# ABSTRACT

| | |
|---|---|
| Title of dissertation: | MULTI-OBJECT TRACKING, EVENT MODELING, AND ACTIVITY DISCOVERY IN VIDEO SEQUENCES |
| | Seong-Wook Joo<br>Doctor of Philosophy, 2007 |
| Dissertation directed by: | Professor Rama Chellappa<br>Department of Electrical and Computer Engineering,<br>Department of Computer Science |

One of the main goals of computer vision is video understanding, where objects in the video are detected, tracked, and their behavior is analyzed. In this dissertation, several key problems in video understanding are addressed, focusing on video surveillance applications.

Moving target detection and tracking is one of the most fundamental tasks in visual surveillance. A new moving target detection method is proposed where the temporal variance is used as a measure for characterizing object motion. Our method is experimentally shown to produce high detection rates while keeping low false positive rates.

In tracking multiple objects, it is essential to correctly associate targets and measurements. We describe an efficient multi-object tracking approach that maintains multiple hypotheses over time regarding the association of targets and measurements. The data association problem is solved by a combinatorial optimization technique which finds the most likely association allowing track initiation, termination, merge, and split. Experimental results show that our method tracks through varying degrees of interactions among the targets with high success rate.

Recognizing complex high-level events requires an explicit model of the structure of the events. Our approach uses attribute grammar for representing such event, which formally specifies the syntax of the symbols and the conditions on the attributes. Events are recognized using an extension of the Earley parser that handles attributes and concurrent event threads. Various examples of recognizing specific events of interest and detecting abnormal events are demonstrated using real data.

Unsupervised methods for learning human activities have been largely based on clustering trajectories from a given scene. However, conventional clustering algorithms are not suitable for scenes that have many outlier trajectories. We describe a method for finding only salient groups of trajectories, using the probability of trajectories accidentally forming a group as the measure of significance of the group. The grouping algorithm finds groups that maximizes significance, while automatically determining the threshold for significance. We validate our approach on real data and analyze its performance using simulated data.

# MULTI-OBJECT TRACKING, EVENT MODELING, AND ACTIVITY DISCOVERY IN VIDEO SEQUENCES

by

Seong-Wook Joo

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

## Acknowledgments

I would like to express my deep gratitude for my advisor, Professor Rama Chellappa for giving me an opportunity to work on such a fascinating subject as computer vision, providing support both academically and financially. It was truly an honor to have studied under his guidance. I also want to thank Dr. Qinfen Zheng who co-advised me during the early years of my doctoral studies and providing an important starting point for my dissertation. I appreciate the constructive suggestions and helpful comments given by my committee members, Professor Larry Davis, Professor David Jacobs, and Professor Amitabh Varshney. I also thank Professor Gang Qu for agreeing to serve as the Dean's Representative. I am grateful for Professor Samir Khuller and Minkyoung Cho for their fruitful discussions on the combinatorial algorithms used in this dissertation. Special thanks goes to Dr. Kyungnam Kim for providing me data used in one of my experiments. Interactions with colleagues at the Center for Automation Research were helpful in many ways for my studies. Discussions with my office mates Naresh Cuntoor, Pavan Turaga, and Feng Guo were especially enjoyable and insightful. Last but not least, I want to thank my father who inspired me to study computer science and my mother's unceasing support through the years of my studies. I cannot thank enough my wife Chag-Hee Lee for always being by my side and encouraging me. Most of all, I thank God who has guided every step of my life.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Surveillance cameras are becoming ubiquitous, however human resources to supervise and process the abundant video data are expensive and limited. Automatic video surveillance address this problem by recognizing activities of interest without human intervention. In this dissertation we discuss several key components of automatic video surveillance.

## 1.1 Motivation

Detecting moving objects is one of the most fundamental tasks in video surveillance. A common approach for detecting moving objects is background subtraction where a statistical model of the background pixels is compared against each image frame in the video. However, this approach typically requires some training period to construct the background model. Although techniques exist for adaptively updating the background model, they are generally not robust to rapid changes in the background. Also, background subtraction does not effectively handle temporally transient spike noise caused by the noise in the image or by abrupt but small displacement of objects in the scene. This is because in this approach, temporal information is not fully utilized.

Persistently tracking the moving objects is also of great interest in video surveillance. A great amount of work in the computer vision community has been focused on tracking a single target. On the contrary, relatively less attention was given to tracking multiple objects simultaneously. In multi-object tracking, the problem of determining which observation arises from which target (which is commonly referred to as *data association*) must be addressed. Moreover, in visual tracking where the targets often occlude each other, targets may appear to merge or split. Many approaches for this problem has been ad-hoc, often extending the single-object tracking methods to multi-object case.

Recognition of short-term events such as human gesture has been studied extensively using statistical pattern recognition techniques, such as Hidden Markov Models. However, it is difficult to apply the same approach to the problem of recognizing events that span extended periods of time with complex structure, often involving multiple interacting objects. By structure we mean the spatial, temporal or other relations among the sub-components of the data. Such complex events can be better recognized by explicitly modeling the structure of an event. It is known from theoretical studies that learning a non-trivial structural model from examples is very difficult [13]. Therefore, the state-of-the-art approaches for these problems are based on human designed models that reflect domain knowledge. The remaining issue then is to devise an appropriate representation of the model such that it is flexible enough to be able to describe complex structures yet simple and intuitive for the human designer. Existing representations for short-term events such as finite state automata or Hidden Markov Models may not have adequate expressive power. On the other hand, generic logic-based representations may not be well suited for expressing temporal structures, though it is possible.

While it is possible to classify relatively simple patterns of activities using supervised training methods, it is desirable to detect and classify frequent patterns of activities in an *unsupervised* manner. Unsupervised clustering approaches have been successfully applied to activity analysis in some domains. But a difficulty lies in the fact that in a general surveillance scene, there may exist many patterns that does not correspond to any frequent activity. For such scenarios, conventional clustering methods may produce non-meaningful clusters and the outliers may bias the characteristics of the clusters that represent true activities. Thus it is important to find only the clusters that are significant using some reasonable measure of significance.

## 1.2 Approach

We employ local temporal variance as a measure for moving object detection. The pixel-wise temporal variance is efficiently computed through a recursive filter with exponential window. The exponentially decreasing nature of the window allows for sensitive detection of consistent change in the pixel values while the relatively large window size suppresses transient noise. Since we estimate the variance online from a temporal window covering the past data, a trail of non-zero variance is measured behind the moving object. To reduce the effect of this artifact, a simple background model is used to suppress the variance in regions that is part of the background with high confidence.

Our approach for multi-object tracking is based on Multiple Hypothesis Tracking (MHT) by Reid [51], which is one of the classical methods for multi-target tracking in the "small target" (including radar or sonar data) tracking community. In small target tracking, the association between targets and measurements are assumed to be one-to-one, meaning at most one measurement is associated with a target and vice versa. We relax this assumption to allow association of a single target with multiple measurements and multiple targets with a single measurement, which is more suitable for visual tracking of object of relatively large size. Following the approach of MHT, a number of the most likely associations are hypothesized and propagated over time. However, the number of possible data association is exponential in the number of targets and exhaustive enumeration is often impractical. We propose an efficient method of searching for the $k$-best hypothesis association given the targets and the object detection information. Further, we deal with the nonlinearity of the association cost for blob targets using a linear approximation method in the hypothesis generation step.

For recognizing high-level events, we propose to use syntactic models. In syntactic models, patterns are expressed by a sequence of symbols which corresponds well with the sequential nature of events. Specifically, we choose to use attribute grammars (context-free grammars with attributes associated with the symbols) for representing the events. In contrast to conventional grammars, attribute grammars are capable of describing features that are not easily represented by syntactic patterns that uses finite symbols. Using an online attribute context-free language parser as the main framework, we describe a complete real-time event recognition system. The motion trajectories generated by tracking moving objects are used as the raw data for recognizing events. Primitive events and their attributes are extracted from the trajectories, which are incrementally parsed according to the attribute grammar to recognize high-level events. Our approach handles multiple concurrent events involving multiple entities by associating object identification labels with event threads. The attribute grammar may be used to model specific events of interest or general events that typically occur in a scene. In the latter case, the modeled events are recognized and labeled with the event category, while events that does not follow the grammar are labeled as abnormal.

We present a method to automatically discover salient activities using a very general model of trajectories. Our approach is motivated by prior work in perceptual organization for detecting salient groups of image features. Specifically, the probability of trajectories accidentally forming a group is used as the measure of significance of the group. The probability is derived from an assumed general model of random background trajectories and a definition of the spatial region determined by a group of trajectories. The grouping algorithm finds maximally non-accidental groups and automatically determines the threshold for significance.

## 1.3 Original Contributions

- A novel measure for detecting moving object is proposed, which is based on a combination of temporal variance-based motion detection and background modeling. In addition, a thorough performance analysis is given for both our approach and a common background modeling (Kernel density estimation) approach.

- A fully automatic real-time multi-object tracker is demonstrated that is capable of handling a comprehensive types of object behaviors and interactions. In particular, the merging and splitting of objects handled by a theoretically sound bipartite graph edge covering model. Through the use of a combinatorial optimization technique (minimum weighted graph edge covering), this method efficiently generates the most probable multiple hypotheses regarding the joint association between targets and measurements.

- We have introduced attribute grammar for high-level event representation, which has not been done before to our knowledge. The Earley parsing algorithm is extended to handle multiple concurrent threads of events and multiple objects in a single event. In addition, the conventional attribute grammar is extended to include uncertainty in the conditions on the attributes, which is used to assign a degree of confidence for an event. A new method for abnormal event detection is proposed using a syntactic model of normal events for constrained surveillance scenes.

- Applying the principle of non-accidentalness to trajectory features for activity recognition has not been done previously to our knowledge. The notion of regions for evaluating the accidental probability is defined in a principled manner, which results in a theoretically plausible means of determining the threshold for significance. We also propose an agglomerative clustering algorithm that directly optimizes the non-accidentalness criterion.

## 1.4 Organization

The next two chapters (2 and 3) correspond to two lower levels of processing temporal visual information, namely detection and tracking. Chapter 2 describes our method of detecting moving object based on temporal variance, and in Chapter 3 the multiple hypothesis approach for tracking multiple objects are discussed. The following two chapters are related to higher level interpretations of the visual information. Chapter 4 deals with the use of attribute grammars for recognizing high-level events and detecting abnormal events. Chapter 5 discusses our approach for discovering patterns of activities using the measure of accidental probability for groups of trajectories. Chapter 6 summarizes the conclusions of this dissertation and discusses the future work. Relevant background material and survey of related literature are included in the beginning each chapter.

# Chapter 2

# Moving Target Detection Based on Temporal Variance

## 2.1 Introduction

Moving target detection and localization is one of the most fundamental tasks in visual surveillance. Assuming that the video is taken from a stationary camera, moving target detection algorithms are mostly based on either some kind of image differencing or background modeling.

A simple moving target detection can be achieved by subtracting and thresholding two successive frames from a sequence. However, frame differencing alone is not robust enough for most applications. Jain and Nagel [32] used an accumulation of the thresholded difference images with respect to a reference frame. A Moving object detection method by Paragios and Tziritas [47] used the consecutive image difference in a Markov Random Field formulation. Rosin [52] compared several different criteria for choosing the threshold for change detection. A comprehensive survey on general change detection in images is given by Radke [50].

Background modeling methods construct a model of the stationary background and then each pixel of a video frame is classified as a part of a moving object, if the pixel is not likely to be from the background. Stauffer and Grimson [54] modeled each pixel value as a mixture of $K$ Gaussians and used an approximate on-line algorithm to update the model. In [19], Elgammal et al. used kernel density estimation to model each pixel of the background and applied a threshold on the probability to obtain the foreground. False detection was suppressed by considering the local spatial information in the model. To achieve further robustness, results from two separate models, a short-term model with selective update and a long-term model with blind update are combined to produce the final result.

Background subtraction methods typically require some training period to construct the background model and are generally not robust to rapid changes in the background. We suggest a novel approach for moving object detection using the local temporal variance as the main criteria. In addition, a simple background model is used to enhance the detection. The algorithm not only requires minimal computation and memory but also quickly adapts to a changing background, eliminating the need for the training period. Our method is also robust to image degradation that is previously unobserved, and thus unmodeled.

To the best of our knowledge, temporal variance has not been used directly as the main measure for target detection. In [28], the local temporal mean of difference images and adaptive background modeling was combined. The Dynamic Retina [49] used the local temporal mean as an intensity normalization factor to measure the intensity offset caused by camera jitter as well as object motion. Temporal variance of the spatial average of consecutive frame difference was used in [2] to determine when to update the threshold for background subtraction.

In this chapter, we describe the use local temporal variance as a measure of moving object detection. The following section describes the temporal variance measure in detail. Next, we show how a simple background model is utilized to remove the artifact coming from the use of a local window. Finally, we describe our performance metric based on the bounding boxes and centroids of the objects, followed by the evaluation results.

## 2.2 The temporal variance measure

Since an object in a video generally occupies a small spatial area and almost always moves with a limited speed, the changes in the pixel values caused by the moving object are localized in the spatio-temporal domain. It is the pixel-wise temporal locality that we wish to exploit. Although consecutive two-frame differencing is highly adaptive to changes in the scene, it is also very sensitive to noise. Therefore, we use information from multiple frames for robustness. One natural way to measure the amount of change in some time interval is the variance. Further, we apply an exponentially decaying weight (window) to the pixel values to save computation and memory. This is easily computed by the recursive filter

$$\begin{aligned} m(t) &= \alpha m(t-1) + (1-\alpha)x(t) \\ m_2(t) &= \alpha m_2(t-1) + (1-\alpha)\{x(t)\}^2 \\ v(t) &= m_2(t) - \{m(t)\}^2 \qquad (2.1) \end{aligned}$$

where $x(t)$ is the value of the pixel at time (frame) $t$, $\alpha$ is the decay rate, $v(t)$ is the variance, and $t = 1, 2, \ldots$. In order to avoid floating point operations, the equations can be rewritten as

$$\begin{aligned} m(t) &= ((N-1)m(t-1) + x(t))/N \\ m_2(t) &= ((N-1)m_2(t-1) + \{x(t)\}^2)/N \\ v(t) &= m_2(t) - \{m(t)\}^2 \end{aligned} \tag{2.2}$$

where $N = 1/(1-\alpha)$ is a measure of the size of the exponential window. The initial value $m(1)$ and $m_2(1)$ are respectively set as $x(1)$ and $x(1)^2$. Moving target detection can be achieved by thresholding this variance. Note that the variance measure is reduced to the consecutive frame differencing by using a uniform window of length 2:

$$\begin{aligned} m(t) &= (x(t-1) + x(t))/2 \\ m_2(t) &= (\{x(t-1)\}^2 + \{x(t)\}^2)/2 \\ v(t) &= m_2(t) - \{m(t)\}^2 \\ &= \{x(t-1) - x(t)\}^2/2 \end{aligned} \tag{2.3}$$

This approach uses the variance *directly* as the measure whereas background subtraction methods model the *background* with the variance information (or probability distribution in general) and use the pixel values as the measure. This strategy is similar in spirit to the Resonant Retina [26] where the variance arising from camera jitter is actively used to collect useful information. The increase in variance of a pixel is caused by an object with a different intensity entering the pixel. Furthermore, since an object usually has texture or is non-rigid, as the object moves through, the pixel tends to have even larger variance. The advantage of using an exponential window, apart from its simplicity is that it is sensitive to the initial entry of the object and suppresses noise to some degree. Large coherent change in the signal is quickly amplified, while temporary noise is smoothed out. Assuming a simplified case of perfectly still background and a moving object of uniform intensity i.e., a square pulse temporal signal, the variance is expressed as

$$v(t) = \begin{cases} 0, & t \leq 0 \\ A^2(1-\alpha^t)\alpha^t, & 0 < t \leq T \\ A^2(1-\alpha^T)\alpha^{t-T}\left\{1-(1-\alpha^T)\alpha^{t-T}\right\}, & t > T \end{cases} \tag{2.4}$$

where the pulse starts right after time 0 and ends at time $T$, A is the intensity difference between the background and the object, and $\alpha$ is the window decay rate. The maximum value of $v(t)$ in the interval $0 < t \leq T$ is

$$\max(v(t)) = \begin{cases} A^2(1-\alpha^T)\alpha^T, & T < -\log 2/\log\alpha \\ (A/2)^2, & T \geq -\log 2/\log\alpha \end{cases} \tag{2.5}$$

meaning the peak in $v(t)$ is suppressed if the pulse duration $T$ is short. Note that the critical point $t = -log2/log\alpha$ coincides with the half-life of the exponential window. As an example, Figure 2.1(a) illustrates two different pulse durations and the square-root of the variance. The short pulse at $t = 10$ results in a smaller peak variance than the case with the longer duration starting from $t = 60$. Figure 2.1(b)

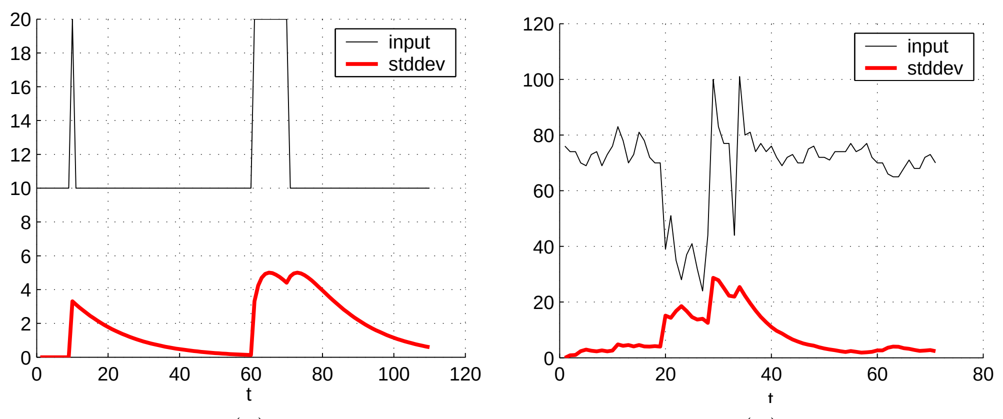


Figure 2.1: One-dimensional example of the variance measure

shows an example taken from a single pixel of a video with a person walking through it. Unfortunately, because the window has a long “tail”, the variance decreases too slowly over time. This results in the detection of the moving object with a long trail behind its trajectory (Figure 2.2(a)). Note that there is a tradeoff between the window size $N$ and the robustness of detection: a small $N$ will shorten the trail but cannot suppress the noise very well.

## 2.3 Removing the trail artifact

To eliminate the trail effect, a simple background subtraction is performed combining the result with the variance. The background is simply modeled by mean and variance, which are also obtained from recursive filtering.

$$\begin{aligned} m_{bg}(t) &= ((N_{bg}-1)m_{bg}(t-1)+x(t))/N_{bg} \\ m_{2bg}(t) &= ((N_{bg}-1)m_{2bg}(t-1)+\{x(t)\}^2)/N_{bg} \\ v_{bg}(t) &= m_{2bg}(t)-\{m_{bg}(t)\}^2 \end{aligned} \tag{2.6}$$

$m_{2bg}(t)$ denotes the background mean and $v_{bg}(t)$ is the background variance. The window size $N_{bg}$ should be large compared to $N$ so that the background model covers a longer period and is robust to noise. The mean and variance is updated only if the pixel is not part of the foreground. In order to avoid including any part of the foreground to the background model, an enlarged foreground region is obtained by thresholding the temporal variance from Equation (2.2) against a small value.

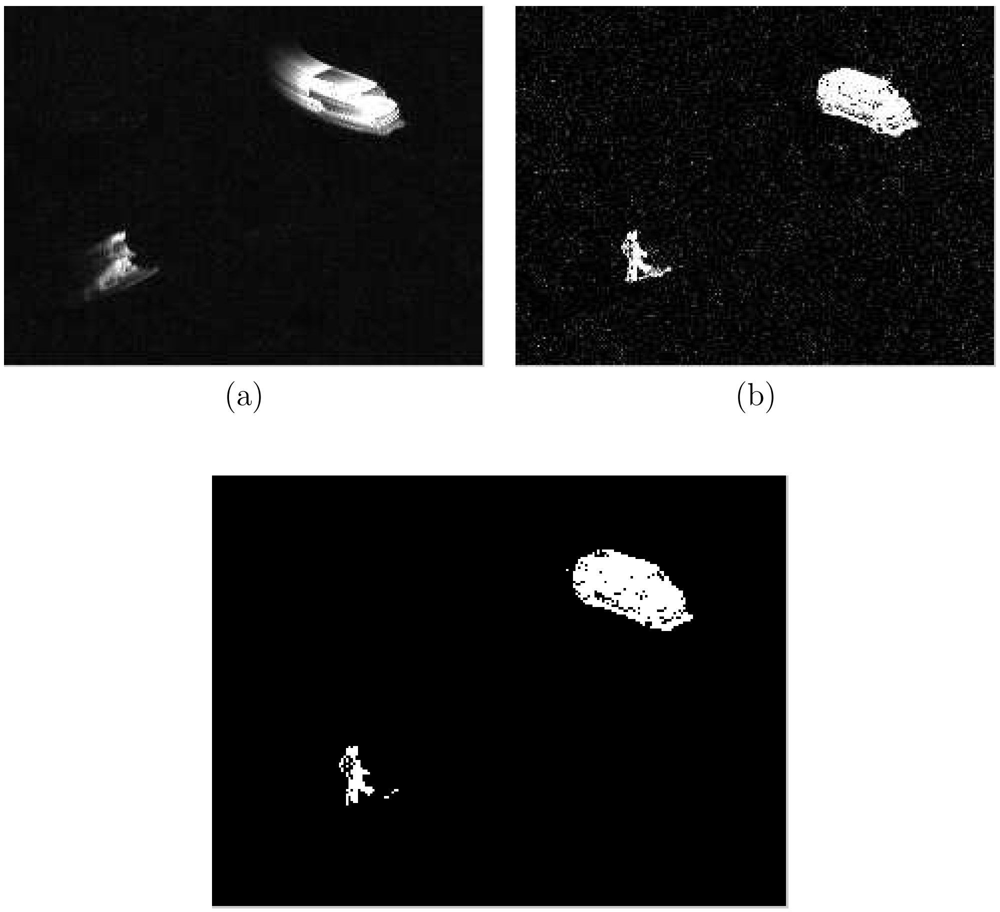
The background model is used to derive a confidence measure of the detected foreground. Assuming a Gaussian distribution for each background pixel, one measure of a given pixel being a foreground can be obtained by the error function

$$f_{conf}(t) = \frac{1}{\sigma}\sqrt{\frac{2}{\pi}}\int_0^d e^{-z^2/2\sigma^2}dz \tag{2.7}$$

where $d = |x(t) - m_{bg}(t)|$ and $\sigma = \sqrt{v_{bg}(t)}$. The error function is close to linear near zero. However, we need a function that is very small near the background intensity so that it strongly suppresses false detection. Experience shows that the following sigmoidal function is adequate for our purpose

$$f_{conf}(t) = \begin{cases} \frac{1}{2}\left(1 - \cos\left(\frac{\pi d}{r\sigma}\right)\right), & 0 \le d \le r\sigma \\ 1, & d > r\sigma \end{cases} \tag{2.8}$$

where $r$ is a scale factor that determines the range of the function having the transition from 0 to 1. Finally, the square-root of the variance is multiplied by the confidence value and thresholded to obtain the detection mask. (From here on, "variance" refers to the square-root of the variance.) The threshold is defined as



(a) (b)

(c)

Figure 2.2: Results of combining variance and background subtraction

some factor of the average background variance over all the pixels. Figure 2.2(a),(b), and (c) shows the variance, the confidence values, and the final detection mask, respectively.

By combining the variance measure and background subtraction, the algorithm retains the robustness of the variance measure while effectively removing false detection in the trail. If background subtraction happens to give a false positive, the variance measure is robust enough to suppress the error. False negatives from background subtraction are less critical since they tend to occur sparsely. Even when background subtraction fails overall, the result would at least be similar to using the variance alone.

## 2.4 Performance evaluation

### 2.4.1 Implementation

The temporal window size for our variance-based method was chosen as $N = 4$ and $N_{bg} = 8$, the range for the confidence function was fixed at $r = 9$. The sigmoidal function in Equation 2.8 is discretized into a lookup table for faster computation. The algorithm implemented in Matlab can process 1.5 to 3 frames per second on a 1.7 GHz Pentium 4 processor, depending on the frame size. Another implementation of a simpler version of the algorithm written in C++ achieves real-time rate of 30 frames per second.

At a higher level, the location of the moving object is determined by finding the bounding box of the object and choosing the centroid of the detection mask inside the box as the center of the object. Instead of looking for connected components, which is computationally expensive, we use 1-dimensional projections of the detection image mask to find the bounding boxes. First the mask is projected on the vertical vector i.e., count the number of pixels in each horizontal scan line, and are segmented into chunks that form horizontal strips in the 2-dimensional mask. Each strip is projected on the horizontal vector to get the left and right sides of the box. A final projection to the vertical vector gives a tight top and bottom bound. The boxes that are close together are merged and boxes with sparse pixels are removed. Although there exist configurations where this algorithm fails, most common situations are handled correctly.

### 2.4.2 Performance metric

Three types of performance metrics were used for sequences that have bounding boxes as ground truth.

