

DETECTING BEHAVIORAL ZONES IN LOCAL AND GLOBAL CAMERA VIEWS

A Thesis

Presented in Partial Fulfillment of the Requirements for
the Degree Master of Science in the
Graduate School of The Ohio State University

By

Matthew Nedrich, B.S.

Graduate Program in Computer Science and Engineering
The Ohio State University

2011

Master's Examination Committee:

Prof. James W. Davis, Adviser

Prof. Richard Parent

© Copyright by
Matthew Nedrich
2011

ABSTRACT

We present a complete end-to-end framework to detect and exploit entry and exit regions in video using behavioral models for object trajectories. We first describe how weak tracking data (short and frequently broken tracks) may be utilized to hypothesize entry and exit regions by constructing the weak tracks into a more usable set of “entity” tracks. The entities provide a more reliable set of entry and exit observations which are clustered to produce a set of potential entry and exit regions within a scene. A behavior-based reliability metric is then used to score each potential entry and exit region, and unreliable regions are removed. Using the detected regions, we then present a method to learn scene occlusions and causal relationships between entry-exit pairs. An extension is also presented that allows our entry/exit detection algorithm to detect global entry and exit regions with respect to the viewspace of a pan-tilt-zoom camera. We provide thorough evaluation of our local and viewspace region discovery approaches, including quantitative experiments, and compare our local method to existing approaches. We also provide experimental results for our region exploitation methods (occlusion discovery and entry \rightarrow exit region relationships), and demonstrate that they may be incorporated to aid in tasks such as tracking and anomaly detection.

For my parents

ACKNOWLEDGMENTS

Beginning as an undergraduate student, I have had the pleasure of working with my advisor, James W. Davis. His direction and inspiration have guided me through my graduate school journey. I will never forget the years spent working under him, constructing my intellectual foundation, and learning how to become a true computer scientist.

I would like to thank the Center for Surveillance Research (CSR), and the Air Force Research Lab (AFRL) for their support throughout my time in graduate school.

Additionally, I have had the opportunity of working beside a great group of peers, without whom I would not have the success that I have today. First and foremost, I would like to thank Karthik Sankaranarayanan, whose patient mentoring, and sincere friendship have influenced my intellectual growth by an invaluable amount. I would also like to thank Kevin Streib for continually challenging me and my work.

Most importantly, I would like to thank my loving family for everything that have done for me, and my wonderful fiancée Sara, without whose companionship I would not be the person I am today.

VITA

November 26, 1985 Born in Seven Hills, OH, USA

March, 2009 B.S., Computer Science and Engineering
The Ohio State University, Columbus,
OH, USA

PUBLICATIONS

Conference Papers and Books

M. Nedrich and J. Davis, “Learning Scene Entries and Exits using Coherent Motion Regions”, *International Symposium on Visual Computing (ISVC)*, November 2010.

Technical Reports

M. Nedrich, K. Sankaranarayanan, and J. Davis, “Geo-registration and Interactive Control for Distributed Camera Networks”, *OSU Dept. Computer Science and Engineering Technical Report OSU-CISRC-6/10-TR13*, 2010.

FIELDS OF STUDY

Major Field: Computer Science and Engineering

Studies in Artificial Intelligence and Computer Vision: Prof. James W. Davis

TABLE OF CONTENTS

	Page
Abstract	ii
Dedication	iii
Acknowledgments	iv
Vita	v
List of Tables	ix
List of Figures	x
Chapters:	
1. Introduction	1
1.1 Motivation	1
1.2 Contribution and Significance	2
1.3 Proposed Approach	5
1.3.1 Weak Tracking	5
1.3.2 Entry and Exit Region Detection	7
1.4 Organization	8
2. Related Work	9
2.1 Weak Tracking	9
2.2 Entry and Exit Discovery	10
2.3 Entry and Exit Exploitation	11
2.4 Camera Viewspace Entry and Exits	12

3.	System Overview	14
3.1	Local View Region Detection	14
3.2	Camera Viewspace Region Detection	15
3.3	Method Applications	15
4.	Entity Detection and Tracking	19
4.1	Entity Discovery	19
4.2	Entity Tracking	21
5.	Entry and Exit Region Detection	23
5.1	Region Shape	23
5.2	Region Reliability	26
6.	Exploiting Entry and Exit Regions	31
6.1	Exit → Exit Occlusion Relationships	31
6.2	Entry → Exit Non-Pathway Relationships	34
7.	Extension To Camera Viewspace	39
7.1	Camera Geometry	40
7.2	Viewspace Data Collection	42
7.3	Viewspace Region Detection	43
7.4	Using the Viewspace Model in a Local View	48
7.5	Viewspace Region Exploitation	50
8.	Experiments and Results	53
8.1	Region Detection Experiments	53
8.2	Ground Truth Evaluation	61
8.2.1	Region Shape Evaluation	63
8.3	Region Exploitation Experiments	68
8.3.1	Occlusion Discovery	71
8.3.2	Entry-Exit Relationships	71
8.3.3	Region Exploitation Applications	73
8.4	Camera Viewspace Experiments	76
8.4.1	Viewspace Region Detection	76
8.4.2	Local vs. Viewspace Approach Comparison	79
8.5	World Space Entry/Exit Regions	81

8.6	Limitations	86
8.6.1	Entities	86
8.6.2	Entry and Exit Region Detection	86
8.6.3	Occlusion Detection and Entry → Exit Region Relationships	87
9.	Conclusion and Future Work	89
9.1	Contributions	91
9.2	Future Work	92
9.2.1	Adaptive Clustering	92
9.2.2	Adaptive Viewspace Sampling	93
9.2.3	Online Update	93
	Bibliography	95

LIST OF TABLES

Table	Page
8.1 Data collection durations in minutes for Scenes 1-7.	55
8.2 Results for Entry 1 Synthetic Ground Truth Experiment. Note, F1, precision and recall scored are the average over five trials	68
8.3 Results for Entry 2 Synthetic Ground Truth Experiment. Note, F1, precision and recall scored are the average over five trials.	69
8.4 Results for Entry 3 Synthetic Ground Truth Experiment. Note, F1, precision and recall scored are the average over five trials.	70
8.5 Results for Entry and Exit 3 Synthetic Ground Truth Experiment, object count varied. Note, F1, precision and recall scored are the average over ten trials for object quantities 25-500. The values for 1000 objects represent the full set of data (which the other object amounts were sampled from). . .	70
8.6 Activity probabilities between entries (rows) and exits (columns) for Scene 5. Region labels shown in Fig. 8.7(a).	75
8.7 Activity probabilities between entries (rows) and exits (columns) for Scene 3. Region labels shown in Fig. 8.7 (b).	75
8.8 Data collection durations for our global region detection approach in minutes for cameras 1-3.	77

LIST OF FIGURES

Figure	Page
3.1 Local camera view entry/exit detection system overview.	17
3.2 Camera viewspace entry/exit detection system overview.	18
4.1 (a) KLT (weak) tracks and (b) their corresponding entity tracks. [Best viewed in color]	21
4.2 (a) Weak tracking start observations, and (b) the corresponding set of entity entry observations.	22
5.1 Outlier removal example. (a) Convex hull area as points are removed. (b) Convex hull area change for each point. (c) Original cluster points, and (d) final cluster points after removing outliers.	25
5.2 Region shapes learned via kernel density estimation using kernel bandwidth values of (a) 25, (b) 15, (c) 10, and (d) 5.	26
5.3 Example entry regions displaying (a) good directional consistency, (b) bad directional consistency, (c) good interaction consistency, and (d) bad interaction consistency.	28
5.4 (a) Plausible entry regions, and (b) reliable entry regions with $\Psi > 0.75$. . .	30
6.1 Series of occlusions. Entry regions are shown in green and exit regions are shown in red. [Best viewed in color]	32
6.2 Occluded paths behind the occlusion. [Best viewed in color]	35
6.3 (a) Estimated 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.	35

7.1	(a) Local camera view, and (b) the view highlighted in the camera viewspace.	42
7.2	(a) Camera viewspace, and (b) spherical panorama projection.	43
7.3	Weak tracks (tracklets) from two local camera views displayed on the panorama.	44
7.4	(a) Sphere geometry for central angle, and (b) spherical polygon.	45
7.5	Entity tracks in the camera viewspace.	48
7.6	(a) Camera viewspace potential entry regions, and (b) reliable entry regions $\Psi > 0.75$	49
7.7	(a) Local camera view highlighted on panorama, and (b) camera activity mask.	51
7.8	(a) Viewspace camera entries and, (b) viewspace exits for a local view with image border activity shown in blue.	51
8.1	Detected Entry and Exit regions for Scenes 1-4.	58
8.2	Detected Entry and Exit regions for Scenes 5-7.	59
8.3	Entry and exit regions using the method in [15] (cols 1 and 2). Entry and exit states using the method in [31] (cols 3 and 4). [Best viewed in color] .	60
8.4	Synthetic entry and exit regions.	63
8.5	Detected entry and exit shapes across all eight noise levels.	64
8.6	Occlusion detection results from Scenes 5, 6, 3, and 7 with arrows drawn from the occlusion exit (red) to the occlusion entry (green). [Best viewed in color]	72
8.7	Most probable entry and exit connections for Scenes 5 and 3. [Best viewed in color]	72
8.8	Covariance tracking results from Scenes 5 and 6. Dashed lines correspond tracks on either side of the occlusion. [Best viewed in color]	74

8.9	Track likelihoods for (a) the ten most likely and (b) five least likely tracks. [Best viewed in color]	74
8.10	Example frames showing objects moving on the ground through the reflection in a building window for Camera 3.	79
8.11	Viewspace entry and exit regions for camera 1.	80
8.12	Viewspace entry and exit regions for camera 2.	80
8.13	Viewspace entry and exit regions for camera 3.	81
8.14	Entry and exit regions from a single view learned locally (row 1) and in the camera viewspace (row 2).	82
8.15	Entry regions for Cameras 1 and 2 plotted on an orthophoto.	84
8.16	Exit regions for Cameras 1 and 2 plotted on an orthophoto.	85

CHAPTER 1: INTRODUCTION

In this thesis we present a novel approach to discover and exploit entry and exit regions in video. The goal of our work is to automatically detect regions where objects enter into or leave a scene defined by a camera view. Such regions may correspond to anything from a doorway or garage opening, to a walkway or street that intersects the edge of the camera view (image border). In addition to the detection of such regions, we will also explore ways in which they may be utilized to learn more about the scene structure and higher level semantic activity. We will also present a novel extension that allows such regions to be learned in the viewspace of an pan-tilt-zoom camera (camera that can move), rather than a local camera view. In this chapter, we begin by motivating the problem, and discuss our contributions. We then describe our approach and outline the organization of the subsequent chapters.

1.1 Motivation

An important step when attempting to understand a scene is to identify regions where activity enters and exits. Such regions can be useful in many visual surveillance applications. For mid-level tasks such as object tracking, entry regions allow for more efficient tracker initialization. If an object being tracked disappears but is not near a known exit, it is likely due to tracker failure. Understanding entry and exit locations may also help to attach semantic meaning to tracking events. If an object enters through a commonly used entry region and leaves through a popular exit the object is probably behaving normally.

However, if uncommon regions are used, or objects enter or exit through areas that do not correspond to entry/exit locations, 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 about the scene “pulse” as pedestrians, cars, and cyclists come and go throughout the day. Knowing where objects typically enter and exit provides insight as to where the scene’s “pulse” should be taken. Such higher level analysis may be useful to understanding how the traffic ebbs and flows over time. If semantic meaning can be attached to each entry or exit (e.g., building doorway), monitoring traffic at these regions can help indicate how populated buildings, or other semantically meaningful areas are. At a region level, monitoring pulses between entry and exit zones can help learn relationships between such regions. As we will demonstrate, depending on the direction of such relationships (e.g., entry \rightarrow exit vs exit \rightarrow entry), they can aid in understanding the dynamics of the scene activity as well as reasoning about occluded paths in the scene.

1.2 Contribution and Significance

Thus far, most existing scene modeling work has focused on learning pathways in the scene. There has, however, been a smaller body of work that focuses on detecting entry and exit locations. We contribute to the existing work by proposing novel approaches to detect and exploit such regions. We also show that our approach may be extended to detect regions within the viewspace of a pan-tilt-zoom camera, rather than just focusing on a local camera view.

First, we introduce a behavior model for object trajectories, and show that utilizing the behavior of a scene is paramount to understanding it. We define scene behavior as the spatio-temporal motion (and interaction) of objects in a scene. In most previous work, the

problem of detecting entry and exit regions is approached by tracking objects and clustering the trajectory endpoints. Dense clusters are kept and sparse clusters regarded as noise. These previous approaches fail to utilize the scene behavior, which we show is very powerful tool, and allows us to adopt a inexpensive tracking approach that is applicable to very busy and crowded environments.

We also show that once such regions are discovered, they may be exploited to learn about the scene structure and behavior. Previous work that explores relationships between entry/exit regions attempts to learn relationships between exit and entry regions across disjoint camera views to track objects between cameras. We show that learning relationships between regions in a single camera view can 1) help explain how objects move through the scene (e.g., objects that enter via entry ‘A’ typically leave through exit ‘B’), and 2) uncover occluded pathways in the scene.

In addition to our contributions to discovering regions in a static camera views, we also introduce a novel extension to allow entry/exit regions to be learned with respect to the viewspace of a pan-tilt-zoom camera. This allows global entry/exit regions to be learned and then applied in any local camera view.

For each of our proposed methods we provide experimental results for numerous scenes and cameras. In addition to these qualitative results we also present an approach to quantify the accuracy of our region detection process, and discuss the challenges of determining ground truth. This is significant as previous approaches consistently ignore the question of ground truth and generally only provide qualitative results. To summarize, the main contributions of our work include the following.

- **Entity tracking:** Our approach constructs “entities” using underlying weak tracking data (tracklets). Entities are important as they represent “coherent motion regions”

in the scene where the objects moving through the scene are uninterrupted. Previous approaches attempt trajectory clustering techniques on tracklets, though such approaches are expensive and require extensive parameter optimization. Our approach is a frugal, yet very useful alternative to extract useful swatches of motion in the scene. We use these “swatches” to accumulate entry and exit observations, though they may also be useful for other applications as well (e.g., person counting).

- **Behavior-based entry and exit detection algorithm:** Previous approaches ignore scene behavior when detecting entry and exit regions. They typically approach the problem as a spatial clustering problem and cluster trajectory endpoints to detect possible entry and exit regions. We show that the inclusion of a behavior model greatly improves region detection capability. Our approach works by analyzing the behavior around each potential entry and exit region and scoring it’s consistency.
- **Quantitative analysis for region detection:** Ground truth is generally ignored when attempting to model or discover any type of high level scene aspect (e.g., entry, exit, pathways, and other semantic regions). We provide experimental results that quantify the accuracy our region detection approach.
- **Method to discover relationships between detected regions:** Very limited exploration has been done in the area of entry and exit region applications. We provide a novel approach to aid in object tracking and anomaly detection using a learned set of entry/exit regions.
- **Extension to detect entry/exit regions in a camera viewport:** All previous work in detecting entry and exit regions (and most other semantic regions) has only focused

on a single camera view. We present a method that works on the entire viewspace of a pan-tilt-zoom camera.

1.3 Proposed Approach

Most scene modeling techniques require some form of object tracking as input. Many existing methods [15], [28], [33], [14] rely on stronger tracking data which consists of a set of reliable and long duration object trajectories. When an object enters the scene, it is detected and tracked until it leaves, producing a single trajectory capturing the motion of the object. While such tracks are very useful, many approaches (stronger trackers) to collect them tend to be unreliable in busy urban environments - especially when the scene is crowded. In addition, such trackers are usually only able to track a small subset of the objects in real-time. This may be problematic as the produced set of object tracks may be unrepresentative of the true scene activity (e.g., activities that occur with low frequency may not be detected, and the balance of activity may be skewed).

1.3.1 Weak Tracking

To compensate for these shortcomings we present an approach that uses *weak* tracking data. We believe that any set of object tracks (or trajectories) exist on a spectrum between strong and weak, depending on the characteristics they exhibit. Stronger tracks exhibit qualities that more closely approximate the ground truth activity that one wishes to capture in the scene. If person tracking is the goal, the ground truth would consist of one track per person that remains on the same location of the person as they move through the scene until they exit. There are many ways to evaluate object tracks [1], [27], [3]. We define stronger tracks as having the following qualities:

- **Persistence:** Strong tracks will continuously follow the motion of an object through the scene for all portions of the scene where the object is visible, and will not fragment.
- **Track Uniqueness:** Each object should have no more than one track representing its movement at any time instant (related to tracker “configuration” in [27]).
- **Label Uniqueness:** Each track should only represent the motion of one underlying object. Thus, tracks should not switch objects (related to tracker “identification” and “purity” in [27]).
- **Stability:** Each strong track observation should correspond to the same location on the object being tracked (e.g., if a person is being tracked, the track should not jump from the person’s foot to head).

Any deviation from the list above will cause a set of tracks to become “weaker”. Thus, tracks may be labeled as weak for any combination of qualities that stray from the specifications listed above. As an example, weak tracks may exhibit qualities such as multiple tracks per target, fragmented tracks, tracks that jump between objects in the scene, etc. The importance of each of these qualities depends on their application, so it is not necessarily possible to produce an absolute rank ordering of importance for these characteristics. Our input consists of multiple and frequently fragmented tracks per target. Thus, each object is usually tracked by multiple tracks that tend to fragment and break (“tracklets”) as the object moves through the scene (shown in Fig. 4.1 (a)). For our work in this thesis we use a modified version of the Kanade Lucas Tomasi (KLT) tracker [24], presented in [31].

1.3.2 Entry and Exit Region Detection

Weak trackers are capable of tracking many objects simultaneously in real-time, and can function well in busy scenes. Given such tracking data, we learn “entities” in each frame by clustering similar weak tracking observations. We define an entity to be a cluster of weak track observations moving in a coherent manner. 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, merge, etc., with other entities in the scene. This allows us to simplify the weak tracking data into a set of entities where each entity’s motion is defined by an uninterrupted “coherent motion region” (shown in Fig. 4.1).

We then cluster the resultant entity entry and exit observations to produce a set of potential scene entry and exit zones. Each zone is scored using a behavior-based reliability metric that analyzes the manner which the entity trajectories leave, enter, and interact with each potential entry/exit region. We define a reliable entry region as one with tracks *emanating out* of it in a semi-directional manner (e.g., tracks should not fan out greater than 180°), and a reliable exit region as one where tracks *flow into* it in a semi-directional manner. Further, other tracks in the scene should not intersect a reliable entry/exit region in the same emanating direction. For a potential entry region, such an action would indicate that another entry region exists behind the current one. Examples of these scenarios are presented in Fig 5.3. After scoring each region, unreliable zones are removed, leaving the final set of entry and exit regions. We then show how the set of learned entry/exit regions may be exploited to learn relationships between entry and exit regions, and also learn occluded paths in the scene. Doing so allows for different applications such as anomaly detection, and aids on object tracking.

We also present an extension which allows our local entry/exit discovery technique to be applied to the entire camera viewspace. We explain how each process used in the local approach may be extended to work in the new global space, and how the discovered global regions may be shared across cameras.

1.4 Organization

We begin with a review of related work in Chapter 2, and provide a general system overview in Chapter 3. The entity learning process which simplifies weak tracking input to a more usable set of entities is described in detail in Chapter 4. In Chapter 5 we explain how we cluster entity entry and exit observations to obtain a set of potential entry and exit regions, and how each region is scored by analyzing scene behavior.

Once a set of entry and exit regions is discovered, Chapter 6 details how we use these regions to discover scene occlusions and learn causal relationships between them. This is accomplished by monitoring the “pulses” between each of the learned regions.

In Chapter 7 we describe how our region learning approach may be extended to learn entry and exit regions with respect to the viewspace of a pan-tilt-zoom camera, rather than focusing on a local camera views.

Our work is evaluated in Chapter 8 where we provide experimental results for our local and camera viewspace region detection methods, our region exploitation methods, and discuss what ground truth means for our problem and how quantitative metrics may be used to gauge success.

CHAPTER 2: RELATED WORK

In this chapter we provide an overview of related work. We start with a discussion on using weak tracking as input. We then elaborate on previous entry/exit detection and exploitations work and describe the novel aspects of our approach. Additionally, we provide a discussion on previous multi-camera scene modeling and camera viewspace related work.

2.1 Weak Tracking

Our entity learning process allows us to use weak tracking data as input. This relaxes the requirement of accurately tracking each object as it moves through the scene (as strong trackers attempt to do). When a scene is very busy or crowded such a task is very difficult. Moreover, the task of simultaneously tracking many objects is expensive. Weak trackers attempt to track salient features (e.g., corners or areas of high texture) as they move through the scene. Such trackers perform well even in crowded environments as they are not tasked with sorting out each object from the crowd. Our only requirement is that there be at least one weak track on each target at every time instance as the target moves through the scene. We define a target as an underlying semantically important entity in the scene (e.g., person, group, cyclist, car). Provided this is the case, we will be able to track each target (entity) as it moves through the scene using our entity detection and tracking method described in Chapter 4.

