A Framework for Activity Recognition and Detection of Unusual Activities

A dissertation
submitted in partial fulfillment
of the requirements for the degree
of
Bachelor of Technology
in
Computer Science and Engineering
by

Dhruv Kumar Mahajan, Nipun Kwatra and Sumit Jain

Under the guidance
of

Prof. Subhashis Banerjee and Dr. Prem Kalra
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

Department of Computer Science and Engineering
Indian Institute of Technology Delhi

May 2004

Certificate

This is to certify that the thesis entitled *A Framework for Activity Recognition and Detection of Unusual Activities*, which is being submitted by *Dhruv Kumar Mahajan*, *Nipun Kwatra* and *Sumit Jain* in fulfillment of the requirements for the award of *Bachelor of Technology* in *Computer Science and Engineering* at the Indian Institute of Technology is a true record of the students' work carried out under our supervision and guidance.

The matter embodied in this dissertation, to the best of our knowledge, has not been submitted for the award of a degree or a diploma elsewhere.

Prof. Subhashis Banerjee **Dr. Prem Kalra**

Department of Computer Science and Engineering
Indian Institute of Technology Delhi
Hauz Khas, New Delhi - 110016

Acknowledgement

We would like to express our gratitude to our supervisors Prof. Subhashis Banerjee and Dr. Prem Kalra for their continued support and valued guidance at every step of our project. Without their timely advice it would have been impossible to make any inroads into the project. Working on this project has been an enriching as well as an enjoyable experience for all of us. Prof. Banerjee and Dr. Kalra provided us with initiation, encouragement and motivation at every stage without which the completion of this project would not have been possible.

We also take this opportunity to thank Prof. Santanu Chowdhary for the insightful sessions of discussions and explanations which helped us clear a lot of doubts. We are also grateful to the members of **Kritikal Solutions** and **Vision and Graphics Lab, IITD** for their valuable time, support and helpful suggestions.

Dhruv Kumar Mahajan **Nipun Kwatra** **Sumit Jain**

Abstract

In this work we present a simple framework for activity recognition based on a model of multi-layered finite state machines, built on top of a low level image processing module for spatio-temporal detections and limited object identification.

The finite state machine network learns, in an unsupervised mode, usual patterns of activities in a scene over long periods of time. Then, in the recognition phase, usual activities are accepted as normal and deviant activity patterns are flagged as abnormal. Results, on real image sequences, demonstrate the robustness of the framework.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Introduction

Analyzing complex spatio-temporal changes in a dynamic scene in order to recognize logical sequences of usual activity patterns and detect unusual deviations has become an important problem in computer vision, especially in the context of visual surveillance. In this work, we present a simple framework for activity recognition based on a model of multi-layered finite state machines, built on top of a low level image processing module for spatio-temporal detections and limited object identification. We fix the modes of interaction in the lower physical layer of the network architecture depending on the context of the problem, and automatically find clusters of high probability sequences of transitions to learn usual behavioural patterns in an unsupervised manner. Low probability sequences which are not recognized by the FSM network are diagnosed as unusual.

We argue that such a model is more suitable for detection and recognition of activity patterns in surveillance situations than some of the more complex models suggested in the literature - based on Hidden Markov Models [16, 12, 17, 14, 15], Bayesian networks [2, 13, 8, 9] and stochastic context free grammars [11].

In contrast with most of the techniques found in the literature which are either hard coded to recognize only specific activities or model activity patterns using supervised learning, we consider two different scenarios -

1. unsupervised learning of usual activity patterns and detection of unusual activities. Any activity which is not recognized as normal is flagged as deviant.

2. explicit programming of the FSM's for recognition of complex activities or training using supervised learning.

Examples of the first of the above may include learning usual activity patterns of movements of people in an airport terminal or of people and cars in a parking lot, and detecting deviant patterns like walking in a wrong direction, straying in to restricted areas, crossing of barriers, wrong parking and suspicious behaviours like taking more than usual amount of time to open the door of a car.

Our framework for this case is generic - only the low level detectors in the image processing layer and the physical layer of the network structure need to be changed with context, the algorithms for activity learning and recognition require no changes whatsoever and can be trained using unsupervised learning, even for multiple types of unknown activities.

