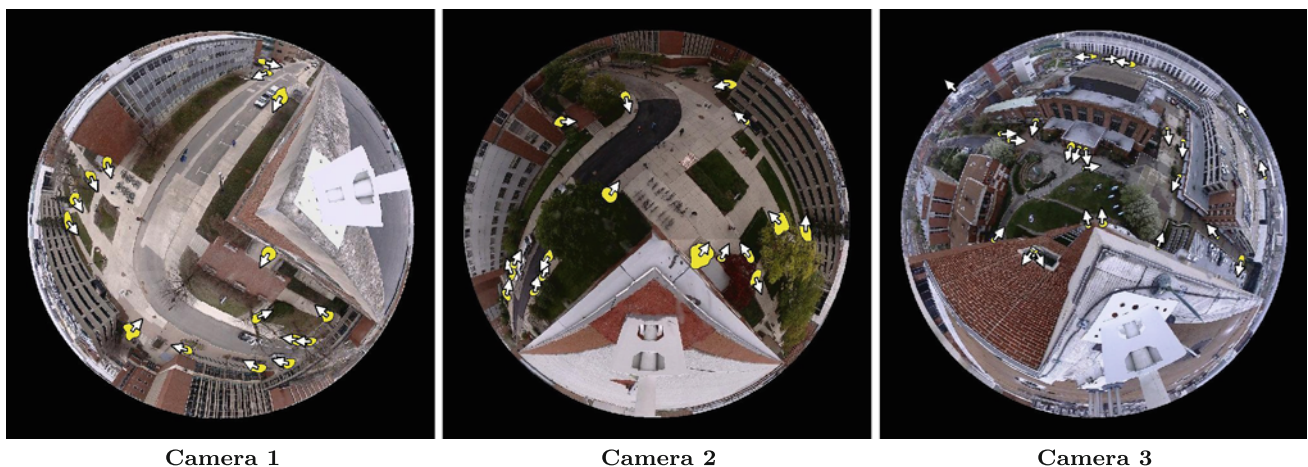
For Camera 1, we detected most of the expected entry and exit regions. We are able to learn regions corresponding to activity entering and exiting two of the buildings, as well as

**Table 6** PTZ viewspace data collection durations (min) for cameras 1–3

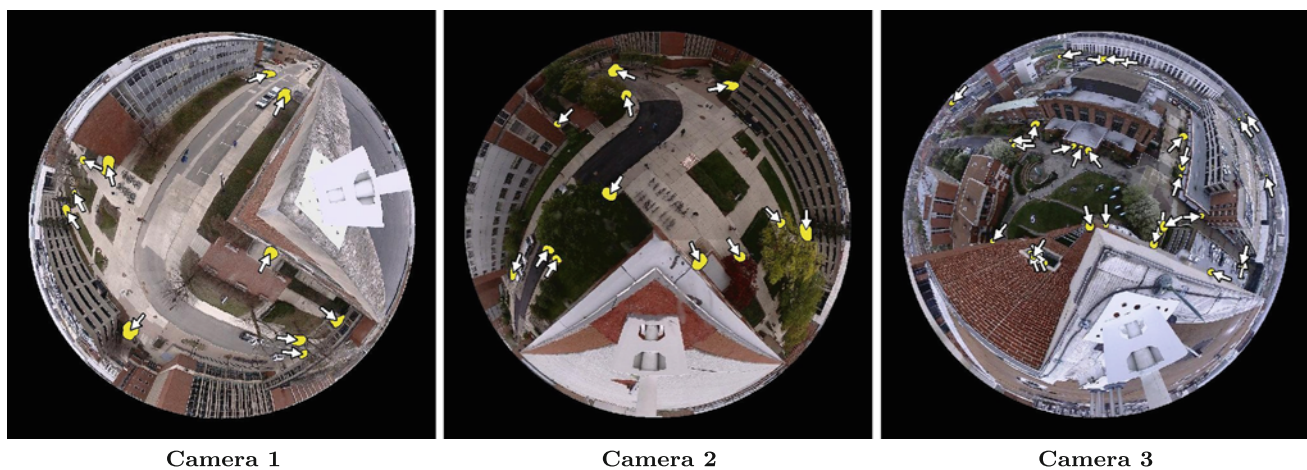
Camera	Passes	Duration (min)
1	14	425
2	8	243
3	8	243