- *Detection rate* is defined as the fraction of the ground truth boxes that are successfully detected by the algorithm. By successful detection we mean the centroid of the detected object is inside the ground truth box.
- *False positive rate* is the total number of detection centroids that does not land on any of the ground truth boxes, divided by the number of frames.
- *Multiple detection* refers to the average number of detection inside the ground truth box that is successfully detected. This indicates the amount of "broken up" detection of an object.

The ground truth bounding boxes were not quite correct—occasionally part of the object protruded out a few pixels from the box. In our experiments, the ground truth boxes were enlarged by 5 pixels in all directions to correct this error.

### 2.4.3 The dataset

The PETS 2001 datasets were used to evaluate the performance since the ground truth information for some of the sequences is publicly available. However, the ground truth is in the form of bounding boxes for the objects. To accurately

| Sequence | Num. of frames | Frame size | Object size | Travel distance (x,y) |
|---|---|---|---|---|
| 1 | 410 | 320 × 490 | 14.8 × 41.1 | (470, 61) |
| 2 | 139 | 392 × 322 | 71.7 × 50.6 | (355, 53) |
| 3 | 341 | 490 × 306 | 18.1 × 53.2 | (413, 94) |

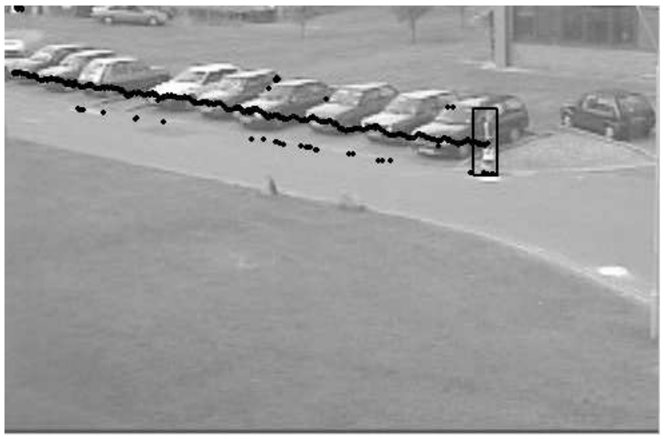
Table 2.1: Characteristics of the test sequences

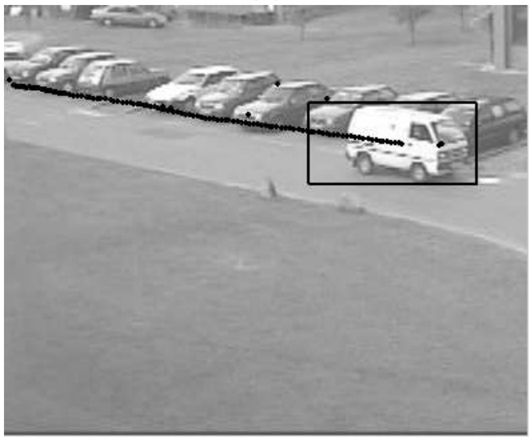
evaluate the performance of the detection algorithm, the sequence is cut and cropped so that only one moving object exists in each sequence, or two objects do not intersect each other. The reason for this is that when objects merge and split, the detected centroid does not always fall inside a ground truth bounding box. This error is caused by the bounding box algorithm, not from the main detection algorithm. Three sequences, 1, 2, and 3, respectively containing a walking person, a moving car, and another person were selected for testing (Figure 2.3). The characteristics of each sequence are summarized in Table 2.1. Units are in pixels. Object size is the average size of the bounding box over all frames. Travel distance refers to the distance of the box centers between starting and ending frames. All the images were converted to 256-level grayscale.

## 2.5 Experimental results

### 2.5.1 The effect of object speed

For the variance measure, the window size should be adjusted according to the speed of the object. A slow object requires a long window; otherwise, the interior of the object becomes hollow with small variance, especially when the object has


(a)


(b)

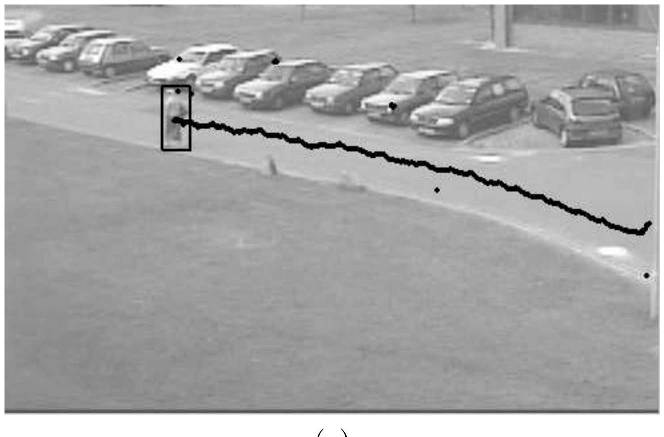
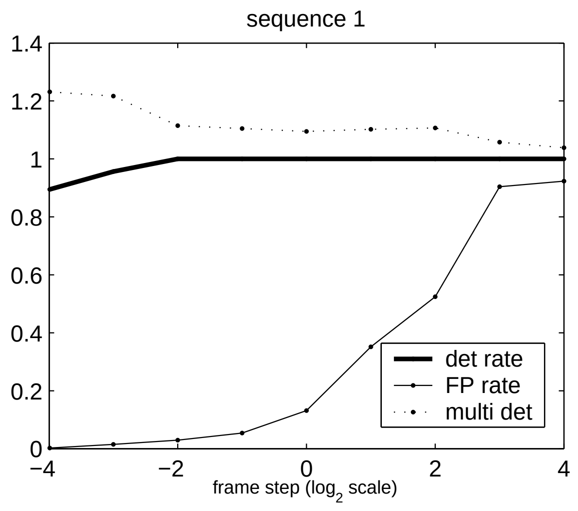

(c)

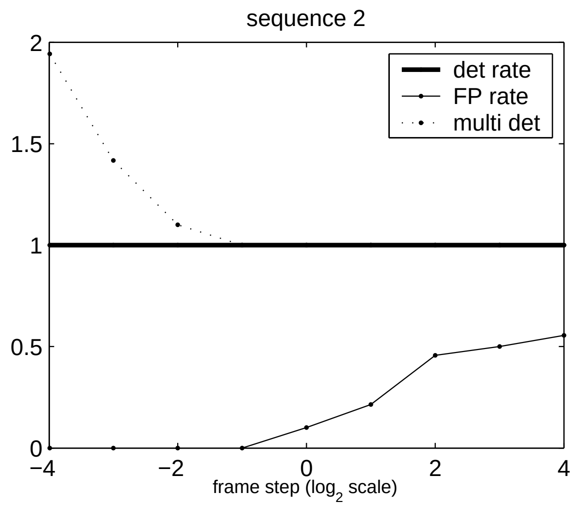
Figure 2.3: Sample frames from the test sequences and the detection results

little texture e.g., a solid colored object (see the vehicle in Figure 2.2(a)) or when an infrared sequence is used. Conversely, a shorter window is desirable for a fast object so that the length of the trail is minimized. For this experiment, the threshold factor was fixed at 5.

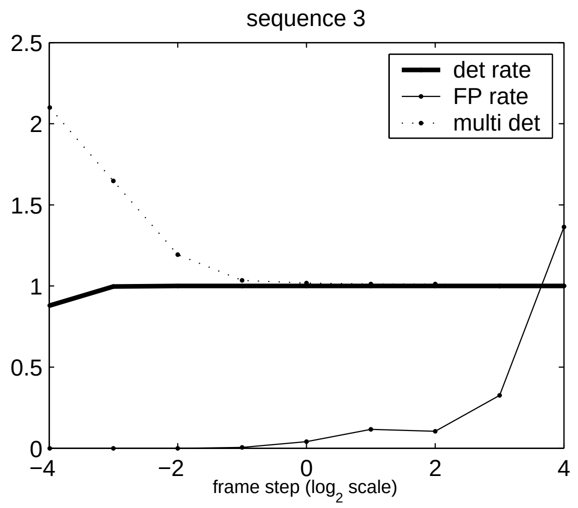
Various object speeds were simulated by skipping or interpolating the original frames. To speed up the object, larger frame steps were taken. Conversely to slow down the object, sub-frames were created by linear interpolation between consecutive frames, in which case the frame step was defined as a fraction. Figure 2.4 shows the detection performance with respect to the frame step. Faster object speed increased the false detection rate but did not degrade the detection rate. This is a consequence of using background subtraction: Even if the detection step based on variance had a long trail, background subtraction suppressed it, avoiding the bias that would have occurred towards the tail. At slower speeds, the detection rate slightly decreased and multiple detection increased. This is because the window size becomes relatively shorter compared to the object speed, and the detection is broken up into front and rear parts. This effect was amplified for larger objects (sequences 2 and 3). Multiple detection is considered relatively less critical in moving target detection applications and the current fixed window size appears to be sufficient. However, for some situation where the object speed is expected to be excessively fast or slow, the window size should be adjusted in advance for better performance.



(a)



(b)



(c)

Figure 2.4: The effect of object speed

### 2.5.2 Comparison with background subtraction

A background subtraction method using kernel density estimation (KDE) [19] was also implemented for comparison. For the training phase, a section of frames was used which contains only the background. The number of training frames used was set to be equal to the number of samples. Shadow detection was not implemented and the same bounding box algorithm was used.

The identical test sequences were given as input for both our variance-based and the background subtraction algorithm. For KDE, 40 samples are used (more samples did not significantly increase the performance for the test sequences) and only the "short term model" is used. For the variance-based algorithm, 5 consecutive frames consisting of only the background were added in front of the test sequence. The background-only frames were not required for our algorithm to work, but was included to remove the errors caused by inaccurate estimates of the parameters during the initial frames. This allows a fair evaluation of the long-term performance. The results are shown in Table 2.2. A set of different thresholds was used for each of the algorithms. For most thresholds, both methods achieved perfect detection. Since it is difficult to compare the two results objectively, the threshold that produced the lowest false positive was selected for each algorithm. It was observed that our algorithm tended to have lower false positives under similar multiple detections.

In order to compare the performance under a temporary image quality degradation, 20 frames from sequence 1 were selected with the last frame recompressed using JPEG compression quality rate of 50%. (The original frame was in JPEG

| Sequence— | Detection rate |  | False Positive rate |  | Multiple detection |  |
|---|---|---|---|---|---|---|
|  | Proposal | KDE | Proposal | KDE | Proposal | KDE |
| 1 | 1.0000 | 1.0000 | 0.0366 | 0.0512 | 1.1098 | 1.2415 |
| 2 | 1.0000 | 1.0000 | 0.0000 | 0.0863 | 1.0072 | 1.0000 |
| 3 | 1.0000 | 1.0000 | 0.0000 | 0.0528 | 1.0323 | 1.0469 |

Table 2.2: Performance measurements for the lowest possible false positive rate obtained

format with a quite low compression ratio at 3.8%.) The resulting frame has a root mean squared error of 2.8966 and relative root mean squared error (with respect to the spatial variance) of 0.0880. Pixel-wise ground truth mask was manually generated for the last frame. The two algorithms were tested with varying thresholds. The ROC curve for the detection on the last frame is shown in Figure 2.5. Our proposed algorithm consistently gave less false positive rate under equal true positive rate. Since background modeling approaches rely on the history of the pixel values, it is likely to give false positives for the values that exceed the modeled noise range. On the contrary, our variance measure does not rely on history and is effective in handling moderate spike noise as described in Equation (2.5).

Next, performance under rapidly changing illumination was compared. The test frames are from another sequence of PETS 2001 dataset for which the ground truth is not available. We used manual selection and interpolation to generate a ground truth bounding box data. The number of frames is 600 with frame size 454 $\times$ 360. Figure 2.6 shows a plot of the average intensity for each frame and two frames with the minimum and maximum mean intensity. For the KDE method, in addition to the "short-term" model, the "long-term" model that covers the entire sequence

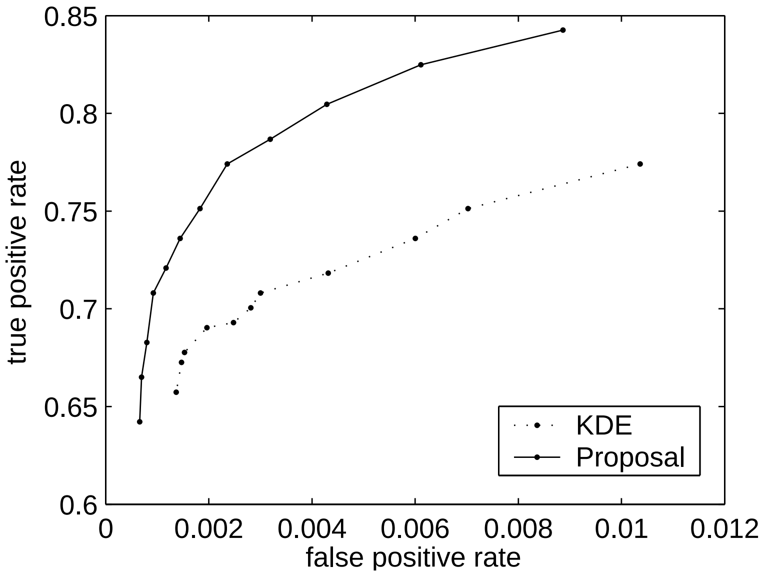
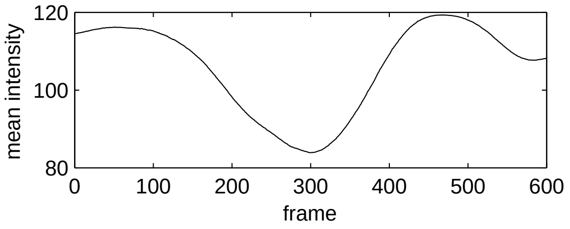


Figure 2.5: ROC curve for pixels detected in a noisy frame

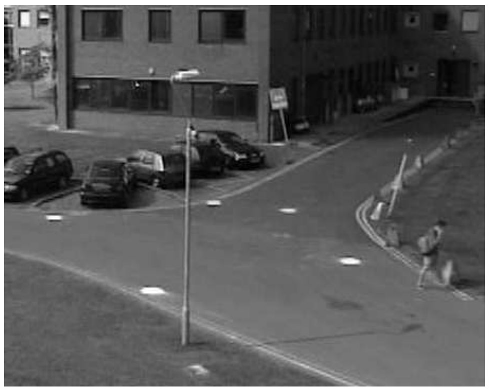
was used to remove persistent false positives. Figure 2.7 plots the false positive rate vs. the detection rate, as defined in Section 2.4.2. For our proposed method, only the threshold was varied whereas for KDE, various thresholds and sample sizes were used. The plot shows that our algorithm gives higher detection rates with low false positive rates. The KDE approach needs to keep a small number of samples to quickly adapt to the changing illumination but on the other hand requires large enough samples to be accurate. However, our approach does not require a large number of samples, therefore is more adaptive.

## 2.6 Summary

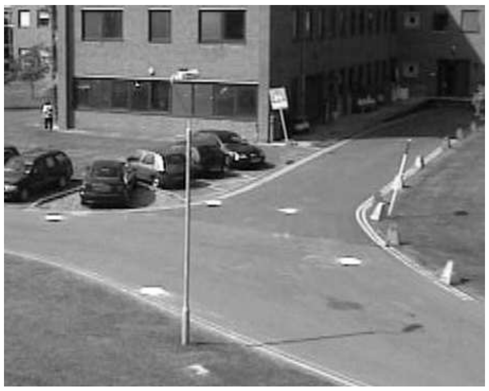
We have proposed a temporal variance-based approach for moving target detection. The variance measure is robust to noise and gives a low false detection rate. The trail artifact is greatly reduced by a simple background modeling method. Ex-



(a)



(b)



(c)

Figure 2.6: The test sequence with rapid illumination changes

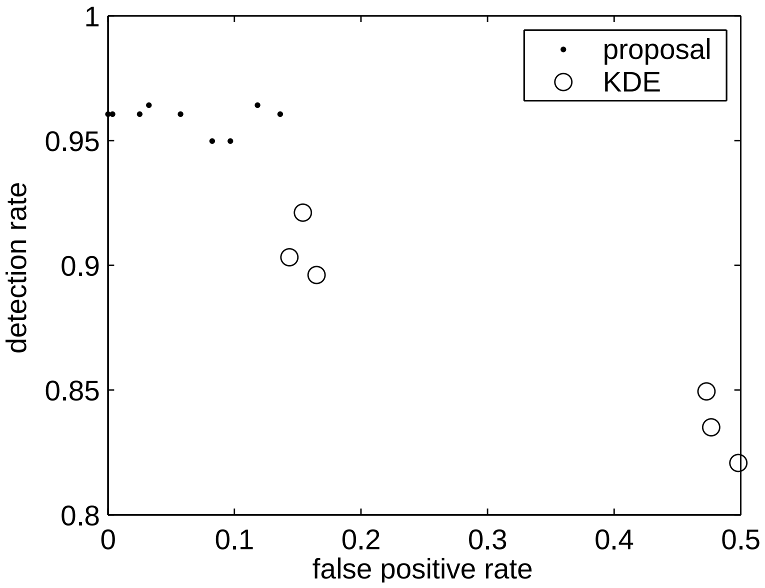


Figure 2.7: Performance under rapidly changing illumination

periments show that a fixed window size can be used for a reasonable range of object speeds. Under normal conditions, our method performed as well as the KDE-based background subtraction approach. In addition, our approach performs better than KDE under temporary compression noise and rapidly changing illumination. The algorithm is simple and fast and is highly adaptive to changing conditions, making it suitable for a wide range of real-time surveillance applications.

# Chapter 3

# A Multiple Hypothesis Approach for Multi-Object Visual Tracking

## 3.1 Introduction

Visual tracking of multiple objects involves not only localizing each object in the scene, but also the problem of data association. Since the observation originating from one target object may interfere with observations from other targets, all targets and observations should be jointly associated in a coherent manner. Often in computer vision applications, object detection information may be given, which greatly simplifies the object localization problem. In this chapter, we use moving object detection to localize objects and focus mainly on the data association problem. Further, we restrict ourselves to tracking relatively small objects that have little useful appearance information.

Data association methods for multiple target tracking have been studied for many years in the small target (e.g., radar or sonar) tracking community. Since targets are assumed to be small in these methods, it is assumed that a target and its measurement is associated in a one-to-one manner i.e., a target is associated with at most one measurement and vice versa. Well known data association approaches used in this context include Joint Probabilistic Data Association Filter (JPDAF) [4] and the Multiple Hypothesis Tracking (MHT) by Reid [51]. The small target tracking problem is very similar to visual object tracking in static cameras in that the observation (measurement) data can be given by the detection algorithm rather than features being searched over some large data (image). However, unlike in the small target tracking problem, objects detected in images may split or merge. We propose a tracking algorithm based on MHT, relaxing the association constraint to allow association of a single target with multiple measurements and multiple targets with a single measurement. We refer to this type of association as *multiple association*. In addition, an efficient $k$-best hypothesis generation algorithm for multiple association is proposed. Further, we deal with the nonlinearity of the association cost for blob targets using a linear approximation method in the hypothesis generation step.

A survey of related work is given in Section 3.2, followed by some discussion on the background work related to this chapter in Section 3.3. The detection method and the tracking model for individual targets are respectively described in Section 3.4 and Section 3.5. Discussions on multiple hypothesis generation for the multiple associations are presented in Section 3.6. In Section 3.7 the implementation issues are addressed and the experimental results are given in Section 3.8.

## 3.2 Related work

We briefly review some related prior work that addressed the data association problem in multi-object tracking.

One type of approach is to consider a finite sequence of measurement sets and create a graph that represents the possible associations from one time step to another, then extract the optimal set of trajectories from the graph. The method used in Medioni et al. [43] used a graph where each edge was assigned a weight based on pixel correlation and spatial distance. The blobs that were split due to noise were merged by clustering tracks in the graph. The trajectory for each object was extracted from the graph by searching for an optimal path where the cost of a path was a function of the similarity measure between the nodes and the length of the path. A similar approach was adopted in [8] where the graph representation implicitly retained multiple hypotheses regarding the possible trajectories. Dynamic occlusion was handled by deferring the association for multiple time steps. The best hypothesis was updated over time from the dynamically created graph. The method by Han et al. [25] may also be viewed as a graph-based approach. A graph that maintains multiple hypotheses was maintained with their corresponding likelihood. The association hypothesis included object split and merge in addition to track initiation, termination and missing measurement. The likelihood of a joint association was defined by individual likelihoods of the trajectories and a global likelihood of the detection image. These approaches often rely on ad-hoc techniques for extracting the trajectories.

Other approaches associate the existing targets with the measurements at each time step. These approaches typically assumes one-to-one association between targets and measurement and thus require techniques to segment the merged measurements caused by dynamic occlusion. Javed and Shah [33] used a suboptimal greedy algorithm for target association. The occluded objects was predicted by the overlap of the target regions, in which case the measurements for each target was estimated from the nearest sub-region of the merged measurement. Chen et al. [7] described a method that uses *bipartite graph matching* for target association, which finds the optimal joint association under the one-to-one assumption. Merged blobs were segmented by finding the modes of the projected intensity distribution. In [21] the problem of merged and split measurements was handled by creating virtual measurements so that each target may be associated with exactly one (virtual) measurement. This is done by respectively splitting and merging the conflicting measurements and extra measurements within the validation gates (feasible regions) of targets. The most likely joint association among all association is selected in each time step.

Khan et al. [37] used a Markov chain Monte Carlo (MCMC) technique to deal with the association problem. Multiple hypotheses were generated regarding joint associations that allow split and merge of targets through MCMC sampling in the joint association space. The joint distribution of the target locations was represented by a set of hybrid samples each corresponding to a different hypothesis. MCMC sampling was also used in [63]. In contrast to the approximate nature of MCMC methods, our approach is based on an efficient combinatorial optimization algorithm that directly finds the best hypotheses.

It is noted that there is a large body of work on particle filtering methods for jointly estimating the locations of targets as well as the associations [29, 6, 40], which is beyond the scope of this chapter.

## 3.3 Background

### 3.3.1 Classical small target tracking

We summarize the two classical methods for small target tracking—JPDAF and MHT. The basic approach described here should serve as the background for this chapter. Both approaches consider a set of feasible hypotheses regarding joint associations between targets and their measurements, along with their joint probabilities. The target states are assumed to follow Gaussian distributions and have linear dynamics thus the Kalman filter is used to track the targets.

JPDAF considers joint association hypotheses with the latest (current) measurements only. Possible associations in the hypothesis are: target-measurement association, undetected target, false measurement. Targets assumed to be initialized by some other means and a fixed number of targets is assumed. The probability of a joint association event is given by

$$P(\theta|Z^{1:t}) = \frac{1}{c} p(Z^t|\theta, Z^{1:t-1}) P(\theta) \tag{3.1}$$

where $\theta$ represents a joint association hypothesis, $Z$ is the measurements, $t$ denotes the current time step, and $c$ is a normalization constant. $p(Z^t|\theta, Z^{1:t-1})$ is the joint measurement likelihood given by a product of the individual (Gaussian) likelihoods, assuming conditional independence of the measurements given the association. The prior probability $P(\theta)$ on the association is based on the number of false measurements and true detections. To combine the multiple hypotheses in the current time step, the association probabilities are *marginalized* over all feasible associations. The probability $P(\theta_{ji}|Z^t)$ of associating measurement $j$ with target $i$ is defined as

$$\beta_{ji} = \sum_{\theta} P(\theta|Z^{1:t})\hat{\omega}_{ji}(\theta) \tag{3.2}$$

where $\hat{\omega}_{ji}$ is 1 if $j$ is associated with $i$ and 0 otherwise. Finally, each target $i$ is updated with the "combined measurement" using $\beta_{ji}$ as the summing weight.

In contrast to JPDAF, Reid's MHT approach considers cumulative history of all feasible association hypotheses up to the current time, resulting in a tree of hypotheses. Since the hypotheses are maintained over the time steps, this approach is capable of initiating new targets. Thus the association of a measurement to a *new target* is considered in addition to the three type of associations used in JPDAF. Let $\Theta_l^{1:t}$ denote the $l$th cumulative hypothesis

$$\Theta_l^{1:t} = \{\Theta_{s(l)}^{1:t-1}, \theta_l^t\} \tag{3.3}$$

where $s(l)$ denotes the index of the previous (parent) hypothesis which the current hypothesis $\theta_l^t$ is based on. The probability of the cumulative hypothesis is given by

$$P(\Theta_l^{1:t}|Z^{1:t}) = \frac{1}{c} p(Z^t|\theta_l^t, \Theta_{s(l)}^{1:t-1}, Z^{1:t-1}) P(\theta_l^t|\Theta_{s(l)}^{1:t-1}, Z^{1:t-1}) P(\Theta_{s(l)}^{1:t-1}|Z^{1:t-1}) \tag{3.4}$$

where the prior probability $P(\theta_l^t|\Theta_{s(l)}^{1:t-1}, Z^{1:t-1})$ on the current hypothesis is evaluated as in JPDAF. Since multiple hypotheses are created in the current time based on multiple parent hypotheses, the cumulative hypotheses form an exponentially growing tree. Thus the number of hypotheses need to be reduced by ignoring the less likely ones. Common techniques [4] include "$N$-scan-back" pruning where all the subtrees except the most probable one are removed from the node $N$ time steps back, and various thresholding methods.