Examples of the second may include *unattended baggage detection*, wherein a person leaves a bag in the scene and walks away and the bag lies unattended for a certain duration of time; and *stealing*, where a person puts a bag down and a second person picks it up and rushes away. Our system can either be explicitly programmed to recognize activities such as these or may be trained to recognize these activities by enacting them a number of times.

The rest of the thesis is organized as follows. In Section 1.2 we review some of the literature on activity recognition and detection and put our approach in perspective. In Chapter 2 we describe our low level image processing module and object detectors. In Chapter 3 and 4 we describe our activity analysis framework based on finite state machines and layered finite state machines. We present some results on examples of the type mentioned above in Chapter 5, and finally, in Chapter 6, we conclude out work.

1.2 Related work

Recently, the problem of activity detection and recognition in the context of visual surveillance has received considerable attention [1]. There has been significant work that span across techniques for low level event detection [5, 20, 23, 4] and activity modeling [16, 12, 17, 14, 15, 11, 2, 13, 8, 9, 11, 6, 21, 3, 10]. Starting from the early work of Yamato et. al. [22], HMM's have been a popular tool for activity modeling, motivated primarily by its successful use in speech recognition. A HMM is a stochastic finite state machine which models an activity pattern by learning transition probabilities among its non-observable states such that the likelihood of observation of a temporal sequence of symbols representing the activity is maximized. HMM's have been used to model simple isolated hand gestures [19], and more complex variants like coupled HMM's [17] and layered HMM's [16] have been proposed to model events such as interaction between multiple mobile objects. Some of these enhancements also implicitly model dynamic time warping. However, unlike in speech (and perhaps in gestures, to a certain extent), where the relationship of the observed output symbol to the internal semantic state is uncertain at any instant of time necessitating the use of 'hidden' states, in surveillance applications the relationship is often more direct. Consider, for example, the stealing example (see above). The internal semantic states are clearly characterized by the sequence 'person A puts a bag down', 'person B approaches the bag', 'person B picks up the bag', 'person B rushes away' - and all of these are directly observable from the image sequence. For representing such activities it is perhaps prudent to use a model where the state representations are 'transparent' with well identified physical meanings rather than 'hidden'. This would not only facilitate direct estimation of the state transition probabilities from the observables, thereby eliminating the need for solving a complex maximum likelihood estimation problem using iterative techniques, but also avoid the problems of over-parametrization and over-fitting especially when the training data is inadequate. In [11] a parsing technique based on stochastic context-free grammars is proposed to compute the probability of a temporally consistent sequence of primitive actions recognized by a HMM model. A model that recognizes a context-free grammar is indeed more powerful than a FSM model which can capture only regular languages. For example, it allows for recognition of languages like $\{a^n b^n, n \geq 0\}$ for logical symbols a and b, which cannot be recognized by FSM's. However, in typical surveillance applications, activity patterns that need to be modeled are seldom of the type that requires bracket matching using a stack, and simple precedence requirements usually suffice in most cases, which can be effectively modeled by FSM's. In [18] it is argued that a finite state machine cannot model temporal inter-leavings of low level events which may occur concurrently and do not have simple temporal precedence relationships in complex multi-actor activities. To deal with such situations the authors propose an multi-layered extension of FSM's called *Propagation net-*

works. However, we show that such situations can be effectively captured by coupled FSM's. Another popular approach for activity recognition is though the use of Bayesian networks. In fact, in an approach closely related to ours Hongeng et. al. [8, 9] propose a multi-layered FSM model for activity recognition. However, they estimate the parameters of their model using supervised training using a Bayesian formulation. The major difficulty with the Bayesian formulation lies in obtaining the apriori estimates, especially for unusual activities which are not part of a closed set. Moreover, in their model the states of the FSM's depend on absolute time clocks causing an explosion of possibilities for state changes. Also, the method cannot be directly extended to unsupervised clustering.