The idea of using a weak tracker in busy and crowded environments, and then using the weak tracks to derive higher level meaning of the scene has been employed in the past

for different tasks. Rabaud and Belongie [20] use weak tracks to count pedestrians in busy scenes. Work by Cheriadat et. al. [5] attempts to leverage the idea of a “coherent motion region” to learn distinct objects using weak tracks (as we do with our entity learning). However, their motion regions are constructed using a user-defined bounding box representing the size of a person. Motion regions are also learned in [6], though they are learned via trajectory clustering ignoring the time that the trajectories were generated, thus two trajectories that cluster together may have been generated at different times and belong to two different people. Our approach is unique in the nature that we construct our entities. When a new entity (set of weak tracks) enters the scene, we track it and allow it to “flow” through the scene (perhaps merging and splitting with other entities) until it leaves, though we still differentiate its entrance and exit from its interactions while it is in the scene. Being able to separate an entity’s initiation and termination from its interactions (e.g., split, merge) provides us with a powerful set of entry and exit observations which we use to estimate entry and exit regions (approximating the set of entry and exit observations that would be produced by a strong tracker).

2.2 Entry and Exit Discovery

While scene modeling has become a very active area of research, particularly in the area of visual surveillance, much of the scene modeling work attempts to model pathways and motion in the scene. In addition to this work, there has been a smaller body of work in relation to entry-exit zone identification. Makris and Ellis [15] and Stauffer [28] assumed that trajectory endpoints are entry and exit observations, which are then clustered to learn the regions. In the approach from Makris and Ellis [15], a density threshold is used to remove noise clusters. However, such approaches require strong tracking data and will

not work on weak tracks. Further, both approaches fail to leverage the scene behavior to learn accurate regions. Wang et. al. [33] described a framework to model semantic regions via trajectory clustering. When learning entry and exit locations they only consider trajectory endpoints that exist near the borders of semantic regions. Streib and Davis [31] presented a grid-based approach where the ratio of tracks through each state (grid cell and direction) vs. tracks that originate in (entries) or terminate in (exits) each state is leveraged to learn entry/exit states. None of these methods attempt to identify regions that result due to occlusions in the scene, and only [31] is able to function using weak tracking data.

2.3 Entry and Exit Exploitation

Fusing occlusion learning directly into scene modeling is one novelty of the proposed method. Often handling occlusions involves detecting and modeling the occluding objects directly. Existing occlusion detection work such as the approach from Guan et. al. [10], attempts to fully model occlusions in 3D. In other works, Guan et. al. [11] attempts to model occlusions directly in image space. Approaches similar to the one by Kaucic et. al [12] use trajectory matching to track objects through occlusions, and attempt to learn plausible occluding structures through image segmentation (requiring manual labeling). Our approach is different in that we do not aim to model the occluding structure in the scene. Rather, we model the regions where objects enter and exit the occluded areas by studying correlated scene activity. For applications such as tracking, our approach can be used to produce a useful predictor for reacquiring an object track. Makris, Ellis, and Black [16] proposed a method to learn relationships between exit and entry regions between disjoint camera views. They use a cross-correlation based approach to associate activity between such regions. However, the goal of their approach is to automatically learn camera

network topology by determining how activity is connected between camera views. They do not attempt to learn connections that result from occluded paths, and do not model forward (entry to exit) relationships as we do. Further, their cross-correlation approach is based on time-lagged activity co-occurrences between regions. That is, they only look at the time differences between observations in each pair of regions. Our approach learns relationships between regions by attempting to estimate the path traveled between them (if any exists). Thus, is it robust to different object classes that may travel at different speeds (e.g., pedestrian, cyclist, cars). As a result, we do not only learn connections between regions but the actual path distance traveled between them.

2.4 Camera Viewspace Entry and Exits

We provide an extension of our local camera view entry and exit detection approach to discover entry/exit regions in the viewspace of a pan-tilt-zoom (PTZ) camera. Outside of approaches that attempt to model a single camera view, there has been some scene modeling effort that utilizes multiple cameras to detect semantic regions. In work by Wang et. al [34], multiple local (static) camera views are used to track objects and learn semantic activities (pathways) that persist across camera regions. Their approach, however, does not require the cameras to be calibrated to utilize data across cameras. Other approaches such as the one by Stauffer et. al. [29] or Lee et. al. [13] use tracking data across multiple camera views to calibrate the views. Such works usually attempt to calibrate camera views using data collected (e.g., tracking data), from different cameras, or to learn semantic regions that span between cameras. None of these works employ pan-tilt-zoom cameras.

Sankaranarayanan and Davis [21], and Sinha and Pollefeys [25] present methods to create panoramic mosaics using pan-tilt-zoom cameras. However, in [25] they do not model

the entire viewspace. Instead, they only focus on stitching a few local images together. In [21] they do model the entire space, though they only apply their camera viewspace panorama and model for active object tracking. In [32] Wada and Matsuyama describe a method to learn a background model for a pan-tilt-zoom camera, however they focus on indoor scenes and do not attempt to extract any additional semantic meaning from the camera viewspace. To the best of our knowledge our extension for learning entry and exit regions in the viewspace of a pan-tilt-zoom camera has not been attempted before. As described, most previous work has mainly focused on learning such regions in local camera views, or across multiple static cameras.

CHAPTER 3: SYSTEM OVERVIEW

In this section we provide an overview of our local and camera viewspace entry and exit region detection processes. Our local approach involves tracking objects in a single camera view, simplifying the collected set of weak tracks into entity tracks, clustering the entity entry and exit observations and then analyzing the behavior in each cluster. The camera viewspace approach uses the same data collection, weak tracking, and entity detection approach as our local approach, however, this process is repeated for many overlapping views that span the camera viewspace. The entity tracks learned in each local view are then pushed to a common space (camera viewspace) where they can be combined. We then use an analog process to our local region detection in the global camera viewspace to detect entry/exit regions.

3.1 Local View Region Detection

Our process for detecting local entry and exit regions is summarized in Fig. 3.1. Given a local camera view, we observe and track objects in the view using a weak tracker. From the collected weak object tracks (tracklets), we learn a set of entity tracks (described in Chapter 4). The entity track entry and exit observations are then used to hypothesize a potential set of scene entry and exit regions. This is accomplished by clustering each set of entry and exit observations independently, removing outlier observations from each cluster (if any exist), and obtaining a region shape. Each potential entry and exit region is then scored using a behavioral-based reliability metric which captures the behavioral consistency of

each region (e.g., ensures entry and exit regions behave as they should). Regions with low reliability score are removed, and regions with high scores are kept, producing a final set of entry and exit regions. The region scoring process is described in detail in Chapter 5.

3.2 Camera Viewspace Region Detection

Our process for detecting entry and exit regions within the viewspace of a pan-tilt-zoom camera is summarized in Fig. 3.2. We collect weak tracking data from a set of overlapping local camera views that collectively span the camera viewspace. Using the set of weak tracks in each local view, we obtain a set of entity tracks (the green input box in Fig. 3.2 is a repetition of the input box in the local approach (Fig. 3.1)). The entity tracks from each view are then projected to a common space (camera viewspace). Using the same processes for our local region approach (but adapted for the camera viewspace) we cluster entity track entry and exit observations, remove outliers, and generate a set of potential entry and exit regions. Then, each region is scored using the same behavior reliability metric as in the local approach, and regions with high reliability score are kept.

3.3 Method Applications

Each method described above is useful under certain conditions. The local view approach is applicable for static cameras (cameras that do not move), and PTZ cameras at a fixed orientation. This approach is desirable if the area of interest is only the local camera view. However, if the camera is a PTZ camera, and there is need for learning the regions in the camera viewspace, then the viewspace region learning approach may be applied. This approach allows the camera to move, but the location of the detected regions to be retained.

Thus, for active camera tasks (e.g., active object tracking, where the camera follows the object), the viewspace approach may be preferable.

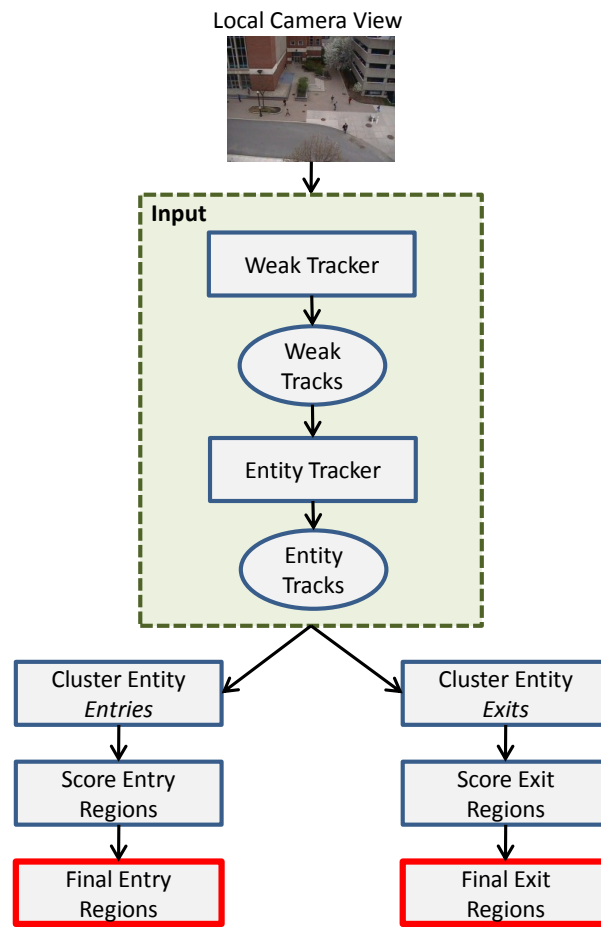


Figure 3.1: Local camera view entry/exit detection system overview.

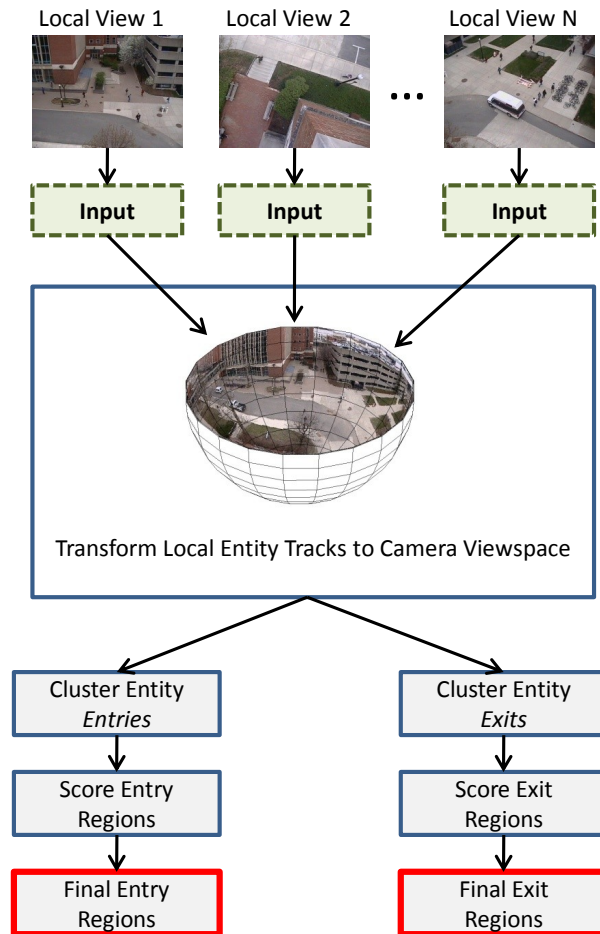


Figure 3.2: Camera viewspace entry/exit detection system overview.

CHAPTER 4: ENTITY DETECTION AND TRACKING

The weak tracking data we collect is far too noisy to say anything about the entry and exit locations in a scene. To obtain a more reliable set of entry and exit observations from the weak tracking data, we simplify the weak tracks into “entities”. In this chapter we explain how we learn a set of entities from weak tracks. This is accomplished by first detecting a set of entities in each frame, and then associating the entities across temporally adjacent frames.

4.1 Entity Discovery

For a given frame, there exists a set of weak track observations (assuming there is object motion in the frame). We wish to learn a set of entities in each frame that were likely to have generated the observed weak track observations. To do so, we first cluster the weak tracking observations in each frame into plausible entities. Entities are represented by a set of spatially close weak tracking observations moving in a similar direction. Thus, an entity may correspond to a person, group of people, vehicle, etc. Though we remain agnostic to what each entity actually corresponds to in the world, we are still able to use them to accumulate reliable entry and exit observations. The resultant entity tracks differ from tracks produced using a strong tracker as we only loosely track entities, allowing them to merge, split, etc. with other entities as they move through the scene.

For a given image frame $f_i \in F$, our weak tracker produces a set of trajectory observations P_i . Our goal is learn a set of entities, T_i , for each frame f_i . To do so we employ

a modified version of mean-shift clustering [4] 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 by introducing a velocity weight to ensure that observations that cluster must be spatially close and traveling in the same direction. For a weak tracking observation $p = (x, y)$, it is shifted to p_{new} until convergence, where p_{new} is computed as

$$p_{new} = \frac{\sum_{i=1}^n p_i \cdot w_{vel} \cdot K\left(\frac{|p_i - p|}{h}\right)}{\sum_{i=1}^n w_{vel} \cdot K\left(\frac{|p_i - p|}{h}\right)} \quad (4.1)$$

Here, K is the chosen mean-shift kernel. We use the Gaussian kernel, defined as

$$K\left(\frac{|p_i - p|}{h}\right) = \frac{1}{\sqrt{2\pi}h} \cdot \exp\left(-\frac{(p_i - p)^2}{2h^2}\right) \quad (4.2)$$

The described velocity weight, w_{vel} , is a function of the velocity angle (ϕ) between p_i and p , and is computed as

$$w_{vel} = \begin{cases} \frac{1}{1 + \exp(-\frac{\cos(\phi)}{\sigma})} & \text{if } |\phi| < \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases} \quad (4.3)$$

Here, σ defines the rate of weight transition (we use $\sigma = 0.07$ for our experiments). Velocity is computed using observations in the previous two frames, and is ignored in the first frame. Thus, only points that are spatially close and traveling in similar directions will seek to the same mode.

We also introduce a blend parameter to the velocity weight computation. Rather than using the initial velocity of p for each iteration (as it seeks to the closest mode), we start by 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 as computing velocities over a short window is subject to noise. To compute the velocity of p at iteration k , we use $dx_p^k = (1 - \alpha) \cdot dx_p + \alpha \cdot dx_{avg}$ and $dy_p^k = (1 - \alpha) \cdot dy_p + \alpha \cdot dy_{avg}$. Here, dx_{avg} and dy_{avg} are weighted averages of the velocities of nearby points.



Figure 4.1: (a) KLT (weak) tracks and (b) their corresponding entity tracks. [Best viewed in color]

The blend parameter α is a linear function of the mean-shift iteration number (increasing from 0 to 1). Thus, after clustering each frame $f_i \in F$ we obtain a set of entities T_i . In our experiments, we use a kernel bandwidth of 15 when detecting entities for our local view approach, and a bandwidth of 30 when detecting entities for our camera viewspace approach.

4.2 Entity Tracking

We next associate entities from frame-to-frame 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 shared trajectory from the underlying weak tracking data. An entity from frame i not connected to any other entity in subsequent frame j corresponds to an entity *exit* event.

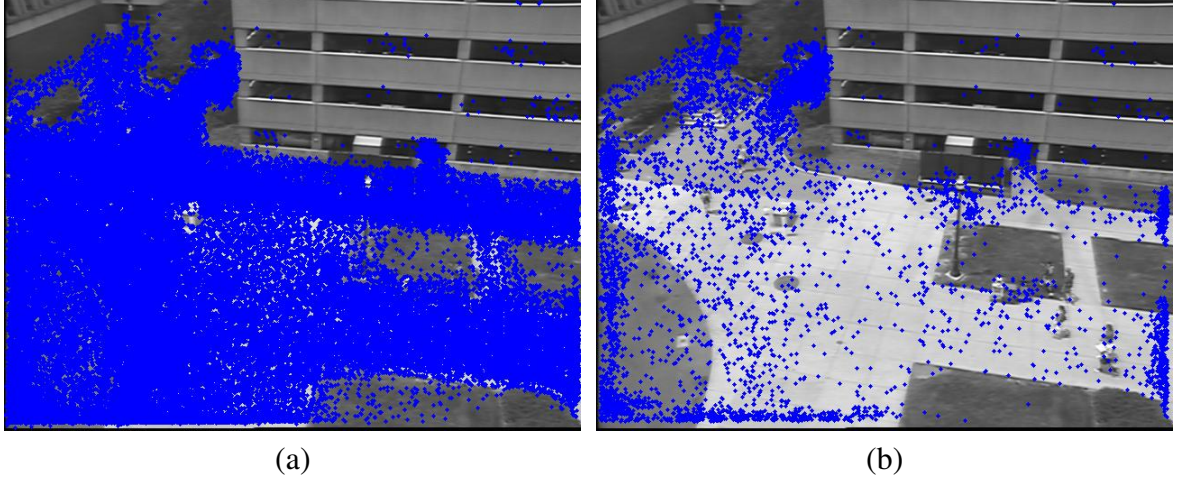


Figure 4.2: (a) Weak tracking start observations, and (b) the corresponding set of entity entry observations.

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* (e.g., split, merge, etc.). Figure 4.1 displays a set of weak tracks (a), and their corresponding entity tracks (b).

Analysis of this graph is used to produce a set of entity tracks whose endpoints correspond to either 1) an entity entry event, 2) an entity exit event, or 3) an entity interaction event. We leverage the entity tracks containing entry and exit event observations to obtain a set of possible scene entry observations, and scene exit observations. We then use these observations to hypothesize a set of potential entry and exit regions. Figure 4.2 displays a set of weak track start observations (a), and the corresponding set of entity entry observations (b). As shown, the entities produce a much more reliable set of entry and exit observations (than using the weak tracking start and stop observations).

CHAPTER 5: ENTRY AND EXIT REGION DETECTION

From our entity detection and tracking framework described above, we accumulate a set of entity entry/exit location observations. We now explain how we detect entry/exit “regions” from these observations, and how we score each region. This is accomplished by first clustering each set of entity entry and entity exit observations. We then remove outlier observations, and extract a region shape from the points within each cluster. Then, each potential entry and exit region is scored using a behavioral-based reliability metric that captures the consistency of the behavior defining each region.

5.1 Region Shape

We first perform standard mean-shift clustering on our set of entry locations (and then exits). The result is a set of entry and exit clusters (we cluster entries and exits independently). We choose mean-shift clustering over an Mixture of Gaussians (MoG) approach (as in [15]) 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. Model selection techniques such as Bayesian Information Criterion (BIC) [23], for a MoG approach that attempt to automatically determine the number of clusters may still sometimes suffer from over fitting (as explained in [7]). Further, the mean-shift clusters better represent the shape of non-Gaussian regions.

After clustering the data we attempt to remove outliers in each cluster, and localize on the area of highest density within each cluster. To accomplish this we employ a convex hull

area reduction technique. We first compute a convex hull around each cluster of entry/exit observations. Then, for each cluster, points on the perimeter are removed in order of ascending density (the density of each point is computed using kernel density estimation). After each point is removed we compute a new convex hull and record the area change that removing the point resulted in. This process is repeated for each point. Doing so results in a distribution of convex hull area changes. The intuition is that outlier points will result in large convex hull area changes. We compute the variance of this distribution (assuming a zero mean), and select observations greater than σ_r standard deviations away. Of the cluster points that produced these outlier convex hull area changes, we choose the point was most recently removed, and discard all cluster points that were removed earlier in time. Thus, we have a new set of points which better represent the true mass of the cluster. An example cluster is shown in Fig. 5.1. Here, Fig. 5.1 (a) shows a plot of convex hull area as points are removed from the cluster shown in 5.1 (c). The red circle on the area change plot denotes the threshold point (most dense point that resulted in a convex hull area change greater than $1.5 \sigma_r$). The area change induced by removing each point is shown in 5.1 (b). The area changes that are $> 1.5\sigma_r$ are highlighted in red. The resultant cluster with outlier points removed is shown in Fig. 5.1 (c).