### 3.3.2 An efficient method of generating association hypotheses

The previously described methods enumerate all feasible associations given the targets and the measurements. This makes the computational complexity of the algorithms exponential. Recently, Cox and Hingorani [9] introduced an efficient method of generating only the $k$-best association hypotheses, which was based on Murty's ranked assignment algorithm [46]. This method was used to track multiple point features using the MHT framework. We briefly describe the formulation of the association problem and the algorithms used in their work.

For tracking point targets, the problem of generating the optimal one-to-one association hypothesis can be mapped to an *assignment* problem as defined below.

**Definition 3.1** *Given an edge weighted graph $G = (V, E)$, A* minimum weight perfect matching *is a subset of edges $E' \subset E$ such that every vertex in $V$ is incident on exactly one edge in $E'$ and the sum of the weights in $E'$ is minimum.*

Polynomial time algorithms exist for minimum weight perfect matching [53]. The problem of finding the minimum weight perfect matching in a bipartite graph is called the *assignment* problem. The target-measurement association can be expressed by a bipartite graph with one set of vertices representing the target and the other set representing the measurements. The weights for the edges in the bipartite graph are assigned as the negative log-likelihoods of the measurement given a target, the sum of which corresponds to the joint likelihood (product of likelihoods).

Murty's algorithm finds $k$ assignments that have the lowest weights among all the possible assignments in polynomial time. The algorithm works by repeatedly finding the next best solution for the assignment problem excluding the solutions already found. The exclusion of solutions are achieved by solving a set of modified assignment problems (we call these the sub-problems) such that each sub-problem excludes some edges in the existing solutions and no solutions for the sub-problems are identical. Each sub-problem effectively partitions the set of all possible assignments (we call this the solution space) where each partition does not contain any of the existing solutions.

## 3.4 Object detection

The silhouettes of the moving objects (blobs) are detected using background subtraction techniques [36, 38]. Connected component analysis is applied to the binary map to obtain the bounding boxes of each measurement represented as

$$\mathbf{z} = (c_x, c_y, d_x, d_y)^{\mathrm{T}} \tag{3.5}$$

where $c_x$ and $c_y$ are the target center, $d_x$ and $d_y$ denote the size of the bounding box. A simple size filter that removes small detections is then applied. We do not attempt to segment the detected objects (for example as in [44]), which is very difficult for small objects or objects with similar appearances.

## 3.5 Tracking model

The standard Kalman filter [4] is used to estimate the individual target states. The process and measurement model are expressed as

$$\mathbf{x}^t = \mathbf{A}\mathbf{x}^{t-1} + \mathbf{w} \tag{3.6}$$

$$\mathbf{z}^t = \mathbf{H}\mathbf{x}^t + \mathbf{v} \tag{3.7}$$

where $\mathbf{x}^t$ and $\mathbf{z}^t$ denote the state and the measurement at time $t$, $\mathbf{A}$ and $\mathbf{H}$ are the process and measurement matrices, and $\mathbf{w}$ and $\mathbf{v}$ denote the process and measurement noise, respectively. The state $\mathbf{x}^t$ is defined as

$$\mathbf{x}^t = (c_x, c_y, d_x, d_y, v_x, v_y)^{\mathrm{T}} \tag{3.8}$$

where $c_x$ and $c_y$ are the target center, $d_x$ and $d_y$ denote the size of the target bounding box, and $v_x$ and $v_y$ represent the velocity of the target (the time index $t$ has been omitted). The following process and measurement matrices are used

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{3.9}$$

where $\mathbf{A}$ represents a constant velocity model with constant object size. In this work, we assume that the targets have little distinguishable features in appearance. Therefore we do not model the appearance of the targets although it can be included in the target model.

## 3.6 Hypotheses generation

We propose a multiple hypothesis approach that handles objects (1) entering and (2) exiting the view, (3) merging and (4) splitting, as well as (5) detection of an object as split parts due to the limitation in background subtraction.

Let $n_T$ and $n_M$ denote the number of targets and measurements, respectively. An association hypothesis can be represented by a bipartite graph as shown in Figure 3.1. The nodes $t_i$ and $m_j$ respectively correspond to targets and measurements. The node $t_N$ is a dummy node that covers all measurements not associated with any existing targets and similarly the dummy node $m_D$ covers any unassociated targets.

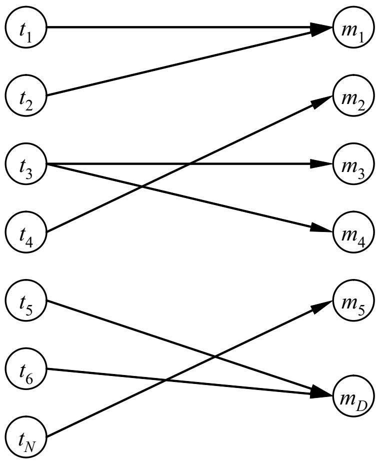


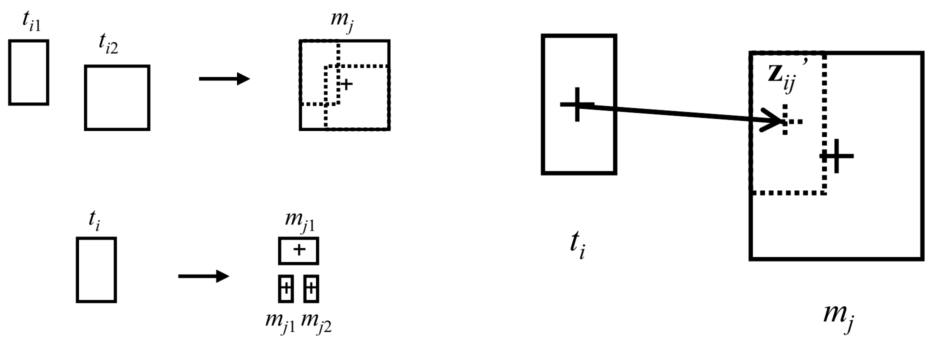
Figure 3.1: An example of a target-measurement association hypothesis represented by a bipartite graph. $t_N$ and $m_D$ are dummy nodes that covers unassociated measurements and targets, respectively

The only constraint in the association is that each node should have at least one incident edge. If all $m_j$'s are associated or all $t_i$'s are associated, $t_N$ is connected to $m_D$. An exhaustive enumeration of all such possible hypotheses would result in $2^{n_T n_M}$ hypotheses, which may become infeasible even for moderately large values of $n_T$ and $n_M$.

The problem of finding an optimal multiple association in a bipartite graph can be posed as a problem of finding the *minimum weight edge cover* (MWEC) in a bipartite graph.

**Definition 3.2** *Given an edge weighted graph $G = (V, E)$, A* minimum weight edge cover *is a subset of edges $E' \subset E$ such that every vertex in $V$ is incident on at least one edge in $E'$ and the sum of the weights in $E'$ is minimum.*

Fortunately, an algorithm exists which reduces the MWEC problem to a minimum weighted graph matching problem in polynomial time [53]. Details on this algorithm are given in Chapter A. Hereafter, the term *association cost* will be used in place



(a) (b)

Figure 3.2: Association of area targets and blob measurements. (a) nonlinearity in merging and splitting into parts. (b) linear approximation

of edge weight. Murty's algorithm can also be adapted to the ranked MWEC problem. However, in tracking bounding boxes of targets it is not reasonable to directly map the measurement log-likelihood to the edge cost (weight), since the problem of multiple association of targets and measurement blobs is not entirely linear in the sense that the sum of individual association costs may not accurately represent the joint association cost. For example, when an object is detected as split parts, the cost of associating a target with all the split parts should be less than the sum of individual association costs. Similarly, for the case of two targets associated with a single measurement (merge), the cost should be less than the individual sum if the appearance—or the size—of the merged blob is significantly different from either of the targets. Figure 3.2(a) illustrates these two situations. We propose a two-pass approach that approximates the optimal $k$-best solution.

## 3.6.1 Linear cost hypothesis generation

In the first pass, a linear approximation to the multiple association problem is used to generate the $k$-best solution. We treat the predicted targets and the measurements as *point* targets by considering only the center coordinates $(c_x, c_y)^{\mathrm{T}}$. This assumption significantly reduces the nonlinearity in the cost function. However, for the merged measurements which often results in a blob larger than either of the individual targets, it is unreasonable to assume that the center of the measurement is the location of the "point". In order to approximate the true distance to a potentially merged measurement, we assume that all measurements are merged measurements and the distance is determined by the following procedure:

- Let $\tilde{\mathbf{x}}'_i$ be the center coordinates of the $i$th predicted target state $\tilde{\mathbf{x}}_i$ and denote $\mathbf{z}'_{ij}$ as the approximated measurement center within the measurement $\mathbf{z}_j$ for association with $\tilde{\mathbf{x}}_i$.
- For each pair $\tilde{\mathbf{x}}_i$, $\mathbf{z}_j$, determine $\mathbf{z}'_{ij}$ by the following rule.
  - If the measurement $\mathbf{z}_j$ is smaller than the target $\tilde{\mathbf{x}}_i$, the center of $\mathbf{z}_j$ is used as $\mathbf{z}'_{ij}$.
  - Otherwise, shift $\tilde{\mathbf{x}}_i$ to the interior of $\mathbf{z}_j$ such that the shifted distance is a minimum and use the shifted $\tilde{\mathbf{x}}'_i$ as $\mathbf{z}'_{ij}$, as shown in Figure 3.2(b).

Using the approximated $\mathbf{z}'_{ij}$, the cost between $\tilde{\mathbf{x}}_i$ and $\mathbf{z}_j$ is defined as

$$\mathbf{C}(i,j) = -2\log(\mathcal{N}(\mathbf{z}'_{ij}; \mathbf{H}'\tilde{\mathbf{x}}'_i, \mathbf{S}')) \tag{3.10}$$

where $\mathcal{N}(\mathbf{x}; \mathbf{m}, \mathbf{\Sigma})$ denotes the normal distribution with mean $\mathbf{m}$ and covariance matrix $\mathbf{\Sigma}$, and $\mathbf{H}'\tilde{\mathbf{x}}_i'$ and $\mathbf{S}'$ denote the predicted measurement and covariance of the point measurement, respectively. Fixed costs are assigned for the dummy nodes.

$$\begin{aligned} \mathbf{C}(i, D) &= c_D,\ i = 1, \ldots, n_T \\ \mathbf{C}(N, j) &= c_N,\ j = 1, \ldots, n_M \\ \mathbf{C}(N, D) &= 0 \end{aligned} \tag{3.11}$$

The cost $\mathbf{C}(N, D)$ for the edge connecting the two dummy nodes are set as zero so that this edge can always be chosen when needed without affecting the cost of the solution. The cost matrix $\mathbf{C}$ represents a fully connected bipartite graph for the MWEC problem. The solution to the MWEC for the bipartite graph can be obtained using the reduction to the assignment problem and any general algorithm for the assignment problem. In our implementation the assignment alogrithm by Jonker and Volgenant [35] is used.

We now describe the method for generating the $k$-best solution to the problem using a modification of Murty's algorithm. The key idea of Murty's algorithm is that given a problem and its solution, the problem may be "partitioned" into a set of sub-problems which partitions the solution space that does not include the solution of the original problem. Here the problem is the MWEC, given by the association cost matrix. Specifically, given the best MWEC solution (set of edges) $E^{(1)} = \{e_1, \ldots, e_n\}$ to a problem $P_0$, the solution space can be partitioned into set $\overline{S}_1$ that does *not* include the first edge $e_1$ of $E^{(1)}$, and set $S_1$ that *does*. $S_1$ can in

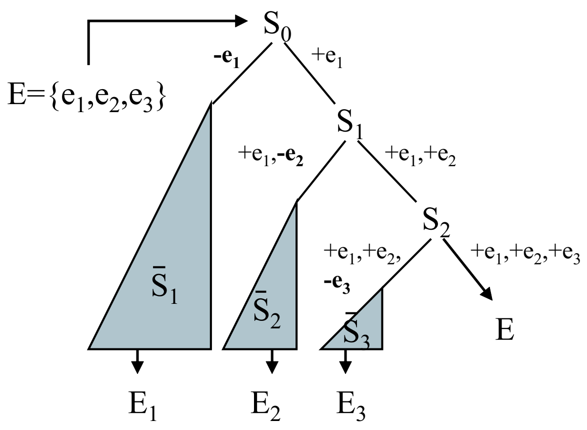


Figure 3.3: Illustration of partitioning the solution space given the current solution $E = \{e_1, e_2, e_3\}$.

turn be partitioned into $\overline{S}_2$ and $S_2$, where $\overline{S}_2$ includes $e_1$ but excludes $e_2$, and $S_2$ includes $e_1$ and $e_2$, and so forth. Figure 3.1 illustrates an example of the partitioning for $E^{(1)} = \{e_1, e_2, e_3\}$.

Let $P_m$ be the sub-problem whose possible solution space is the $m$th partition $\overline{S}_m$. We observe that the sub-problem $P_m$ needs to be constrained such that some edges are forced in the solutions and other edges are excluded from the solutions. Excluding an edge $(i, j)$ is done simply by removing the edge from the cost matrix $\mathbf{C}_m$ of the problem (set $\mathbf{C}_m(i, j) = \infty$). To force an edge $(i, j)$, we set $\mathbf{C}_m(i, j) = 0$ and solve the MWEC problem, after which we add $(i, j)$ to the solution. Note that the MWEC solution will not always include $(i, j)$ but this does not affect the cost of the solution, therefore adding $(i, j)$ gives the optimal solution with the forcing constraint. Thus a constrained problem $P_m$ is defined by the pair $\langle F_m, \mathbf{C}_m \rangle$ where $F_m$ is a list of forced edges.

To obtain the second-best solution, we solve all subproblems $\{P_1, \ldots, P_n\}$ of $E^{(1)}$. Let $\{E_1, \ldots, E_n\}$ be the respective solutions of $\{P_1, \ldots, P_n\}$. The minimum cost solution $E_m$ among $\{E_1, \ldots, E_n\}$ is the second best solution $E^{(2)}$. Then the sub-problem $P_m$ corresponding to $E_m = E^{(2)}$ is partitioned into its sub-problems. The next best solution is the one with the least cost from the sub-problems of $E^{(2)}$ *and* all previous sub-problems (from $E^{(1)}$ in this case). This process is repeated until $E^{(k)}$ is obtained.

So far we assumed that only one parent hypothesis existed from which the problem $P_0$ was constructed. When multiple hypotheses $\Theta_1, \ldots, \Theta_k$ exist from the previous time step, the best solution from each of the hypotheses is added to a list from which the minimum cost solution is selected. A new hypothesis $\theta'_l$ consists of a solution $E$ together with the parent hypothesis $\Theta_{s(l)}$ from which the problem of the solution was constructed. Also, the *cumulative cost*, which is the sum of the cost of $\theta'_l$ and $\Theta_{s(l)}$, is used to select the $k$-best hypotheses. Figure 3.4 describes the algorithm precisely.

The inner for-loop systematically creates sub-problems by temporarily removing one edge from the existing solution $E$ and cumulatively forcing the edge to subsequent sub-problems. Note that the dummy-to-dummy edge $(N_s, D)$ is never removed from the problem, where $N_s$ denotes the index of the dummy target in the hypothesis $\Theta_s$. Also, once an edge $(i, D)$ is forced i.e., target $i$ is not associated with any measurement, all other edges from target $i$ are removed. This is to prevent any contradictory solution which associates a target to both a real measurement and the dummy measurement. Similarly, appropriate edges are removed when $(N_s, j)$ is forced. In this work, we do not consider any solution that includes a path longer than two i.e., a target that is both split and merged at the same time. The last two

```
for each existing hypothesis $\Theta_s$
    create problem $P$ from $\Theta_s$ and $\{\mathbf{z}_j | j = 1, \ldots, n_M\}$
    $E \leftarrow$ solution of $P$
    add $\langle P, E, \Theta_s \rangle$ to the list $L$

for $l = 1 \ldots k$ or until $L$ is empty
    extract next best $\langle P, E, \Theta_s \rangle$ from $L$
    output hypothesis $\theta'_l \leftarrow \langle E, \Theta_s \rangle$
    for each non-forced edge $e = (i, j) \neq (N_s, D)$ in $E$
        $P' \leftarrow P$
        remove $e$ from $P'$ and solve $P'$
        if solution $E'$ for $P'$ exists
           add $\langle P', E', \Theta_s \rangle$ to $L$
        add $e$ to forced edges $F$ of $P$
        if $j = D$
           remove $(i, j')$, $j' = 1, \ldots, n_M$
        else if $i = N_s$
           remove $(i', j)$, $i' = 1, \ldots, n_T$
        else if $F$ includes a path longer than two
           break (stop partitioning from $E$)
        else if $F$ includes a tree
           for each leaf node $l$ of the tree
               from $P$, remove all edges $\notin F$ incident on $l$
```

Figure 3.4: The algorithm for generating $k$-best hypotheses given multiple parent hypotheses

if-statements attempt to avoid solutions that include such cases by limiting certain configurations of the forced edges. (Note that there is still a small chance of generating such an association. However in our experience, such solutions were rarely generated.)

Now we analyze the complexity of the algorithm given in Figure 3.4. First we show that the number of edges in any sub-problems in the algorithm is linear in the number of nodes in the graph.

**Lemma 3.1** *The number of edges in the solution of any sub-problem in Figure 3.4* *is $O(n)$, where $n$ is number of nodes in the graph defining the sub-problem.*

**Proof** Suppose that the solution $E$ of problem $P$ contains no cycles so the edges forced in each sub-problem $P'$ contains no cycles. The new edges added to the solution by solving MWEC contains no path connecting two nodes that are both incident on any of the forced edges (otherwise, it is not a minimum cost solution), thus the solution $E'$ is acyclic. Since the initial best solution is acyclic, by induction, none of the solutions of the sub-problems are acyclic. Undirected acyclic graph of $n$ nodes have at most $n-1$ edges therefore the number of edges in any solution is $O(n)$.

The MWEC problem has a time complexity of $O(n^3)$ since the reduction to the assignment problem takes $O(n^2)$ time [53] and Jonker and Volgenant's algorithm [35] takes $O(n^3)$, where $n$ is the number of nodes. Therefore from lemma 3.1, creating and solving sub-problems for each best hypothesis takes $O(n^4)$ time. We repeat this $k$ times so the total time complexity is $O(kn^4)$.

### 3.6.2 Hypothesis revision and update

In the second pass, each hypothesis from the first pass is revised to handle targets splitting into parts and to use the full state and measurement vectors for updating the targets and the cumulative hypotheses. The following process is applied to each hypothesis $\theta'_l = \langle E, \Theta_s \rangle$.

For each predicted target $\tilde{\mathbf{x}}_i$ in $\Theta_s$, it is determined whether $\tilde{\mathbf{x}}_i$ has been split or merged according to $E$. If $\tilde{\mathbf{x}}_i$ is *split*, each measurement $\mathbf{z}_j$ associated with $\tilde{\mathbf{x}}_i$ is initially treated as a split part from the same target and is a candidate to be merged back. Specifically, the minimum enclosing bounding box is formed by adding the $\mathbf{z}_j$'s in the order of increasing center distance from $\tilde{\mathbf{x}}_i$. The merging process continues as long as the size of the enclosing box satisfies the condition

$$\left(B_{dx} < \tilde{\mathbf{x}}_{dx} + \beta\sqrt{\mathbf{S}_{dx,dx}}\right) \wedge \left(B_{dy} < \tilde{\mathbf{x}}_{dy} + \beta\sqrt{\mathbf{S}_{dy,dy}}\right) \tag{3.12}$$

where $B$ denotes the enclosing bounding box, $\mathbf{S}$ is the predicted measurement covariance, subscripts $dx$ and $dy$ denote the width and height component of the vector, and $\beta$ is a tolerance parameter. The nearest measurement is added to $B$ regardless of the condition. Finally, $B$ is used as the revised measurement $\hat{\mathbf{z}}_i$ for $\tilde{\mathbf{x}}_i$ and the rest of the measurements are treated as unassociated. If any $\mathbf{z}_j$ is also associated with another $\tilde{\mathbf{x}}_{i'}$ i.e., $\tilde{\mathbf{x}}_i$ is split and merged at the same time, $\theta'_l$ is discarded.

For targets that are *merged* according to $\theta'_l$, their $\hat{\mathbf{z}}_i$'s are estimated by the following process: Let $\mathbf{z}$ be the (merged) measurement that the targets are associated with. We exploit the constraint that each of the four sides of $\mathbf{z}$ should be tangent to at least one target. For simplicity, it is assumed that exactly one target is tangent to each side. Assuming targets are roughly elliptical, we define the *tangent points* of a target as the center points on the sides of the bounding box. To determine which target is tangent to each side of $\mathbf{z}$, the tangent points on the sides are estimated using the local center of mass of the blob along each side as shown in Figure 3.5. Then for each side, the target with the nearest distance between its tangent point and the estimated tangent point is chosen. The targets that are chosen to be tangent

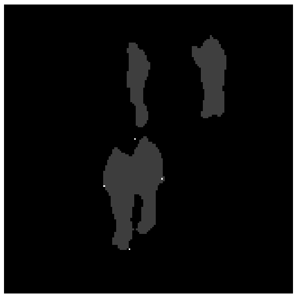


Figure 3.5: Estimated tangent points on the bounding box of the merged measurement.

to the sides of $\mathbf{z}$ are given appropriate measurement $\hat{\mathbf{z}}_i$. For each target that is not tangent to any sides, $\hat{\mathbf{z}}_i$ is given by the procedure for $\mathbf{z}'_{ij}$ in section 3.6.1, using as the measurement the region in the blob that is not occupied by any tangent targets. Lastly if the target is *neither split nor merged*, we let $\hat{\mathbf{z}}_i = \mathbf{z}_j$.

Next, each target in $\theta'_l$ is updated using $\hat{\mathbf{z}}_i$ as the measurement. In addition, if the target is merged, its velocity is exponentially decreased over time. We re-initialize the state of the target from which a new target is split, instead of updating it. Unassociated targets are removed and new targets are created for unassociated measurements in this step. (We assume that false negative measurements are rare and remove a target once it is unassociated. However, this decision may be deferred by either counting the number of consecutive unassociations of each target or by hypothesizing both the removal and the temporary unassociation of a target.) Let $\theta_l$ be the revised hypothesis that reflects this new set of target states. The updated cumulative hypothesis is expressed as

$$\Theta_l^{1:t} = \{\Theta_{s(l)}^{1:t-1}, \theta_l^t\} \tag{3.13}$$

where $t$ is the time index and $s(l)$ denotes the parent hypothesis index of $\theta_l$.

Finally, the cumulative cost of $\Theta_l^{1:t}$ is updated. The probability of the cumulative joint association hypothesis can be expressed as

$$p(\Theta_l^{1:t}|Z^{1:t}) = \frac{1}{c} p(Z^t|\theta_l^t, \Theta_{s(l)}^{1:t-1}, Z^{1:t-1}) p(\theta_l^t|\Theta_{s(l)}^{1:t-1}, Z^{1:t-1}) p(\Theta_{s(l)}^{1:t-1}|Z^{1:t-1}) \tag{3.14}$$

where $Z^t$ denotes the set of measurements at time $t$. The normalizing constant $c$ may be ignored for comparison between hypotheses since it is common across all hypotheses. We formulate the first two factors as

$$\prod_{i=1}^{n_T} \left( [\mathcal{N}(\hat{\mathbf{z}}_i; \mathbf{H}\tilde{\mathbf{x}}_i, \mathbf{S})]^{\delta(i)} \right) (P_D)^{n_D} (P_N)^{n_N} (P_S)^{n_S} \tag{3.15}$$

where $\delta(i)$ is 1 if $\tilde{\mathbf{x}}_i$ is associated and 0 otherwise. $P_D$, $P_N$, and $P_S$ respectively represent the density of Poisson distributed events for unassociated targets, unassociated measurements, and targets split into parts, which are given as constant parameters. $n_D$, $n_N$, $n_S$ denote the respective number of such events determined by $\theta_l^t$ [51]. (The dependency on $\theta_l$ and $\Theta_{s(l)}$ in (3.15) is omitted for conciseness.) We take the logarithm of the probability to derive the cost of $\Theta_l^{1:t}$

$$\begin{aligned} C(\Theta_l^{1:t}) &= -2\ln(p(\Theta_l^{1:t}|Z^{1:t})) = \\ &- \ 2\left( \sum_{i=1}^{n_T} \ln[\mathcal{N}(\hat{\mathbf{z}}_i; \mathbf{H}\tilde{\mathbf{x}}_i, \mathbf{S})]\delta(i) + n_D \ln(P_D) + n_N \ln(P_N) + n_S \ln(P_S) \right) \\ &+ \ C(\Theta_{s(l)}^{1:t-1}) \end{aligned} \tag{3.16}$$

## 3.7 Implementation

The costs $c_D$ and $c_N$ for the dummy nodes in the first pass should be given values such that it generates diverse hypotheses. If they are too small, normal targets may be hypothesized as terminated and the correct measurement associated with a new target. This may prevent more useful hypotheses from being generated such as merging and splitting. In particular, $c_N$ should be large enough so that targets split in parts are hypothesized as split, rather than being associated with only one part. This will ensure that the measurements are given a chance to be merged back in the second pass. On the other hand, the probabilities $P_D$ and $P_N$ should be given values that more or less reflect the actual probabilities. Note that $P_S$ acts as a penalty for a target being split into parts multiple times. When a target slowly splits into multiple targets, it may be difficult to distinguish between a real split ($\theta_1$) and a temporary split into parts ($\theta_2$) in the current time. However as time progresses, the cost of splitting multiple times in $\theta_2$ outweighs the cost of creating a new target in $\theta_1$ and the correct hypothesis gets selected.