Recently, there have been some interesting approaches to detection of unusual activities. In [21, 3] moving objects on a 2D plane are treated as point clusters, whose deformable shapes in time represent usual patterns of activities. Deviations are then detected from the shape changes. In [10], usual activities in a video stream are learned using unsupervised clustering. However, the application scenarios of these approaches are different from ours.

In contrast to some of the other finite state models, the states in our multi-layer FSM framework have well identified physical meanings. They represent single or multiple objects, their positions and distance vectors. There is no explicit need for fixing the size of the network apriori. Also, the state transition probabilities, at any layer, can be directly estimated from the observables. As our results clearly demonstrate, our framework can also handle complex activities involving concurrent and interleaved actions by multiple actors in a scene.

1.3 Overview of our model

Our entire system of activity recognition is divided into four different layers as shown in the Fig. 1.1.

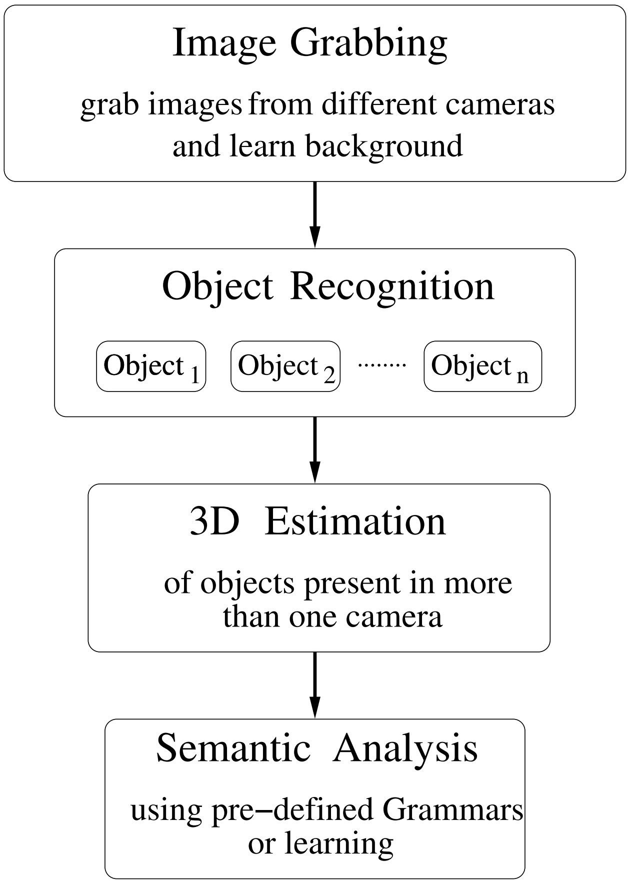

Figure 1.1: Overall Structure

1. **Image grabbing layer**: This is the basic low level module which provides an interface to frame grabbing from multiple cameras. It also gives foreground from previously learnt backgrounds.

2. **Object recognition layer**: Different vision and learning techniques are available for object recognition based on previously trained samples.

Methods like harr wavelets [REF], SIFT[REF] descriptors, SVMs can be used for this purpose. This layer uses these methods to detect the various objects using the input image and the extracted foreground.

3. **3D estimation**: For objects visible in more than one cameras, simple triangulation can be used to evaluate their approximate 3D positions. Ground plane homography can be used for more accurate estimation of 3D in objects known to lie on the ground plane.

4. **Semantic analysis**: Last but not the least, this is the most important layer. The purpose of this layer is to analyze the various objects and other data from the lower layers to recognise activities. Various approaches are possible for solving the problem. Pre-defined grammars may be defined to recognise various kinds of activities. It is desirable that the system is made programmable for recognising generic activities.

Chapter 2

Image analysis modules

As mentioned above, our state descriptions are derived from physically observable attributes like object types and their positions on the ground plane. We use standard image processing techniques to detect and compute these from on-line sequences [5]. Our system works in real-time at full frame rate. We assume the ground plane to be calibrated in terms of a XY grid whose resolution is dependent on the context. We also assume that the homography of the ground plane to be known from calibration [7]. Our low level image processing layer consists of the following modules:

1. segmentation using adaptive multi-layer background subtraction [5].