After removing outlier observations we are left with a final set of entry/exit observations for each entry/exit cluster. We now wish to fit a shape to each cluster. One approach to do this is to compute an alpha-shape [8] for the cluster points. For the 2D case, this involves choosing a circle radius (α) and fitting circles tightly around the cluster point mass. Points that are touched by the same circle get connected with a line. The collection of such points make up the perimeter of the shape. One can imagine, depending on the value of α , the

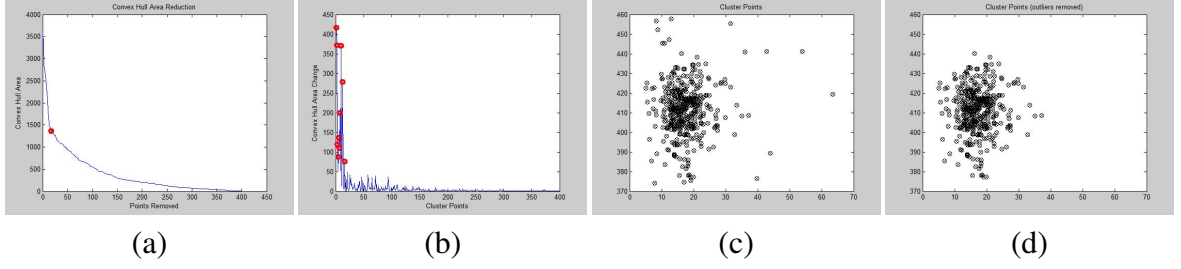


Figure 5.1: Outlier removal example. (a) Convex hull area as points are removed. (b) Convex hull area change for each point. (c) Original cluster points, and (d) final cluster points after removing outliers.

estimated shape will vary. Large values of α will produce a convex hull around the shape. As α is decreased, the shape will become more concave.

Rather than using alpha-shapes, we employ a different approach. We compute a density surface for the points in each cluster using kernel density estimation (KDE) [18]. 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. Smaller kernel bandwidths will force the shape to wrap tightly around the point mass. Example region shapes obtained using kernel density estimation are shown in Fig. 5.2. The black outline represents the region learned as the kernel bandwidth is varied. Thus, unlike [15], our entry and exit clusters reflect the true spatial density and distribution of their underlying observations (which may not be Gaussian).

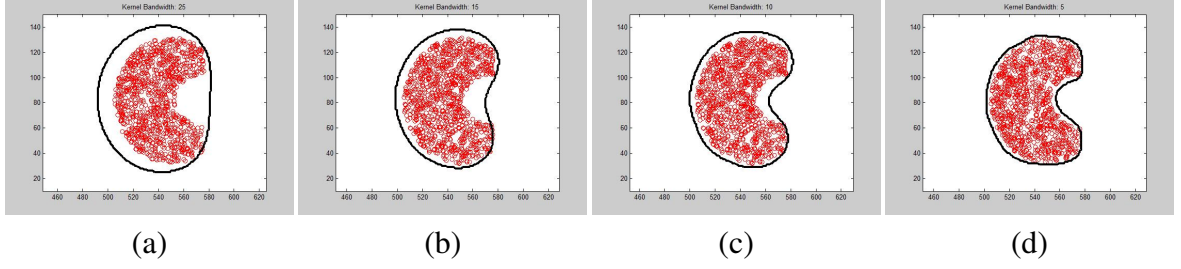


Figure 5.2: Region shapes learned via kernel density estimation using kernel bandwidth values of (a) 25, (b) 15, (c) 10, and (d) 5.

5.2 Region Reliability

We now describe how we validate our entry and exit regions to distinguish reliable entries and exits from those that are the result of noise or partial scene occlusions. In [15], they compute an entry/exit region density and then label regions with density below an arbitrary threshold as noise. Such an approach will not work well if scene traffic is imbalanced, as entries/exits with low popularity (and thus low density), may be regarded as noise. Further, if a scene is very noisy, this method may also classify noise as being a good entry/exit region. We define a good entry region as one with entity tracks emanating out of it, and a good exit region as one with entity tracks flowing into it. Entry regions whose entry-only tracks (or exit regions whose exit-only tracks) exhibit bidirectional activity are unreliable regions, and may be the result of areas with a high rate of tracking failure, partial scene occlusions, or scene noise (trees or other such movement that the tracker may pick up). An example of an entry region exhibiting good and poor directional consistency is provided in Fig. 5.3. Further, for entry regions, other tracks in the scene should not intersect the region in the same emanating direction that defines the region (i.e., the entry region should not be a “through” state, as shown in Fig. 5.3 (d)). Such a scenario would indicate that another

entry region exists behind the current one. The same idea extends to exit regions. Thus, we attempt to capture both the consistency of the entry/exit entity tracks that define each region, as well as the consistency of the interaction between other entity tracks and each entry/exit region.

Using the entry/exit entity tracks that define each region, we learn the distribution of directions that these tracks leave (for entries), and enter (for exits), the region by quantizing the velocity angle at the location that the track intersects the region into one of b bins (we use $b = 8$ in our experiments). The velocity angle for an entity at an intersection location is computed as an average of the velocities of the underlying weak tracking observations that the entity was constructed from (this makes it more robust to noise). This histogram is normalized to provide a probabilistic measure for the directions that entry/exit tracks leave/enter each region. From this distribution q , we compute a directional consistency function \hat{q} , which accounts for any symmetry of the entity track distribution for each entry and exit region in the following manner. For a bin i with probability $q(\theta_i)$, 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 weight (as they correspond to bi-directional behavior), and bin angles close to i receive lower weight. For a region k ,

$$\hat{q}_k(\theta_i) = \frac{\max \left[0, \sum_{j=1}^b w_{ij} \cdot (q_k(\theta_i) - q_k(\theta_j)) \right]}{\sum_{j=1}^b w_{ij} \cdot q_k(\theta_i)} \cdot q_k(\theta_i) \quad (5.1)$$

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

$$w_{ij} = \begin{cases} \exp(-|1 + \cos(\theta_i - \theta_j)|) & \text{if } \cos(\theta_i - \theta_j) < 0 \\ 0 & \text{otherwise} \end{cases} \quad (5.2)$$

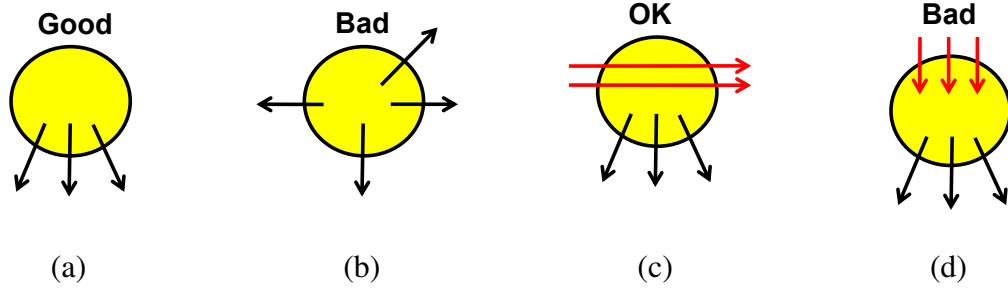


Figure 5.3: Example entry regions displaying (a) good directional consistency, (b) bad directional consistency, (c) good interaction consistency, and (d) bad interaction consistency.

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

In addition to modeling the consistency of behavior corresponding to each entry and exit region, we also incorporate the consistency of how each entry and exit region interacts with other activity in the scene. An entry region that emanates activity in a particular direction should not be intersected by other tracks also traveling in the same emanating direction (the same logic extends to exits). For an entry or exit region k , let D_k be the 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 θ_i . Further, let M_k be the set of outside entity tracks that intersect region k , and $M_k(\theta_i)$ be the number that intersect region k at angle θ_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 interaction consistency may be combined with the previously defined directional consistency to create a single reliability score ψ_k for region k as

$$\psi_k = \left(\sum_{i=1}^b \hat{q}_k(\theta_i) \right) \cdot \left(1 - \min \left[1, \frac{\sum_{i=1}^b \hat{q}_k(\theta_i) \cdot M_k(\theta_i)}{\sum_{i=1}^b \hat{q}_k(\theta_i) \cdot D_k(\theta_i)} \right] \right) \quad (5.3)$$

Here, $\sum_{i=1}^b \hat{q}_k(\theta_i)$ is the directional consistency term (Eqn. (5.1)) across all angles. This score will be low (approach 0) if the defining tracks leaving an entry, or entering an exit, are very symmetric. The interaction consistency score (second term) reflects the manner in which other tracks in the scene intersect an entry or exit region. As the number of intersecting tracks that could discredit a region (M_k) approach 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 1. The resultant regional consistency score is then passed through a sigmoid function, allowing the model to be smoothly adaptive to various noise levels. The final region score, Ψ_k is computed as

$$\Psi_k = \frac{1}{1 + \exp\left(-\frac{\psi_k - \mu_\Psi}{\sigma_\Psi}\right)} \quad (5.4)$$

where μ_Ψ and σ_Ψ should be determined based on scene noise. Figure 5.4 (a) displays a set of potential entry regions, and (b) entry regions with a reliability score $\Psi > 0.75$.

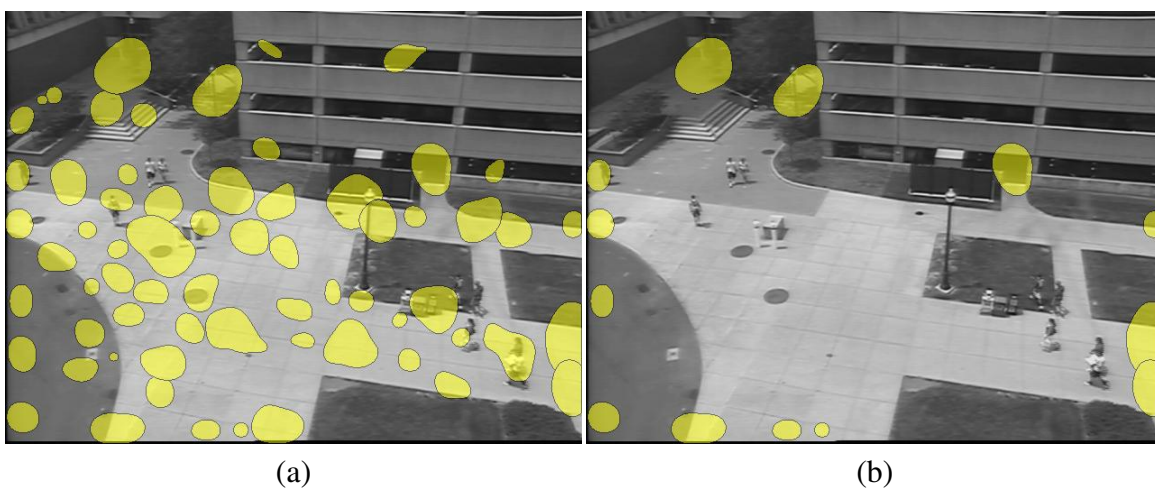


Figure 5.4: (a) Plausible entry regions, and (b) reliable entry regions with $\Psi > 0.75$.

CHAPTER 6: EXPLOITING ENTRY AND EXIT REGIONS

In this chapter we discuss applications of the detected entry and exit regions. Most previous work dealing with entry and exit regions has not focused on anything outside of region detection. In these works, the entry/exit regions are typically regarded as part of the scene model, but more attention is focused on learning pathways between regions than exploring what is capable if only the entry/exit regions are used. We explore this idea in detail in this chapter, and show that relationships between regions can be used to model activity through occlusions, and model common actions (e.g., objects that enter entry “A”, typically exit . Using these oc and provide tracking and anomaly detection applications using only the set of discovered entry and exit regions.

6.1 Exit → Exit Occlusion Relationships

Given the detected set of entry and exit regions, we introduce a method to detect the location of static scene occlusions. An occlusion may be characterized by a set of exit regions where activity disappears into an occluded region, and a corresponding set entry regions where activity reappears in the scene. In the simple case, an occlusion is captured by one exit region and a corresponding entry region (see Fig. 6.1). Consider a more difficult scenario where traffic enters an occlusion region and splits (see Fig. 6.2 (a)), reappearing at two different locations in the scene. Another scenario shown in Fig. 6.2 (b) where activity enters an occluded region via two different locations, merges together while occluded, and reappears in the scene at one location, and Fig. 6.2 (c) shows a scenario where activity

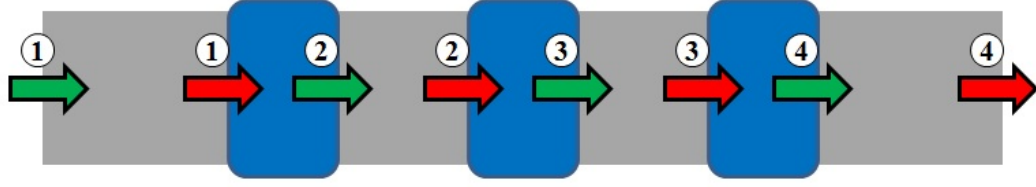


Figure 6.1: Series of occlusions. Entry regions are shown in green and exit regions are shown in red. [Best viewed in color]

enters an occluded region from two locations, crosses, and re-enters the scene at two locations. There are many possible scenarios, and as such we generalize our model to allow for any number of scene exit regions where activity enters an occlusion and any number of corresponding scene entry regions where activity reappears from the occluded region.

As described, behavior representing an occluded region can be represented by a connection from a set of exit regions to a set of entry regions. We detect such regions by constructing a graph, G_o , capturing causal relationships between any 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 exit event at exit region $x \in X$, we look forward in time over a time window of w frames. If an entry event occurs at time lag w_i for scene entry region r , we estimate the distance $\hat{d}(x, r)$ the entity would have traveled given its velocity and the number of frames between the exit observation at x and the entry observation at r , 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, and w_i is the number of frames between the entity exit and entry observation. We do this for all entries that occur within

w frames of the exit event. This formulation is more robust than using simple time-lagged event correlation, as entities may travel at different speeds (e.g., consider pedestrian vs. cyclist). By using estimated distances, we not only can detect whether an occlusion exists, but we can also learn the distance of the path traveled through the occlusion region without actually observing the path.

Given a set of distance estimates between an exit and entry region (x, r) from multiple examples, we determine if there exists a specific distance that is strongly voted for 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. 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). Example distributions from our experiments are shown in Fig. 6.3. The distribution for the true occlusion displays a strong peak, corresponding to the distance traveled by entities through the occluded region. We first smooth the distribution to generalize the data and compute the exit-entry entropy score $H(x, r)$ as

$$H(x, r) = - \sum_{i=1}^n \hat{d}_i(x, r) \cdot \log \left(\hat{d}_i(x, r) \right) \quad (6.1)$$

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

Consider a scenario where occlusions exist in series, one after the other, as shown in Fig. 6.1. In such a scenario, exit 1 would match entries 2, 3, and 4 as they are causally related by traffic that passes through the series of occlusions. Similarly, exit 2 would match entries 3 and 4. To account for possible scenarios as the one shown in Fig. 6.1, and remove

the higher order connections, we observe that traffic going through exit 1 must go through entry 2 and exit 2 before it reaches entry 3. 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 an occlusion. We define a first-order relationship, as a relationship that exists between two regions, A and B, such that there do not exist any alternative paths to get to B from A beside going straight from A to B.

Using this idea, we examine all entry-exit, exit-entry, exit-exit, and entry-entry regions to model the region relationships in the scene. We do so using the same approach used 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 to find the relationships.

The algorithm for our occlusion detection is presented in Alg. 1. Here we use all potential exit-entry regions (not just the reliable regions found in Chapter. 5). Any occlusion exit-entry match found is included in the final set of reliable regions for the scene.

6.2 Entry \rightarrow Exit Non-Pathway Relationships

As we found the correlated behavior between exits and entries to locate occlusions, we can use a similar approach to find the relationship between the entry and exit states. For each non-occlusion entry-exit pair (r, x) we compute a distribution of distance estimates from entry region r to exit region x . If there exists a common pathway of traffic from r to x , the resultant distribution will have a strong mode for the distance of the path traveled

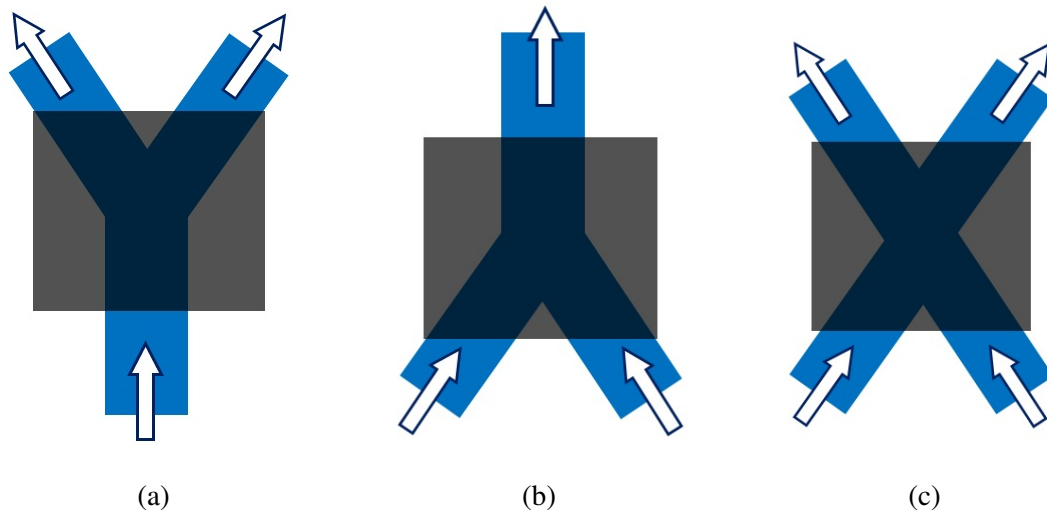


Figure 6.2: Occluded paths behind the occlusion. [Best viewed in color]

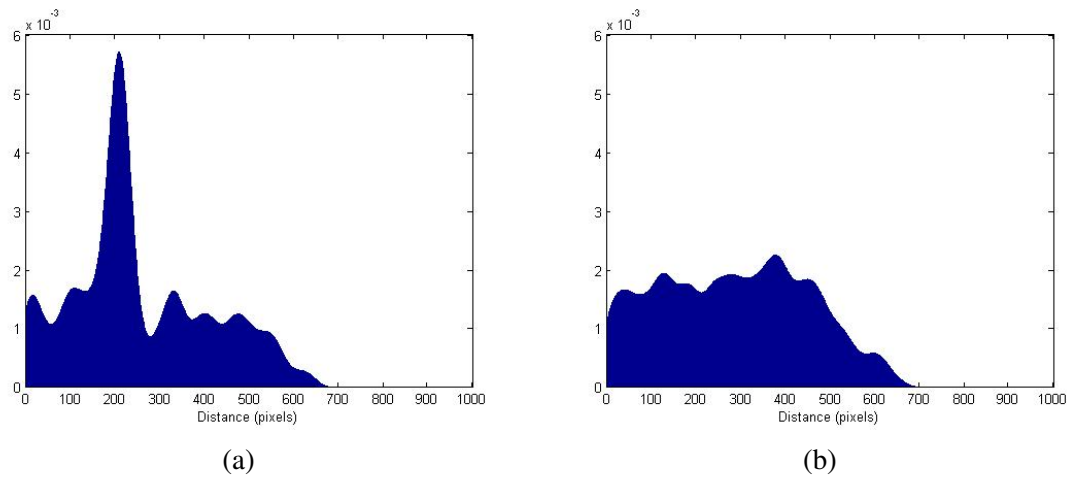


Figure 6.3: (a) Estimated distance distribution between an occlusion exit region and corresponding entry region, and (b) estimated distance between an exit region and entry region not corresponding to an occlusion.

Data: Graph G_o

Result: Set of occlusion pairs $occlusionSet$

```
// Construct  $G_o$ ;  
foreach exit-entry pair  $(x_i, r_j)$  do  
  if  $H(x_i, r_j) < H_{thresh}$  then  
    | add edge  $x_i \rightarrow r_j$  to  $G_o$   
  end  
end  
foreach entry-exit pair  $(r_i, x_j)$  do  
  if  $H(r_i, x_j) < H_{thresh}$  then  
    | add edge  $r_i \rightarrow x_j$  to  $G_o$   
  end  
end  
foreach exit-exit pair  $(x_i, x_j)$  do  
  if  $H(x_i, x_j) < H_{thresh}$  then  
    | add edge  $x_i \rightarrow x_j$  to  $G_o$   
  end  
end  
foreach entry-entry pair  $(r_i, r_j)$  do  
  if  $H(r_i, r_j) < H_{thresh}$  then  
    | add edge  $r_i \rightarrow r_j$  to  $G_o$   
  end  
end  
  
// Search  $G_o$  for valid occlusions;  
foreach potential occlusion pair  $(x, r)$  do  
   $pathCount \leftarrow$  number of paths from  $x$  to  $r$  in  $G_o$   
  if  $pathCount == 1$  then  
    |  $occlusionSet \leftarrow occlusionSet \cup (x, r)$   
  end  
end
```

Algorithm 1: Occlusion detection via graph reduction.

between the regions. 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

$$p(x|r) = \frac{\hat{d}_{mode}(r, x)}{\sum_{x_c \in X} \hat{d}_{mode}(r, x_c)} \cdot \left[1 - \exp\left(-\frac{H_{max} - H(r, x)}{\sigma_p}\right) \right] \quad (6.2)$$

where \hat{d}_{mode} is the raw histogram count for the distance voted for most strongly in the accumulated distance estimate distribution, and $1 - \exp\left(-\frac{H_{max} - H(r, x)}{\sigma_p}\right)$ captures the reliability of the mode in the distribution. Here, H_{max} is the maximum possible entropy score given the number of bins used to model the distribution, and σ_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 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 , Eqn. (6.2) gives a likelihood of the object leaving through each scene exit $x \in X$. From the estimated distance distribution constructed for (r, x) we have an expected distance that the object must travel between r and x . With this, we can score the likelihood of any trajectory entering entry region r and exiting through exit region x as

$$p(traj|x, r) = p(x|r) \cdot \exp\left(-\frac{(\hat{d}_{mag}(r, x) - |traj|)^2}{\sigma_t^2}\right) \quad (6.3)$$

Here, $|traj|$ is the length of the trajectory (in pixels), $\hat{d}_{mag}(r, x)$ is the expected distance traveled between r and x (distance value at $\hat{d}_{mode}(r, x)$), and σ_t allows for some variance between the expected distance and the actual trajectory distance. For our experiments and image size (640x480) we use $\sigma_t = 50$, though it could be learned by observing scene activity.