In order to increase the efficiency and generate more useful hypotheses, we apply two different hypothesis pruning strategies. First, hypotheses with costs exceeding the best cost by more than the threshold $T_c$ are removed in the generation process. Second, we keep track of the age of each cumulative hypotheses and prune the ones with age $T_a$ or older. The age is incremented when the cumulative hypothesis is updated. (The best hypothesis is always given age zero.) Further, the hypothesis generation algorithm is modified so that only the best cumulative hypothesis may branch into multiple child hypotheses. This ensures that alternative hypotheses from the best hypothesis, which tends to be the useful ones, survive long enough.

## 3.8 Experimental results

We first present results on video segments from a soccer game. A subset of the VS-PETS 2003 “football” dataset is used to validate the effectiveness our multiple hypothesis approach. The frame size is $320 \times 240$ pixels and lasts 600 frames. For this video, the algorithm written in C++ achieved real time performance. Background subtraction was done by a simple variant of the method described in [36]. Using at most $k = 20$ hypothesis, all 18 targets including the ball were correctly tracked. Figure 3.6 (a)–(c) show sample frames from the tracking result containing merged targets and targets split into parts. Figure 3.6 (d) shows object detection results corresponding to Figure 3.6 (c). The black boxes and the white boxes represent the measurements and the target states, respectively. The numbers below the white boxes denote unique identification numbers for the targets. During the entire sequence, there were 7 instances of two objects merging and then separating correctly. 2 of the instances involved near-complete overlap between two people, each lasting 38 frames and 104 frames.

To validate the effectiveness of maintaining cumulative multiple hypotheses over time, we compared the result between using multiple hypotheses and single hypothesis. The single hypothesis implementation used multiple hypotheses ($k =$

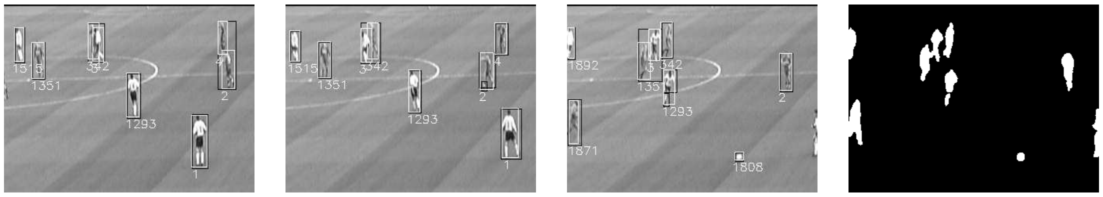
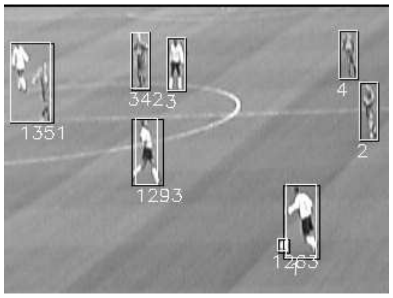

(a) frame 445 (b) frame 475 (c) frame 549 (d) frame 549

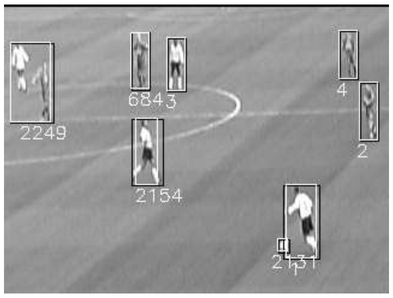
Figure 3.6: Sample frames from the multiple hypothesis tracking result.

20) in the first pass, but kept only the single best hypothesis at each time step. The previous video was used for both algorithms and to simulate false positive measurement, we chose a small area in frame 410 and lowered its intensity. Figure 3.7 highlights the difference in the results between the two approaches. In frame 398, a target initially appeared from the left as a group of two persons merged together then a target was split from the group in frame 399. After several frames (in frame 403) the multiple hypothesis approach correctly split the group, as mentioned in Section 3.7. However, using a single hypothesis, the two persons in target 2249 kept getting merged back due to the absence of alternative hypotheses. Also, the multiple hypothesis algorithm correctly removed the noise (target 1557) in frame 410 after 3 frames. This was possible because both the removal of the noise and merging of the noise with target 3 were hypothesized. Due to the accumulated cost of moving the false target to target 3, eventually the hypothesis that removed the false target was chosen. The single hypothesis algorithm failed to remove the false target since at each time step, moving the false target cost less than removing it.

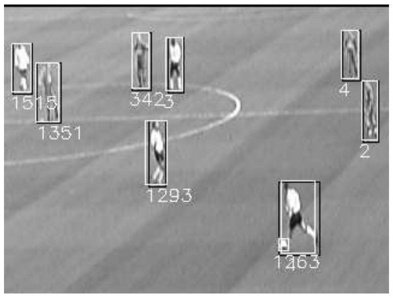
To evaluate the performance under different degrees of interaction between targets, we tested our method using the entire video from the VS-PETS 2003 dataset for which the ground truth data was available. This video has frames of size 720 ×



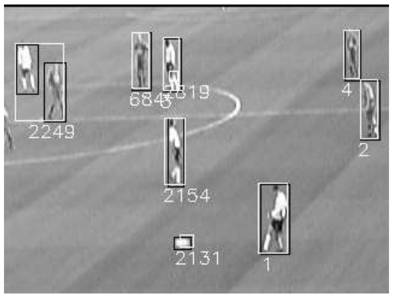
(a) frame 398



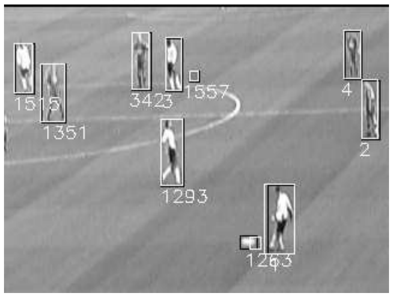
(e) frame 398



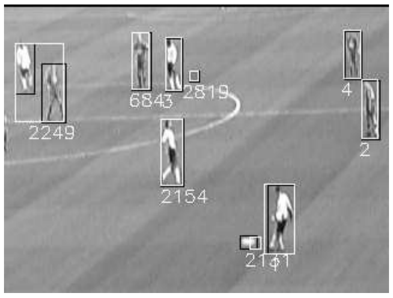
(b) frame 403



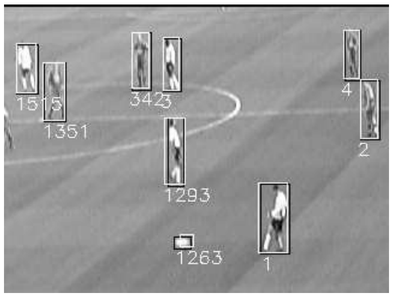
(f) frame 403



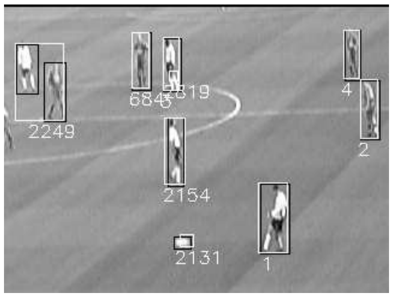
(c) frame 410



(g) frame 410



(d) frame 413



(h) frame 413

Figure 3.7: Comparison between multiple hypothesis (a)–(d) and single hypothesis (e)–(h). Frames 398 and 403 show before and after target split. Frame 410 contains false positive measurement.

576 pixels, lasts 2500 frames, and contains 59 unique targets in the ground truth data. Figure 3.8 shows a sample output frame from this video. The performance regarding occlusion handling is measured by the success of tracking over the duration of pairwise occlusion events. An occlusion event is defined by an instance where two initially separated ground truth targets overlap then separate. We regard the tracking as successful if the two corresponding tracked targets exist before the event and have centers that are each within their ground truth targets after the event. 179 occlusion events were identified from the video excluding events with very short duration or very small overlap. Our method successfully tracked the targets in 136 events using $k = 20$. Figure 3.9(a) shows the number of events and the number of successfully tracked events, classified by different degrees of occlusion. The degree of occlusion for an event is defined as the ratio

$$\text{overlapped area}/\min(\text{target 1 area}, \text{target 2 area}) \tag{3.17}$$

Figure 3.9(b) shows the performance depending on the total number of targets involved in an event. By involved we mean a target spatially and temporally overlaps any target in the pairwise occlusion event. (The "4" in the x-axis represent four or *more* involved targets.) The performance decreased with increasing degree of occlusion and the number of targets involved. However, under reasonable degrees of target interactions, the performance was very good. Tracking errors were typically caused by severe overlap combined with either a sudden change in the velocity during the overlap i.e.,"bouncing" targets (Figure 3.10(a)), or a bad estimate of

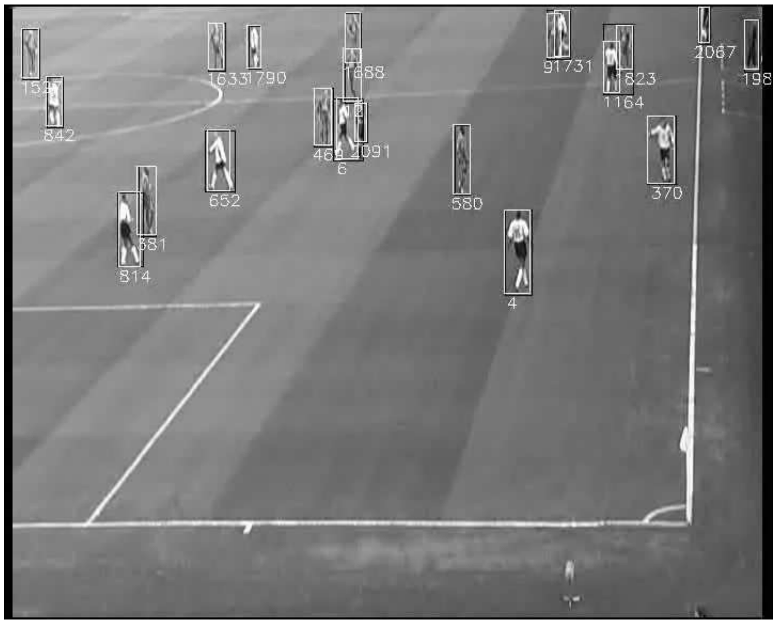


Figure 3.8: Sample frame from the full soccer video.

measurements within a merged measurement (Figure 3.10(b)).

In the next experiment, our method was tested using two videos, each containing a complex interaction among a group of four persons. The two videos were taken from two different outdoor security cameras. Both have a resolution of $320 \times 240$ pixels and last 180 frames. For these videos, a more sophisticated background subtraction algorithm [38] is used. Using $k = 5$ hypotheses, our algorithm successfully tracked all targets in both videos. Figure 3.11 shows the plot of the average overlap over time, along with sample frames roughly corresponding to peaks in the plot. The average overlap for a frame is defined as

$$1 - \frac{\text{total target area that is not overlapped}}{\sum_i \text{area of target}_i} \tag{3.18}$$

The results show the effectiveness of the algorithm for estimating measurements for

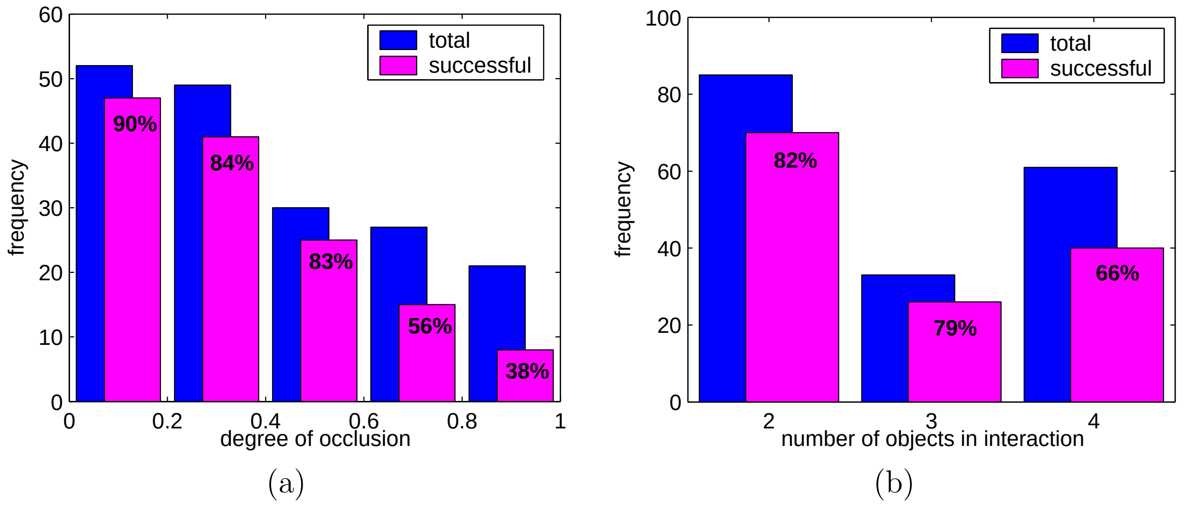


Figure 3.9: Tracking success rates on the full soccer video under (a) varying degrees of occlusion, (b) different number of targets involved in an event.

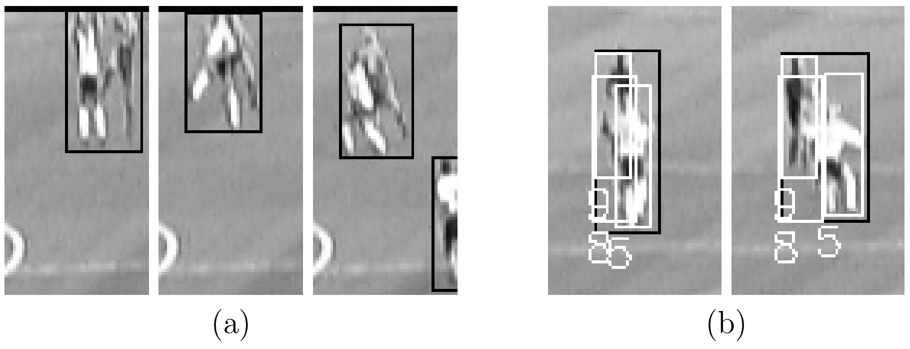
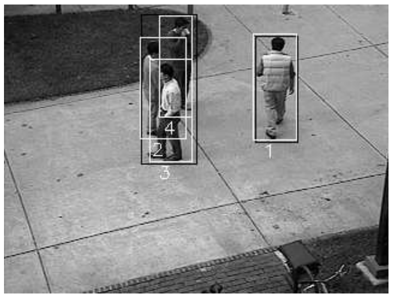


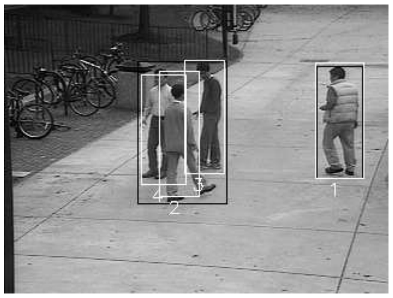
Figure 3.10: Sample interactions for which tracking failed. (a): bouncing targets (target bounding boxes are not shown for clarity), (b): bad target location estimate within a merged measurement.

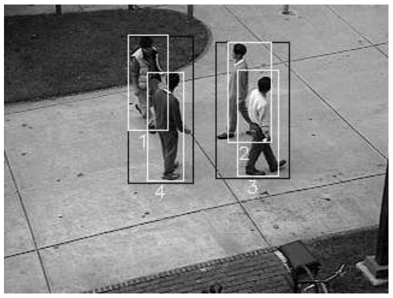
each target within a merge measurement given in section 3.6.2.

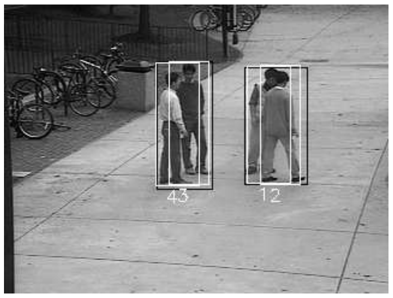
## 3.9 Summary

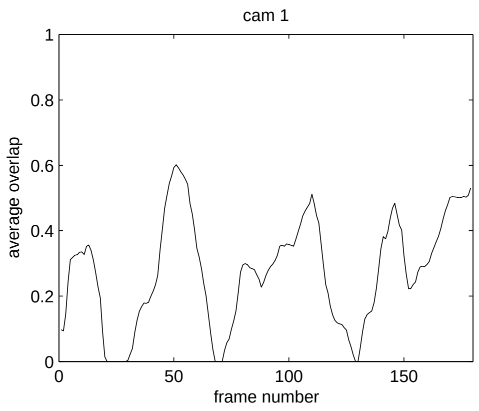
We have presented a multi-object tracking approach based on the multiple hypothesis tracking method by Reid. Since in visual tracking the objects may split and merge, the data association problem was posed as a minimum weight edge cover problem. A polynomial-time algorithm was developed for generating the $k$-best hypotheses regarding the multiple association. Our implementation achieved real-time performance for relatively low-resolution videos. The experiment results show that the multiple hypotheses approach is effective in recovering from data association errors and that our approach is robust against varying degrees of target interactions.


(a) frame 48

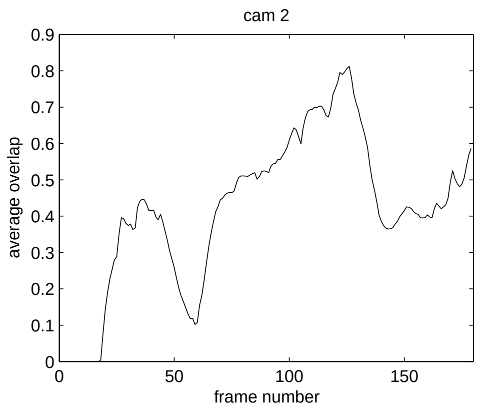

(d) frame 36


(b) frame 147


(e) frame 129



(c)



(f)

Figure 3.11: Tracking results from security videos. (a)–(c): camera 1, (d)–(f): camera 2.

# Chapter 4

# Visual Event Recognition Based on Syntactic Models of Events

## 4.1 Introduction

Surveillance cameras are becoming ubiquitous, however human resources to supervise and process the abundant video data are limited. Automatic visual surveillance addresses this problem by recognizing events of interest without human intervention. Recognition of short-term events such as human gesture has been studied extensively using statistical pattern recognition techniques, such as Hidden Markov Models. However, it is difficult to apply the same approach to the problem of recognizing events that span extended periods of time with complex structure involving multiple interacting objects. A major difficulty is that for complex activities, very little training data is available compared to the huge dimensionality of its feature space. Also, semantically equivalent activities may often have feature values that are quite far apart. Such complex events can be more suitably represented by an explicit model of the structure in the pattern.

There has been substantial amount of work on structural model-based event recognition. Vu et al. [59] defined a description of an activity as consisting of actors, logical predicates, and temporal relations between sub-events. The activity recognition problem was then posed as a constraint satisfaction problem, where the search process for the temporal constraints was optimized. Ghanem et al. [22] proposed the use of high-level Petri-Nets for representing and recognizing events. Petri-Nets have the ability to represent sequentiality, concurrency, and synchronization of events. In their framework, tokens represent objects and firing of a transition corresponds to the occurrence of a primitive event subject to conditions imposed on the transition. Ivanov and Bobick [30] described a method of modeling and recognizing activities using stochastic context-free grammars (CFG) and stochastic parsing. The input symbols were generated by low-level event detectors based on HMMs. The parser corrected substitution and insertion errors and handled concurrent tracks from separate objects using insertion error correction. Consistently associating interacting objects with an activity was enforced by simple spatial-temporal conditions. The authors demonstrated results from gesture recognition and outdoor video surveillance. Stochastic CFG was also used by Moore and Essa [45] to recognize events involving multiple entities where stochastic parsing was applied to each independent interactions in blackjack games.

Recent work on detection of anomalous activities were mostly based on statistical learning. Stauffer and Grimson [55] extracted feature prototypes from tracked objects and classified activities by hierarchically clustering the prototypes using the co-occurrence statistics of the prototypes within a track. Unusual activities are detected by measuring the deviation from the learned prototype density and the co-occurrence statistics. Similar approaches include jointly clustering image features and video segments [64] and clustering $n$-grams of pre-defined events [24]. Vaswani et al. [58] modeled a group activity as a dynamic "shape" (object trajectories with translation, rotation, and scale removed). The shape dynamics is modeled by a continuous HMM and an abnormal activity is defined as a change in the model. Dee and Hogg [14] modeled pedestrian behavior as a series of linear trajectories that are directed towards a point in space. Interesting behaviors are detected by measuring the degree of fit to this model.

In this chapter, we present a real-time event recognition system based on models of events represented by attribute grammars. In contrast to purely syntactic grammars, attribute grammars are capable of describing features that are not easily represented by finite symbols. Primitive events and their attributes are extracted from the video by detecting and tracking moving objects in the scene. This intermediate data is incrementally parsed according to the attribute grammar to recognize high-level events. Our approach handles multiple concurrent events involving multiple entities by associating object identification labels with event threads. Uncertainty in the semantic conditions on the attributes are expressed using probabilities, which are used to generate confidence measure of recognized events. The attribute grammar may be used to model specific events of interest or general events that typically occur in a scene. In the latter case, the modeled events are recognized and labeled with the event category, while events that does not follow the grammar are labeled as abnormal. Experiment results are presented for recognizing specific events as well as abnormal events occurring in the parking lot.

In section 4.2 the detection and tracking method used in the system is described. Section 4.3 describes how the the primitive events and attributes are extracted from the tracking data. The attribute grammar and the parsing algorithm in the context of our system is discussed in sections 4.4 and 4.5, respectively. Next, the implementation of the system and experiment results are respectively described in sections 4.6 and 4.7.

## 4.2 Detection and tracking

Moving blobs are detected using background subtraction and are tracked by associating the blobs with targets. A pixel-wise adaptive Gaussian background model is used for object detection. Also for each pixel, the local temporal variance $tvar$ is used as an indicator of motion caused by a moving object. A background update counter $UC$ is set to $UC_{max}$ when motion is detected and is decremented at each frame, which is used to delay the adaptation of a foreground pixel into the background. Objects are detected from the foreground mask using connected component analysis. We shall refer to the connected components as measurements.

Tracking is done by associating existing targets from the previous frame with the measurements in the current frame. A set of associations can be represented by a bipartite graph where each node has a degree of at least 1, as shown in Figure 3.1. A node of degree greater than 1 correspond to a target split or merge. A dummy target and a dummy measurement node respectively covers entered and exited targets. The optimal association is obtained using a minimum weight bipartite graph edge covering algorithm [53] where the edge weights in the graph are given by the Euclidian distance between target and measurement pairs. Target removal are delayed by a small number of frames and the measurements that are fragmented into parts for a short duration are merged back. The bounding box of the measurement is used to update the target state which is tracked by a Kalman filter with a constant velocity motion model.

Undesirable artifacts caused by the slow adaptation of the background model are reduced by selectively updating the background in the object level. To prevent a stationary target from being gradually fragmented, the background corresponding to the target region is updated simultaneously when its average value of $UC$ is small. Similarly, to avoid the "ghost" effect where a newly revealed background region is falsely detected as an object, the background is updated when the average $tvar$ of a newly detected target is small. This approach is similar to [11] but is selectively applied using feedback from the tracker.

## 4.3 Primitive events and attributes

In syntactic pattern recognition [20], the given data is represented by a string of input symbols from an alphabet (a finite set of symbols). For event recognition, the symbols correspond to what we call primitive events extracted from the tracking data. For example, a primitive event might be generated when a moving object stops. In a purely syntactic approach, all the information used for recognition is contained in the string of primitive events. However, it is often desirable to use additional attributes or features associated with the primitive events. For example, the exact location in which the primitive event occurs may be significant for describing an event, but this may not be effectively encoded in the primitive event set. Attributes are also useful where the number of primitive events is unbounded such as in an event involving arbitrary number of objects each having distinct primitive events associated with it. In this case, an object identification label can be used as the attribute of the primitive event.

The basic primitive events used in this work are *appear*, *disappear*, *start*, and *stop*. Additional variants of these primitive events with different constraints are used depending on the application. For each existing target, a set of states is maintained including the target location in the image, the width and height, the velocity, age (the number of frames the target has been tracked), and the object type. To reduce the effect of noise in the tracking data, the location and the velocity are smoothed by temporal averaging. When a target is created by the tracking algorithm, the target is given an *inactive* state. The *appear* event is generated when its age reaches a threshold and its bounding box is not on the boundary of the frame i.e., the target has completely emerged from the frame boundary. However, if the target is split from another target, the *appear* event is generated immediately. The type of the object is also determined when the *appear* event is generated. A simple classification such as person and vehicle is achieved based on the dimensions of the target. The *disappear* event is generated when the target is no longer tracked. The events *start*, and *stop* are generated by hysteresis thresholding on the speed of the target, where a high and a low threshold is respectively used for *start* and *stop*.