activity entering and exiting the camera viewspace periphery. The regions on the edge of the camera viewspace correspond to the farthest areas from the camera that we can track reliably with our weak tracker (at a fixed zoom). Objects moving in distant areas (especially moving toward or away from the camera) are not tracked well due to the minimum frame-to-frame displacement constraint in the motion-based tracker. In addition, we also learn a few regions due to occlusions (e.g., trees, streetlight), but fail to learn an entry and exit region corresponding activity of the building seen in the bottom of the camera panorama. Again, this is due to tracking difficulty.

For Camera 2, we are also able to learn many of the expected entry and exit regions. We again learn regions cor-



**Fig. 24** Viewspace entry regions for Cameras 1–3



**Fig. 25** Viewspace exit regions for Cameras 1–3

responding to the entry and exit behavior for two of the buildings in the camera view space. We also learn regions around two tree occlusions. For Camera 3, we are able to detect entry and exit regions corresponding to the three doorways of the building in the center of the view space. In addition, we learn an entry and exit region corresponding to a tunnel walkway to the left of the building entrance and a doorway coming out of the building directly below the camera. We also learn regions on a window (on the building below the camera) due to the ground traffic being reflected on the window. Lastly, we also learn a few noisy regions on the periphery of the camera view space (e.g., from traffic several blocks away).

We are able to learn most of the expected regions in each camera view space, although we do occasionally learn noisy regions and miss some expected regions. We found that strong camera perspective prevents us from learning reliable regions near the periphery of the camera view space, which may be addressed using (1) adjustable tracking parameters that vary with camera tilt, or (2) a zoom-adaptive sampling of the space (we used a constant zoom level).

### 8.3.2 Local versus view space comparison

We also ran an experiment to compare results attained from our local fixed-view approach to our view space detection method. Using one data set (Scene 3 from our local fixed-view experiments) we first detected the regions in the local camera view and then for comparison we projected the local view entity tracks to the camera view space (Scene 3 was collected from a single view of a PTZ camera), and ran our view space region detection algorithm. Comparative results are shown in Fig. 26, where the view space results are displayed inside the highlighted local view on a panorama.

As shown, we were able to learn the same set of regions in the view space that we learned locally. The learned region shapes are quite similar between the two spaces (the local

space is an approximation of the global view space). The main direction of motion in and out of each region is also very accurate between each space. It is worth noting that we increased the angular histogram bin count used to capture the directional and interaction consistency for each region (Eq. 7) from 8 to 16 for the view space approach to more accurately compute and compare the main motion direction of the regions.