Such a formulation allows for anomalous tracks to be detected. Such tracks may correspond to an object taking an abnormally long meandering path 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 a short track not leaving any exit (resulting from a tracker failure or from leaving at an unknown exit).

CHAPTER 7: EXTENSION TO CAMERA VIEWSPACE

Thus far we have only dealt with a single camera view. While many commercial surveillance cameras are static view cameras (meaning they do not move), there are also many active pan-tilt-zoom (PTZ) cameras. Such cameras are typically attached to a pan-tilt motor which gives them two degrees of freedom (pan and tilt) to move around and see different parts of the world. While learning the entry/exit regions for a specific camera view is useful, when the camera is moved all of the zones learned from that view will not be applicable to the new view. One possible solution to this problem is to choose a set of fixed pan-tilt-zoom orientations and learn entry and exit regions for each of those camera views. Such a solution is still limited as the camera must be moved to an orientation that it has learned a set of entry/exit regions in and then it must recall the appropriate set of regions for that view. Any activity that requires the camera to be moving, such as actively tracking an object as it moves around by moving the camera as the object moves, will not be able to take advantage of the learned set of regions. Additionally, learning a different set of regions for local views is expensive. To account for these issues, we explain how our region detection approach may be extended to the viewspace of a PTZ camera, rather than just focusing on a single local view.

To achieve this we track objects in many overlapping local camera views, learn entities in the local views, and then fuse the local data into a global camera viewspace where it can be combined. This requires a model to map each local image pixel (x,y) to the global (pan,tilt) space of the camera. We then cluster the entity entry and exit observations in

the global space as we did for the local approach, and then analyze the behavior in this new space to learn a global set of entry and exit regions. Clustering these observations in the camera viewspace and analyzing each cluster requires an analog approach to our local scene analysis as we are now working in a non-Euclidean space. The resulting global entry and exit regions capture entry and exit behavior with respect to the camera viewspace rather than a single local camera view.

We first explain how the geometry of our local view Euclidean space extends to the pan-tilt space of an active camera.

7.1 Camera Geometry

As stated earlier, we require a model to map pixels (x, y) from a local camera view to a common space where the data from different local camera views may be fused together. To achieve this we employ the camera model presented in [21]. This model presents a method to map local (x, y) image coordinates to their corresponding pan-tilt (θ, ϕ) coordinates in the cameras pan-tilt viewspace. Suppose the camera is oriented at pan-tilt location (θ, ϕ) . For an (x, y) coordinate in the local view, the change in pan and tilt to re-orient the camera to focus at that (x, y) location may be computed using the following two equations.

$$\delta\theta = \tan^{-1} \left(\frac{x}{y \cdot \sin \phi + f \cdot \cos \phi} \right) \quad (7.1)$$

$$\delta\phi = \tan^{-1} \left(\frac{y + a}{f \cdot \cos \left(\tan^{-1} \left(\frac{a}{b} \cdot \frac{x}{y+a} \right) \right)} - \frac{a}{f} \right) \quad (7.2)$$

Here, f is the focal length of the camera, $a = \frac{f}{\tan \phi}$, and $b = \frac{a}{\sin \phi}$. Thus, the global coordinate for a local image (x, y) is computed as

$$\theta_{(x,y)} = \theta + \delta\theta \quad (7.3)$$

$$\phi_{(x,y)} = \phi + \delta\phi \quad (7.4)$$

where (θ, ϕ) represent the camera orientation of the local view, and $(\delta\theta, \delta\phi)$ specify the change in pan and tilt for any local image coordinate (x, y) . This mapping allows us to combine data collected at different camera orientations.

The pan-tilt space of the camera can be modeled using a hemisphere, as a sphere is a natural way to model the two degrees of freedom of a PTZ camera and we only require the bottom half of the sphere (the camera is the sphere center and cannot tilt above the equator). Using this model, each pan-tilt (θ, ϕ) camera orientation corresponds to a location on the viewspace hemisphere. An example is illustrated in Fig. 7.1, where a local camera view (Fig. 7.1 (a)) is highlighted on the camera viewspace hemisphere (Fig. 7.1 (b)).

One way to visualize the camera viewspace rather than using a hemisphere is through a spherical panorama. In [21], they present a method to generate spherical panoramas to visualize the camera viewspace (shown in Fig. 7.2 (b)). Here, the pan of the camera increases counter-clockwise and the tilt increases radially from the image center. These panoramas may be thought of as a projection of the camera hemisphere on to the circle that defines the top of the hemisphere (slices through the middle of the sphere that the hemisphere comes from). We will use this panorama representation of the camera viewspace when displaying results throughout this text, however, we remind the reader that this is really a projection of the viewspace and that our work takes place on the viewspace hemisphere (not the panorama image).

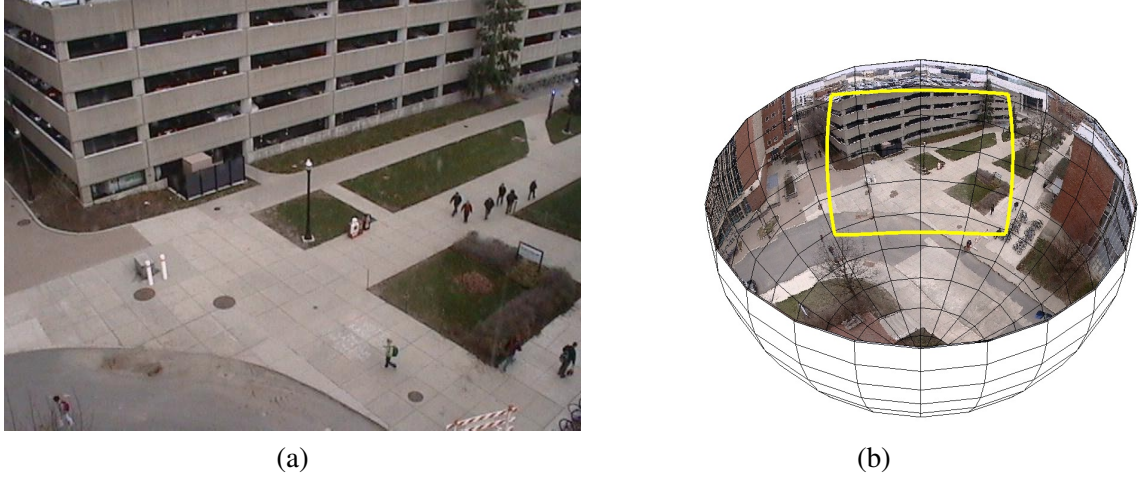


Figure 7.1: (a) Local camera view, and (b) the view highlighted in the camera viewspace.

7.2 Viewspace Data Collection

Previously, we just collected data for the camera view of interest. However, a single view from a PTZ camera only captures a small portion of the total camera viewspace (as illustrated in Fig. 7.1). To extend our approach to the camera viewspace, we first generate a list of pan-tilt-zoom orientations that overlap and cover the entire camera viewspace. We then randomly sample a view from the set, move the camera to that orientation, and track objects via a weak tracker (same as for the local view approach) for a short duration of time (300 frames, roughly 40 seconds). A different view is then sampled from the collection and the process repeats until all views are used. We regard this chunk of data as a “pass” as it is a pass of the entire viewspace composed of local samples, and employ multiple “passes” for our approach. We choose a random sampling of the space rather than a sequential one to prevent any bias when observing the scene. There may be durations of time when the scene is more crowded than others, and sequentially sampling may bias

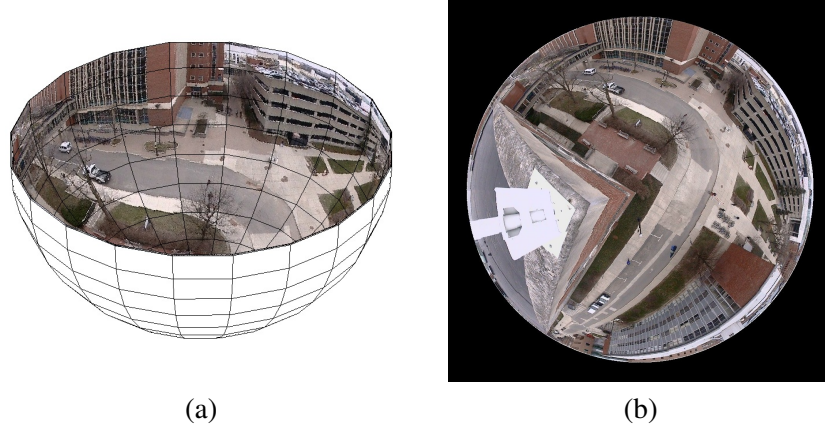


Figure 7.2: (a) Camera viewspace, and (b) spherical panorama projection.

the range of adjacent views we sample when the scene is crowded to appear as if they are more frequently traveled. Figure 7.3 shows weak tracks collected from two different local camera views, and their projection onto a spherical panorama (camera viewspace).

7.3 Viewspace Region Detection

After collecting multiple passes of tracks from a set of local camera views, the tracks in each view are clustered into entity tracks using the approach described in Chapter 4 (as in the local approach). The entity tracks from each local view are then transformed into the global viewspace using the camera model described in Sect. 7.1. This provides us with a set of entity entry and entity exit observations in the camera viewspace. As before, we wish to cluster these entry and exit observations and then score each cluster by analyzing its behavior. To accomplish this, we must adapt our previous approach to work in the non-Euclidean pan-tilt space of the camera.

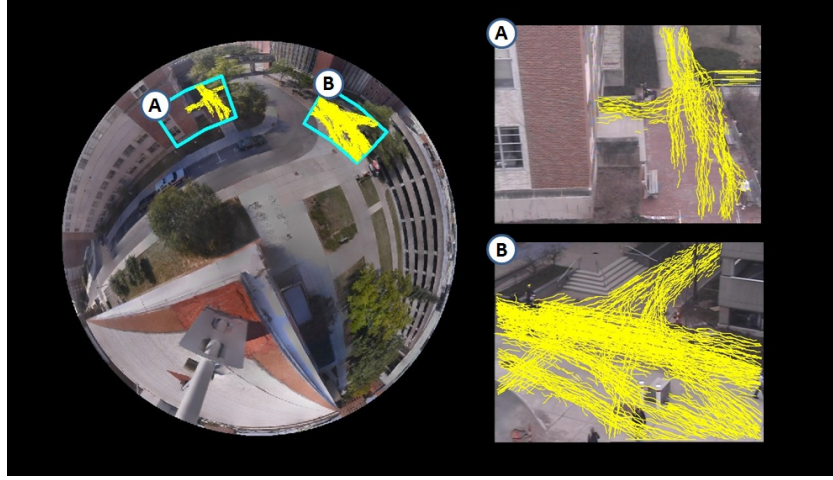


Figure 7.3: Weak tracks (tracklets) from two local camera views displayed on the panorama.

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 between them (on the image plane). To extend this approach to the entry and exit observations that sit on the hemisphere surface, we require a distance metric to compute the distance between points on a sphere. For this, we employ a geodesic approach and define the distance between two points as the shortest arc that connects the points on the sphere surface. The length of this arc may be computed as the great circle distance. The great circle distance between two points p and q on a sphere surface may be computed as

$$d_{sphere} = R \cdot \Delta\hat{\sigma} \quad (7.5)$$

Here, R is the radius of the sphere, and $\Delta\hat{\sigma}$ is the central angle (see Fig. 7.4 (a)) between the two points along the great circle that connects them on the sphere surface. The central

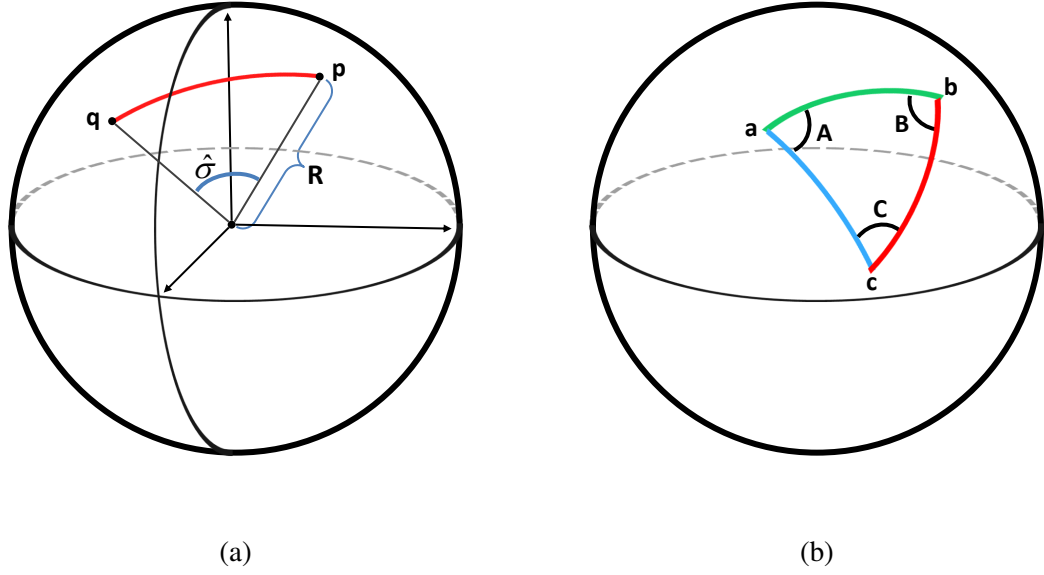


Figure 7.4: (a) Sphere geometry for central angle, and (b) spherical polygon.

angle may be computed using the haversine formula [26] as

$$\Delta\hat{\sigma} = 2 \cdot \sin^{-1} \left(\sqrt{\sin^2 \left(\frac{\Delta\phi}{2} \right) + \cos \phi_p \cos \phi_q \sin^2 \left(\frac{\Delta\theta}{2} \right)} \right) \quad (7.6)$$

The θ and ϕ in the haversine formula usually specify longitude and latitude coordinates, respectively, though they naturally extend to the pan-tilt coordinates of our camera view-space.

This extension allow us to cluster the entry and exit observations in the camera view-space as we did previously for a local camera view. After we obtain a set of entry and exit clusters, we must then remove any outliers in each cluster using the convex hull area-reduction technique described in Chapter 5. This requires a convex hull be fit to each cluster and the points on the perimeter of the convex hull be removed iteratively resulting in a distribution of convex hull area changes which we analyze to determine outlier observations.

The convex hull for a cluster of points on a sphere 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. There are several ways to obtain this convex hull. We employ an approach using the ideas from [17]. For each entry and exit cluster, we compute a 3D convex hull for the points in the cluster and the sphere origin point. Points in the cluster that are connected to the sphere origin produce the set of points for the convex hull on the sphere surface. Other points that are not connected to the sphere origin are interior points. We then compute the area of the spherical polygon formed by the set of points that make up the spherical convex hull. The area of a spherical polygon may be computed as

$$S = (\Theta - (n - 2) \cdot \pi) \cdot R^2 \quad (7.7)$$

where Θ 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 may be computed using the law of spherical cosines. For three points on a sphere 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. 7.4 (b)). Then, any angle in the triangle (A for example) may be computed as

$$A = \cos^{-1} \left(\frac{\cos(d_{bc}) - \cos(d_{ab}) \cos(d_{ac})}{\sin(d_{ab}) \sin(d_{ac})} \right) \quad (7.8)$$

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

The last extension required for working on the hemisphere is to be able to determine if a given point (θ, ϕ) on the hemisphere is contained in an entry or exit region (spherical polygon). This is used when determining entity track intersection with entry or exit regions (each region is a spherical polygon in this space). This can be accomplished similar to the Euclidean approach where a line from the point in question is drawn to a point known to be outside of the polygon. If the line intersects an even number of edges of the polygon, the

point is not in the polygon. Otherwise the point is inside of the polygon (odd number of intersections). This approach works for both concave and convex polygons. To test whether a point in pan-tilt space is within a spherical polygon, intersection with each polygon edge (arc) must be tested. To test intersection with respect to an arc, we employ the approach described in [2]. First, the pan value of the point in question can be compared with the pan values of the arc endpoints. If the pan of the point in question does not sit within the end points of the arc, then the point is sure to not intersect the arc. Otherwise, if it does sit within the pan range of the arc, another test must be performed. Here, an arc is constructed from the North Pole (known to be outside of the spherical polygon) to the point in question. The next step is to determine if the arc between the North Pole and point in question intersects the polygon edge arc. To achieve this, an angular comparison can be performed. Let N be the point at the North Pole, P be the point in question, A be one end of the polygon arc, and B be the other end. First, $\angle NAB$ is computed. Then $\angle NAP$ is computed. If $\angle NAP$ is greater than $\angle NAB$ then arc \overline{NP} must intersect arc \overline{AB} . This angle comparison takes advantage of the fact that we can confidently choose the North Pole as a point known to be outside of the spherical polygon in question, as we only require the southern hemisphere of the sphere.

Using the above processes to work on the spherical surface, our previously defined local image approach may be applied at the camera viewspace scale. The entity tracks used within the camera viewspace (displayed in Fig. 7.5 (a)) originated in many different overlapping camera views. As such, the edge of each local camera view will create an artificial entry/exit region due to traffic crossing the view borders. However, our approach is able to eliminate these regions as tracks from other (overlapping) camera views will run through these artificial (artifacts of the local view border) regions, driving down their



Figure 7.5: Entity tracks in the camera viewspace.

interaction consistency score (e.g., these regions will resemble through states). Thus, by utilizing the scene **behavior** we are able to extend our local view approach by intelligently selecting overlapping local views whose behavior will discredit regions that result from local view borders. Using this process, we are able to learn the true environmental entry/exit regions (e.g., building doorways) within each camera viewspace, as well as regions at the border of the camera viewspace. Thus, regardless of where the camera is oriented within its viewspace, a check may be performed to see if any global entries or exits exist within the local camera views.

7.4 Using the Viewspace Model in a Local View

We have shown how viewspace entry and exit regions may be detected (with respect to the viewspace of a camera). The global regions are useful as they help distinguish

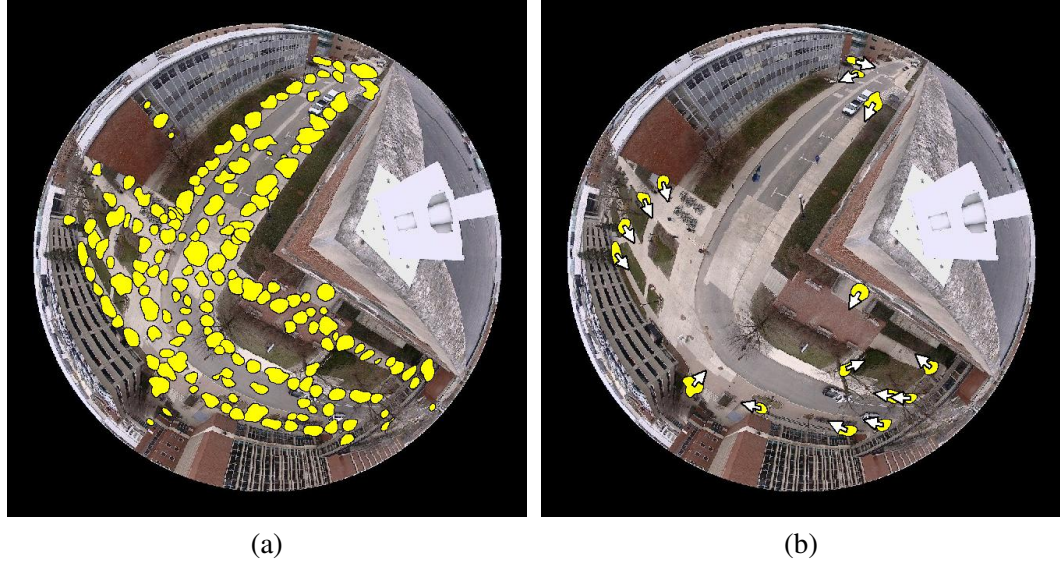


Figure 7.6: (a) Camera viewspace potential entry regions, and (b) reliable entry regions $\Psi > 0.75$.

an environmental entry or exit from one that results from the edge of the image in any local camera view. Regardless of how the camera is oriented, we now have the ability to determine if any global entry or exit regions exists within the local camera view (previously we could only learn regions for a specific view). Depending on ones application, however, the local view regions that correspond to the edge of the image may still be useful. To supplement the learned global set of regions, we introduce a method to detect areas of activity near the edge of any local camera view. Thus, for any local view, any learned global regions that exist in that view may be utilized, but the user can also have an idea of the image border activity as well.