Each primitive event is assigned a set of primitive attributes which includes *id* (target identification label), *loc* (location in 2d coordinates), and *timestamp* (the frame number). In some applications, relational attributes such as the *id* of and distance to the nearest object may be used.

In addition to tracked objects, special objects named *contextual objects* that represent semantically significant regions are pre-defined in the given scene. Examples contextual objects include parking spaces and building entrance.

## 4.4 Representation of events using attribute grammars

### 4.4.1 Attribute grammars

Attribute grammars, first introduced by Knuth [39], have been used in syntactic pattern recognition [20] as well as natural language processing [3] and programming languages [1]. An attribute grammar $AG$ is a five-tuple

$$AG = (G, SD, AD, R, C) \tag{4.1}$$

where

- $G = (V_N, V_T, P, S)$ is the underlying context-free grammar. $V_N$ and $V_T$ represent the non-terminal and terminal symbols, respectively, $P$ is the set of productions, and $S$ is the start symbol.
- $SD$ denote a semantic domain consisting of a set of types (e.g., integers or coordinates) and a set of functions operating on the types.
- $AD$ is a set of attributes associated with each symbol occurring in the productions in $P$. Each attribute is of a certain type.
- $R$ is a set of *attribute evaluation rules* associated with each production $p \in P$.

Attributes are evaluated using functions defined on attribute values of symbols in $p$.

- $C$ is a set of *semantic conditions* associated with each $p \in P$, which are predicates defined on the attribute values.

*Synthesized attributes* are initially given with the terminal symbols, which are passed up the parse tree during parsing, whereas *inherited attributes* are evaluated top-down from the parents of the node. The semantic conditions impose constraints on the value of the attributes such that the production $p$ can be used only when the conditions are satisfied. (The term *semantic* here refers to the non-syntactic nature of the attributes and does not necessarily refer to the meaning of an event.)

We extend the predicates in the semantic condition to real valued functions with range $[0, 1]$. The function may have a value of either 0 or 1 (for hard condition) or have the probability value that the condition is satisfied (soft condition). We refer to the latter case as the soft predicate. Typical examples of predicates used in this work include $Near(loc_1, loc_2)$ and $Inside(loc, area)$ where the former imposes a constraint that $loc_1$ and $loc_2$ are near each other and the latter refers to a condition that $loc$ lies within a region denoted by $area$. Soft predicate names are prefixed by the letter $s$ e.g., $sNear(X1.loc, X2.loc)$. A conjunction of boolean predicates is extended to a product of probabilities (assuming independence between the attributes) and a set of conditions is regarded as satisfied if the product of their probabilities is non-zero. Each production may be given an optional weight, reflecting the relative frequency that the production is used to generate the language. The production weights are considered as a “prior” probability on the semantic conditions and are multiplied to the predicate values.

We use the following notation for the attribute grammar. Words in capital letters and lower case letters represent nonterminal and terminal symbols, respectively. The symbol in a production for which an attribute is associated is denoted as $Xi$ where $i$ is the index to the symbol; $X0$ refers to the symbol on the left hand side (lhs), $X1$ denotes the first symbol on the right hand side (rhs), and so on. An attribute $a$ associated with symbol $Xi$ is denoted $Xi.a$. Attribute evaluation rules are listed following each production. For example, $Xi.a := f(Xj.b)$ denotes that the value of attribute $a$ of $Xi$ is given by the evaluation function $f(\cdot)$ with the value of attribute $b$ of $Xj$ as the argument. Semantic conditions are expressed as a conjunction of predicates enclosed in parentheses.

### 4.4.2 Thread consistency

We handle arbitrary number of concurrent events each of which consist of a sequence of primitive events which in turn may originate from different objects. Therefore to keep track of the identity of the object (represented by the attribute *id*) that each symbol in the grammar is associated with, a special attribute named *thread consistency id* (*tid*) is implicitly defined for each symbol in the grammar. Subscripts on the symbols in the rhs of the production describe the implicit evaluation rules and conditions regarding the *tid*. The subscript $i$ in a symbol $A_i$ represents the index to other symbols in the production, starting from 0. For a terminal symbol $b_i$

the implicit condition is ($b.id = Xi.tid$) where $Xi$ denotes the $i$th symbol. An index $N$ refers to a "wildcard" symbol whose $tid$ matches any $tid$. For example, in the production $A \rightarrow B_0C_1D_NE_3$, the implied condition is $B.tid = C.tid = A.tid$ and $E.tid = D.tid$. For the terminal symbol, $tid$ is given by the attribute $id$ assigned to the primitive event. Details on how the $tid$'s are assigned to nonterminals are described in the next section. The wildcard $tid$ allows us to associate multiple objects to an event thread involving multiple objects. A typical example is a thread consisting of a vehicle being parked and then a person exiting from the vehicle. Ivanov and Bobick [30] used a simple fixed rule to determine whether the change of object has occurred in a thread. However, our representation is more flexible since arbitrary conditions can be explicitly described by the attribute grammar.

## 4.5 Recognition of events by parsing

For real-time surveillance applications it is important to detect events as they happen, which requires an online parsing algorithm. We have implemented a parsing algorithm for attribute grammars based on the Earley parser [18]. We assume the order in which the synthesized attributes are evaluated are such that the arguments of each evaluation function are always defined previously. Also, the inherited attributes for a symbol $A$ are assumed to depend only on the attributes on the left of $A$. These assumptions are very natural for describing an event that unfolds over time (it is natural to refer to attributes that are known from the past rather than unknown attributes in the future). In such a case, it is straightforward to evaluate attributes and check for conditions in the Earley's parsing framework.

### 4.5.1 Online parsing

Earley's algorithm reads terminal symbols sequentially, creating a set of all the pending derivations (potential event threads) that is consistent with the current input terminal symbol. A pending derivation is represented by a state which is expressed as

$$i : {}_kX(0) \rightarrow X(1) \ldots X(j) \bullet X(j+1) \ldots X(n) \tag{4.2}$$

where $X(0), \ldots, X(n)$ represent the symbols in the pending production, $i$ is the index of the current state set, and $k$ refers to the state set from which the nonterminal $X(0)$ was expanded. The dot marks the current position ($j$ in this state) in the pending derivation. In addition, a list of attributes and its values are stored for each symbol in the state. Given the next input terminal symbol, the parsing algorithm iteratively performs one of the following three operations for each state in the current state set. We assume the current state is given by (4.2).

- Prediction: If the symbol after the dot, $X(j+1)$ is a nonterminal $A$, evaluate the set of inherited attributes $\{a_m\}$ for $X(j+1)$. For each production that expands $A$ (let $Y(0)$ be the lhs for this production), add the following state in the current state set $i$, and assign $\{a_m\}$ to $Y(0)$:

$$i : {}_iY(0) \rightarrow \bullet Y(1) \ldots Y(n) \tag{4.3}$$

- Scanning: If $X(j+1)$ is a terminal and matches the next input $b$, copy the current state to set $i+1$, moving the dot to the right of $X(j+1)$, and assigning all the attributes that are given with $b$ to $X(j+1)$:

$$i+1 : {}_kX(0) \rightarrow X(1) \ldots X(j+1) \bullet \ldots X(n) \tag{4.4}$$

- Completion: If the dot is in the rightmost position ($j = n$), check for all conditions on the attributes of $X(0), \ldots, X(n)$. If the conditions are satisfied, evaluate all synthesized attributes $\{a_m\}$. For each state in state set $k$ (the set where $X(0)$ was predicted) whose nonterminal $Y(h)$ on the right of the dot is identical to $X(0)$,

$$k : {}_{k'}Y(0) \rightarrow Y(1) \ldots \bullet Y(h) \ldots Y(n') \tag{4.5}$$

  copy the state to the current set $i$, move the dot over, and assign $\{a_m\}$ to $Y(h)$:

$$i : {}_{k'}Y(0) \rightarrow Y(1) \ldots Y(h) \bullet \ldots Y(n') \tag{4.6}$$

### 4.5.2 Managing concurrent events

The preceding algorithm is modified to effectively maintain multiple independent event threads using the thread consistency id (*tid*). In the prediction step, *tid* is inherited to the added state as

$$\begin{cases} Y(0).tid := NID, \text{if} d = N \\ Y(0).tid := X(d).tid, \text{otherwise} \end{cases} \tag{4.7}$$

where $d$ is the subscript of $X(j+1)$ in the current state.

In the scanning step, the input symbol is skipped (ignored by the current state) if the *id* of the input is not consistent with the *tid* specified by the grammar. Specifically, given the next input symbol $b$, the procedure shown in Figure 4.1 is used to determine whether to skip or scan or do both. The skipping process simply copies the current state to the next state set $i+1$. This procedure ensures that only the symbol from the expected object, specified by *tid*, is considered for the current thread, but if the expected *tid* is a "wildcard", any object is considered. Note that in the case of a wildcard, the current state is also skipped since the newly associated object may not be the "correct" object (perhaps not satisfying a condition in a later step). This nondeterministic behavior enables the parser to initiate and maintain multiple candidate threads in a common framework. In particular, by specifying the *tid* of the start symbol as a wildcard, the parser looks for a potential new thread each time an input symbol is read. This approach of explicitly associating primitive events with appropriate threads is in contrast to the approach in [30], where symbols from concurrent threads are treated as insertion noise. It should be noted that in case 1 of the algorithm, the object associated with the previous symbol is not allowed as the new object. Also in the scanning step in case 1, the symbol from a previously

```
// case 1: tid is a wildcard
if ( d=N or X(d).tid=NID )
    if ( X(j+1)=b )
        do scan, skip
    else
        do skip
// case 2: tid matches
else if ( X(d).tid=b.id )
    if ( X(j+1)=b )
        do scan
// case 3: event from other threads
else
    do skip
```

Figure 4.1: Algorithm for handling multiple event threads in the scanning step.

skipped object is prevented from being scanned. These exceptions are omitted from the algorithm in Figure 4.1 for conciseness.

Finally in the completion step, the *tid* of the last symbol is passed to the completed nonterminal symbol as a synthesized attribute. The probability for the condition is treated as the "inner probability" [56] for stochastic parsing and is accumulated for each of the pending parses.

An event may be classified as abnormal when none of the states successfully scans the current terminal symbol i.e., the current primitive event is not explained by any pending event threads. This is a case of a syntactic abnormality. An event that is recognized as a normal event but with a probability lower than a threshold is also labeled as abnormal. This can be referred to as a semantic abnormality. In the case of recognizing specific events of interest, the parser does not label any events as abnormal.

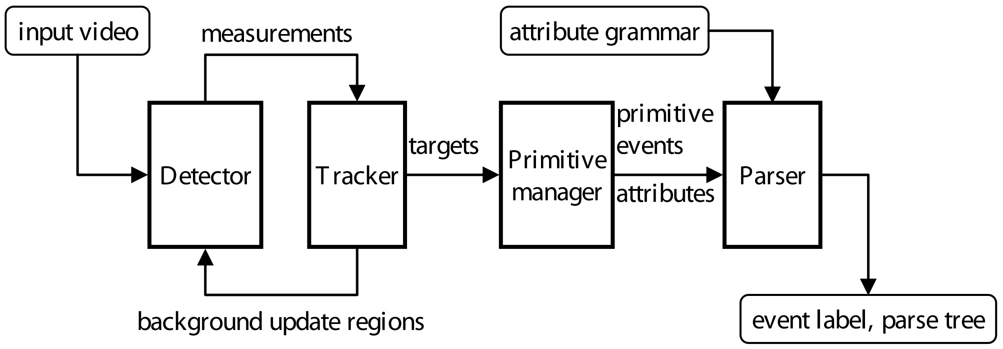


Figure 4.2: A block diagram of the event recognition system

## 4.6 System implementation

The recognition system consists of the detection and tracking module, the primitive manager, and the parser as shown in Figure 4.2. At each frame, the tracker provides a list of current targets to the primitive manager. The primitive manager passes primitive events to the parser only when they are detected. When the parser recognizes an event, the label of the event is immediately displayed along with its computed probability. Also, the primitive events that are used in the derivation are displayed in the location where they have occurred, summarizing the details of the event. In case of a syntactic abnormality, the primitive event that caused the parse to fail is displayed. After an event is recognized, the system continually monitors the video for other events. The entire system, written in C++, runs in real-time on a common PC platform.

## 4.7 Experiments

As an example application of detecting a specific event of interest, we demonstrate the task of detecting a person casing cars in the parking lot. Next, an example is presented for the application of recognizing normal events and detecting abnormal events in the parking lot.

### 4.7.1 Detection of specific events

The primitive events used in this experiment are *perapp* (person appears), *carapp* (car appears), *disappear*, *stop*, and *start*. The *stop* event is given the attribute *dist*, which denotes the distance to the nearest vehicle when the event occurred. A contextual object *Fov* specifying the field of view of the camera excluding some margin is used.

Table 4.1 shows the attribute grammar for this event. The casing behavior, described by the productions expanding *CASING* and *CASING2*, involves a person stopping near a vehicle two times or more. The attribute *mindist* keeps track of the minimum among the set of distances to the nearest vehicle over all the stopping events. Note that the grammar describes not only the key event of casing, but also the context of the event such as how the person enters and leaves the scene. Specifically, the low weights ($P$=0.1) assigned to the last two productions expanding *CASEVEHICLES* reflect the low relative frequency that a person following typical behaviors in a parking lot (driving in and walking away or walking in and driving away) will case vehicles. To prevent the parser from assigning low probabilities to long strings, we give large probabilities to recursive productions. The computed probability for the parse using these production weights will not be a probability measure. This is acceptable, since we are only interested in the relative likelihood of an event. For associating two different objects with a single event thread such as a person exiting from a car, a hard predicate is used which involves a threshold that is predetermined depending on the scene. This is a reasonable choice since the upper bounds on the size and speed of an object are more or less fixed, given a particular camera configuration. However, we note that the criterion is fuzzy regarding how close one should be to a vehicle in order to conclude that the person is casing. Thus in the condition for *CASING*, we use a soft predicate $sDistNear()$ defined by a Gaussian function.

The last production in the grammar shows the error correcting feature of the grammar. The tracker as described in section 4.2, may lose track of a person and shortly regain track. However, the two instance of the tracks—and consequently the event threads that originates from the tracks—are not associated with each other. The association is done in the parser using the production that links a sequence of *disappear* and *perapp* events into a single thread, given that the two events are spatially ($Near()$) and temporally ($TimeNear()$) close.

We show the result from 3 surveillance videos for recognizing the vehicle casing event. Figure 4.3 shows a representative frame (top row) and the recognition result (bottom row) for each sequence. The lightly colored region depicts the contextual object. In sequence (a), a person walks into the lot, cases vehicles, and walks out of the lot. Sequence (b) consists of a person driving into the parking lot, casing vehicles, and driving out of the lot. Sequence 3 contains a person parking a car, and walking into a building. The person stops twice with no intention of casing vehicles. Sequences (a) and (b) were correctly recognized as the casing event with

| | P | Attribute rules and Semantic conditions |
|---|---|---|
| S → CASEVEHICLES$_N$ | 1.0 | |
| CASEVEHICLES → perapp$_0$CASING$_1$disappear$_2$ | 0.5 | NotInside(X1.loc, Fov), NotInside(X3.loc, Fov)) |
| CASEVEHICLES → DRIVEIN$_0$CASING$_1$DRIVEOUT$_1$ | 0.3 | |
| CASEVEHICLES → perapp$_0$CASING$_1$DRIVEOUT$_2$ | 0.1 | NotInside(X1.loc, Fov)) |
| CASEVEHICLES → DRIVEIN$_0$CASING$_1$disappear$_2$ | 0.1 | (NotInside(X1.loc, Fov)) |
| DRIVEIN → carapp$_0$DSTOP$_1$perapp$_N$ | 1.0 | Near(X3.loc, X2.loc) |
| DSTOP → start$_0$stop$_1$DSTOP$_2$ | 0.9 | X0.loc := X3.loc |
| DSTOP → start$_0$stop$_1$ | 1.0 | X0.loc := X2.loc |
| DRIVEOUT → disappear$_0$start$_N$DEXIT$_2$ | 1.0 | (Near(X2.loc, X1.loc)) |
| DEXIT → stop$_0$start$_1$DEXIT$_2$ | 0.9 | |
| DEXIT → disappear$_0$ | 1.0 | |
| CASING → start$_0$PERSTOP$_1$start$_2$CASING2$_3$ | 1.0 | X0.mindist:=min(X2.dist,X4.mindist) (sDistNear(X0.mindist)) |
| CASING2 → PERSTOP$_0$start$_1$CASING2$_2$ | 1.0 | X0.mindist := min(X1.dist,X3.mindist) |
| CASING2 → PERSTOP$_0$start$_1$ | 1.0 | X0.mindist := X1.dist |
| PERSTOP → stop$_0$ | 1.0 | X0.mindist := X1.dist |
| PERSTOP → disappear$_0$perapp$_N$start$_2$stop$_3$ | 0.9 | X0.mindist := X1.dist (TimeNear(X2.timestamp,X1.timestamp), Near(X2.loc,X1.loc)) |

Table 4.1: Attribute grammar for casing vehicles in a parking lot.

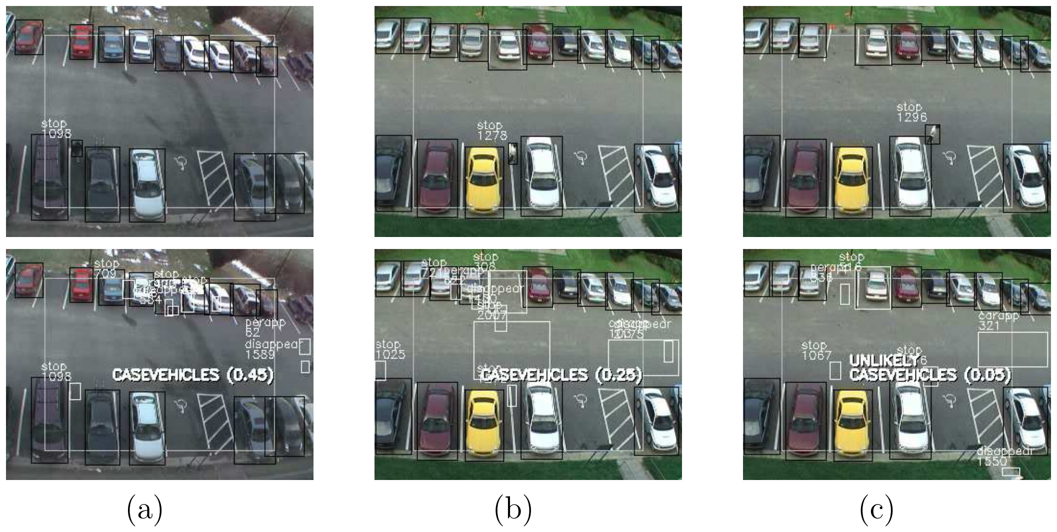


Figure 4.3: Recognition results for casing vehicles in the parking lot. (a) A person walks in, cases vehicles, and walks away. (b) A person drives in, cases vehicles, and drives away. (c) A person drives in, stops twice, and walks into the building.

high likelihood. In particular, in sequence (a) the tracker briefly lost track of the person but using the error correcting grammar, the parser correctly recognized the entire sequence as a single event. Sequence (c) was syntactically parsed as a casing event but with a low likelihood compared to sequence 1 and 2.

### 4.7.2 Detection of abnormal events

In this experiment, we demonstrate the application of detecting abnormal events as well as recognizing normal events in a parking lot adjacent to a building. The primitive events used are *carapp*, *perapp*, *disappear*, *carstop*, *carstart*, and *carstat*. *carstat* denotes the instance where a previously moving car has been stationary for a long time.

Table 4.2 shows the attribute grammar for typical events that happen in a parking lot. The first nonterminal symbol derived from the start symbol *PARK*-

*INGLOT* represents a specific event. The events described by the grammar are:

- *PARKING*: A person parks the car and enters the building.
- *PARKOUT*: A person exits the building, gets in a parked car, and drives away.
- *DROPOFF*: A car enters the lot, drops off a person, who enters the building.
- *PICKUP*: A person exits the building, gets in a car, which leaves the lot.
- *WALKTHRU*: A person moves through the lot.
- *CARTHRU*: A car moves moves through the lot.

Again, the grammar describes not only the key event such as a person getting out of the car, but also the context of the event. For example, when a person parks a car in the lot, he or she is expected to enter the building, assuming the lot is only for people who have a business in the building. By taking the entire context of the event, the class of abnormal events that can be detected are expanded. For instance, the two individual events of parking a car and walking out of the lot without entering the building are not considered abnormal, but the combination of the two makes it abnormal.

The contextual objects *PkSpace1* and *PkSpace2* represent the parking areas in the scene, and *BldgEnt* indicates the location of the building entrance. Soft predicates are used where the attributes are expected to vary among the individuals who perform the event. The soft predicates $sNearPt()$ and $sFar()$ are defined as Gaussian functions on the distance between two points. $sInside(loc, area)$ is defined

| Grammar productions | Attribute rules and Semantic conditions |
|---|---|
| PARKINGLOT → PARKING$_N$ $\mid$ PARKOUT$_N$ $\mid$ DROPOFF$_N$ | |
| PARKINGLOT → PICKUP$_N$ $\mid$ WALKTHRU$_N$ $\mid$ CARTHRU$_N$ | |
| PARKING → CARPARK$_0$perapp$_N$disappear$_2$carstat$_1$ | (Near(X2.loc,X1.loc), sNearPt(X3.loc, BldgEnt)) |
| PARKING → CARPARK$_0$perapp$_N$carstat$_1$disappear$_2$ | (Near(X2.loc,X1.loc), sNearPt(X4.loc, BldgEnt)) |
| CARPARK → carapp$_0$carstart$_1$carstop$_1$ | X0.loc := X3.loc (NotInside(X1.loc,Fov), sInside(X3.loc, PkSpace1, PkSpace2)) |
| CARSTOP → carstop$_0$carstart$_1$CARSTOP$_1$ | X0.loc := X3.loc |
| CARSTOP → carstop$_0$ | X0.loc := X1.loc |
| PARKOUT → perapp$_0$disappear$_1$carapp$_N$CARSTART$_3$disappear$_3$ | (sNearPt(X1.loc,BldgEnt),Near(X3.loc,X2.loc), NotInside(X5.loc,Fov)) |
| CARSTART → carstart$_0$carstop$_1$CARSTART$_1$ | X0.loc := X1.loc |
| CARSTART → carstart$_0$carstop$_1$ | X0.loc := X1.loc |
| CARSTART → carstart$_0$ | X0.loc := X1.loc |
| DROPOFF → CARSTAND$_0$perapp$_N$disappear$_2$CARSTART$_1$ | (Near(X2.loc,X1.loc), sNearPt(X3.loc,BldgEnt)) |
| DROPOFF → CARSTAND$_0$perapp$_N$CARSTART$_1$disappear$_2$ | (Near(X2.loc,X1.loc), sNearPt(X4.loc,BldgEnt)) |
| CARSTAND → carapp$_0$carstart$_1$CARSTOP$_1$ | X0.loc := X3.loc (NotInside(X1.loc,Fov)) |
| PICKUP → perapp$_0$disappear$_1$CARSTART$_N$disappear$_3$ | (sNearPt(X1.loc, BldgEnt), Near(X3.loc,X2.loc), NotInside(X4.loc,Fov)) |
| WALKTHRU → perapp$_0$disappear$_1$ | (NotInside(X1.loc,Fov), NotInside(X2.loc,Fov), sFar(X2.loc,X1.loc)) |
| CARTHRU → carapp$_0$CARSTART$_1$disappear$_1$ | |

Table 4.2: Attribute grammar for normal events in a parking lot.

by the “probit” distribution which is defined as

$$\Phi(d) = \int_{-\infty}^{d} \mathcal{N}(x; 0, \sigma)dx \tag{4.8}$$

where $\mathcal{N}(x; 0, \sigma)$ denotes the Gaussian distribution, $\sigma$ is a softness parameter, and $d$ is the distance from *loc* to the nearest boundary of a set of areas. This can be viewed as a hard threshold that has been perturbed by Gaussian noise.

Figure 4.4 shows recognition results for normal events in a parking lot. The video include three events *PARKING*, *WALKTHRU*, and *PARKOUT*, which overlaps with one another. The system correctly recognized the three events in the order that each event has terminated. Note that in Figure 4.4(b) two persons exit from the car after it was parked. In this case the parser recognized two different *PARKING* events, one for each person. However only one instance is shown for readability. Recognition of *DROPOFF*, and *PICKUP* events are shown in Figure 4.5.

Figure 4.6 gives examples of detecting abnormal events. Each row corresponds to a different abnormal event. The first row represents an illegal parking event. The event is parsed as a *PARKING* event but since the location of the parked car is away from the parking area, a low probability is given and is labeled as abnormal. In the second event the person exits through a location that is far from the (expected) building entrance. The third event involves a vehicle that is parked but no person appears from the vehicle for a long time. Since the *carstat* symbol is not anticipated by any pending event thread, it causes a detection of a syntactic abnormality.