2. object tracking using Kalman filters and inter-frame object matching using volume intersection of hue-saturation histograms of detected objects.

3. splitting and merging of image segments.

4. object identification using principle component analysis. Currently, we detect only three types of objects - humans, cars and normal bags.

5. detection of the position of the object on the XY grid on the ground plane or computing the 3D point if object is present in multiple cameras.

2.1 Object tracking

We also need to track the moving objects so as to find out which object in the previous frame corresponds to which object in the current frame. Also if two objects merge together (e.g. if two humans cross each other) we need to find the boundary between them and identify them as two separate components. Similarly in case of splitting we have to do consistent label assignments.

2.1.1 Features used for tracking

Distance between centroids of objects: In general, the centroid of an object in the current frame is closer to the corresponding object in the previous frame as compared to other objects. As a result distance between corresponding centroids in consecutive frames acts as a good metric for tracking.

Color histograms of objects:To detect merging and finding separating boundaries or splitting and finding consistent label assignments, we also need another metric to incorporate the appearance model of the different objects. We also maintain the average colored histogram of each object as an estimate of its colour distribution. The histogram is averaged over previous few, say N frames to incorporate gradual changes in the orientation of various objects, e.g. a person turning around. Histogram matching (Algorithm 1) between the averaged and present histogram of objects then acts as a metric of similarity.

Algorithm 1 Histogram matching-$Hmatch$

/* $H1$ and $H2$ are assumed to be normalised histograms */

for all colour components $M \in \{R, G, B\}$ **do**

  for all bins $n \in \{0, \ldots, MAX\_PIXEL\_VALUE - 1\}$ **do**

    $err_M[n] = (H1_M[n] - H2_M[n])^2 \times H1_M[n]$

  end for

end for

$Match\_error = \Sigma_M \Sigma_n err_M[n]$

2.1.2 Cost matrix for tracking

Let d_{ij} be the distance between centroids of i^{th} object in current frame and j^{th} object in previous frame. Let $hist\_current_i$ denote histogram of i^{th} object in current frame and $hist\_previous_j$ denote average histogram of j^{th} object in previous frame. We define the cost matrix (symmetric) elements c_{ij}, the cost between the i^{th} object in current frame and the j^{th} object in previous frame as

$$c_{ij} = w * d_{ij} + (1 - w) \times Hmatch(\ hist\_previous_j,\ hist\_current_i) \quad (2.1)$$

where w signifies the relative weight assigned to centroid matching and histogram matching. c_{ij} then is an estimate of error involved if we assume that i^{th} object in current frame corresponds to j^{th} object in previous frame.

2.1.3 Basic tracking algorithm

Algorithm 2 Basic tracking algorithm-*no merging and splitting*

/* Let M denote number of objects in the current frame and N denote number of objects in the previous frame */

Case 1: $M < N$, $\Rightarrow$ some objects have left the scene.

1. for each object m in current frame, find n in the previous frame such that $n = argmin(c_{mj})$, $j \in \{1, \ldots, N\}$.
2. assign m^{th} object in current frame, the label of n^{th} object in previous frame ie.

$$label(current\_object_m) = label(previous\_object_n)$$

Case 2: $M > N$, $\Rightarrow$ some objects have entered the scene.

1. for each object n in previous frame, find m in the current frame such that $m = argmin(c_{in})$, $i \in \{1, \ldots, M\}$.
2. assign m^{th} object in current frame, the label of n^{th} object in previous frame ie.

$$label(current\_object_m) = label(previous\_object_n)$$

3. assign new labels to the unassigned objects in the current frame.

Case 3: $M = N$, $\Rightarrow$ same objects in both the frames.

apply either **Case 1** or **Case 2**

2.1.4 Tracking algorithm handling merging of image segments

We now propose an algorithm for detecting merging of objects (foreground lobes).

Potential merging

A decrease in the number of foreground lobes will correspond to one of the two possibilities, *viz*

1. an object has left the scene.

2. two objects have merged in the foreground.

Algorithm 3 below illustrates the procedure for tracking objects even if the foreground lobes merge.