We achieve this by constructing an activity mask for each camera, and then intersecting the current local camera view with the viewspace activity mask. The camera viewspace is first gridded (using pan by tilt grid). The entity tracks for that camera (learned in a set of

local camera views) are then mapped to a set of bins in the viewspace grid. This creates a histogram of activity at each pan-tilt (θ, ϕ) grid cell. A threshold is then applied to keep bins with some activity. This mask is then projected to a panorama image where image processing techniques may be applied more easily to determine the final activity mask. First, a median filter is used to remove salt and pepper noise in the mask. The mask is then dilated to generalize its shape and a connected components algorithm is employed to learn the segments in the mask. Segments with sufficient area are kept to create the final activity mask. An example activity mask is shown in Fig. 7.7 (b) for the viewspace represented by the panorama in Fig. 7.7 (a).

For any local camera view, the intersection of the view border with the activity mask may be used to determine areas of image border activity. These regions may be combined with any global entry or exit regions that exist in the local view to create a more extensive summarization of activity occurring in the view (example shown in Fig. 7.8 for the view highlighted in Fig. 7.7).

7.5 Viewspace Region Exploitation

Being able to detect entry/exit regions in a camera's viewspace can be very useful. Defining such regions in the viewspace allows the camera to be moved anywhere while still retaining the entry/exit location information. Thus, as described in the previous section, for any local camera view, a test may be performed to determine if any viewspace entry or exit regions exist in the view. This relaxes the limitation of only learning regions in a local camera view (as they are bound to that view). However, learning regions in the viewspace does place certain limitations on which region exploitation applications can be applied to the viewspace model.

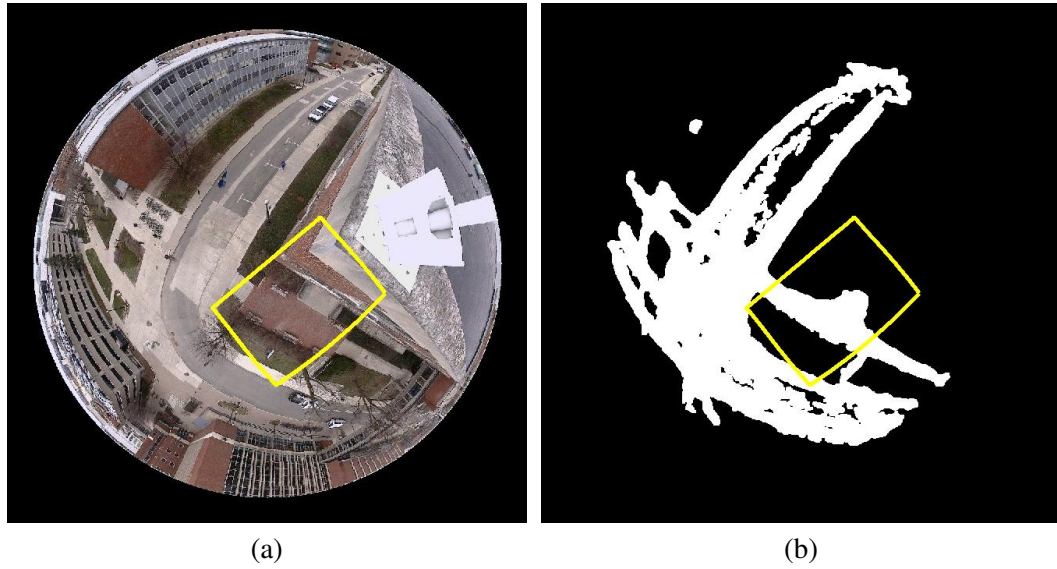


Figure 7.7: (a) Local camera view highlighted on panorama, and (b) camera activity mask.

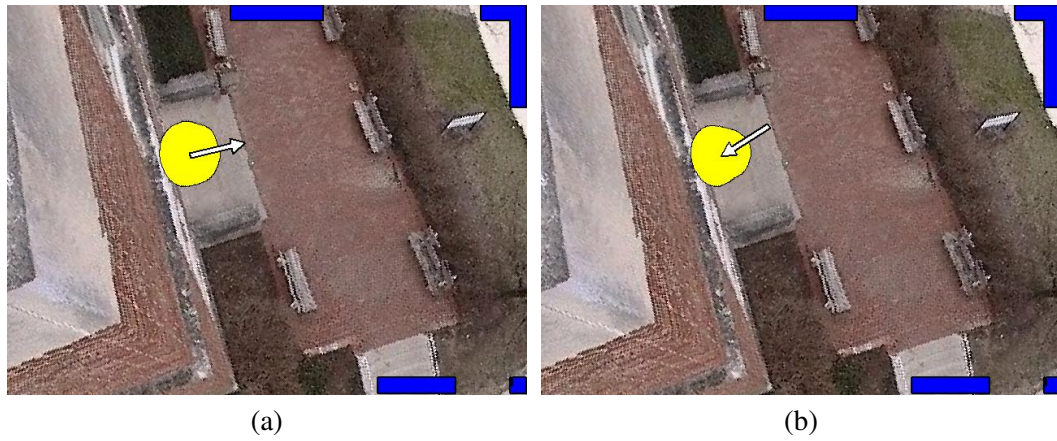


Figure 7.8: (a) Viewspace camera entries and, (b) viewspace exits for a local view with image border activity shown in blue.

Previously we relied on detecting relationships between regions to discover occlusion activity and region relationships in a static scene (e.g., entry \rightarrow exit and exit \rightarrow entry). With our viewspace approach, we can still perform these relationship tests, though only for region pairs that are visible in a local camera view. Thus, we can still detect occlusion activity. Though we can also theoretically recover entry \rightarrow exit relationships as well (provided the entry and exit can be seen in a local view), such a model will not be as useful unless all entry and exit regions are used, which will typically not be the case due to entry and exit regions being spread far across the viewspace. Thus, we will only focus on occlusion discovery as it is very likely for occlusion activity to be captured by a single camera view, though we acknowledge that there may be longer occlusions that cannot be captured by a single view.

For any exit and entry pair from the set of regions detected in the viewspace, the method described in Chapter 6 may be applied to determine if a causal relationship exists between the region pair. This can be accomplished using the data used to detect the viewspace regions, or from new data collected after the regions are learned. Collecting new data after the regions are detected may allow for a test to be performed for regions that are visible in a single camera view, though happened to not be visible in any camera view used to detect the regions. Using the causal relationships that are learned from this process, occlusion activity may be detected in the same manner as the local camera view approach.

CHAPTER 8: EXPERIMENTS AND RESULTS

In this chapter we provide experimental results for our entry and exit region discovery methods. We detect entry and exit regions for seven local scenes of varying difficulty, and compare our local region detection method to two existing approaches. We also provide a discussion on ground truth and offer a quantitative analysis of our region shape detection method. Additionally, we evaluate our, occlusion detection, and entry \rightarrow exit non-pathway relationship methods, and evaluate our viewspace region detection method on three different pan-tilt-zoom cameras. We also compare our local and viewspace approaches, and explain how our viewspace regions may be aggregated in world space.

8.1 Region Detection Experiments

We ran our single view entry and exit region detection method on seven scenes of varying difficulty. We collected data using actual surveillance cameras located on four and eight story buildings. All seven scenes were captured at a resolution of 640x480, and were recorded for durations listed in Table 8.1. We learned entry and exit regions for each scene, and kept regions with a reliability score $\Psi > 0.75$ and with at least 10 tracks leaving/entering each region. The results we show used a kernel bandwidth of 10 to cluster entry and exit observations for Scenes 2-4 and 7, and a kernel bandwidth of 15 for Scenes 1, 5 and 6 (as they show a more zoomed in view). Additionally, when performing our convex hull area-reduction technique to remove outliers we chose a threshold of $\sigma_r = 1.5$. This value should be determined by the reliability of the tracking data (e.g., how likely are

noisy entry and exit observations). For computing the final region score, Ψ , for each entry and exit region we used parameter values $\mu_{\Psi} = 0.04$ and $\sigma_{\Psi} = 0.10$. For each Scene we also display an arrow corresponding to the most popular direction of motion out of (entries) or into (exits) each detected region. Results for Scenes 1-4 can be seen in Fig. 8.1, and results for Scenes 5-7 can be seen in Fig. 8.2. A discussion of results for each scene is given below.

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 learn all of the expected entries and most of the expected exits. For the entry regions, we split up a few of the regions along the bottom of the image which should probably be merged semantically. For the exits, we fail to learn the region in the top left. This is due to the tracking in that part of the scene being more unreliable due to both camera perspective and shadows. We also fail to learn a region corresponding to motion exiting into the parking garage.

In Scene 3 we were able to learn 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 walkway even though it is infrequently traveled with respect to the other regions in the scene.

We also do a good job detecting regions in Scene 4, and even learn activity coming and going from the parking garage on the left of the scene. The one region we do have problems 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 shadow on the sidewalk that our weak tracker does not perform well with.

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

Table 8.1: Data collection durations in minutes for Scenes 1-7.

In Scenes 5 and 6 we also learn all of the expected regions. Both regions exhibit an occlusion (bridge walkway in 5, and tree in 6) and we learn entry and exit regions at locations where objects walk behind and re-appear from these occluding structures. The other interesting aspect to these scenes is how the regions tend to adhere to the right side of the walkways, demonstrating the manner that pedestrians typically travel.

Lastly, we again do a good job detecting 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 compared our results to the methods described in [15] and [31] for Scenes 1-4 (results shown in Fig. 8.3). In [15] they use an EM-based Mixture of Gaussians (MoG) approach to cluster trajectory start and end points to obtain a set of potential entry and exit regions (described by Gaussian ellipses). They then use a density metric to determine which clusters to keep, 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

$$E = \pi \cdot I_1 \cdot I_2 \quad (8.1)$$

where I_1 and I_2 are the eigenvalues of the covariance matrix. When comparing to this approach we used our *entity* entry and exit observations, as it provides for a fairer comparison. Using the weak tracking start and stop observations would be too noisy as this approach expects strong tracking trajectory endpoints. When clustering each entry and exit set we chose a large number of clusters (25) for each scene (as they do). We also experimented with using Bayesian Information Criterion (BIC) [23], to automatically determine the number of clusters. However, for our experiments we chose not to use BIC as it often overfit the data (as BIC may do [7]), producing worse results. We kept clusters with weight $W/E > 2e^{-5}$ (produced the best results). Those clusters are plotted as yellow ellipses in Fig. 8.3.

In [31] they partition the scene into a grid of states, where each state is defined by a grid cell location and motion direction. They also use a binary activity mask for force entry/exit states to be on the border of the scene activity, though we do not employ this technique to allow for a fair comparison (neither our method nor the method in [15] have such constraints). They map each weak track to a set of states, and compute an entry and exit weight score for each state s_i . The entry and exit weight (W_E and W_X) are computed as

$$W_E = C_{start} \cdot \max\left(0, 1 - \left(\frac{C_{in}}{C_{start}}\right)\right) \quad (8.2)$$

$$W_X = C_{stop} \cdot \max\left(0, 1 - \left(\frac{C_{out}}{C_{stop}}\right)\right) \quad (8.3)$$

Here, C_{start} is the number of weak tracks that start in state s_i , C_{stop} is the number of weak tracks that stop in state s_i , C_{in} is the number of weak tracks that transition into state s_i , and C_{out} is the number of weak tracks that transition out of state s_i . Entry and exit states with low weight scores are removed to obtain a final set of entry and exit states. We chose to

keep states with at least 0.75 of the max entry/exit weight for each scene. Those states are denoted by arrows on the gridded scene in Fig. 8.3.

The approach from [15] had difficulty distinguishing reliable entry/exit regions from ones that resulted in tracking noise in many of the scenes. In Scene 1 their approach failed to detect any of the expected entry regions and only detected one of the three expected exits. Most of the regions it detected resulted from noise due to bushes blowing in the wind. We were able to distinguish such regions as being unreliable by analyzing the consistency of the scene behavior with respect to each region. In Scene 2, the approach from [15] over-clustered the entry/exit region near the top middle of the scene. Though 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).

The method from [15] was able to detect many of the expected entry and exit regions in Scene 3, though it failed to learn the middle doorway of the building, and the walkway in the top left of the scene. Additionally, it detected motion around the streetlight as an entry and exit region (due to tracking failures occurring around this location).

In Scene 4 the approach from [15] was able to detect most of the expected regions. Though they fail 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 on the top left of the scene which we miss due to tracking inconsistency in this part of the scene.

In general, choosing a robust means to distinguish reliable regions from unreliable ones for the approach in [15] was difficult. As described, we ended up choosing a threshold of $W/E > 2e^{-5}$. However, changing this threshold can drastically change the results, and it

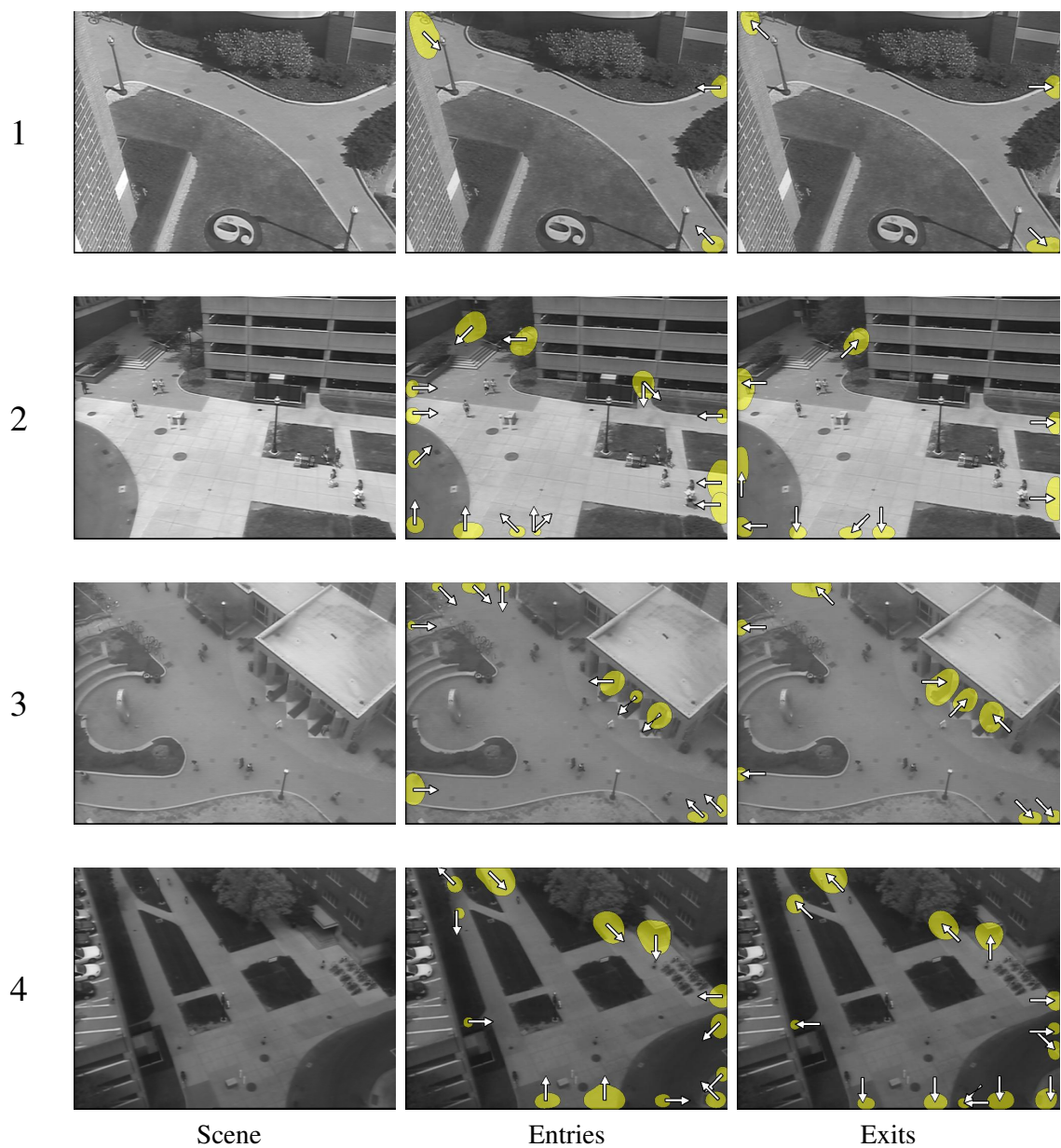


Figure 8.1: Detected Entry and Exit regions for Scenes 1-4.

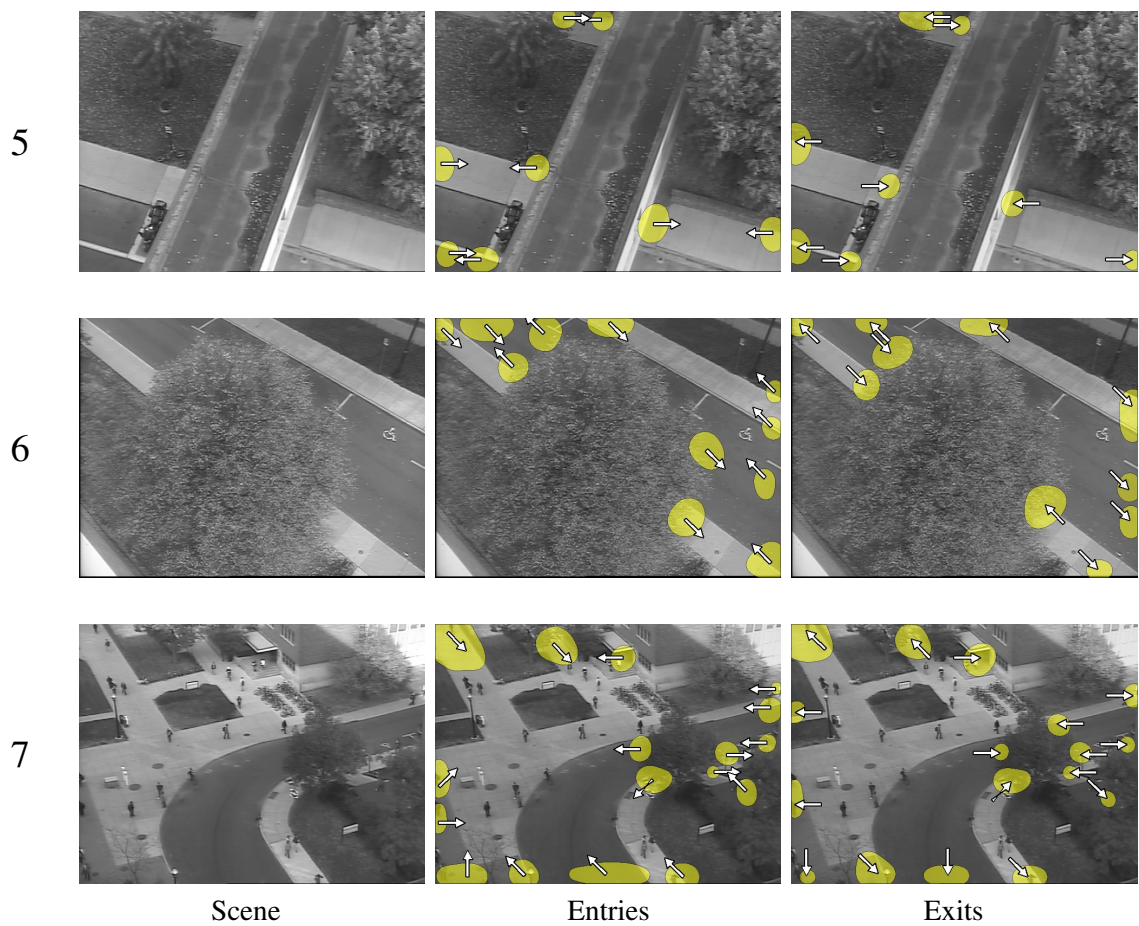


Figure 8.2: Detected Entry and Exit regions for Scenes 5-7.