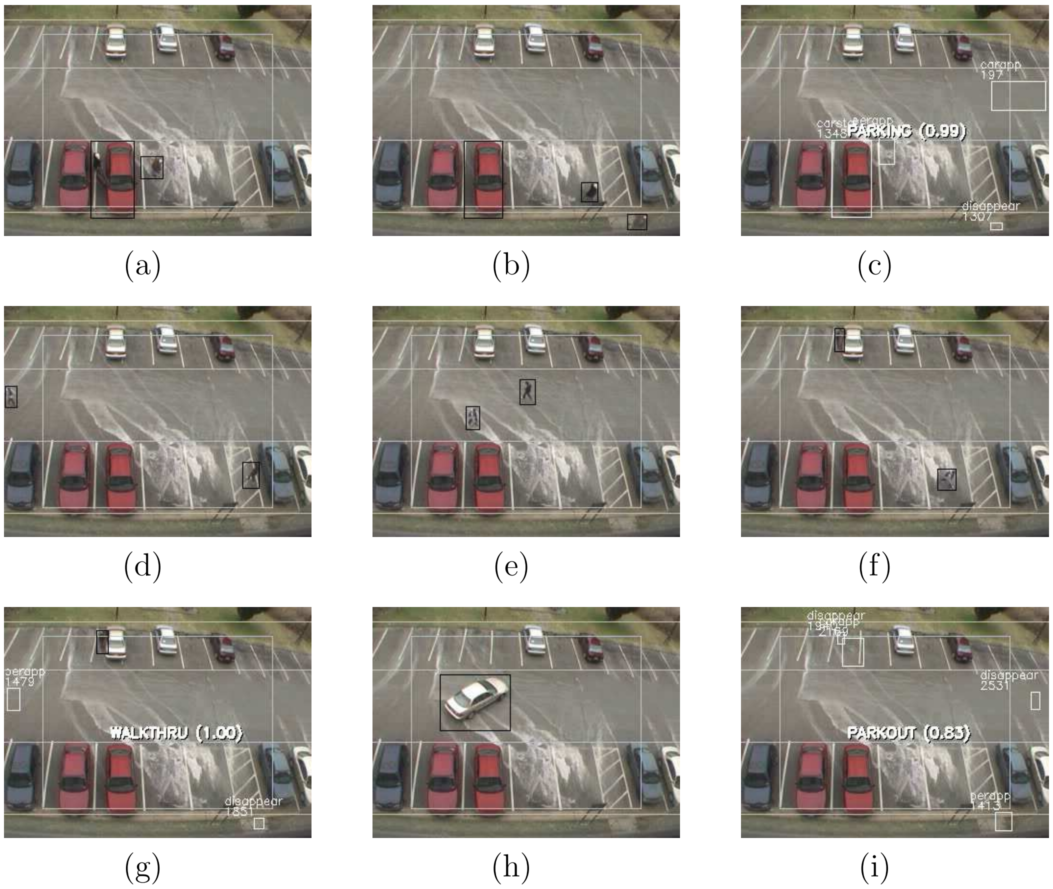


Figure 4.4: Normal events in the parking lot.

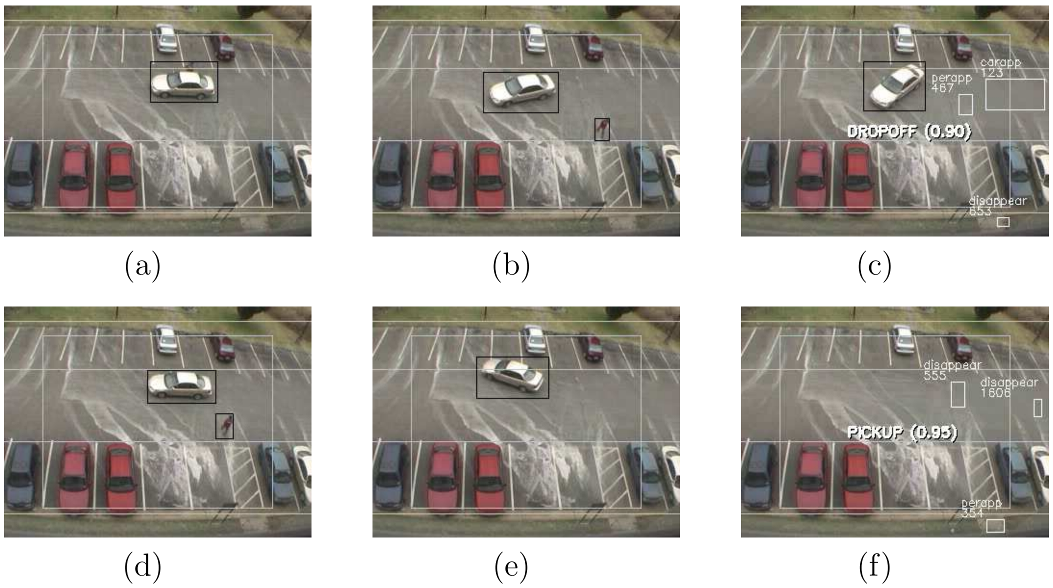


Figure 4.5: Normal events in the parking lot. (a)–(c): Dropping off a person. (d)–(f): Picking up a person.

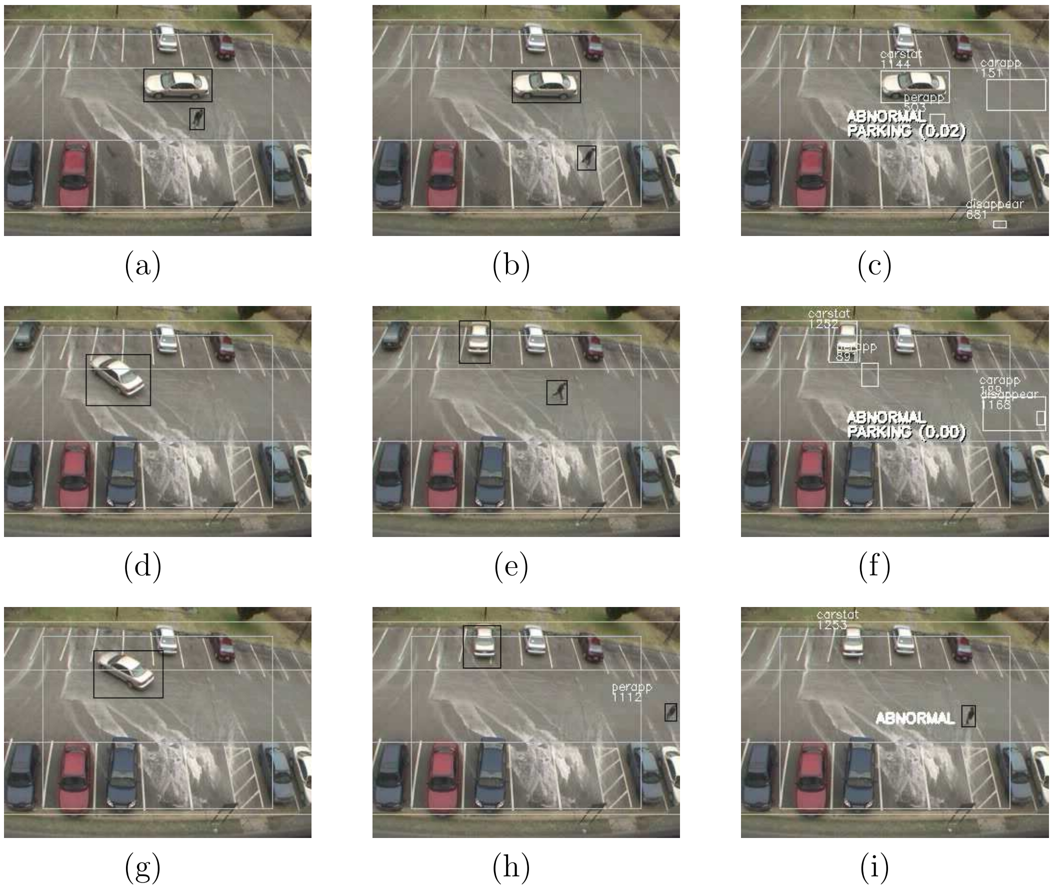


Figure 4.6: Abnormal events in the parking lot. (a)–(c): Illegal parking. (d)–(f): Parking and not entering the building. (g)–(i): Parking and not exiting the vehicle for a long time.

## 4.8 Discussion

We have presented a surveillance system for recognizing specific events and detecting anomalies in a video using attribute grammars. The system is robust to tracking errors and also handles the variations on the execution of events by the production weights in the grammar and by allowing uncertainty in the attributes. Specific events critical to security are detected by directly modeling the event. Alteratively, when it is possible to model most of the events expected to occur in a scene, anomalies may be defined as all events that does not fit the model to some degree of certainty. This method may be applied to areas where strict protocols must be followed such as a high security zone.

# Chapter 5

# Activity Discovery by Finding Salient Groups of Trajectories

## 5.1 Introduction

It is often desirable to automatically learn various human activities given a video containing many such examples. Unsupervised methods for learning activities have been largely based on clustering trajectories of features [55, 57, 27]. The assumption behind the clustering approach is that repeating patterns of similar trajectories is an indicator of a meaningful activity. These methods use partitional clustering strategies assuming that each trajectory can be classified into a specific category. However, this assumption is not always valid when the trajectories are influenced by not only the structures in the scene but also by various other intentions of the human. For example in a relatively open space such as parking lot, a person often does not choose to walk along a common path but move in a seemingly arbitrary path for some reason e.g., getting into a car or meeting someone. Therefore blindly classifying all the trajectories may result in clusters that do not accurately represent the characteristics of a truly repetitive activity.

We propose a method for finding salient groups of trajectories, inspired by prior work on perceptual organization in computer vision [41, 15]. Our method uses the probability of trajectories accidentally forming a group as the measure of significance of the group, assuming a model of random background trajectories.

The grouping algorithm finds maximally non-accidental groups and automatically determines the threshold for significance. To the best of our knowledge, this principle of non-accidentalness has not been exploited in human activity analysis.

The following section reviews some prior work related to this chapter. In Section 5.3 we describe our approach and present a generic algorithm for discovering significant clusters based on the non-accidentalness principle. An example of finding significant clusters of 2-$D$ points is presented in Section 5.4. In Section 5.5, the problem of discovering human activities by grouping trajectories is addressed. Experimental results from real and synthetic data are given in Section 5.6, followed by some discussions in Section 5.7.

## 5.2 Related work

We list some of the prior work where unsupervised clustering methods have been applied using spatial motion trajectory as the feature. In [34], trajectories expressed as motion vector sequences are clustered by a self-organizing neural network with leaky neurons. Makris and Ellis [42] described a method for learning models of scene structures from observed trajectories. Entry, exit, and stop zones modeled by a mixture of Gaussians are learned by Expectation-Maximization. Routes are represented by a spline-like main axis with varying widths, which are incrementally fitted to the set of trajectories using a heuristic merging algorithm. A similar goal was pursued in [60], where trajectories are classified into vehicles and pedestrians by clustering the size, taking the perspective effect into consideration. Each trajectory class is further clustered using the spectral method with a combination of a variant of the Hausdorff distance and motion direction as the distance measure. Scene structures were modeled with densities of points and motion directions from each cluster. In [48] feature trajectories such as sequences of coordinates and orientations were represented by Hidden Markov Models (HMM). Spectral clustering was applied to the affinity matrix where the distance measure was based on the fitness of a trajectory to other HMM's. A method for estimating the number of clusters was also described. Hu et al. [27] used fuzzy $k$-means to cluster the trajectories in two stages. Spatial clustering was applied to vectors of sampled points, then each cluster was further clustered according to its temporal characteristics. The clusters were modeled with a series of Gaussians, which allowed incremental anomaly detection and behavior prediction.

The notion of randomness against which significance is tested has been used in *cluster validity* analysis where the results of clustering are evaluated in an objective fashion. In cluster validation, some random distribution is assumed as the null hypothesis regarding the data or the distances between the data samples. A *statistic* evaluated from a clustering result is tested against the null hypothesis, where the rejection of the null hypothesis indicates a valid clustering. Jain and Dubes [31] discussed in detail various indices used for this statistical test, focusing mainly on ordinal data. A closely related problem is *clustering tendency* [31] which is the problem of deciding whether the data are clustered rather than purely random.

The principle of non-accidentalness for perceptual organization was advocated by Witkin and Tenenbaum [61] and adopted by Lowe [41]. Perceptual organization refers to a process of grouping image features into structures without prior knowledge of the contents. This process closely corresponds to the grouping phenomena in human vision studied by the Gestalt psychologists [41]. Desolneux et al. [15] used the probability of the number of accidentally aligned edge features as a measure of "meaningfulness" of the group (extended line segment). They proposed to impose a threshold on the expected number of "false alarm"s i.e., the groups that are accidentally aligned in the image. The authors later demonstrated applications of their framework for several different problems [16].

In the context of model-based object recognition, Grimson and Huttenlocher [23] developed a method for setting the threshold on the number of image features that consistently match the model features. They assumed a uniformly distributed (similarity) transformation of features and derived the probability that a given number of features coincide (fall inside a regularly spaced buckets) in the transformation space by chance. A similar approach was described in the work by Cao et al. [5] where the framework in [15] was applied to a general single-link agglomerative clustering scheme. In contrast to Grimson and Huttenlocher [23], this method considers regions of different size in the transformation space. Also the distribution of the random "background" transformation is empirically estimated.

## 5.3 Approach

Our goal is to identify groups of trajectories that correspond to activities without any prior knowledge about the activities except for a very general assumption on the nature of trajectories. Since no statistical model of an activity is present, classical decision theory is not effective. This is analogous to the background subtraction approach for detecting foreground objects, where only the model of the background is used for the classification of foreground vs. background.

The problem of evaluating the significance of a group of general features is cast as a statistical test of hypothesis. Let $G$ be a given set of $k$ features selected from $n$ available features (e.g., points in a multi-dimensional space) and $R_G$ be a parameterized geometric region of a given class (e.g., hyper-sphere) that tightly encloses $G$. The null hypothesis $H_0$ is that the features are independent and identically distributed according to the background model (e.g., uniform distribution). The test statistic $x$ is the number of features that are within $R_G$ and its probability mass function is given by the binomial distribution

$$b(x; n, p(R_G)) = \binom{n}{x} p(R_G)^x (1 - p(R_G))^{n-x} \tag{5.1}$$

where $p(R_G)$ is the probability that a feature lies within $R_G$ under $H_0$. Let $P_a(G)$ be the probability of accidentally having $|G|$ (the cardinality of $G$) or more features in $R_G$, which is given by the binomial tail

$$P_a(G) = \sum_{x=|G|}^{n} b(x; n, p(R_G)) \tag{5.2}$$

Then $H_0$ can be rejected at the confidence level $1-\alpha$ if the $P_a(G)$ is less than $\alpha$. The value of $\alpha$ is chosen such that the expected number of false positives is $N_{fp}$, which may be fixed at $N_{fp} = 1$. Since the number of false positives is given by $N_R P_a(G)$, where $N_R$ is the number of possible regions over the entire feature set, the threshold is set as $\alpha = N_{fp}/N_R$. Rather than assuming a fixed set of possible regions (e.g., based on regular grids) and thus the value of $N_R$ *a priori* as in [15, 5], we propose to estimate $N_R$ depending on the total number of features $n$. $N_R$ is estimated by the expected number of unique geometric regions that can be defined by all possible subsets of $n$ features assuming the random background model. It should be noted that the choice of the threshold $N_{fp}/N_R$ is not sensitive to the successful detection of salient groups. In fact, it is shown that the threshold on the number of features $k$ in $G$ asymptotically depends on the logarithm of $N_{fp}$ and $N_R$ [15].

The remaining task is to search for groupings that are significant according to the previous criterion. As there might be multiple nested subsets whose accidental probabilities are all below $\alpha$, we choose to find the grouping that have minimal probability among them. Exhaustive search over all $2^n$ possible groupings of $n$ features is clearly not feasible. We employ an agglomerative hierarchical clustering approach where the criterion function is the sum of accidental probabilities of each cluster. The algorithm starts by initializing the individual features $\mathbf{x}_i$ as singleton clusters. At each iteration, a pair of clusters that satisfies the following condition is found and merged: Among all pairs $G_i$, $G_j$ whose individual probabilities both decrease after merging, i.e. , $P_a(G_i \cup G_j) < min(P_a(G_i), P_a(G_j))$, the value of $P_a(G_i \cup G_j)$ is the minimum. ($P_a(\mathbf{x}_i)$ is defined as a constant greater than 1.) The detailed procedure is described in Algorithm 1. Agglomerative clustering has also been used in [5] but the space in which groupings were searched was restricted to subgraphs of the minimum spanning tree derived from an *ad-hoc* distance measure.

**Algorithm 1** Generic salient cluster detection algorithm based on the non-accidentalness principle

```
$\mathcal{G} \leftarrow \{G_i = \mathbf{x}_i\}, i = 1, \ldots, n$
**while** $|\mathcal{G}| \geq 1$ **do**
  find a pair of clusters $G_{i^*}$, $G_{j^*}$ from $\mathcal{G}$ such that $(i^*, j^*) = \operatorname*{argmin}_{i<j} P_a(G_i \cup G_j)$
  subject to $P_a(G_i \cup G_j) < min(P_a(G_i), P_a(G_j))$
  **if** $(i^*, j^*)$ exists **then**
    $\mathcal{G} \leftarrow \mathcal{G} - \{G_{i^*}, G_{j^*}\} \cup \{G_{i^*} \cup G_{j^*}\}$
  **else**
    stop
  **end if**
**end while**
output $\{G_i | G_i \in \mathcal{G}, P_a(G_i) < 1/N_R\}$
```

## 5.4 An illustrative example: 2-$D$ points

As an illustration of our cluster detection algorithm, we give an example of clustering 2-$D$ points in the presence of background noise. The background distribution is assumed to be uniform inside the 2-$D$ image boundary. The class of geometric region is chosen to be a disc, parameterized by the center coordinates $(x_c, y_c)$ and the radius $r$. Given a set of points $G$, the region $R_G$ that tightly encloses $G$ can be defined as the smallest enclosing disc [12] (abbreviated as *SED*) of $G$. The probability of a point being inside $R_G$ is given by $P(R_G) = \pi r^2/A$ (ignoring the effects of the image boundary), where $A$ is the area of the image. However, in order to take into consideration the measurement error in the points, it is more reasonable to add some margin $r_\delta$ to $r$ while evaluating $P(R_G)$. We define $r_\delta$ as half the average of maximum margin that may be added without enclosing another point, which is approximated by the average nearest neighbor distance of a point given by $1/(2\sqrt{n/A})$ [10]

$$r_\delta \approx 1/(4\sqrt{n/A}) \tag{5.3}$$

For this example, $N_R$ is the number of unique *SED*s defined by subsets (with at least two points) of $n$ points. It is known that the *SED* of $G$ is determined by (assuming general position) either three points on the circle that form an acute triangle or two points that form the diameter of the circle on the boundary. Noting that a constrained set of three points uniquely defines a *SED* of *some* point set, and any pair of points similarly defines a unique *SED*, the expectation of $N_R$ may be expressed as

$$E[N_R] \approx \beta\binom{n}{3} + \binom{n}{2} \tag{5.4}$$

where $\beta$ is the average fraction of three-point subsets whose triangles formed by the points are acute. While it is possible to obtain $E[N_R]$ analytically assuming uniformly distributed points, it is enough for our purpose to estimate its upper bound. Assuming large $n$ ($n > 3/(1-\beta)+2$ to be exact) it can be shown that the expectation is bounded above by $\binom{n}{3}$ thus we use $N_R = \binom{n}{3}$. (The empirical value of $\beta$ was around 0.27 for a square image.)

Figure 5.1 shows the results of applying the clustering algorithm on synthetic data. The data was created by sampling $n/4$ points each from two Gaussian distributions and $n/2$ points uniformly distributed in $(0,1)^2$. Column (a) represents the input data, (b) shows the detected clusters, and (c) is the dendrogram showing all the (intermediate) clusters, where the height of the (sub)tree denotes the probability and the lightly colored horizontal line represent the computed threshold. The top

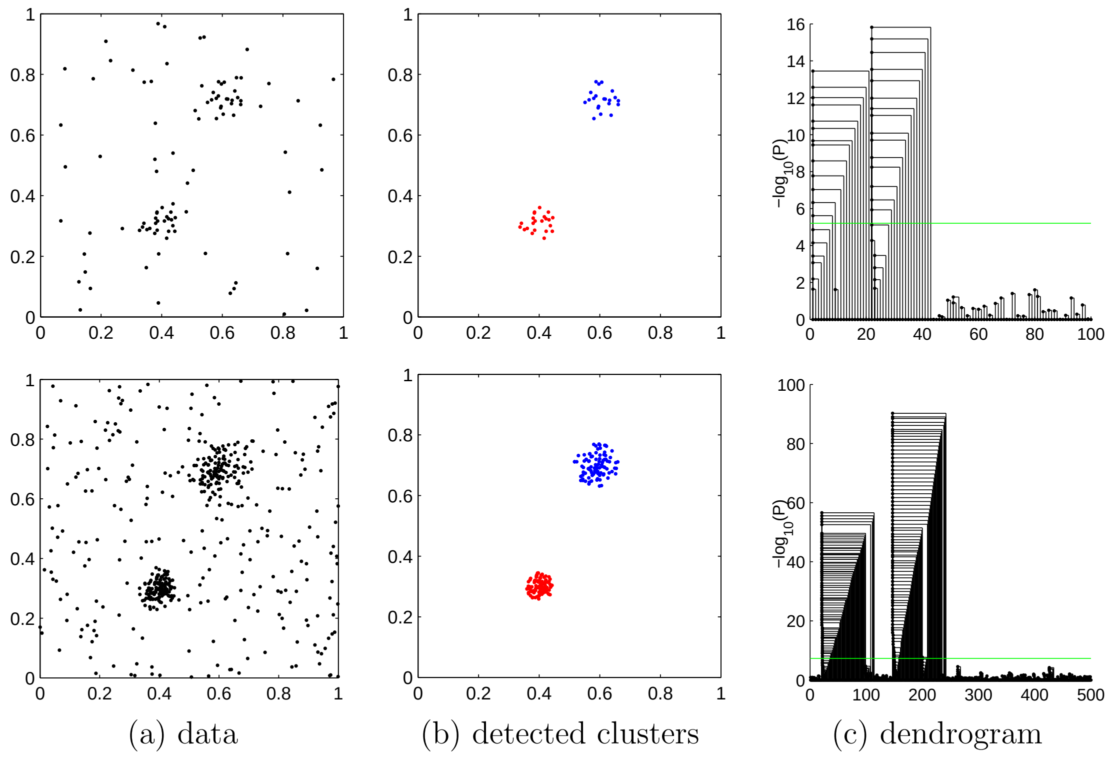


Figure 5.1: Clustering result for 2-$D$ point data.

and bottom rows respectively show the results from a data of $n = 100$ and $n = 500$ points. Both the results were visually reasonable and no false clusters were detected, which shows the threshold adapts well across wide range of densities.

## 5.5 Finding salient groups of trajectories

We now describe the main problem that is addressed in this chapter. Given a set of trajectories of individual humans in a fixed field of view, many of which are rather arbitrary and not well clustered, our goal is to find (if any) the salient clusters. The saliency of a group of trajectories is defined by the probability that the trajectories occur by chance within some region defined by the group. A group with a low accidental probability is regarded as a result of a common underlying cause and thus is likely to correspond to a semantically meaningful activity. The following subsections describes our representation of trajectories and the distribution of background trajectories. The last subsection derives expressions for the accidental probability $P_a(G)$ that a group of trajectories ($G$) is formed, and the number of regions ($N_R$) enclosing subsets of $n$ trajectories. Using these results, Algorithm 1 is applied to find the salient clusters.

### 5.5.1 Representation for trajectories

Representations for trajectories used in the literature include sequence of points sampled over regular time intervals [27, 48], sequence of motion vectors [34], and set of points [60]. We choose to represent each trajectory as a sequence of a constant number of points such that they form line segments of equal length. This is denoted by a list of $s$ points

$$\mathbf{t} = \langle (x_0, y_0), \ldots, (x_{s-1}, y_{s-1}) \rangle$$

with the implicit constraint $\|\overrightarrow{(x_{j-1}, y_{j-1})(x_j, y_j)}\| = l, (j = 1, \ldots, s-1)$, or equivalently by the initial point, segment length, and the orientations of the segments

$$\mathbf{t} = \langle (x_0, y_0), l, \theta_0, \phi_1 \ldots, \phi_{s-2} \rangle$$

where $l$ is the segment length, $\theta_0$ is the direction of the first segment, and $\phi_j$'s denote the turning angles of subsequent segments, defined by

$$\phi_j = \angle(\overrightarrow{(x_{j-1}, y_{j-1})(x_j, y_j)}, \overrightarrow{(x_j, y_j)(x_{j+1}, y_{j+1})}) \tag{5.5}$$

where $\angle(\vec{u}, \vec{v}) = arccos(\vec{u} \cdot \vec{v} / \|\vec{u}\| \|\vec{v}\|)$. This representation approximates the shape of the trajectory and also expresses the direction. (The term "shape" here refers to a curve with constant velocity and does not refer to the representation that is invariant under similarity transform.) Other features such as speed and object size are thought to contribute less to the distinctiveness of an activity. Limiting the dimensionality of the feature is also necessary for correctly estimating the background distribution from limited amount of data.

The points can be extracted from the constant velocity curve derived from the tracking data. A simple way is to measure the entire length $L$ of the curve by accumulating the magnitude of the motion vectors and sample points at an interval of length $L/(s-1)$ along the curve. The lengths of the line segments obtained by this method will slightly vary depending on the local curvature of the trajectory but the error is found to be negligible when a reasonable number of points is used.