Breaking the merged lobe: Now for each merged lobe, we want to find a distinguishing boundary between the constituent objects in the lobe. In our model we find the best vertical line that separates the lobe into two so that an appropriate cost function is minimized. Algorithm 4 below illustrates the procedure of distinguishing between the two objects in the merged lobe.

2.1.5 Tracking algorithm handling splitting of image segments

Potential splitting

An increase in the of number of foreground lobes will correspond to one of the two possibilities, *viz*

1. Either a person enters the scene.

Algorithm 3 Tracking algorithm-*handling merging*

/* assumption: only two lobes can merge together into one, however any number of merges may take place. In each frame, we maintain a list of merged objects as $merge\_objects$ */

/* Let Δn denote the diference between number of objects in previous frame and number of lobes in current frame.*/

1. First of all, detect the *'potential'* lobes in the previous $(i-1)$ th frame that can merge (not belonging to $merge\_objects$) by thresholding the inter centroid distance below a certain threshold T_{DIV}.
2. Find (m_{ij}), the mean of the centroids of humans i and j for each of these potential mergers.
3. For each element (i, j) in the potential list and each lobe k in the current (i th) frame, find the distance cost $d_{ij,k} = ||m_{ij} - centroid_k||_2$ and select a k' for each (i, j) such that $d_{ij,k'} = min_k(d_{ij,k})$
4. To save computation, consider costs $d_{ij,k'}$'s below a threshold $T_{CENTROID}$.
5. For each obtained pair (ij, k'), merge the histograms h_i and h_j to form h_{ij} and obtain histogram matching error $err_{ij,k'} = Hmatch(h_{ij}, h_{k'})$.
6. Evaluate the cost function $cost_{ij,k'} = d_{ij,k'} + err_{ij,k'}$.
7. Form a list L of such tuples (ij, k') whose cost lie below a threshold T_{MATCH}.
8. Let M= $min(\Delta n, size(L))$. Then $L(1..M)$ gives the list of merges and $\Delta n - M$ gives the number of objects that have left the scene.

Algorithm 4 Breaking the merged lobe

1. For each element k' of the tuple (ij, k') the list L, assign the entire histogram $h_{k'}$ to h_{right} and empty to h_{left}. Also find a bounding box to achieve computational efficiency.

2. Start from the leftmost vertical line of the bounding box and move it in increments of S pixels wide, updating the histograms and computing the cost of the histogram match (defined later).

3. Update the histogram h_{right} and h_{left} incrementally. i.e. add the pixels in the incremented strip of pixels $(S \times height(bounding\ box))$ to h_{left} and subtract the same from h_{right}.

4. Choose the line for which the cost function is minimized.

5. Computing the cost of match. For a given separating line, and tuple (ij, k'), compute the cost of match as follows:

   - Compute the histogram matching error between h_i and h_{left} added to matching error between h_j and h_{right}. Call it $err_{il,jr}$. i.e. $err_{il,jr} = Hmatch(h_i, h_{left}) + Hmatch(h_j, h_{right})$.
   - Similarly compute $err_{jl,ir} = Hmatch(h_j, h_{left}) + Hmatch(h_i, h_{right})$.
   - Take minimum of these two costs $min(err_{il,jr}, err_{jl,ir})$ as the cost of match for computing the separating line. Also remember the order of match for each of these costs.

6. Finally the labels of the two constituent lobes are assigned according to this order for the line corresponding to global minimum error.

2. Or two persons split in foreground.

Algorithm for detecting probable splitting of foregorund lobes is similar to merge detection. Splitting can be detected by treating the previous frame as the current and vice-versa.

2.2 Position estimation of objects

For objects visible in more than one camera, simple triangulation can be used to evaluate their approximate 3D positions. Ground plane homography can be used for more accurate estimation of 3D in objects known to lie on the ground plane. For humans, we compute their position pn the XY grid on the ground by identifying the position of the feet on the ground plane as the lowest point in the segmented region along the vertical direction (whose vanishing point is known from calibration) and then applying the homography of the ground plane. For top views of cars we compute the centroid of the segmented region and apply the homography. For perspective view of cars we locate the lowest point.