### 8.4 Region exploitation

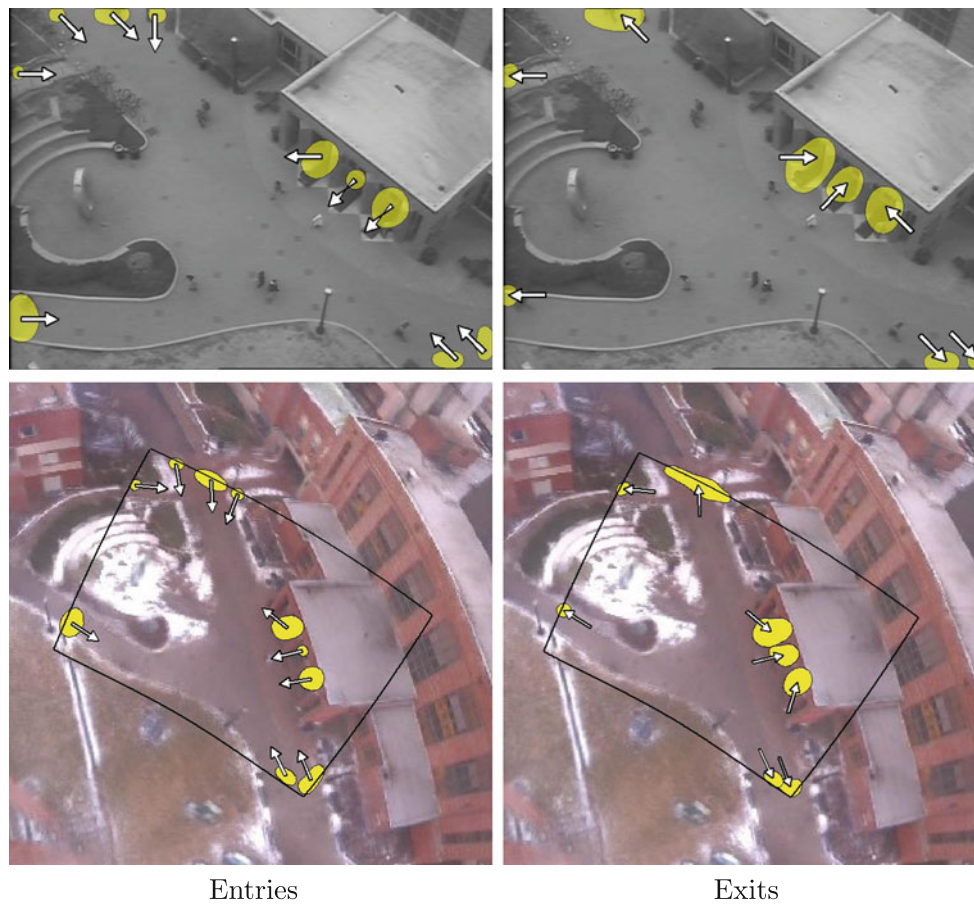
In addition to evaluating the detected entry/exit regions, we also evaluated approaches that exploit the relationships between the regions (as described in Sect. 7).

#### 8.4.1 Occlusions

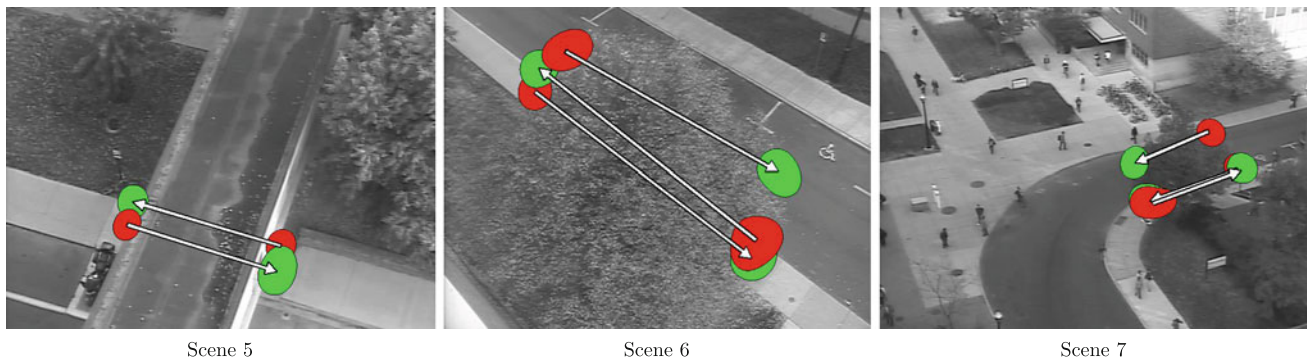
We tested the proposed exit  $\rightarrow$  entry occlusion detection method (Sect. 7) on Scenes 5, 6, and 7 (as they contain natural occlusions). We used an entropy threshold of  $0.98 \cdot H_{\max}$  for all four scenes, where  $H_{\max}$  is the maximum possible entropy score for the distribution (for our histograms  $H_{\max} = 9.45$ ). The results are presented in Fig. 27.

In Scene 5, we successfully learned the bridge occlusion. In Scene 6, we successfully learned the tree occlusion on the sidewalk and the tree occlusion on the street in the direction of the one-way traffic. In Scene 7, we learned the tree occlusion in both directions on the near sidewalk and in one direction on the one-way street.