### 5.5.2 Background model of trajectories

It is desirable to impose minimal *a priori* constraint on the distribution of background trajectories so that it represents a wide variety of trajectories. Accordingly, each trajectory is modeled by the following simple generative model: Randomly select an initial point $(x_0, y_0)$ and the segment length $l$, then randomly choose the direction of segments $\theta_0$ and $\phi_j$ sequentially. $(x_0, y_0)$ is modeled as uniform over some region of the image representing the entry and exit regions. It is difficult to precisely define such a region without prior knowledge of the scene. However, it is sufficient for our purposes to use only its approximate area, which is estimated by considering regions formed by the trajectory end points from a training set of typical trajectories. $l$ is assumed uniform across an interval which is also estimated from the training data. The distribution of $\theta_0$ is modeled as uniform over the possible range given $(x_0, y_0)$ and $l$ i.e., over the range that keeps the segment within the image boundary. Each subsequent turning angle $\phi_j$ is modeled as having a common uniform distribution whose support is estimated from a set turning angles from all segments in the training data. The random variables $(x_0, y_0)$, $l$, and $\phi_j$ are assumed independent.

### 5.5.3 Region defined by a trajectory group

Given a set of trajectories $G = \{\mathbf{t}^1, \ldots, \mathbf{t}^k\}$, we define its "enclosing region" $R_G$ as the volume in the space of variables representing the trajectories. In the space of $(x_0, y_0)$, the region is defined as the smallest enclosing disc parameterized by $(x_c, y_c)$ and $r$ as in Section 5.4. The range of $l$ is defined by $[l^a, l^b] = [min_i\{l^i\}, max_i\{l^i\}]$. The remaining variables $\theta_0$, $\phi_1 \ldots, \phi_{s-2}$ are independent, hence the corresponding volume is defined by a hyper-rectangle having orthogonal ranges in each dimension.

To estimate the ranges of $\theta_0$ and $\phi_j$, we use the mean trajectory of $G$ given by

$$\bar{\mathbf{t}} = \langle(\bar{x}_0, \bar{y}_0), \ldots, (\bar{x}_{s-1}, \bar{y}_{s-1})\rangle \tag{5.6}$$

where $\bar{x}_j = \Sigma_{i=1}^{k} x_j^i / k$, which is viewed as a representative trajectory of $G$. The range of $\theta_0$ is approximated by angles from the mean initial point to all the 2nd points in $G$

$$[\theta_0^a, \theta_0^b] = [\min_{i=1,\ldots,k}\{\arctan(x_1^i - x_0, y_1^i - y_0)\}, \max_{i=1,\ldots,k}\{\arctan(x_1^i - x_0, y_1^i - y_0)\}] \tag{5.7}$$

where the circularity of angles are correctly handled. The range $[\phi_j^a, \phi_j^b]$ is defined by the deviation of the angles from a point in $\bar{\mathbf{t}}$ to all the immediately following points in $G$

$$[\phi_j^a, \phi_j^b] = [\min_{i=1,\ldots,k}\{\Delta\phi_j^i\}, \max_{i=1,\ldots,k}\{\Delta\phi_j^i\}] \tag{5.8}$$

where $\Delta\phi_j^i = \angle(\overrightarrow{(\bar{x}_{j-1}, \bar{y}_{j-1})(\bar{x}_j, \bar{y}_j^i)}, \overrightarrow{(\bar{x}_{j-1}, \bar{y}_{j-1})(x_j^i, y_j^i)})$.

Using the defined volume of $R_G$, the probability of a trajectory being inside $R_G$ is given by

$$P(R_G) = \frac{\pi r^2}{R_0} \cdot \frac{l^b - l^a}{R_l} \cdot \frac{\theta_0^b - \theta_0^a}{R_\theta(\bar{x}_0, \bar{y}_0, \bar{l})} \prod_{j=1}^{s-2} \frac{\phi_j^b - \phi_j^a}{R_\phi} \tag{5.9}$$

where $R_0$, $R_l$, and $R_\phi$ are the area (length) of support of the respective distribution estimated from the training data, and $R_\theta(\bar{x}_0, \bar{y}_0, \bar{l})$ is the length of the possible range of $\theta_0$ assuming that the initial point is $(\bar{x}_0, \bar{y}_0)$ with segment length $\bar{l} = \|(\bar{x}_1 - \bar{x}_0, \bar{y}_1 -$

$\bar{y}_0)\|$. Note that appropriate margins are added to the volume as done in Section 5.4 but are not included in (5.6)–(5.8) for conciseness. The margin for $r$ is the same as in Section 5.4. Using similar arguments (average neighbor distance) the margins for the other parameters are given by

$$\delta_x = \frac{R_x}{4n} \tag{5.10}$$

where $x$ represents $l$, $\phi$, or $\theta$.

The upper bound on the number of possible regions $N_R$ defined by $n$ trajectories is obtained by separately considering (a) the number of enclosing discs for the initial points (as shown in Section 5.4), (b) the number of tight segment length intervals given by $\binom{n}{2}$, and (c) the number of $(s-1)$-dimensional hyper-rectangles enclosing the $s-1$ angles. The upper bound on the number of unique hyper-rectangles tightly enclosing $n$ $d$-dimensional points can be shown to be $\binom{n}{2d}$ by reasoning that the hyper-rectangle can have at most $2d$ points on the boundary assuming general position. An upper bound of $N_R$ can be obtained by the product of the three components (a)–(c)

$$N_R \lesssim \binom{n}{3}\binom{n}{2}\binom{n}{2(s-1)} \tag{5.11}$$

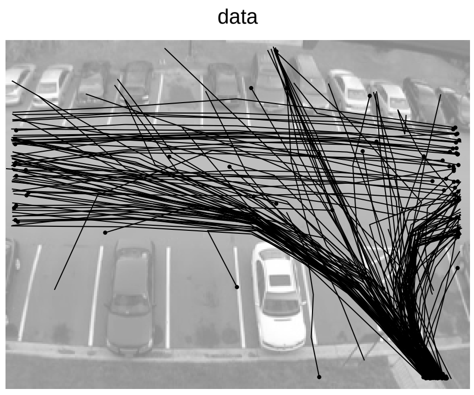
This ignores the dependencies between the three components, which results in over-estimating $N_R$ by a negligible constant factor.

## 5.6 Experimental results

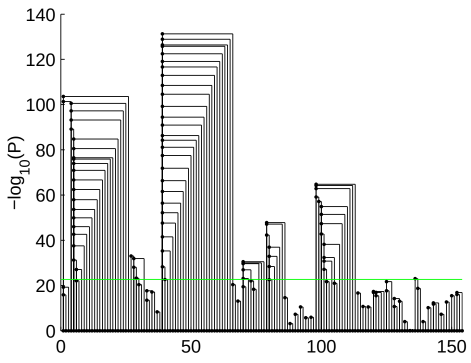
### 5.6.1 Results from real surveillance videos

We validated our method using videos that contain a scene of a parking lot adjacent to a building. Two videos, each lasting 90 minutes, were collected from different days. Humans were tracked from the two videos each resulting in $n = 154$ and $n = 159$ trajectories. The training data for estimating the parameters of the background model was selected from the first tracking data by removing a few unusual trajectories such as a person turning back. The number of sampled points was chosen to be $s = 5$.

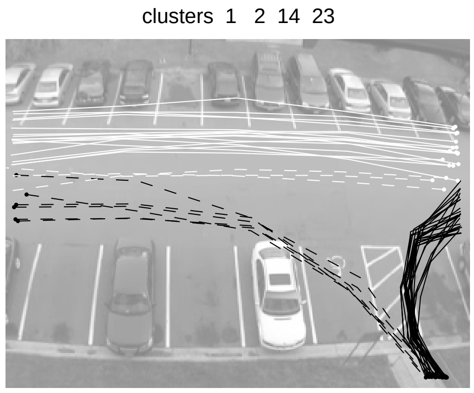
Figures 5.2, and 5.3 show the results from each dataset. The salient clusters are shown in sub-figures (c) and (d). Clusters within each sub-figure are labeled as black, dashed black, white, and dashed white, corresponding to the increasingly ordered cluster indices. The dot in one end of the trajectory indicates the initial point. The scene includes the building entrance/exit in the lower right corner, a walkway leading to a trail in the top, and two open spaces on the left and right leading to roads. Each cluster corresponded to natural activities such as exiting from the building and turning towards a road, or walking straight towards the walkway, etc. Other activities such as a person walking out of or into a car are not grouped into a cluster. Although this kind of activity might be meaningful to the human observer, it could not be detected by the algorithm since the trajectories were not significantly near each other. One obvious error was found in cluster 3 in Figure 5.3 where trajectories of people coming from the left and people coming



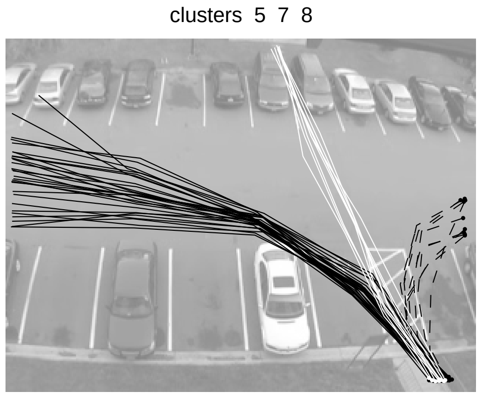
(a) data



(b) dendrogram

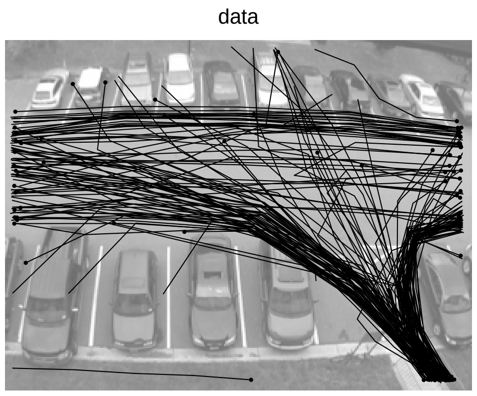


(c) salient clusters

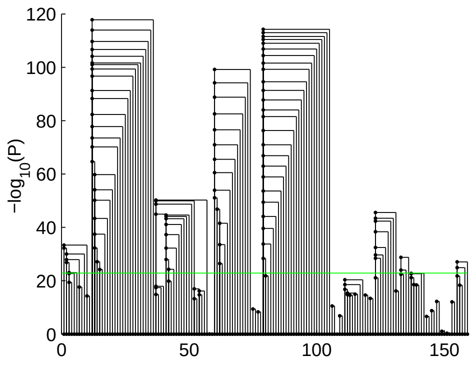


(d) salient clusters

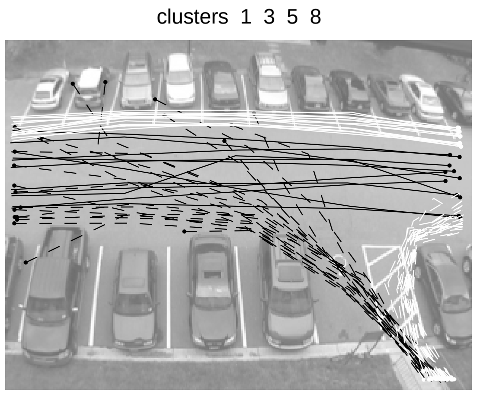
Figure 5.2: Salient clusters detected in the parking lot dataset 1.



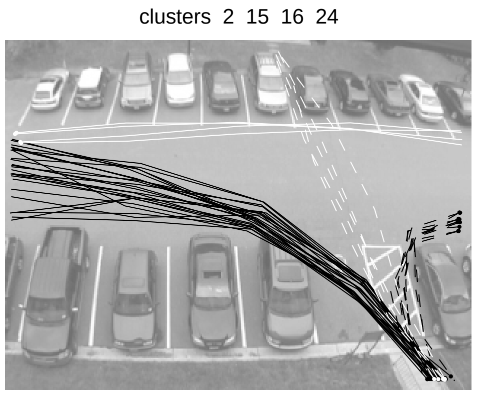
(a) data



(b) dendrogram



(c) salient clusters



(d) salient clusters

Figure 5.3: Salient clusters detected in the parking lot dataset 2.

out of nearby cars were mixed together due to the proximity of the initial points. More or less the same set of activities were discovered from both datasets except that in Figure 5.2, no cluster was found corresponding to cluster 16 in Figure 5.3 (straight trajectories from left to right). In fact, one such cluster *was* found from the first dataset by the algorithm but the probability was above the threshold since only three trajectories were in the cluster. It is also observed that the activity representing straight trajectories from right to left is detected as two clusters in both results, as seen by clusters 14, 23 in Figure 5.2 and clusters 1, 5 in Figure 5.3. This is because our method employs the spatial proximity as the grouping criterion, which is not effective for groups that are spatially spread out.

For comparison of the results, we applied a spectral clustering and two standard hierarchical clustering methods on the same datasets and features. The affinity matrix for spectral clustering was constructed from the Euclidean distances between trajectories. Then the top $k-1$ eigenvectors of the normalized Laplacian matrix were used to map the features into $(k-1)$-dimensional space [62], where $k$ was determined by visually inspecting the ranked eigenvalues. Finally, 20 repetitions of $k$-means clustering was applied to this new data with random initialization, and the best clustering result was selected using the following criterion:

- Define the "goodness" of a cluster $G$ as $f(G) = \sqrt{\mathrm{var}(G)}/|G|$, where $\mathrm{var}(G)$ is the average eigenvalue of the covariance of the $2s$-dimensional vectors representing trajectories in $G$.

- For each $k$-means result, choose the $c$ best clusters that have the lowest $f(G)$

values, where $c$ is the number of clusters determined by our algorithm.

- Select the best $k$-means result that has the least sum of the $c$ best $f(G)$

This process reduces the effect of the background trajectories by trying to assign $k - c$ clusters to them. Note that the number of salient clusters ($c$) were chosen to be the same as our algorithm for comparison, but in general, it is difficult to choose $c$ in spectral clustering.

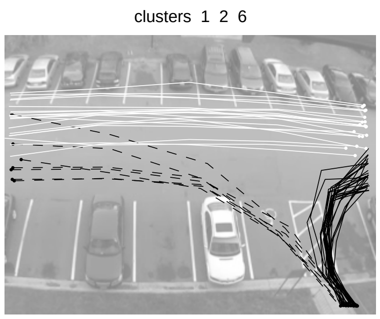
For hierarchical clustering, the single-linkage and the complete-linkage algorithms [17] were applied using Euclidean distance. Instead of choosing a stopping criterion, a wide range of threshold was applied by considering all the intermediate results from every iteration.

To quantitatively analyze the results, ground truth clusters for both datasets were manually labeled by subjectively judging the activity that each trajectory belongs to. For example, if a person appeared to enter a car, the corresponding trajectory was not labeled as "walking towards a path" even if it ended near the path. The resulting number of clusters was 6 for dataset 1 and 7 for dataset 2. Figures 5.4 and 5.7 show the ground truth clusters for each dataset.

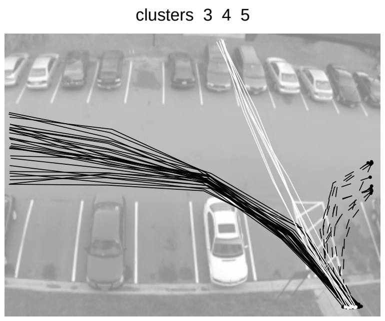
### 5.6.2 Performance measures

We define the following performance measures for quantitative analysis:

- False positive rate ($FPR$) and false negative rate ($FNR$), which are errors in the classification between foreground (the set of salient clusters) and background.

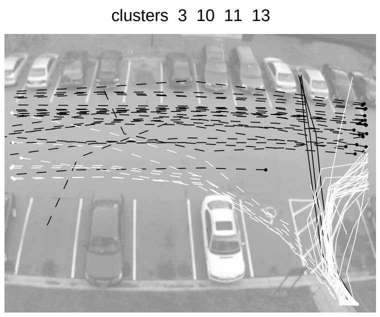


(a)

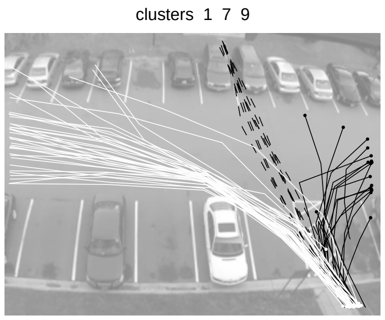


(b)

Figure 5.4: Ground truth clusters for the parking lot dataset 1.

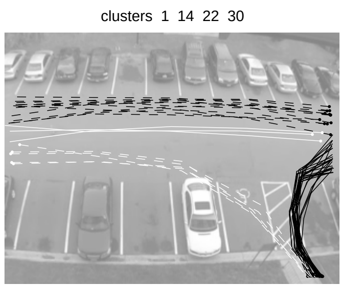


(a)

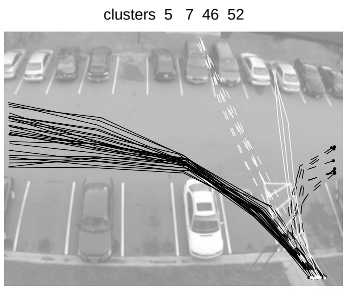


(b)

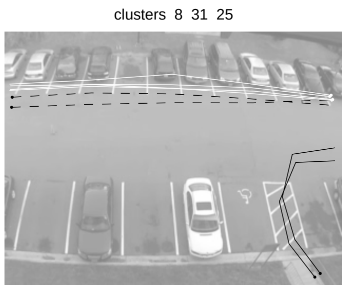
Figure 5.5: Spectral clustering result for the parking lot dataset 1.



(a)

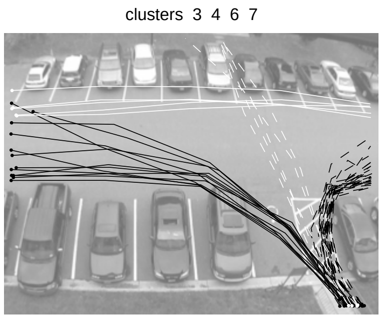


(b)

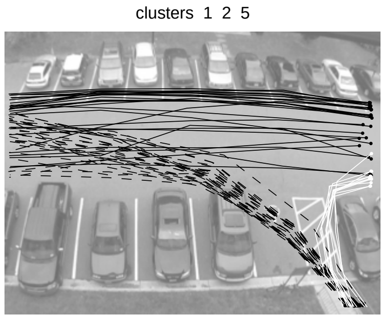


(c)

Figure 5.6: Single-link clustering result for the parking lot dataset 1.

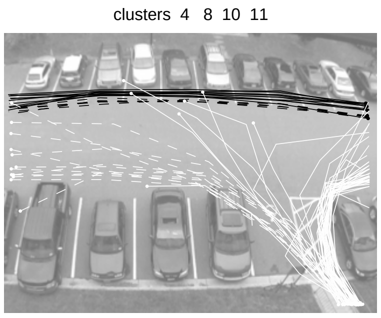


(a)

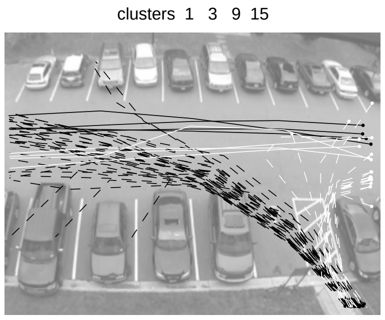


(b)

Figure 5.7: Ground truth clusters for the parking lot dataset 2.

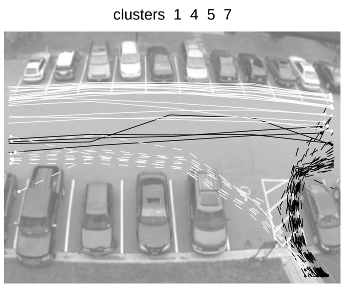


(a)

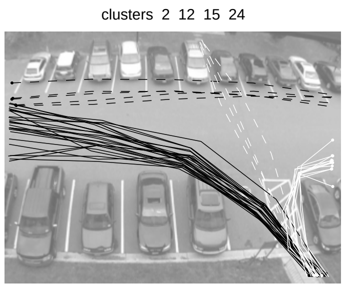


(b)

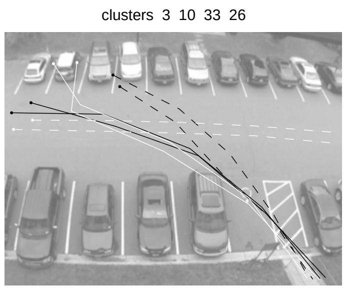
Figure 5.8: Spectral clustering result for the parking lot dataset 2.



(a)



(b)



(c)

Figure 5.9: Single-link clustering result for the parking lot dataset 2.

- Number of split clusters ($NS$) and merged clusters ($NM$): A large $NS$ indicates that many sub-clusters are detected within a true cluster, whereas a high value of $NM$ means that each detected cluster tend to include multiple true clusters.

Let $G_0$ and $G_0^*$ respectively denote the set of background trajectories in the result of an algorithm and in the ground truth. Similarly, let $\mathcal{G} = \{G_1, \ldots, G_c\}$ and $\mathcal{G}^* = \{G_1^*, \ldots, G_{c^*}^*\}$ be the set of detected clusters and the set of ground truth clusters, respectively. The error rates are formally defined by

$$FPR = \frac{|\bigcup_{i=1}^{c} G_i - \bigcup_{i=1}^{c^*} G_i^*|}{|G_0^*|} \tag{5.12}$$

$$FNR = \frac{|\bigcup_{i=1}^{c^*} G_i^* - \bigcup_{i=1}^{c} G_i|}{|\bigcup_{i=1}^{c^*} G_i^*|} \tag{5.13}$$

$NS$ and $NM$ are defined by first associating the clusters in $\mathcal{G}$ with the ones in $\mathcal{G}^*$ then counting the multiplicity of the association from each cluster. The association is done by determining at each cluster $G$, which cluster in the other set overlaps the most with $G$. This association can be viewed as a directed bipartite graph where each node $i$ in $\mathcal{G}$ has at most one outgoing edge $(G_i, G_{j'})$ to $\mathcal{G}^*$ where $j$ is given by

$$j' = \underset{j}{\operatorname{argmax}}\{|G_i \cap G_j^*|\} \tag{5.14}$$

The outgoing edges from $\mathcal{G}^*$ are also defined symmetrically. The number of split (merge) is defined by counting the number of nodes in $\mathcal{G}^*$ ($\mathcal{G}$) that have in-degree of more than 1.

$$NS = \sum\nolimits_{j=1}^{c^*} min(deg^-(G_j^*), 1) \tag{5.15}$$

$$NM = \sum\nolimits_{i=1}^{c} min(deg^-(G_i), 1) \tag{5.16}$$

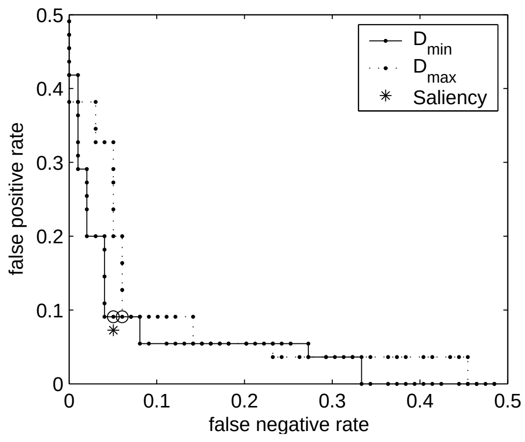
## 5.6.3 Quantitative results

The values of $FPR$, $FNR$, $NS$, and the total number of clusters found by each of the algorithms for both datasets are summarized in Tables 5.1 and 5.2.

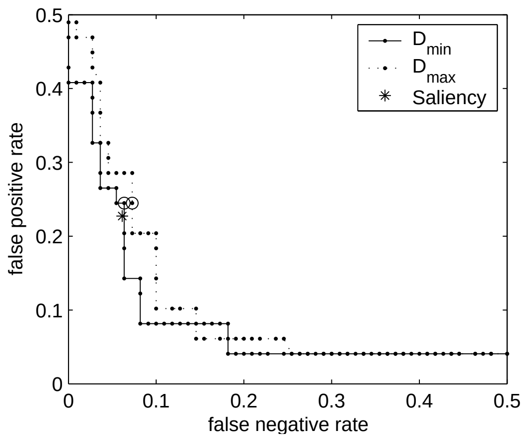
In our method, the false positives were caused by trajectories that are spatially similar to a cluster but does not represent the activity of the cluster e.g., a person appearing from a car which happened to be near the path along which many people walk. The high false positive rate in our method for the dataset 2 is mainly due to the previously mentioned error in cluster 3. The false negatives were trajectories in the true clusters that were unusually far from the center of the cluster.

For the spectral clustering method, the false positive rates were near 50% for both datasets. The false negative rate for dataset 2 was high because the method failed to detect two of the true clusters. $k = 13$ and $k = 15$ were respectively chosen for datasets 1 and 2 and the same number of clusters as our result was selected from the candidate clusters, as mentioned. The clustered trajectories are shown in Figures 5.5 and 5.8. As can be seen in Figure 5.8, one true cluster in dataset 2 was detected as 4 sub-clusters.