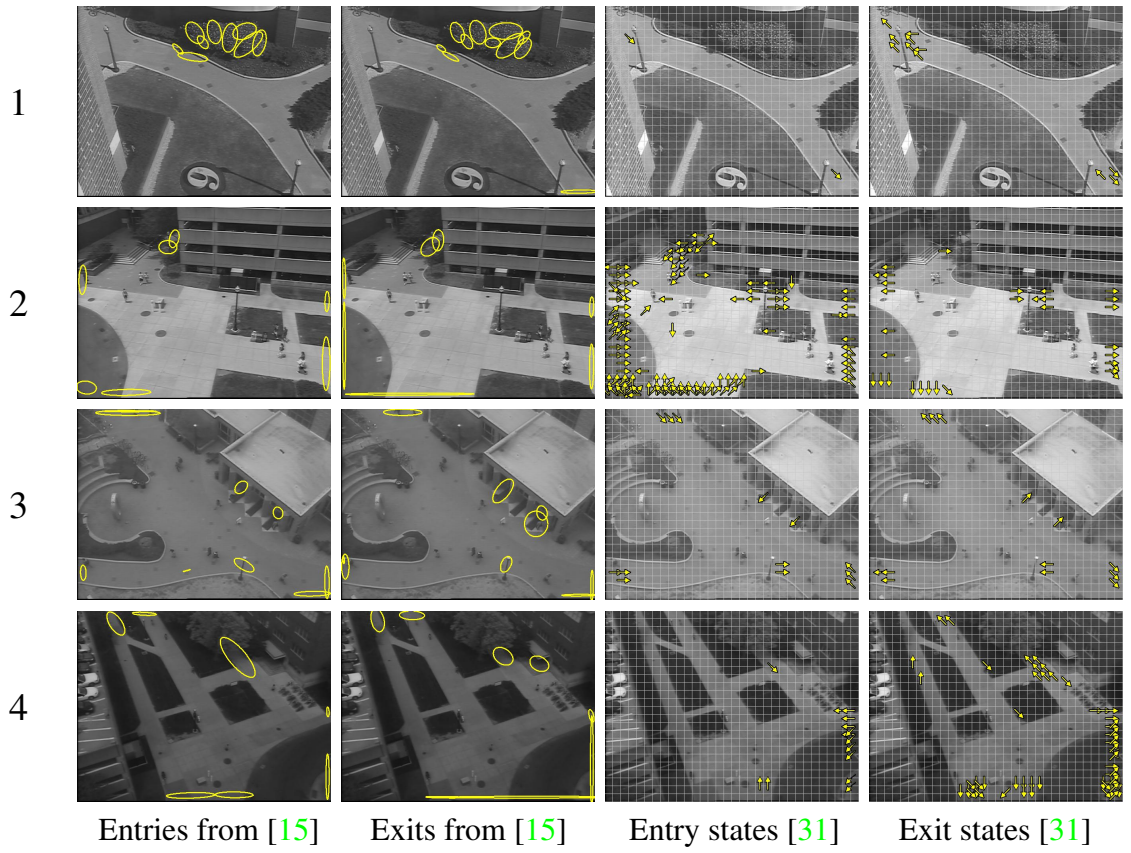


Figure 8.3: Entry and exit regions using the method in [15] (cols 1 and 2). Entry and exit states using the method in [31] (cols 3 and 4). [Best viewed in color]

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).

The approach from [31] failed to detect most of the entry activity in Scene 1, though was able to detect most of the exit activity (though they detect multiple states per region). In Scene 2 they detected states that result from the streetlight occlusion, and learn noisy states in the middle of the scene.

The results produced 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 is not detected.

In Scene 4 they fail to detect most of the entry activity on the left half of the scene. Though they are able to detect a fair amount of valid exit states, they also miss the parking garage entry/exit and the building doorway entry/exit.

In general, the approach from [31] tended to detect a lot of unexpected regions (resulting from noise). It is also unclear how each group of states should be group after they are detected to represent a “holistic” entry/exit region. Overall, our approach produced the best results across all of the scenes.

8.2 Ground Truth Evaluation

Ground truth is an important topic to discuss when doing any type of high-level scene analysis. As discussed in Chapter 2, there is a significant amount of research that attempts to learn or model semantic qualities of a scene. These works typically avoid the question of ground truth. Rather, they usually support their work by showing that the models they learn are useful in certain applications (e.g., anomaly detection, tracking). This can be attributed to the difficulty quantifying such work. For example, if modeling pathways is the goal,

one could certainly show that a model is able to detect anomalous paths that do not adhere to normal pathways, but proving that the learned pathway model is correct is a much more difficult task.

Other works that attempt to detect high level concepts such as groups of people, face a similar problem. In [9], they attempt to detect groups of people in video, and offer a quantitative analysis by comparing their results to the average of nine peoples manually labeled results. This approach of “ground truth through consensus” is common in many similar problems. It is not clear, however, that obtaining ground truth in this manner is desirable. There may be disagreements among manual labels. If this is the case, accepting that the idea of ground truth may be ambiguous for certain problems does not seem correct. Rather, the problem may need to be better redefined to remove ambiguity. Even so, using manual labels for ground truth is still probably preferable to nothing.

When it comes to detecting entry and exit regions, we face similar challenges. Previous approaches to learn such regions tend to ignore the ground truth question. We could estimate ground truth by asking subjects to manually mark the entry and exit zones for a camera view (or even for a camera viewspace). A subject who is given an image of a scene and asked to mark the entry/exit locations could surely produce something reasonable. They would probably mark doorways, pathway entrances, etc. However, in this scenario they would be concentrating solely on the scene structure and ignoring the behavior. They may mark a doorway that is never used, or miss an entrance that results from a shortcut traveled by pedestrians. Additionally, they may mark a sidewalk that intersects the image border as an exit while in reality the exit really does not span the entire width of the sidewalk (due to people only walking on the right side of the sidewalk). Such subtleties can only be realized when the behavior of the scene is carefully analyzed. The authors from [28] share

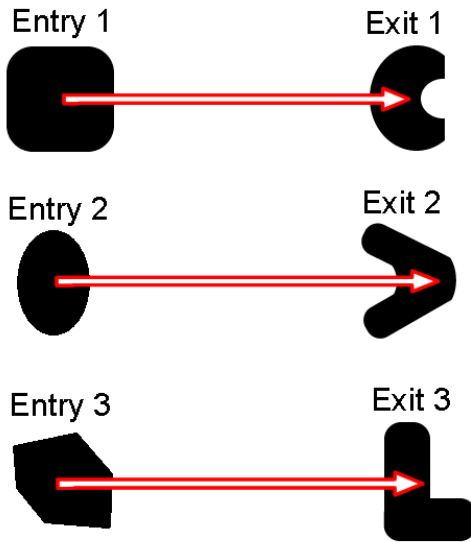


Figure 8.4: Synthetic entry and exit regions.

our view and state 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. We offer a quantitative evaluation of our method to learn region shapes, as well as an analysis of the amount of data required to detect regions shapes reliably.

8.2.1 Region Shape Evaluation

We designed an experiment to test the shape of the regions that we learn with respect to scene noise. To do so, we defined three entry-exit region pairs (shown in Figure 8.4), and generated synthetic weak tracks between each pair. This was achieved by generating synthetic “entities” that moved from each entry to exit for each pair of regions.























































								
								
								
								
								
								
Ground Truth	1	2	3	4	5	6	7	8

Figure 8.5: Detected entry and exit shapes across all eight noise levels.

The motion of each underlying “entity” was generated in the following manner. We first created a base track by randomly sampling a starting location within an entry shape, and an ending location in its corresponding exit shape. Using a pre-defined entity size, represented as a circle around the base start and end locations, we randomly generated 1-3 more weak tracks for the entity. These additional tracks were generated with constraints similar to those of our KLT tracker (e.g., weak tracks cannot be too close to each other). We repeated this process 1000 times for each entry \rightarrow exit pair to simulate 1000 entities. We then fragmented the weak tracks for eight different noise levels (1-8) by adding gaps to the synthetic tracks. Here, the noise level corresponds to the average number of gaps per weak track for each entity (e.g., for noise level 3 each weak track contains on average 3 gaps). When adding each gap, a random location and track was chosen. If the gap could not be added at that location, a new random location was sampled. If after 50 attempts the gap was still unable to be added, we did not add the gap (meaning the tracks were already very fragmented). Additionally, we ensured that at least one weak track still captured the underlying “entity” entry and exit location (we did not delete all of the ends from the weak tracks). From this process we obtained eight different 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 the synthetic weak tracks. From this we learned entities, learned potential entry/exit regions, scored the regions, and kept regions with a reliability score $\Psi > 0.75$. We then compared the learned entry and exit regions with the ground truth regions that the weak tracks were sampled from. For our method we used a kernel bandwidth of 25 to cluster entity entry/exit observation (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 each region score. Results for each entry \rightarrow exit pair can be

seen in Tables 8.2, 8.3, and 8.4. For each entry/exit pair we display the F1 score, precision, and recall (averaged over five trials) capturing how our detected regions compared to the ground truth regions. We also display a signal to noise ratio (SNR), computed as the ratio of the number of ground truth entry/exit observations to total entry/exit observations (averaged over five trials). As we add more gaps to the tracks (noise increases), the number of noise-induced entry/exit observations increases. Thus, the SNR is lower for higher noise levels. A progression of how the region shapes we learned changed with respect to noise is also shown in Fig. 8.5.

Our method was able to learn the correct number of entries and exits for each trial. As the number of gaps per track was increased from 1-8 (the noise increased), our method performed worse. Though our precision scores were high even as the noise increased, the recall score were negatively impacted by noise. This can be attributed to the noise starting to approximate the ground truth entry/exit observations as the amount of noise increased. When removing outlier observations in each cluster, our method attempts to analyze changes in convex hull area as perimeter points are removed. As the noise becomes more dense, the change in convex hull area becomes smaller and it becomes increasingly more difficult to differentiate outlier observations from valid entry/exit observations. Our scores are noticeably worse for exit 2 than the rest of the regions. This can be attributed to our method being unable to perform well when a region has a concavity filled with sparse noise, as we rely on our convex hull approach which has difficulty with concavities. Such a problem could be corrected by replacing the convex hull in our convex hull area reduction approach with a representation that could accommodate concave shapes - such as alpha shapes. Additional techniques to analyze the consistency of each cluster could also be applied, such as the method presented in [30] by Streib and Davis. However, we do not feel

this is a huge shortcoming as such concave shapes do not occur frequently in practice (from our observations).

We did not include results for noise level 0, as it is a degenerate case for our algorithm. Our convex hull area-reduction attempts to remove noise from each cluster. For scenarios where there is not any noise, and the data is uniformly distributed (as in our synthetic experiment), our method will perform very poorly (as expected). There will be natural variation in the area changes induced by each point (due to the nature of the uniformly distributed data) which our algorithm will incorrectly perceive as noise. For real-world data, which is not exactly uniform and will always consist of noise, our algorithm will perform well.

We next varied the number of objects to determine how the number of “entities” impacts the ability of our algorithm to detect the region shape. We first chose a fixed noise level (3), and a set of synthetic weak tracks for that noise level (from the previous experiment). We defined five object quantities (25, 50, 100, 250, and 500) and randomly sampled objects of each amount for ten trials. We then ran our entry/exit region detection method for each set of objects. Results for entry/exit pair 3 can be seen in Table 8.5.

As the number of tracks was increased from 25 to 1000, we were able to recover the underlying entry and exit shape with greater accuracy (F1 score) and reliability (less variance between trials). The set of 1000 objects represents the entire set from which each other amount was sampled from (thus it could only be ran one time and has no variance). The exit shape (the ‘L’) was a more difficult shape for our method to learn than the polygon-shaped entry region. The recall score was generally high for all object amounts. This can mostly be attributed to the entities being uniformly sampled within each region. Without



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

Table 8.2: Results for Entry 1 Synthetic Ground Truth Experiment. Note, F1, precision and recall scored are the average over five trials

many samples, however, it is more difficult to determine the object border. This is reflected by the (generally) poorer precision score for lower object counts.

8.3 Region Exploitation Experiments

After evaluating our entry and exit region detection method, and generating the above quantitative results, we ran our occlusion detection algorithm, and learned forward entry → exit non-pathway region relationships. Additionally, using the detected occlusions and region relationships we demonstrate how they may aid in object tracking and anomaly detection.



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

Table 8.3: Results for Entry 2 Synthetic Ground Truth Experiment. Note, F1, precision and recall scored are the average over five trials.



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

Table 8.4: Results for Entry 3 Synthetic Ground Truth Experiment. Note, F1, precision and recall scored are the average over five trials.



	Objects	F1 Score / St. Dev.	Precision	Recall
	25	0.798 / 0.179	0.838	0.839
	50	0.816 / 0.142	0.780	0.920
	100	0.844 / 0.077	0.760	0.969
	250	0.896 / 0.034	0.830	0.978
	500	0.910 / 0.027	0.868	0.962
	1000	0.940	0.99	0.89
	Objects	F1 Score / St. Dev.	Precision	Recall
	25	0.692 / 0.200	0.612	0.852
	50	0.736 / 0.105	0.620	0.942
	100	0.755 / 0.120	0.632	0.974
	250	0.791 / 0.088	0.681	0.968
	500	0.821 / 0.012	0.757	0.902
	1000	0.90	0.940	0.866

Table 8.5: Results for Entry and Exit 3 Synthetic Ground Truth Experiment, object count varied. Note, F1, precision and recall scored are the average over ten trials for object quantities 25-500. The values for 1000 objects represent the full set of data (which the other object amounts were sampled from).

8.3.1 Occlusion Discovery

We tested the proposed occlusion detection method, and display results for Scenes 3, 5, 6, and 7 (as they contain 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. Results are presented in Fig. 8.6.

In Scene 5 we successfully learned the bridge occlusion region. In Scene 6 we successfully learned the tree occlusion region on the sidewalk, and learned the tree occlusion in the street in the direction of the one-way street traffic. In Scene 7 we learned the tree occlusion in both directions on the near sidewalk, and in one direction in the street (as it is one-way). No natural occlusions existed in Scene 3 so we created a set of eight synthetic occlusions and added them to the scene (shown in black). Some were arranged in series along the bottom and top walkways. The last exit region in the middle of the scene along the bottom actually is associated with two entry regions due to traffic splitting under the occlusion.

8.3.2 Entry-Exit Relationships

For each scene we also found the relationships between non-occlusion-entry and non-occlusion-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 Eqn. (6.2). Entry-exit relationship results for Scenes 3 and 5 are shown in Fig. 8.7 where we display an arrow from each entry region to the most likely exit region. In Scene 3, we learned the entry-exit relationships corresponding to the sidewalk traffic. In Scene 5 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 region has a strong relationship with the walkway exit on the left of the scene, and the right region has a strong relationship with the nearest

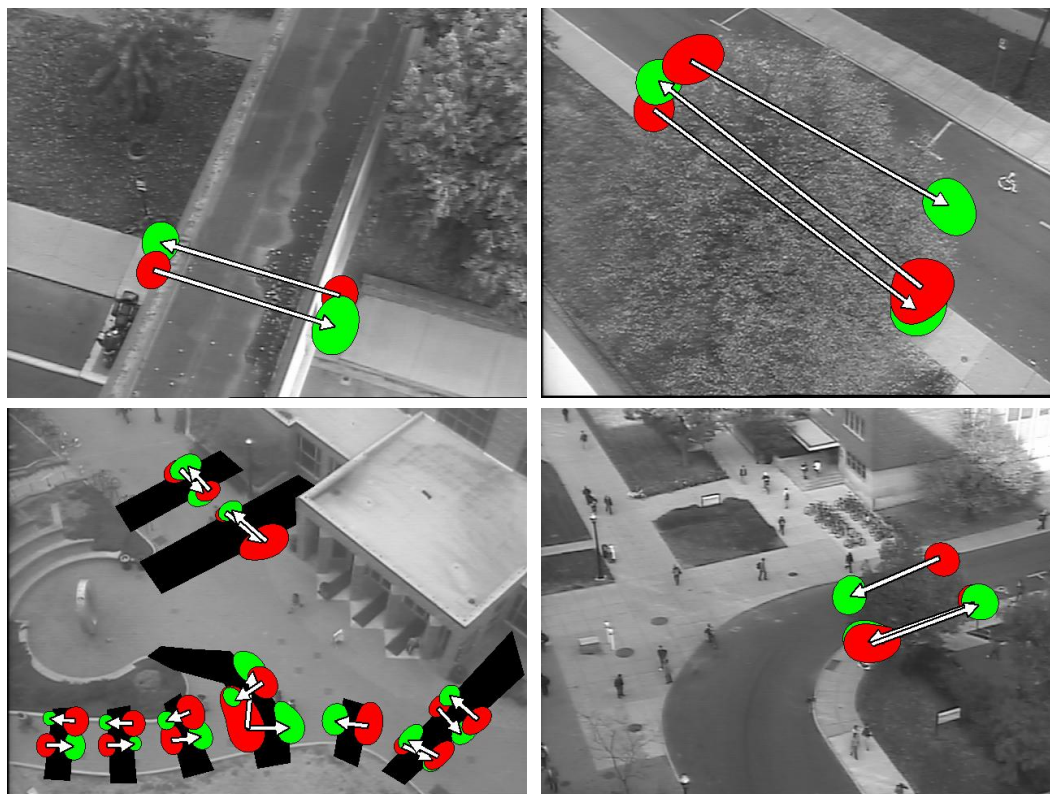


Figure 8.6: Occlusion detection results from Scenes 5, 6, 3, and 7 with arrows drawn from the occlusion exit (red) to the occlusion entry (green). [Best viewed in color]

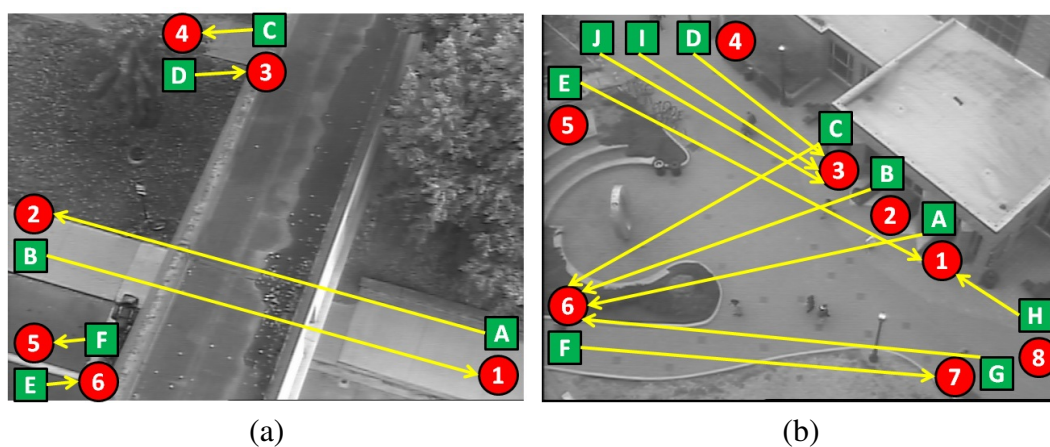


Figure 8.7: Most probable entry and exit connections for Scenes 5 and 3. [Best viewed in color]

door exit to the building. More in depth results can be seen in Table 8.6 and Table 8.7, where individual entry-exit pair probabilities are provided.

8.3.3 Region Exploitation Applications

Next, we applied our learned occlusion and entry-exit relationship models to tracking and anomaly detection applications.

Tracking

We ran a covariance tracker [19] (stronger tracker) along with the learned occlusion model to demonstrate how the model may be used to aid in tracking. We used the feature set $f_k = [x, y, R, G, B, I_x, I_y]$ to build a covariance descriptor for the tracker. The tracker was set up to automatically initialize on motion occurring in the learned scene entry regions, providing that the motion was moving in the same direction as the expected activity for leaving the region. If the tracker is able to initialize, it attempts to track the object until it either 1) leaves the Scene or 2) loses the object. If the object enters an occlusion exit, an expected wait time is estimated from the learned occlusion distance and object speed, and then the tracker attempts to re-acquire the target at the learned entry location(s) for the occlusion. It searches the expected occlusion re-entry area until it finds a match within 2σ from the learned covariance model. Some results are presented in Fig. 8.8 (a) and (b). As shown, the tracker was able to track objects despite the occluded region, and was able to reacquire the object at the occlusion entry via cues derived from our learned model (expected wait time). The results are especially significant for Scene 6, as the tree covering the sidewalk is large and would be difficult for traditional tracking methods to track through such an occlusion.