Of course, segmentation is problematic to say the least, and the low level image processing is clearly the weakest link of our method. In the present state of our implementation dense situations, severe occlusions and unfavourable lighting result in failures more often than not. Consequently, our experiments are mainly restricted to sparse situations and good illumination conditions.

Chapter 3

Semantic analysis

As mentioned before, the purpose of this layer is to analyze the various objects and other data from the lower layers to recognise activities. We made use of *regular grammars* or *FSMs* to specify various activities and successfully demonstrated their use for detecting activities like *unidentified object detection* and *theft detection*.

3.1 Analysis using Regular Grammars/FSMs

Activity recognition using *FSMs* consists of two parts.

1. Symbol Generator

2. Explicit programmed finite state machine

3.1.1 Symbol Generator

This is the non-programmable part and determines how powerful an FSM can be specified on it. The Symbol Generator takes input from lower layers and generates meaningful symbols. The symbols basically define the present *physical state* of the system. e.g. number of humans, position of various objects, etc.

3.1.2 The actual finite state machine

This is the programmable part and can be defined on the symbol set generated by the symbol generator. *FSMs* for various activities can be given as input to the semantic layer to detect specific activities.
We give a couple of examples to demonstrate the concept.

Unidentified object detection The aim is to detect the activity of a person entering a scene with a bag and leaving the scene with the bag behind. The system raises an alarm if the bag is found unattended for some period of time. The following symbol set is used

Symbol	Interpretation(Physical state represeted)
0	No object in scene
1	Human with Bag(combined)
2	Human and Bag(two separate components)
3	Only Bag

The corresponding *FSM* is shown in Fig. 3.1.

Theft detection The aim is to detect theft in places like railway stations. The corresponding *FSM* is shown in Fig. 3.2.