To demonstrate the use of these occlusion region ties to aid strong tracking, we ran a covariance tracker [21] with the addition of the learned occlusion model. The tracker was set up to automatically initialize on motion occurring in the learned scene entry regions, provided that the motion was moving in the same direction as the expected activity for leaving the region. We used a fixed-size bounding box and the



**Fig. 26** Entry and exit regions from a single view learned locally (row 1) and in the camera viewspace (row 2)



**Fig. 27** Occlusion detection results from Scenes 5, 6, and 7 with *arrows drawn* from the occlusion exit to the occlusion re-entry

feature vector  $f_k = [x, y, R, G, B, I_x, I_y]$  to build a covariance descriptor for the target. Once the tracker initializes, it attempts to track the object until (1) the object leaves the scene or (2) the tracker loses the object. If the object enters an occlusion exit, an expected wait time is estimated from the learned occlusion distance and current object speed, and the tracker then attempts to re-acquire the target at the corresponding re-entry location(s) for the occlusion. Our approach

searches the expected occlusion re-entry area until it finds a match within  $2\sigma$  from the learned covariance model (constructed from the covariance match scores of the target in past frames). The results are presented in Fig. 28a and b for Scenes 5 and 6. As shown, the tracker was able to track objects despite the large occluded region and was able to reacquire the object at the occlusion re-entry via information provided by our model. The results are especially signifi-

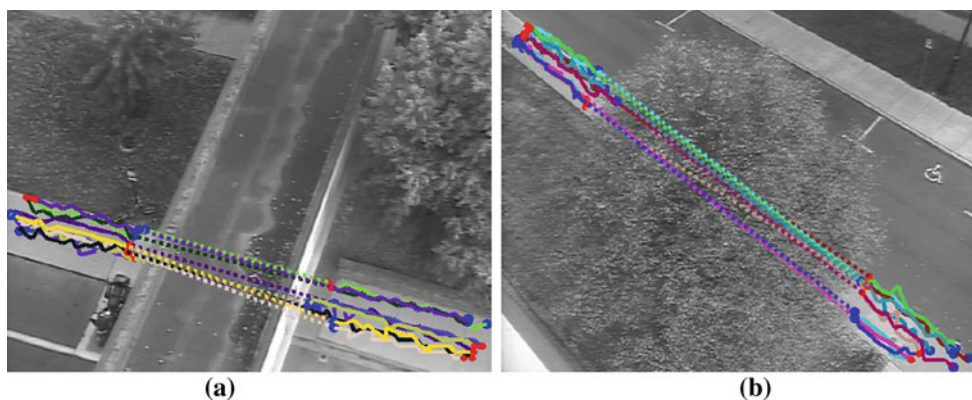


Fig. 28 Covariance tracking results from Scenes 5 and 6. Dashed lines connect corresponding tracks on either side of the occlusion

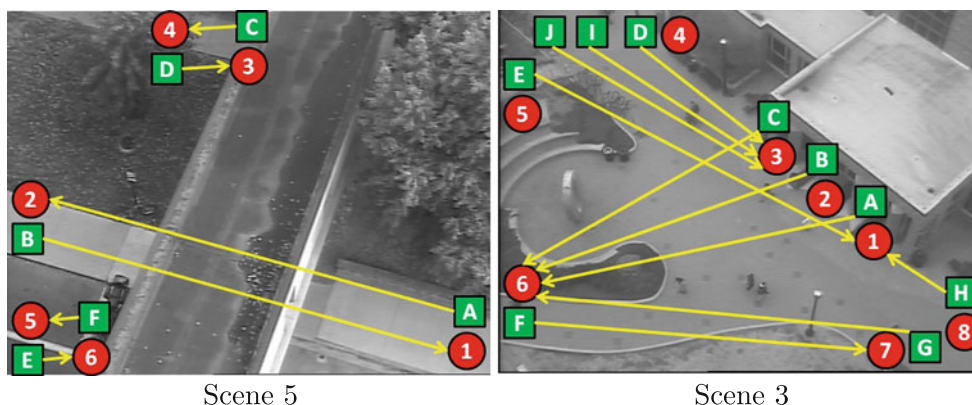


Fig. 29 Most probable entry and exit connections for Scenes 5 and 3

cant for Scene 6, as the tree covers a significant portion of the sidewalk and it would be difficult for traditional tracking methods to track confidently through such an occlusion.