Results from the hierarchical clustering methods are shown in the ROC plots in Figure 5.10(a) and (b), each corresponding to dataset 1 and 2. In each plot, the solid



(a) dataset 1



(b) dataset 2

Figure 5.10: ROC plots for the hierarchical clustering methods.

line and the dotted line represent the results from the single-linkage (denoted $D_{min}$) and the complete-linkage (denoted $D_{max}$) algorithms, respectively. Our results, indicated by an asterisk ($*$) in each figure, had lower error rates compared with both hierarchical algorithms for dataset 1. For dataset 2, our method performed similar to some choice of parameter in the $D_{min}$ algorithm, in terms of classifying between foreground and background. Each column for $D_{min}$ and the $D_{max}$ method in Tables 5.1 and 5.2 is from the result that have the closest error rates as ours, which is indicated by a circle in each plot of Figure 5.10. Both hierarchical algorithms in both datasets produced much higher number of clusters at error rates similar to ours. (The number of clusters did not change much for other choices of parameters near the circle in the ROC plots.) Also, more activity groups were detected as split clusters compared to our results. Figures 5.6 and 5.9 respectively show the clustered trajectories corresponding to $D_{min}$ in Tables 5.1 and 5.2.

|  | saliency | spectral | single-link | complete-link |
|---|---|---|---|---|
| $FPR$ | 7.3% | 50.9% | 9.1% | 9.1% |
| $FNR$ | 5.0% | 0% | 5.1% | 6.1% |
| split groups | 1 | 0 | 2 | 4 |
| clusters found | 7 | 7 | 11 | 21 |

Table 5.1: Quantitative results from dataset 1.

|  | saliency | spectral | single-link | complete-link |
|---|---|---|---|---|
| $FPR$ | 22.7% | 49.1% | 24.5% | 24.5% |
| $FNR$ | 6.1% | 15.2% | 6.4% | 7.3% |
| split groups | 1 | 1 | 2 | 5 |
| clusters found | 8 | 8 | 12 | 22 |

Table 5.2: Quantitative results from dataset 2.

### 5.6.4 Results from synthetic data

For further quantitative evaluation of our method, experiments were performed using synthetic data. We used the six ground truth clusters for the first parking lot dataset as the reference for generating new datasets. Clusters of foreground trajectories were generated by randomly choosing a trajectory from a reference cluster and adding Gaussian perturbation noise along the normal of the trajectory at each point. The standard deviation of the noise given at each point was proportional to the mean of the nearest neighbor distances of points in the reference cluster. The number of trajectories in each cluster was assigned proportionally to those in the reference data. Each background trajectory was independently generated such that one endpoint is shared by one of the clusters using the following procedure: A reference trajectory is randomly chosen, then either its initial point $\bar{p}_0 = (x_0, y_0)$ or its final point $\bar{p}_s = (x_{s-1}, y_{s-1})$ is selected at random. Denote this point as $p_0$.

Using the perturbed $p_0$ (as done previously) as the initial point of the background trajectory, the segment length $l$ and the angles $\theta_0, \phi_1 \ldots, \phi_{s-2}$ are sampled from the background distribution to create a trajectory. Finally, the order of the points in this trajectory is reversed if $\bar{p}_{s-1}$ was selected.

The following sections describe the performance of our algorithm under varying characteristics of the dataset. The performance in each experiment was measured by averaging results from 100 synthetic datasets.

#### 5.6.4.1 The effect of the number of sampled points in a trajectory

We first experimented with the number of sampled points $s$ in the trajectory representation. Synthetic datasets were generated with $s$ varying from 4 to 8. The total number of trajectories was fixed at $n = 200$, 133 of which was clustered foreground trajectories. Figure 5.11 shows the performance of the results. Figure 5.12(a) plots the $FPR$ scaled by a factor of 10 and the $FNR$. $FPR$ values were much less than $FNR$ because the generated background 'noise' was more or less random compared to the real datasets. (All subsequent plots for $FPR$ are scaled by a factor of 10.) The $FPR$ decreased with $s$ while the $FNR$ increased with sufficiently large $s$. This may be loosely explained by the exponential decrease of the volume of $R_G$ with respect to $s$. As $s$ increases, the accidental probability $P_a(G)$ becomes minimal with only a small number of trajectories in $G$ thus the algorithm stops short of detecting many foreground trajectories. Although the estimated background distribution counters this effect by some degree by giving a higher probability to $R_G$, the assumption of independence in the variables in the trajectory representation causes the underestimation of the probability. For small $s$, the $FNR$ was slightly high partly due to sparse clusters with insufficiently high $P_a(G)$. The average $NS$ value shown in Figure 5.12(b) increased with $s$, which is an indication of over-segmentation of some clusters due to smaller $P_a(G)$'s as explained above. The number of sampled points used in the subsequent experiments was $s = 5$ with produces low error rates while keeping the average value of $NS$ lower than 1. No clusters were merged in this set of experiments, which is expected since the clusters are far apart.

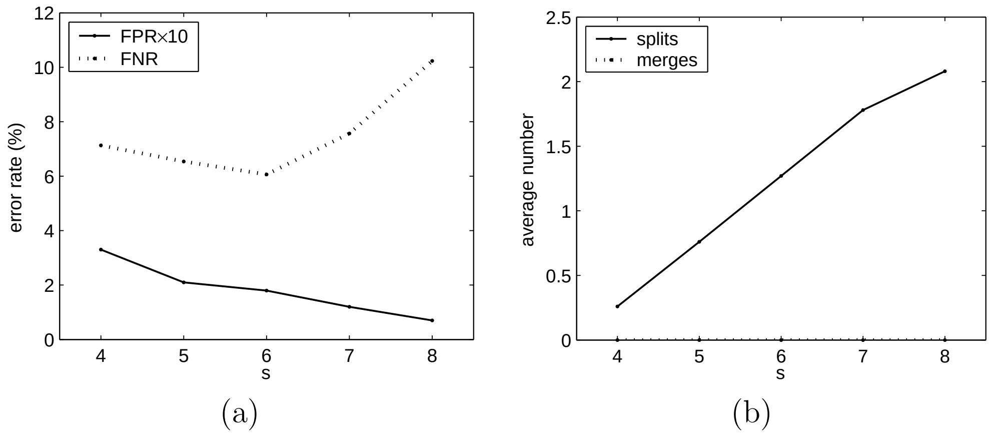
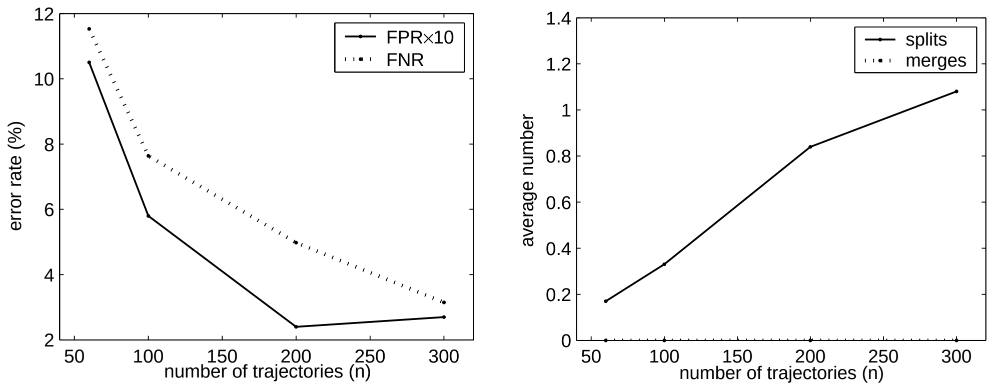


Figure 5.11: The effect of the number of sampled points in a trajectory. ($n = 200$, $n_f = 133$)

### 5.6.4.2 The effect of the number of trajectories

Synthetic datasets with increasing number of total trajectories $n$ were generated for this experiment. The ratio of foreground trajectories were fixed at 1/3 for each dataset. Figure 5.12 shows the performance on datasets with $n$ ranging from 60 to 300. The error rates were high when $n$ was small which means the clusters were not dense enough to be detected as salient. Both $FPR$ and $FNR$ gener-



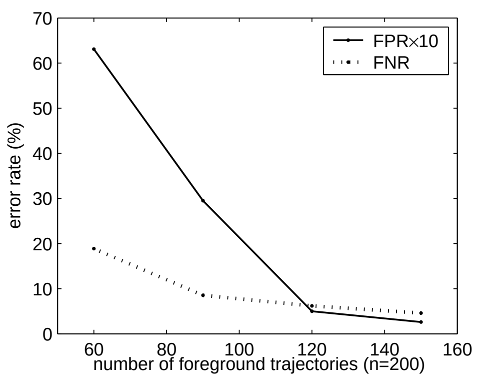
(a) (b)

Figure 5.12: The effect of the number of total trajectories in the dataset.

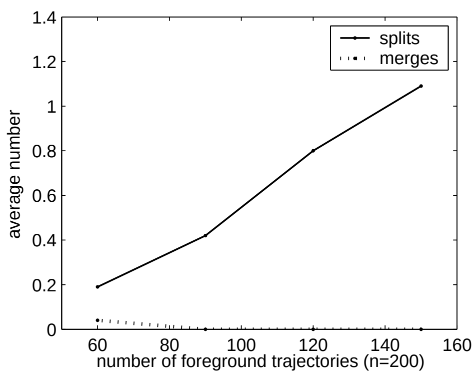
ally decreased with increasing $n$ except at $n = 300$ where $FPR$ slightly increased. This is thought to be caused by the increased number of background trajectories that are similar to the foreground clusters. Split clusters ($NS$) occurred at most approximately once per dataset, which is consistent with the real dataset. The average number of $NS$ increased with $n$ because of the widely spread nature of one reference cluster (clusters 14 and 23 combined in Figure 5.2).

#### 5.6.4.3 The effect of the relative number of background trajectories

In this experiment, the ratio of foreground trajectories was varied while the total number of trajectories was fixed at $n = 200$. Let $n_f$ be the number of foreground trajectories. As can be seen in Figure 5.13(a), the error rates were very high at $n_f = 60$. In particular, both $FPR$ and $FNR$ values were substantially higher compared with $n = 100$ in Figure 5.12(a) where its number of foreground trajectories is close to 60 (67 to be precise) but with much less background trajectories. The high $FPR$ is due to background trajectories accidentally forming clusters among



(a)



(b)

Figure 5.13: The effect of the relative number of background trajectories.

themselves or being near true foreground clusters. The cause for the higher $FNR$ is thought to be a combination of the interference from the background trajectories and the fact that the foreground trajectories was slightly sparser in this experiment. The errors decreased as the relative $n_f$ increased, as expected. It is seen that for $n = 200$, $n_f$ should be more than half of $n$ to obtain $FPR < 1\%$ and $FNR < 10\%$. At $n_f = 60$ some merging of clusters were observed in Figure 5.13(b), which is due to the relative sparseness of the foreground.

## 5.7 Discussion

We have presented a general framework for finding significant clusters of data samples in the presence of unclustered background samples. The non-accidentalness criterion was used to measure the significance of a candidate cluster, assuming a simple statistical model of the background. For the application of discovering salient clusters of trajectories, we defined a representation for the trajectory and the region defined by a group of trajectory representations. It was shown through experiments on real and synthetic data that our method performs well on data with different characteristics without requiring any tuning of the parameters.

In our method, the absolute location and orientation of the trajectories were utilized for discovering activities. This approach is valid when the activities are more or less determined by the fixed structures in the scene e.g., entrance/exits and pathways. However, there are types of activities that are less dependent on the spatial location such as getting into a car or jogging in arbitrary direction. These types of activities are well clustered in different sets of features such as the initial point and general direction or the (similarity-invariant) trajectory shape and speed. One future direction is to study the applicability of the non-accidentalness principle in discovering features that cause the data to be well clustered.

# Chapter 6

# Conclusion and Future Work

We have presented four new approaches that deals with different levels of problems in computer vision applicable to automatic video surveillance. For detecting moving objects in a video, we introduced the temporal variance as a robust measure of motion in a video. The problem of tracking multiple objects given the object detection information was cast as a combinatorial optimization problem of associating measurements with targets, which provided the theoretical basis for an efficient multiple hypothesis tracking technique. For recognizing high-level events, we proposed to model the events using representations based on attribute grammars. In particular, the tracking data was represented by a series of symbols and their associated attributes, which were classified into different type of events using an incremental parsing algorithm. It should be noted that the methods described in later chapters depends on the techniques developed in the earlier chapters. In addtition, the methods presented in Chapters 2 through 4 are online algorithms applicable to a real-time video processing system, an example of which was given in Chapter 4. Finally, we have proposed a novel approach for discovering frequent patterns of activities based on the principle of non-accidental grouping. Although this is an off-line method, it may be applied to incremental chunks of data, which would be appropriate for surveillance applications.

Our approaches have been validated using real and simulated data to ensure that they are useful in practical applications. In particular, the performance evaluation for the temporal variance-based detection method showed that it is effective and performs better than a background modeling method under various conditions. Our multi-object tracking approach is demonstrated to handle various degrees of object interactions for different types of videos. We have shown multiple examples of high-level event recognition through an implementation of a fully automatic surveillance system. The activity discovery method is shown to perform well in real data as well as in synthetic data with various characteristics, whereas a straightforward application of traditional clustering method produced undesirable results.

There are certainly more work to be done regarding these approaches. Our object detection method suffers from fragmentation of objects when they move very slowly and detection would fail if an object stops moving for some time. This is a disadvantage shared by motion-based object detection approaches. For applications where these kinds of object behaviors are common, background modeling approaches should produce better results as they work independently of time, thus object speed. A combination of both approaches, seamlessly integrating the advantages of the two would be one future direction.

In our multi-object tracking method, the appearance (e.g., shape, color, or texture) of each target is not modeled in the tracking framework, assuming that the appearance between targets are indistinguishable such as in the case with the camera positioned against the light or an infra-red camera. However, there are cases where the appearance of the targets are more or less distinct. In such a case the appearance feature extracted from the measurements may be incorporated into the association cost. This may extend the capability of reacquiring targets that moves out of the view for a short time and reappear nearby. The formulation of minimum weight edge cover should be useful for other vision problems such as matching model features to non-rigid objects.

Non-accidentalness is a general principle that may be applied to different levels of features [41, 15]. For activity recognition, the patterns of activities found by our method may itself be a feature which may be grouped over a longer time scale. By focusing on the "outlier" activities, we should be able to detect anomalies. It would also be interesting to apply the principle to the temporal relationships among the trajectory features, possibly enabling to learn the patterns of interactions between objects. Another direction we have not pursued is to apply other perceptual organization principles to trajectories, such as parallelism and similarity.

# Chapter A

# Algorithm for the minimum weight edge cover

We describe the algorithm for reducing the minimum weight edge cover problem to the minimum weight perfect matching. In the following, $V$ and $E$ represent the set of vertices and edges in a graph. $w(e)$ or $w(x, y)$ denotes the weight associated with edge $e$ or with the edge $(x, y)$ defined by two vertices $x, y$. $w(G)$ denotes the sum of edge weights in graph $G$.

1. Given a connected graph $G = (V, E)$ with edge weights $w(e), e \in E$, we construct a graph $H$ in the following way:
    - Add $G$ and its identical copy $G' = (V', E')$ to $H$. Assign weights to the copy as $w(e') = 0, e' \in E'$.
    - Connect each vertex $v \in V$ and its counterpart $v' \in V'$ and assign weights as $w(v, v') = \min_{x \in N(v)} w(v, x)$, where $N(v)$ is the set of vertices adjacent to $v$.
2. Find the minimum weight perfect matching $M_H$ of $H$.
3. Construct an edge cover $C_G$ by replacing each connecting edges $(v, v')$ in $M_H$ with the edge $(v, x)$ whose weight was the origin of $w(v, v')$, then deleting $G'$ from $M_H$.

**Theorem A.1** *$C_G$ is a minimum weight edge cover of $G$.*

**Proof** By the above construction of $H$,

$$w(M_H) = w(C_G) \tag{A.1}$$

Since $M_H$ is the minimum weight perfect matching of $H$, given any perfect matching $\tilde{M}_H$,

$$w(M_H) \leq w(\tilde{M}_H) \tag{A.2}$$

Now given *any* edge cover $\tilde{C}_G$ of $G$ we can (reversely) construct a perfect matching $\tilde{M}_H$ of $H$ by the following:

1. Remove all paths in $\tilde{C}_G$ with 3 edges by removing the edge in the middle. (Note this still results in a valid edge cover.)

2. For each vertex $v$ with degree $deg(v) > 1$, remove any edge $(v, x)$ from $v$ and add $(x, x')$ to $\tilde{M}_H$, repeating until $deg(v) = 1$.

3. For each remaining edge in $\tilde{C}_G$, add the two corresponding edges in $H$ to $\tilde{M}_H$.

Since by construction of $H$, $w(x, x') \leq w(v, x)$ and $w(G') = 0$,

$$w(\tilde{M}_H) \leq w(\tilde{C}_G) \tag{A.3}$$

From (A.1), (A.2), and (A.3), for any $\tilde{C}_G$ and $\tilde{M}_H$

$$w(C_G) = w(M_H) \leq w(\tilde{M}_H) \leq w(\tilde{C}_G) \tag{A.4}$$

# Bibliography

[1] H. Ablas. Introduction to attribute grammars. In H. Ablas and Melichar B., editors, *Attribute Grammars, Application and Systems*, volume 545 of *LNCS*. Springer-Verlag, 1991.

[2] A. Albiol, C. Sandoval, V. Naranjo, and J. M. Mossi. Robust motion detector for video surveillance applications. In *Proc. International Conference on Image Processing*, volume 2, pages 379–82, 2003.

[3] J. Allen. *Natural Language Understanding.* Benjamin/Cummings, 1995.

[4] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association.* Academic Press, 1987.

[5] F. Cao, J. Delon, A. Desolneux, P. Muse, and F. Sur. A unified framework for detecting groups and application to shape recognition. *To appear in Journal of Mathematical Imaging and Vision.*

[6] C. Chang, R. Ansari, and A. Khokhar. Multiple object tracking with kernel particle filter. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1:566–573, June 2005.

[7] H.-T. Chen, H.-H. Lin, and T.-L. Liu. Multi-object tracking using dynamical graph matching. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages II:210–II:217, 2001.

[8] Y. S. Chia and W. Huang. Multiple objects tracking with multiple hypotheses dynamic updating. In *Proc. IEEE International Conf. on Image Processing*, pages 569–572, 2006.

[9] I. J. Cox and S.L. Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(2):138–150, February 1996.

[10] N. Cressie. *Statistics for Spatial Data.* Wiley, 1993.

[11] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(10):1337–1342, October 2003.

[12] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, 2nd edition, 2000.

[13] C. de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 38(9):1332–1348, 2005.

[14] H. Dee and D. Hogg. Detecting inexplicable behaviour. In *Proc. of the British Machine Vision Conference*, 2004.

[15] A. Desolneux, L. Moisan, and Morel J.-M. Meaningful alignments. *International Journal of Computer Vision*, 40(1):1–23, October 2000.

[16] A. Desolneux, L. Moisan, and J.-M. Morel. A grouping principle and four applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(4):508–513, April 2003.

[17] R. Duda, P. Hart, and D. Stork. *Pattern Classification.* Wiley Interscience, 2nd edition, 2001.

[18] J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[19] A. Elgammal, D. Harwood, and Davis L. Non-parametric model for background subtraction. In *Proc. European Conference on Computer Vision*, volume 2, pages 751–767, 2000.

[20] K. S. Fu. *Syntactic Pattern Recognition and Applications.* Prentice Hall, 1982.

[21] A. Genovesio and J.-C. Olivo-Marin. Split and merge data association filter for dense multi-target tracking. In *Proc. International Conf. on Pattern Recognition*, volume 4, pages 677–680, 2004.

[22] N. Ghanem, D. DeMenthon, D. Doermann, and L. Davis. Representation and recognition of events in surveillance video using petri nets. In *Proc. of the IEEE Workshop on Event Mining*, 2004.

[23] W. E. L. Grimson and D. P. Huttenlocher. On the verification of hypothesized matches in model-based recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(12):1201–1213, 1991.

[24] R. Hamid, A. Johnson, S. Batta, A. Bobick, C. Isbell, and G. Coleman. Detection and explanation of anomalous activities: representing activities as bags of event n-grams. In *Proc. Computer Vision and Pattern Recognition, CVPR*, volume 1, pages 1031–1038, 2005.

[25] M. Han, A. Sethi, and Y. Gong. A detection-based multiple object tracking method. In *Proc. IEEE International Conf. on Image Processing*, volume 5, pages 3065–3068, 2004.

[26] M.-O. Hongler, Y. L. de Meneses, A. Beyeler, and J. Jacot. The resonant retina: exploiting vibration noise to optimally detect edges in an image. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(9):1051–1062, September 2003.

[27] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. A system for learning statistical motion patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(9):1450–1464, September 2006.

[28] S. Huwer and H. Niemann. Adaptive change detection for real-time surveillance applications. In *Proc. IEEE International Workshop on Visual Surveillance*, pages 37–45, 2000.

[29] M. Isard and J. MacCormick. Bramble: a bayesian multiple-blob tracker. In *Proc. IEEE International Conf. on Computer Vision*, volume 2, pages 34–41, 2001.

[30] Y. A. Ivanov and A. F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):852–872, August 2000.

[31] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[32] R. C. Jain and H. H. Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1(2):206–213, April 1979.

[33] O. Javed and M. Shah. Tracking and object classification for automated surveillance. In *Proc. European Conf. on Computer Vision – Part IV*, pages 343–357, 2002.

[34] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. In *Proc. British Machine Vision Conference*, volume 2, pages 583–592, 1995.

[35] R. Jonker and A. Volgenant. A shortest path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[36] S.-W. Joo and Q. Zheng. A temporal variance-based moving target detector. In *Proc. IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, January 2005.

[37] Z. Khan, T. Balch, and F. Dellaert. Multitarget tracking with split and merged measurements. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Vol. 1*, pages 605–610, June 2005.

[38] K. Kim, T. H. Chalidabhongse, H. David, and L. Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11(3):172–185, June 2005.

[39] D. E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2(2), 1968.

[40] O. Lanz. Approximate bayesian multibody tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(9):1436–1449, September 2006.

[41] D. Lowe. *Perceptual Organization and Visual Recognition.* Kluwer, 1985.

[42] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Trans. on Systems, Man and Cybernetics, Part B*, 35(3):397–408, June 2005.

[43] G. Medioni, I. Cohen, F. Bremond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(8):873–889, August 2001.

[44] A. Mittal and L. S. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal of Computer Vision*, 51(3):189–203, Feb/Mar 2003.

[45] D. Moore and I. Essa. Recognizing multitasked activities from video using stochastic context-free grammar. In *Proc. of Eighteenth National Conference on Artificial Intelligence*, 2002.

[46] K. G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.

[47] N. Paragios and C. Tziritas. Detection and location of moving objects using deterministic relaxation algorithms. In *Proc. International Conference on Pattern Recognition*, volume 1, pages 201–205, 1996.

[48] F. Porikli and T. Haga. Event detection by eigenvector decomposition using object and frame features. In *Proc. IEEE Workshop on Event Mining*, 2004.

[49] P. Prokopowicz and P. Cooper. The dynamic retina: Contrast and motion detection for active vision. *International Journal of Computer Vision*, 16(3):191–204, November 1995.

[50] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: a systematic survey. *IEEE Trans. on Image Processing*, 14(3):294–307, March 2005.

[51] D. Reid. An algorithm for tracking multiple targets. *IEEE Trans. on Automatic Control*, 24(6):843–854, 1979.

[52] P. L. Rosin. Thresholding for change detection. *Computer Vision and Image Understanding*, 86(2):79–95, May 2002.

[53] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume A. Springer, 2003.

[54] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 246–252, 1999.

[55] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.

[56] A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201, 1995.

[57] Xiang T. and Gong S. Video behaviour profiling and abnormality detection without manual labelling. In *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2005.

[58] N. Vaswani, A.K. Roy-Chowdhury, and R. Chellappa. "shape activity": a continuous-state HMM for moving/deforming shapes with application to abnormal activity detection. *IEEE Trans. on Image Processing*, 14(10):1603–1616, 2005.

[59] V. T. Vu, F. Bremond, and M. Thonnat. Automatic video interpretation: A novel algorithm for temporal scenario recognition. In *Proc. of The Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, August 2003.

[60] X. Wang, K. Tieu, and W. E. L. Grimson. Learning semantic scene models by trajectory analysis. In *Proc. European Conference on Computer Vision*, 2006.

[61] A. P. Witkin and J. M. Tenenbaum. On the role of structure in vision. In J. Beck, B. Hope, and A. Rosenfeld, editors, *Human and Machine Vision*, pages 481–543. Academic Press, 1983.

[62] Weiss Y. Segmentation using eigenvectors: A unifying view. In *Proc. IEEE International Conference on Computer Vision*, pages 975–982, 1999.

[63] Q. Yu, I. Cohen, G. Medioni, and B. Wu. Boosted markov chain monte carlo data association for multiple target detection and tracking. In *Proc. International Conf. on Pattern Recognition*, volume 2, pages 675–678, 2006.

[64] Hua Zhong, Jianbo Shi, and M. Visontai. Detecting unusual activity in video. In *Proc. of Computer Vision and Pattern Recognition, CVPR*, volume 2, 2004.