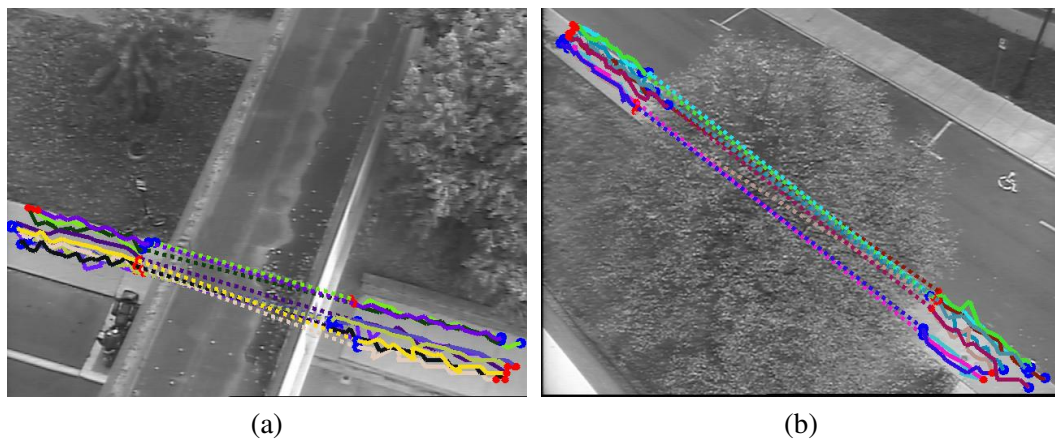


Figure 8.8: Covariance tracking results from Scenes 5 and 6. Dashed lines correspond tracks on either side of the occlusion. [Best viewed in color]

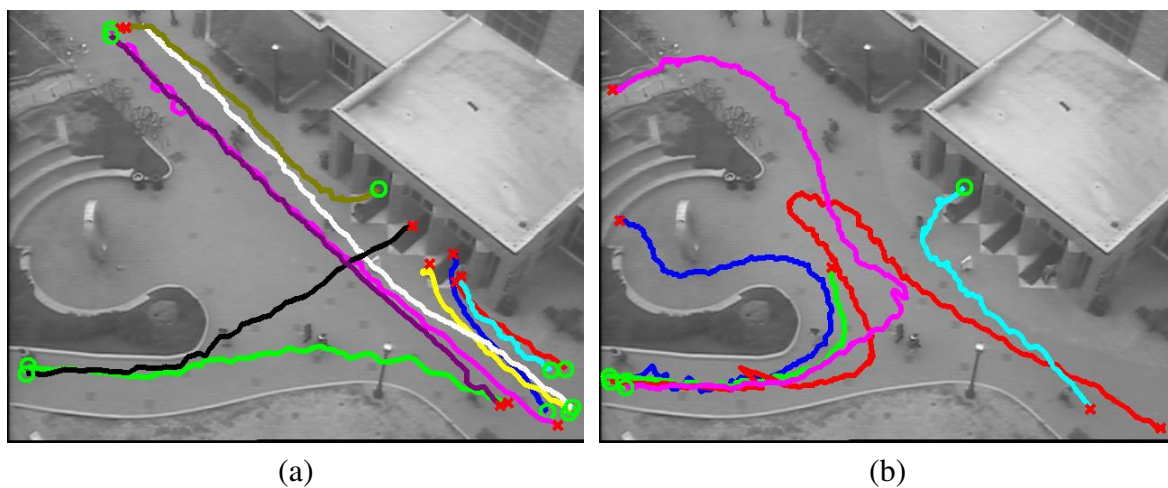


Figure 8.9: Track likelihoods for (a) the ten most likely and (b) five least likely tracks. [Best viewed in color]

	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

Table 8.6: Activity probabilities between entries (rows) and exits (columns) for Scene 5. Region labels shown in Fig. 8.7(a).

	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

Table 8.7: Activity probabilities between entries (rows) and exits (columns) for Scene 3. Region labels shown in Fig. 8.7 (b).

Anomaly Detection

We also used the covariance tracker to track multiple people in Scene 3 and rank their trajectory likelihood. We smoothed the trajectories and then obtained a likelihood score for each trajectory using Eqn. (6.3). We display the ten most likely trajectories in Fig. 8.9 (a), and the five least likely trajectories in Fig. 8.9 (b). The least likely trajectories in Fig. 8.9 (b) correspond to an object taking a long meandering path between regions (red), exiting the scene at a location not corresponding to an exit region (blue), taking a path between regions that is infrequently traveled (magenta, cyan) and a short trajectory resulting from tracker failure (green). Our model is able to detect such anomalous behavior as we not only learn activity relationships between entry-exit pairs, but also the distance of the path taken between them. If desired, a threshold could be learned/employed to automatically detect rare events.

8.4 Camera Viewspace Experiments

We ran various experiments to test the extension of our region detection method to work on a camera viewspace. We present entry and exit detection results for three different pan-tilt-zoom cameras, and a comparative result of a local scene learned on the camera viewspace. We also display a projection of our learned viewspace regions from multiple cameras to an orthographic ground plane for two cameras.

8.4.1 Viewspace Region Detection

We tested our global entry/exit region detection algorithm on three pan-tilt-zoom cameras each located on 4, 4, and 8 story buildings respectively. We first collected data for

Camera	Pass Count	Duration (minutes)
1	14	425
2	8	243
3	8	243

Table 8.8: Data collection durations for our global region detection approach in minutes for cameras 1-3.

each camera by developing a list of overlapping camera orientations that cover the camera viewspace. We used two different lists - one slightly shifted from the first to generate more overlap among the camera views. Each camera’s viewspace was sampled multiple times (multiple viewspace passes). The number of passes for each camera along with an estimated total collection duration is provided in Table 8.8. We used a kernel bandwidth of 0.02 to cluster entry and exit observations on a unit sphere for Camera’s 1 and 2, and a kernel bandwidth of 0.01 for Camera 3 (8 story building vs. 4). We used a threshold of $\sigma_r = 2.0$ to remove outliers from each cluster, and $\mu_\Psi = 0.4$ and $\sigma_\Psi = 0.05$ when computing reliability scores for each potential entry and exit region. We kept regions with a reliability score $\Psi > 0.75$ and at least 10 tracks leaving/entering each entry/exit region. Results are shown in Figures 8.11, 8.12, and 8.13.

For Camera 1 we learn most of the expected entry and exit regions. We are able to learn regions corresponding to activity entering and exiting two of the building, as well as activity entering and exiting the camera viewspace. The regions on the periphery of the camera viewspace correspond to the furthest areas away from the camera that we can track reliably with our weak tracker. Object moving in areas further away (especially moving toward or away from the camera) are not tracked well due to a minimum displacement constraint for the weak tracker (features being tracked must move a minimum amount between frames).

In addition to the expected regions we are able to learn, we also learn a few regions due to partial and full scene occlusions (e.g., trees, streetlight), and fail to learn an entry and exit region corresponding to entry and exit activity of the building in the bottom of the camera view. This is due to the perspective of objects coming and going from that view restricting the amount of displacement each object moves between frames, preventing our weak tracker from reliably tracking such objects.

For Camera 2 we are also able to learn many of the expected entry and exit regions. We again learn regions corresponding to entry and exit behavior of two of the building in the camera viewspace. We also learn regions around two tree occlusions which, using our local occlusion detection approach, could be detected as occlusion regions.

For Camera 3 we learn many expected regions. We are able to detect entry and exit regions corresponding to the three doorways of the building in the center of the view. Additionally, we learn an entry and exit regions 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 of the building that the camera is mounted on. This is due to the scene being reflected in the window. Sample frames for the reflection are shown in Fig. 8.10. Lastly, we also learn some noisy regions on the periphery of the camera viewspace (e.g., from traffic motion many blocks away).

Overall we do a good job detecting entry and exit regions in all three cameras. Though we do learn noisy regions occasionally, and fail to learn some expected regions we are able to learn most of the expected regions in each camera viewspace. We feel that perspective prevents us from learning reliable regions near the periphery of the camera viewspace, and may be corrected using 1) tracking parameters that vary with camera tilt, or 2) a zoom



Figure 8.10: Example frames showing objects moving on the ground through the reflection in a building window for Camera 3.

adaptive sampling of the space (we use a constant zoom level). These ideas are discussed further in chapter 9.

8.4.2 Local vs. Viewspace Approach Comparison

We ran an experiment to compare our local camera view region detection method to our analog camera viewspace approach using the same local scene data. Using the data for Scene 3 in our local region detection experiments, we detected entry and exit regions for Scene 3 using our viewspace approach. To do so, we projected the entity tracks for Scene 3 to the camera viewspace (Scene 3 was actually collected from a single view of a pan-tilt-zoom camera), and ran our region detection algorithm in that space. Results can be seen in Fig. 8.14.

As shown, we were able to learn the same set of regions globally that we learned locally. While the learned region shapes are slightly different (the local space is an approximation of the global space), they are significantly similar between the two spaces. The main direction

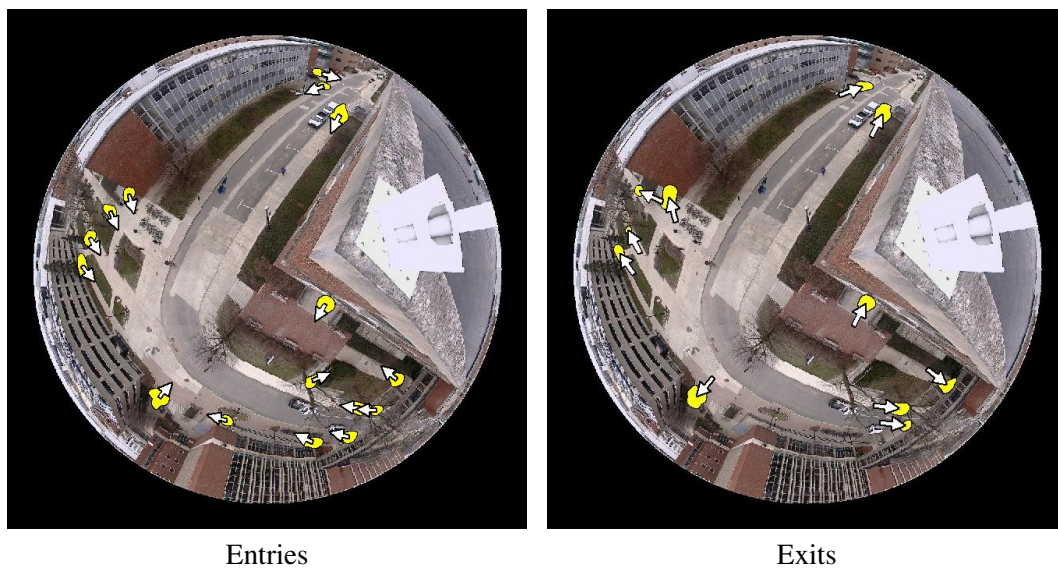


Figure 8.11: Viewspace entry and exit regions for camera 1.

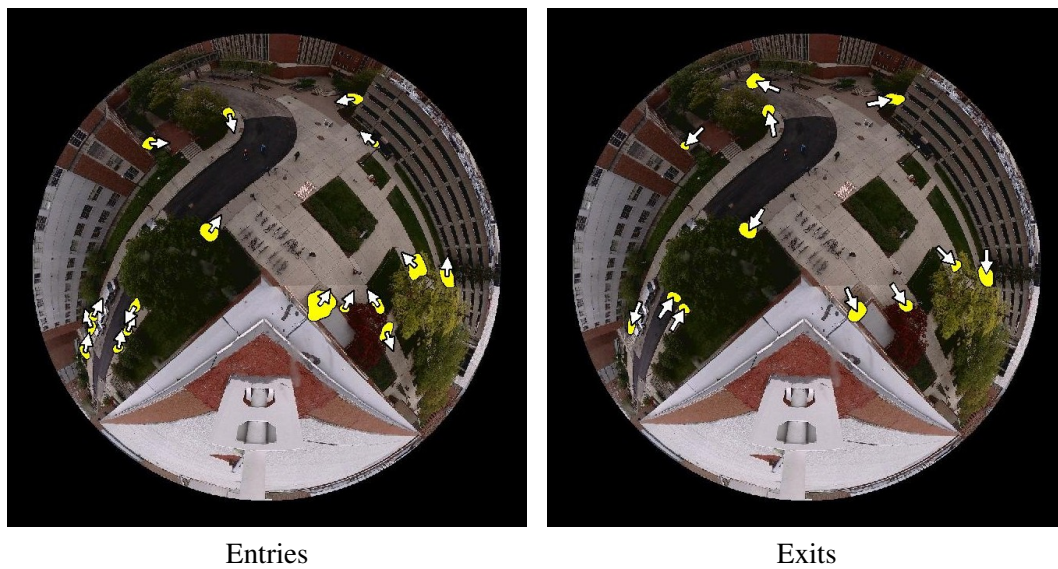


Figure 8.12: Viewspace entry and exit regions for camera 2.

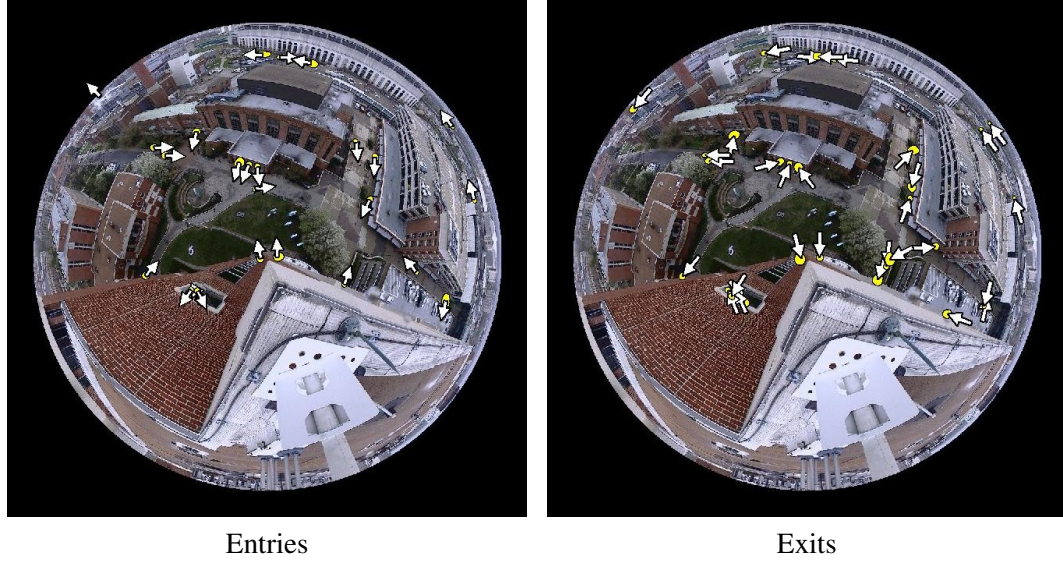


Figure 8.13: Viewspace entry and exit regions for camera 3.

of motion in and out of each region is also very accurate between each space. It is also worth nothing that we increased the angular histogram bin count used to capture the directional and interaction consistency for each region from 8 to 16 for the viewspace approach to more accurately compute the main direction of motion in the camera viewspace.

8.5 World Space Entry/Exit Regions

Global entry and exit regions are useful for tasks within the scope of the camera viewspace. For tasks that may require interaction between cameras (e.g., tracking objects between cameras), regions learned from each camera viewspace may be projected to a global world space (e.g., lat-lon or UTM). For this we require a mapping between the pan-tilt viewspace of a camera and the global (lat-lon or UTM) world space. To achieve this mapping we employ the registration technique described in [22]. They present a method to

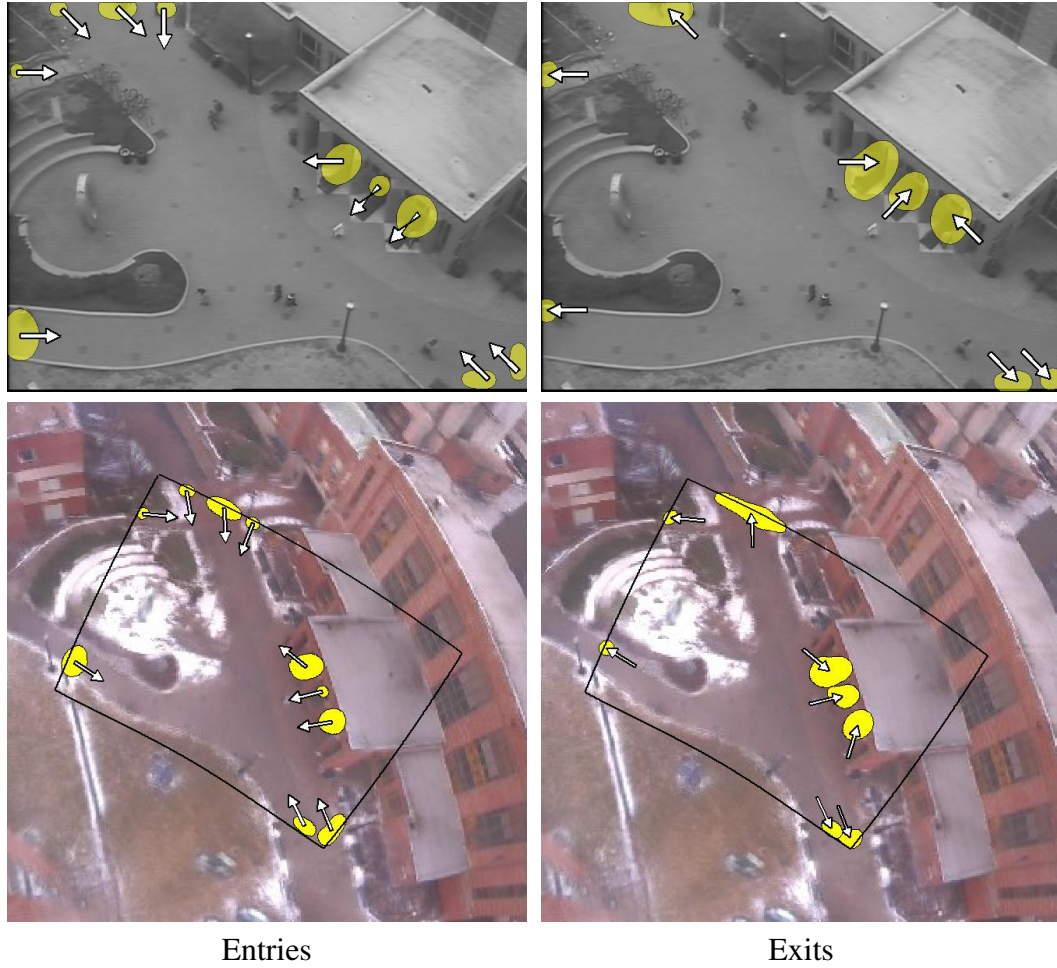


Figure 8.14: Entry and exit regions from a single view learned locally (row 1) and in the camera viewspace (row 2).

register the viewspace of a pan-tilt-zoom camera with a common ground plane. We use an aerial orthophoto as our ground plane. The registration process is composed of two steps, 1) “defishing” the panorama (which representees the camera viewspace), followed by 2) learning a transformation between the defished panorama and the orthophoto. Combining these two steps produces a registration matrix (shown below) that captures the relationship between (θ, ϕ) camera space locations, and (x_g, y_g) locations on the ground plane.

$$\begin{bmatrix} x_g \\ y_g \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & t_x \\ a_3 & a_4 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tan \phi \cdot \cos \theta \\ \tan \phi \cdot \sin \theta \\ 1 \end{bmatrix} \quad (8.4)$$

Here, the six parameters $(a_1, a_2, a_3, a_4, t_x, \text{ and } t_y)$ can be learned via a least squares formulation by manually corresponding (x_p, y_p) points on the panorama image to (x_g, y_g) points on the ground plane.

Using this method, we projected the entry and exit regions learned for Cameras 1 and 2 to an aerial orthophoto. The center of each entry/exit region are displayed in Figures 8.15, and 8.16. We chose to only display the centroid of each region due to possible projection distortion from the mapping model. The mapping model maps points between planes, so distortion is possible if detected regions happen to not exist directly on the ground plane. Displaying the centroid provides a rough estimate of the region location.