### 8.4.2 Entry → Exit non-pathway relationships

For each scene we also found the relationship from entry to exit regions. For each scene entry region  $r$ , we obtain the likelihood that an object entering the scene at  $r$  will leave the scene via each scene exit region  $x \in X$  using Eq. (16). Entry → exit relationship results for Scene 5 (simple scene) and Scene 3 (more complicated scene) are shown in Fig. 29, where we display a straight line (non-pathway) arrow from each entry region (squares) to the most likely exit region (circles). In Scene 5, we learned the entry → exit relationships corresponding to the traffic on the sidewalks. In Scene 3, it can be seen that the two entry regions learned at the bottom-right of the scene are semantically meaningful, as traffic entering the left-side entry region has a strong relationship with the walkway exit on the left of the scene, and the right-side entry region has a strong exit relationship with the nearest door of the building. More in-depth results are

Table 7 Probabilities between entries (rows) and exits (columns) for Scene 5

	1	2	3	4	5	6
A	0	0.84	0.08	0.06	0.01	0
B	0.87	0	0.08	0.03	0.02	0
C	0.09	0.08	0	0.83	0	0
D	0.08	0.15	0.74	0	0.03	0
E	0	0	0	0	0	1
F	0.06	0.08	0.05	0	0.81	0

Region labels shown in Fig. 29

provided in Tables 7 and 8, where individual entry → exit pair probabilities are given for these scenes.

We also used the covariance tracker to track multiple people in Scene 3 and then ranked their trajectory likelihood (for anomaly detection). We collected and smoothed the trajectories and then obtained a likelihood score for each trajectory using Eq. (17). We display the ten most likely trajectories in Fig. 30a and the five least likely trajectories in Fig. 30b. The least likely trajectories (Fig. 30b)



**Table 8** Probabilities between entries (rows) and exits (columns) for Scene 3

	1	2	3	4	5	6	7	8
A	0	0.07	0.06	0.17	0.04	0.35	0.13	0.18
B	0.18	0	0.19	0.19	0	0.22	0.07	0.16
C	0.17	0.07	0	0.25	0.05	0.32	0.05	0.09
D	0.13	0	0.48	0	0	0.20	0.08	0.11
E	0.24	0	0.23	0.20	0	0.16	0	0.17
F	0.20	0.05	0.14	0.04	0.03	0	0.37	0.17
G	0.12	0.05	0.06	0.20	0.03	0.41	0	0.14
H	0.46	0.04	0.04	0.15	0.03	0.21	0.08	0
I	0.13	0.05	0.54	0	0.06	0.06	0.05	0.10
J	0.18	0	0.30	0.13	0	0.19	0.09	0.11

Region labels shown in Fig. 29

correspond to a person taking a long meandering path between regions, another person exiting the scene at a location not corresponding to an exit region, someone using an uncommon entry–exit pair, and a short trajectory resulting from tracker failure. Our model is able to score such anomalous behavior as we not only learn activity relationships between entry  $\rightarrow$  exit pairs, but also the expected travel distance of the path taken between them. If desired, a threshold could be learned/employed to automatically detect such rare events.