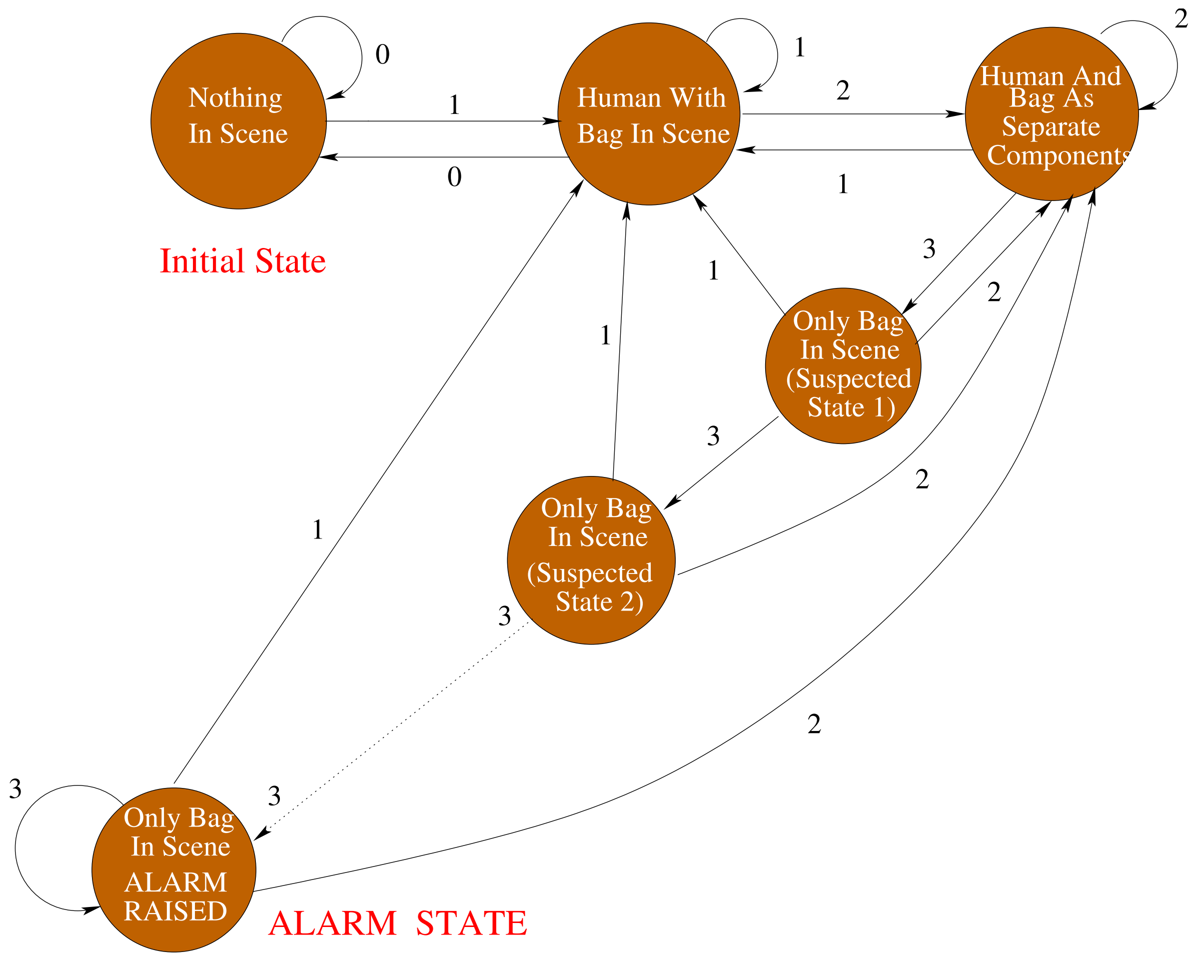

Figure 3.1: Unidentified object detection

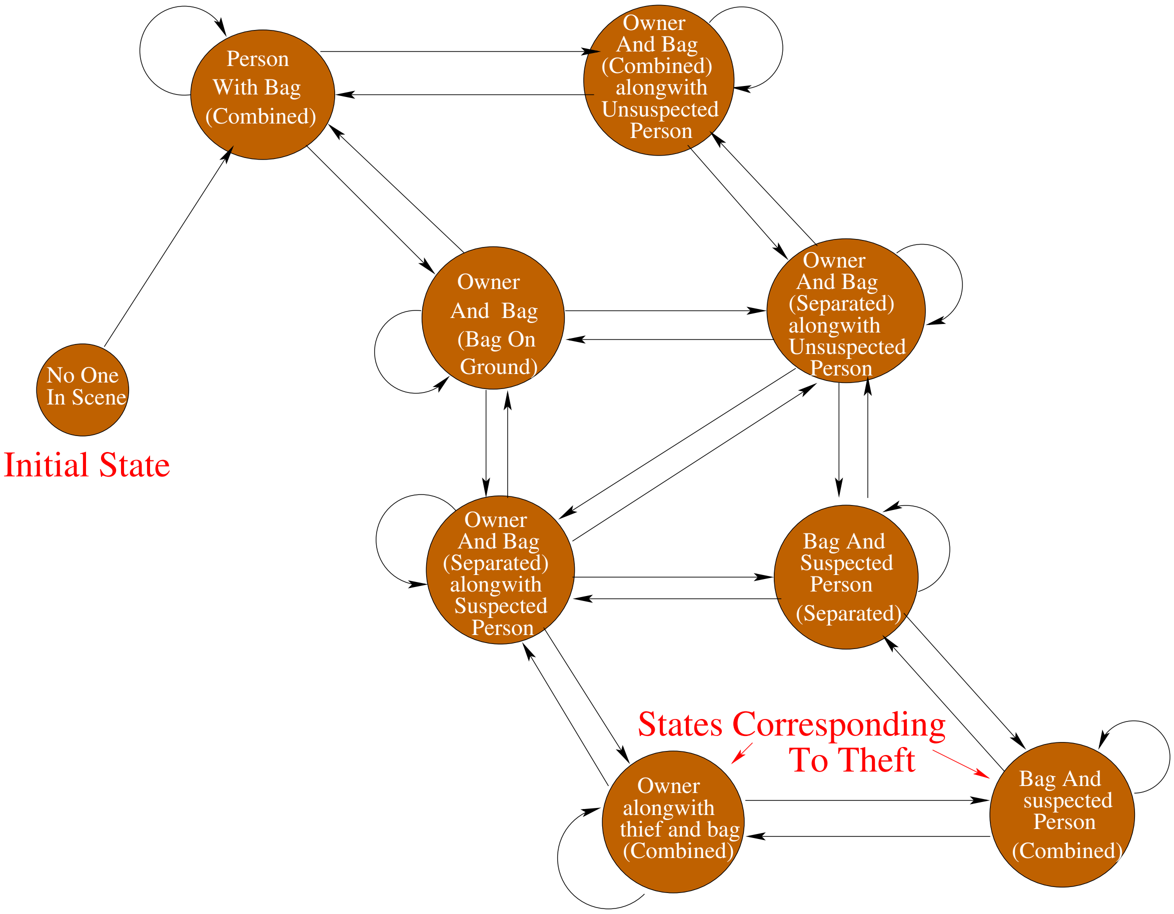

Figure 3.2: Theft detection

3.2 Hierarchichal approach for learning the activities

Using *regular grammars* or any grammars for that matter requires specifications from the user for specific activities. A self learning system will learn activities from prior training to classify normal events and unusual activities at future times. A transductive approach may be followed to learn activities along with the classification phase.

The intuition for our approach comes from an attempt to learn the *states* of *FSMs*. These states give a logical interpretation of the scene. The low-level image processing methods described in the previous section can provide only a physical description of the dynamic scene. The semantic interpretation of the physical events is carried out in the FSM's in the *physical layer* of the architecture. The interpretation at this level is in terms of the number of objects of different types present in the scene, their split and merge history, their geometric positions, and changes in these positions with time. These correspond to the *symbols* of the *symbol generator* in the above *FSM* model. High probability sequences of single object motions and multi-object interactions at the physical layer represent logical interpretations of the physical actions. Examples of such logical interpretations may be a person walking or running in a certain direction, a person walking toward a car, a person putting a bag down, two people moving toward each other etc. These logical actions in time represent the symbols and states in the higher *logical layer*. The types of interactions between objects that are to be considered are specified apriori depending on the context. The logical states are determined automatically as high probability sequences of physical transitions. In many situations, where interactions between multiple objects are unimportant, the scene interpretation can be completed at the logical layer level itself. For example, consider the monitoring of proper/improper parking of individual cars, or straying of individuals in to restricted areas. Such logical events can be flagged without even considering inter-object interactions. However, for