Here, the yellow circles correspond to regions from Camera 1, and the blue triangles correspond to regions from Camera 2. The location of each camera is represented by a square (yellow for Camera 1, blue for Camera 2). These two figures display the coverage overlap between these two cameras. The regions are learned with respect to the view of each camera, thus some of the regions do not correspond to true global entry/exit regions, but rather occlusions with respect to each camera view. Such regions are still important, however, for inter-camera tasks. For example, if an object being tracked by Camera 1

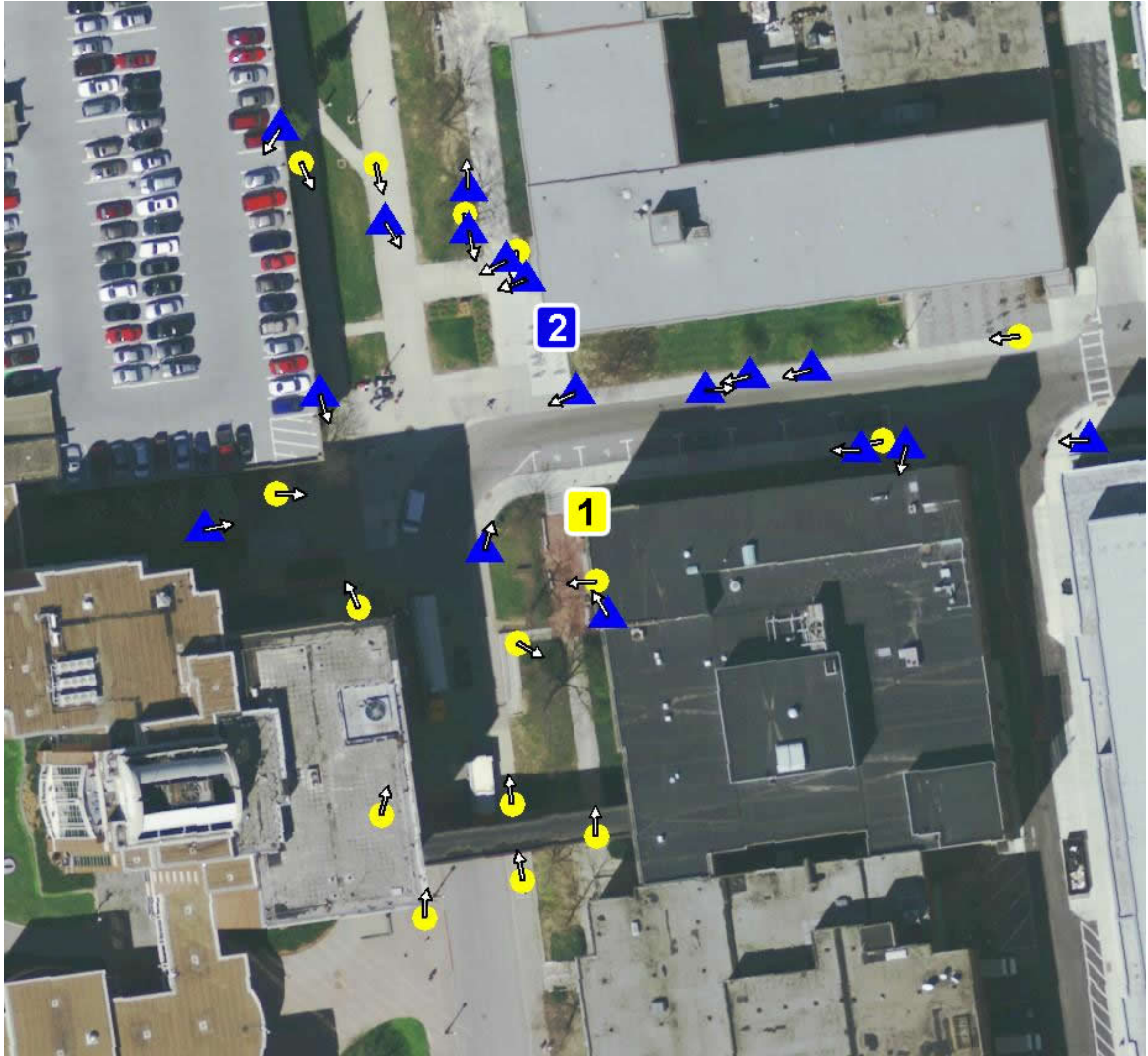


Figure 8.15: Entry regions for Cameras 1 and 2 plotted on an orthophoto.

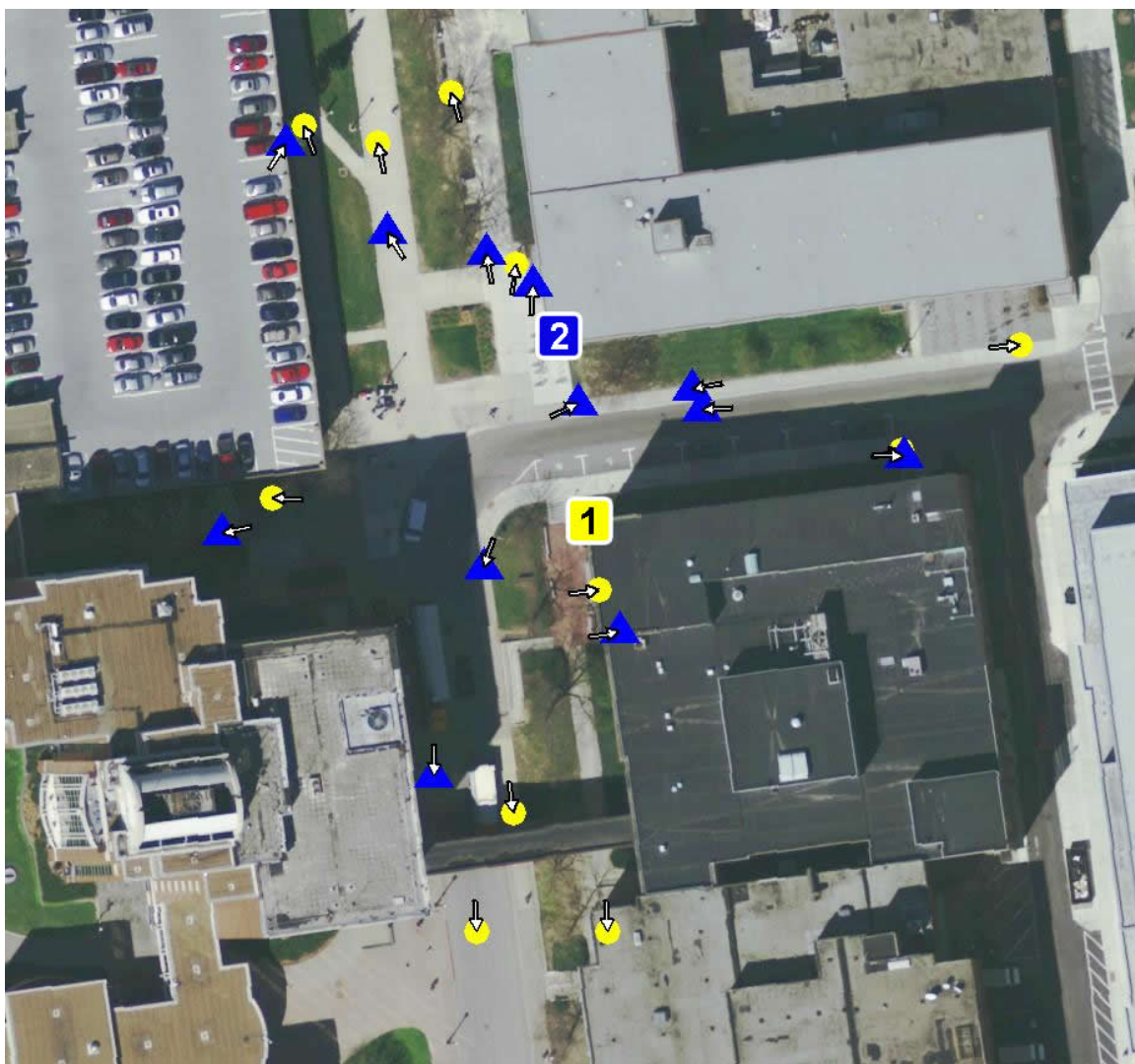


Figure 8.16: Exit regions for Cameras 1 and 2 plotted on an orthophoto.

wishes to be handed off to Camera 2 to continue tracking, it is important to understand where camera 2’s entry/exit regions exists in the world space to ensure a successful handoff.

8.6 Limitations

We have shown entry and exit detection results using our approach on many scenes for both local and global views. As seen in the results, we our method is able to consistently detect the set of expected entry and exit regions in each scenario with great accuracy. However, like all algorithms, there are situations and scenarios where our approach does not perform perfectly. In this section we discuss limitations of our entity tracking technique, our entry and exit detection process, and our occlusion discovery method.

8.6.1 Entities

In order for us to reliably detect and track entities from the underlying weak tracking data, each semantic entity must be tracked by at least one weak track during each frame. If there is a frame where no weak tracks are tracking the underlying entity, we consider this to be an entity exit, and the subsequent frame where a weak track appears on the entity again a new entity entry. Thus, as an object moves through the scene, it must be tracked by at least one underlying weak track between each pair of frames for us to classify the motion of the object into one semantic entity track.

8.6.2 Entry and Exit Region Detection

We are limited to detecting entry and exit regions that adhere to our entry/exit behavior model. That is, tracks that leave (for entries) or enter (for exits) each region must do so in a semi-directional manner. Thus, we are not able to detect entry (or exit) regions where activity leaves (or enters) the region in all directions (e.g., a people coming out of

or moving into a manhole on the ground). Additionally, due to the nature of the weak tracker we use, the entry and exit regions we learn are the consequence of object motion initialization and termination in the scene. While the weak tracker we use is a feature point tracker, we require the features being tracked to move in order to track them. Using our current weak tracking approach as input, regions of a scene where objects typically stop moving, and other objects do not pass through (e.g., at a park bench) would be learned as exit (and entry) region by our approach. Depending on the application, such regions may be important, though they do not correspond to true entry or exit regions where objects appear or disappear from the scene. To prevent such regions from being detected, a different type of object tracker could be used to generate input; one that does not stop tracking objects/features when they stop moving.

8.6.3 Occlusion Detection and Entry → Exit Region Relationships

For our approach to reliably detect scene occlusions we must first be able to learn entry and exit regions and the entrance and exit of the occluding path. Thus, if an occlusion that some people walk behind and other walk in front of exists in the scene, we will be unable to detect entry and exit regions for such an occlusions. In this scenario the occlusion exit and entry would appear as a “through state” as we are unable to differentiate the object tracks that move behind and in front of the occlusion as differing in depth.

Both our occlusion discovery and region relationship detection methods, attempt to correlate activity between regions by estimating the path distances between co-occurring events between regions. This approach requires one distance estimate to strongly stand out for us to consider two regions to be related. Thus, we assume that objects traveling between

two regions use roughly the same path to travel between them. The more paths that are used will introduce more uncertainty to the distribution of estimated travel distances.

CHAPTER 9: CONCLUSION AND FUTURE WORK

We have presented a novel method to detect, and exploit entry and exit regions in video. We have also introduced a novel extension that allows such regions to be detected in the viewspace of a pan-tilt-zoom camera, rather than within a local camera view. After introducing the problem in Chapter 1, we provided a discussion of related work in Chapter 2. A system overview was presented in Chapter 3, and a description of our entry/exit region detection and scoring method was presented in Chapter 5. We then explained how the learned regions can be exploited to learn areas of motion through occluding structures in the scene, and relationships between entry and exit regions that capture higher level semantic “actions” in the scene (e.g., people travel from entry A to exit B). Next, we presented an extension of our local view region detection algorithm to allow for such regions to be detected in the viewspace of a pan-tilt-zoom camera. Thorough experimental results and discussion for each aspect of our approach were presented in Chapter 8, including a quantitative analysis of our region detection method, and a discussion regarding ground truth.

We have shown that utilizing the scene behavior is paramount to detecting semantic regions in a scene. Often such problems are approached in a mechanized manner where data (e.g., object trajectories, entry/exit observations) are collected and then thought of as points or curves in an arbitrary space, completely ignoring the rules that defined the world that they came from. Behavior models have been used previously for a handful of scene modeling tasks (e.g., group detection) and often borrow ideas from sociology or psychology, but such models have not been applied to entry/exit region detection explicitly.

Most previous approaches to entry/exit region detection pose the problem as a strict spatial clustering problem, ignoring the scene behavior that generated the input data used. We have shown that by applying a behavior model for object trajectories, we are able to detect entry and exit regions with much higher accuracy and also require less data, providing the data adhere to the expected entry/exit behavior (compared to approaches that score regions based on density). This can be attributed to the behavior model supplementing the raw trajectory data with higher level information.

We have also demonstrated weak tracking data can be powerful when working with very busy scenes. Though the actual weak tracks cannot be directly tied to semantic objects in the world, we have shown that a grouping of such tracks (entities) in a consistent manner allows for a simplification of the weak track set on which more robust inference can occur. This is especially the case when accumulating a set of entry and exit observations.

Our occlusion discovery method demonstrates that directly modeling the occluding structures in the scene is not always necessary. Rather, we have shown that learning regions where activity exits the scene (moves behind an occlusion), and re-enters a scene can aid in tracking objects through occlusions. Additionally, we have demonstrated that pathways are not needed to model actions in the scene. Rather than learning pathways we model forward relationships between entry and exit regions, and produce a probabilistic model of where an object will exit the scene, depending on where it entered. Such a model also allows for anomaly detection applications (as we have shown in the previous chapter).

We have also demonstrated that it is possible to detect entry and exit regions in the camera viewspace reliably using a collection of overlapping local camera views, and we have also shown that these regions may be pushed to an even more global space to combine information between different cameras.

9.1 Contributions

We have presented several techniques that demonstrate the ideas described above. We have also compared our approach to previous approaches and shown that our entry and exit detection method outperforms previous approaches, especially in crowded and busy scenes. Additionally, we have presented a method to quantitatively evaluate the accuracy of our region detection method with respect to the amount of scene noise, and the number of objects required to accurately estimate the shape of each entry/exit region. Our quantitative results are especially important as most previous works only offer qualitative analysis and anecdotal evidence that their methods perform well. Overall, the main contributions of our work may be summarized in the following manner.

- **Entity tracking:** Our method to construct weak tracking data into “entities” creates a more usable set of scene entry and exit observations. The “entities” represent “coherent motion regions” in the scene which may also be useful for various other visual surveillance tasks (e.g., person counting).
- **Behavior-based entry and exit detection algorithm:** We have presented a method to score the behavioral consistency of each potential entry and exit region. Scoring such regions based on behavior allows for more information to be gained from less data, as the behavior model helps to define the entry and exit regions in the scene which we detect.
- **Quantitative analysis for region detection:** Our quantitative region detection experiments help to evaluate our method outside of our qualitative results. They demonstrate the ability of our method to detect entry and exit regions when the ground truth

regions are known, and show how our approach performs with respect to noise and object count.

- **Method to discover relationships between detected regions:** We have presented an approach to detect causal relationships between exit and entry regions that result from occlusion activity, and have presented a method to model forward entry \rightarrow exit relationships in the scene.
- **Extension to detect entry/exit regions in a camera viewspace:** Lastly, we have presented a novel extension of our local camera view entry/exit detection method to allow for such regions to be detected with respect to the viewspace of a PTZ camera. Learning such regions within a camera’s viewspace frees the camera to move to view different parts of the world, and still retain semantic information about the area it is monitoring.

9.2 Future Work

Below is a discussion of future work that may be interesting to explore to extend or improve upon our current approach.

9.2.1 Adaptive Clustering

Our entry/exit observation clustering technique may over and under-cluster the space when attempting to localize on entry and exit regions. This is especially apparent for the global viewspace approach. Here, the scales that the entry and exit regions exist on may vary greatly (e.g., small doorway vs. large walkway). As such, it may be beneficial to explore alternative clustering approaches where either 1) a mean-shift clustering kernel bandwidth is varied throughout the space, or 2) an alternative method to cluster/partition

the scene is used. For the local image approach, attempting to account for perspective for certain scenes may also be beneficial. This could be accomplished again by varying the kernel bandwidth depending on location of the space after a perspective model is determined for the view.

9.2.2 Adaptive Viewspace Sampling

We generally have difficulty obtaining a reliable set of entry/exit observations at periphery of the camera viewspace (for our camera viewspace approach). This is due to the weak tracking parameters (e.g., feature displacement between frames, minimum distance between observations) being tuned to perform well near the camera, to reduce noise (e.g., trees swaying). These parameters tend to degrade the tracking performance as the camera tilts toward the horizon. This is due to 1) the perspective of objects moving toward the camera appear as if they're not moving, thus causing small displacement between frames, and 2) objects being smaller and not having as many weak tracking observations on them due to the minimum distance between features constraint. One possible solution to these problems is to vary the tracking parameters with respect to camera tilt. This can still be problematic, however, as objects in a single camera view may vary greatly in depth. Additionally, the zoom of the camera could be increased with respect to tilt. This is probably the most robust solution, though it requires many more views to sample the camera viewspace for our viewspace region method.

9.2.3 Online Update

Our approaches to learn both local and global regions require a one time data collection. However, the structure of a scene may vary over time. Aside from re-running our approach

to account for seasonal variations, it would be worth exploring a method to slowly adapt our results via an online update that could be run periodically.

BIBLIOGRAPHY

- [1] F. Bashir and F. Porikli. Performance evaluation of object detection and tracking systems. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2006. 5
- [2] M. Bevis and J.L. Chatelain. Locating a point on a spherical surface relative to a spherical polygon of arbitrary shape. *Mathematical Geology*, 21(8), 1989. 47
- [3] J. Black, T. Ellis, and P. Rosin. A novel method for video tracking performance evaluation. In *In Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2003. 5
- [4] Y. Cheng. Mean shift, mode seeking, and clustering. *PAMI*, 17(8), 1995. 20
- [5] A.M. Cheriyyadat, B.L. Bhaduri, and R.J. Radke. Detecting multiple moving objects in crowded environments with coherent motion regions. In *POCV*, 2008. 10
- [6] A.M. Cheriyyadat and R.J. Radke. Automatically determining dominant motions in crowded scenes by clustering partial feature trajectories. In *Proc. Int. Conf. on Distributed Smart Cameras*, 2007. 10
- [7] N. Cruz-Ramirez and et al. How good are the Bayesian information criterion and the minimum description length principle for model selection? A Bayesian network analysis. In *Proc. 5th Mexican Int. Conf. on Artificial Intelligence*, 2006. 23, 56
- [8] H. Edelsbrunner, D.G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory.*, 29(4), 1983. 24
- [9] W. Ge, R. T. Collins, and B. Ruback. Automatically detecting the small group structure of a crowd. In *IEEE Workshop on Applications of Computer Vision*, 2009. 62
- [10] L. Guan, J.-S. Franco, and M. Pollefeys. 3D occlusion interference from silhouette cues. In *CVPR*, 2007. 11
- [11] L. et. al. Guan. Visual hull construction in the presence of partial occlusion. In *3rd Int'l Symp. on 3D Data Proc., Vis, and Transmission.*, 2007. 11

- [12] R. Kaucic, A.G.A. Perera, G. Brooksby, J. Kaufhold, and A. Hoogs. A unified framework for trackig through occlusions and across sensor gaps. In *CVPR*, 2005. 11
- [13] L. Lee, R. Romano, and G. Stein. Monitoring activities from multiple video streams: Establishing a common coordinate frame. *IEEE Trans. Patt. Analy. and Mach. Intell.*, 22(8), 2000. 12
- [14] D. Makris and T. Ellis. Path detection in video surveillance. In *Image and Vis. Comp.*, 2002. 5
- [15] D. Makris and T. Ellis. Automatic learning of an activity-based semantic scene model. In *AVSS*, 2003. xi, 5, 10, 23, 25, 26, 55, 56, 57, 60
- [16] D. Makris, T. Ellis, and J Black. Bridging the gaps between cameras. In *CVPR*, 2004. 11
- [17] H.S. Na, C.N. Lee, and O. Cheong. Voronoi diagrams on the sphere. *Computational Geometry: Theory and Applications*, 23(2), 2002. 46
- [18] E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33(3), 1962. 25
- [19] F. Porikli, O. Tuzel, and P. Meer. Covariance tracking using model update based on means on Riemannian manifolds. In *CVPR*, 2006. 73
- [20] V. Rabaud and S. Belongie. Counting crowded moving objects. In *CVPR*, 2006. 10
- [21] K. Sankaranarayanan and J.W. Davis. An efficient active camera model for video surveillance. In *Proc. Wkshp. Applications of Comp. Vis.*, 2008. 12, 13, 40, 41
- [22] K. Sankaranarayanan and J.W. Davis. A fast linear registration framework for multi-camera gis coordination. In *Advanced Video and Signal Based Surveillance*, 2008. 81
- [23] G. Schwarz. Estimating the dimension of a model. *Ann. of Statist.*, 6(2), 1978. 23, 56
- [24] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994. 6
- [25] S.N. Sinha and Marc Pollefeys. Pan-tilt-zoom camera calibration and high-resolution mosaic generation. *Comp. Vis. and Image Understanding*, 103:170 – 183, 2006. 12
- [26] R.W. Sinnott. Virtues of the haversine. *Sky and Telescope.*, 68(2), 1984. 45
- [27] K. Smith, D. Gatica-perez, Odobez J.M., and S. Ba. Evaluating multi-object tracking. In *In Workshop on Empirical Evaluation Methods in Computer Vision*, 2005. 5, 6

- [28] C. Stauffer. Estimating tracking sources and sinks. In *In Proc. Second IEEE Event Mining Workshop*, 2003. 5, 10, 62
- [29] C. Stauffer and K. Tieu. Automated multi-camera planar tracking correspondence modeling. In *CVPR*, 2003. 12
- [30] K. Streib and J. Davis. Using ripley's k-function to improve graph-based clustering techniques. In *CVPR*, 2011. 66
- [31] K. Streib and J.W. Davis. Extracting pathlets from weak tracking data. In *AVSS*, 2010. xi, 6, 11, 55, 56, 60, 61
- [32] T. Wada and T. Matsuyama. Appearance sphere: Background model for pan-tilt-zoom camera. In *Proc. Int. Conf. Pat. Rec.*, 1996. 13
- [33] X. Wang, K. Tieu, and E. Grimson. Learning semantic scene models by trajectory analysis. In *ECCV*, 2006. 5, 11
- [34] X. Wang, K. Tieu, and E.L. Grimson. Correspondence-free activity analysis and scene modeling in multiple camera views. *IEEE Trans. Patt. Analy. and Mach. Intell.*, 1(1), 2009. 12