### 8.5 Method limitations

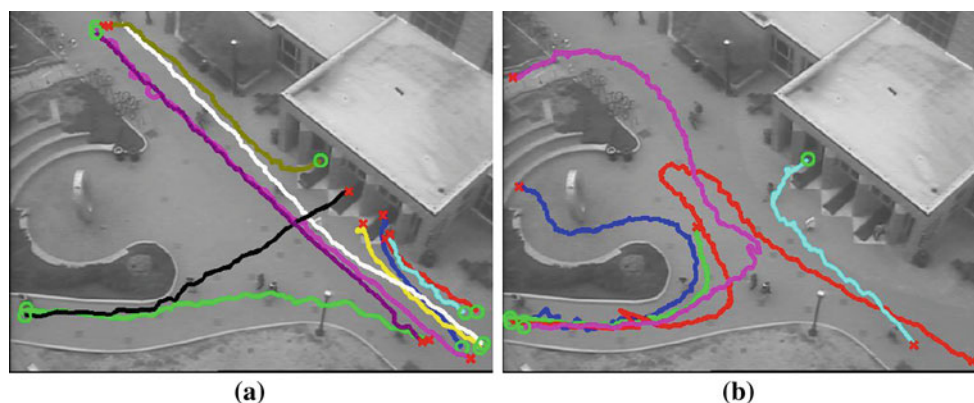
As demonstrated, our approach is able to consistently detect expected entry and exit regions with high accuracy in both local and viewspace data collections. However, like all algorithms, there are scenarios where our approach can be improved. We detect entry and exit regions that adhere to our behavior model, specifying that desired regions have tracks that leave (for entries) or enter (for exits) in a mostly direc-

tional manner and do not have outside tracks that intersect in the direction of the defining motion. Thus, we are not able to detect entry (or exit) regions where activity leaves (or enters) the region in multiple directions (e.g., people coming out of or moving into a manhole in all directions). However, these cases are rare. To detect occlusions, we detect exit and entry regions that characterize the occlusion. If an occlusion exists where people can walk behind or in front of it, we will currently be unable to detect the entry and exit regions (and thus the occlusion). In this scenario, the occlusion exit and entry would each appear as a “through state”, as the current approach does not incorporate depth. We could also improve the tracking method to obtain better data when there are large amounts of perspective and lack of object motion, which prevents us from learning reliable regions in the periphery of the camera viewspace.

## 9 Conclusion

We presented novel methods to detect, exploit, and evaluate entry and exit regions in surveillance video, and presented a novel extension to the viewspace of a PTZ camera. Our approach employs weak tracking data which are transformed into a set of entity tracks for more reliable entry and exit observations. These observations are then clustered to produce a set of potential entry and exit regions, and each region is scored and thresholded using a behavioral-based reliability metric based on the directional and interaction consistency. We then described how relationships can be learned between the detected entry and exit regions to discover common connections between the regions and occlusions in the scene.

Our approach demonstrates that utilizing scene behavior is paramount to detecting semantic regions in a scene. We have also shown that weak tracking data can be useful when working with very busy scenes, and that such data may be used to detect a reliable set of entry and exit regions (in


**Fig. 30** Track likelihoods for **a** the ten most likely and **b** five least likely tracks

both local and global views) and such regions could be useful for inter-camera tasks, such as persistent object tracking. Knowing what regions are occluded with respect to a camera's view space could allow for a more successful object handoff between cameras. In addition, in our evaluation, we presented a method to quantify our method, which is especially important due to the lack of quantitative experimentation in most scene modeling work.

**Acknowledgments** This work was supported in part by the US Air Force Research Laboratory's Human Effectiveness and Sensors Directorates, and the NSF Center for Surveillance Research (IIP-0968910). Initial portions of this work have been published in [19]. We would also like to acknowledge discussions with Raphael Wenger on spherical geometry methods.

## References

1. Bevis, M., Chatelain, J.: Locating a point on a spherical surface relative to a spherical polygon of arbitrary shape. *Math. Geol.* **21**(8), (1989)
2. Beyer, W.: CRC standard mathematical tables, 28th edn. CRC Press (1987)
3. Cheng, Y.: Mean shift, mode seeking, and clustering. *PAMI* **17**(8), (1995)
4. Cheriyyadat, A., Bhaduri, B., Radke, R.: Detecting multiple moving objects in crowded environments with coherent motion regions. In: *POCV* (2008)
5. Cheriyyadat, A., Radke, R.: Automatically determining dominant motions in crowded scenes by clustering partial feature trajectories. In: *Proceedings of International Conference on Distributed Smart Cameras* (2007)
6. 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: *Proceedings of 5th Mexican International Conference on Artificial Intelligence* (2006)
7. Edelsbrunner, H., Kirkpatrick, D., Seidel, R.: On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory.* **29**(4) (1983)
8. Guan, L., Franco, J.S., Pollefeys, M.: 3D occlusion interference from silhouette cues. In: *CVPR* (2007)
9. Guan, L.E.A.: Visual hull construction in the presence of partial occlusion. In: *3rd International Symposium on 3D Data Proceedings, Visualization and Transmission* (2007)
10. Hu, W., Hu, M., Zhou, X., Tan, T., Lou, J., Maybank, S.: Principal axis-based correspondence between multiple cameras for people tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(4) (2006)
11. Kaucic, R., Perera, A., Brooksby, G., Kaufhold, J., Hoogs, A.: A unified framework for tracking through occlusions and across sensor gaps. In: *CVPR* (2005)
12. Keck, M., Davis, J.: 3D occlusion recovery using few cameras. In: *CVPR* (2008)
13. Khan, S., Shah, M.: Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Trans. Pattern Anal. Mach. Intell.* **25**(10) (2003)
14. Li, J., Gong, S., Xiang, T.: Scene segmentation for behaviour correlation. In: *ECCV* (2008)
15. Makris, D., Ellis, T.: Path detection in video surveillance. *Image Vis. Comput.* **20**, 895–903 (2002)
16. Makris, D., Ellis, T.: Automatic learning of an activity-based semantic scene model. In: *AVSS* (2003)
17. Makris, D., Ellis, T.: Learning semantic scene models from observing activity in visual surveillance. *IEEE Trans. Syst. Man Cybernet. Part B Cybernet.* **35**(3) (2005)
18. Makris, D., Ellis, T., Black, J.: Bridging the gaps between cameras. In: *CVPR* (2004)
19. Nedrich, M., Davis, J.: Learning scene entries and exits using coherent motion regions. In: *International Symposium on Visual Computing* (2010)
20. Parzen, E.: On estimation of a probability density function and mode. *Ann. Math. Stat.* **33**(3) (1962)
21. Porikli, F., Tuzel, O., Meer, P.: Covariance tracking using model update based on means on Riemannian manifolds. In: *CVPR* (2006)
22. Rabaud, V., Belongie, S.: Counting crowded moving objects. In: *CVPR* (2006)
23. Russell, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*, 2nd edn. Prentice Hall, New Jersey (2003)
24. Sankaranarayanan, K., Davis, J.: A fast linear registration framework for multi-camera GIS coordination. In: *IEEE International Conference on Advanced Video and Signal Based Surveillance* (2008)
25. Sankaranarayanan, K., Davis, J.: PTZ camera modeling and panoramic view generation via focal plane mapping. In: *Asian Conference on Computer Vision* (2010)
26. Schwarz, G.: Estimating the dimension of a model. *Ann. Stat.* **6**(2) (1978)
27. Shi, J., Tomasi, C.: Good features to track. In: *CVPR* (1994)
28. Sinnott, R.: Virtues of the Haversine. *Sky Telescope* **68**(2) (1984)
29. Stauffer, C.: Estimating tracking sources and sinks. In: *Proceedings of Second IEEE Event Mining Workshop* (2003)
30. Streib, K., Davis, J.: Extracting pathlets from weak tracking data. In: *AVSS* (2010)
31. Wang, X., Ma, X., Grimson, E.: Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**, 539–555 (2009)
32. Wang, X., Tieu, K., Grimson, E.: Learning semantic scene models by trajectory analysis. In: *ECCV* (2006)

## Author Biographies



**Matthew Nedrich** received his M.S. (2011) and B.S. (2009) degrees in Computer Science and Engineering from Ohio State University. His research interests include computer vision, machine learning, and computer architecture.



**James W. Davis** is an Associate Professor of Computer Science and Engineering at Ohio State University. He received his Ph.D. from the Massachusetts Institute of Technology in 2000. His research specializes in computer vision approaches to video surveillance and human activity analysis.