more complex actions one requires to analyze high probability sequences of such primitive logical actions. Examples include *a*) a person walking toward a car, opening the door of the car and merging with the car and driving away - a sequence of three high probability events in the logical layer involving a person and a car, or *b*) stealing as described before involving two people and a bag. The high probability sequences in the logical layers are extracted as events which are states in the *event layer*. Even more complex events which are characterized by high probability sequences of simpler events are detected at the event layer.

Note that when trained in the unsupervised mode, the states at the higher layers are just high probability sequences at the lower layers detected automatically by clustering. While these states have clear semantics, they are still nameless. Recognition of usual behavioural patterns and detection of deviant ones can still be carried out regardless. In the next chapter we describe the details of our framework.

Chapter 4

Framework of multi-layered finite state machines

Our architecture of multi-layered FSM's for learning and recognition of complex actions is depicted in Figure 4.1.

4.1 The physical layer

The structure pf the physical layer is specific to the context. Let O_p be the p^{th} object detected in the scene which may be one of K possible types. For each such O_p we dynamically create a FSM at the lowest physical layer. The states of the FSM's are the geometric positions XY in the rectangular grid partitioning the ground plane. The edges in these FSM's represent transitions to neighbouring grid positions. We fix apriori what type of inter-object interactions are important from the context. If objects O_p and O_q interact, we also create a FSM corresponding to O_pO_q, the states of which represent all possible distance vectors from positions of O_p to positions of O_q measured in the grid coordinates, and their merging and splitting history. For example, in a parking lot scenario, we create a FSM for each $(car_p, person_q)$ pair of detected cars and people, in addition to individual FSM's for each car_p and $person_q$ (assuming that we are not interested in modeling possible interactions between two cars or two people, like accidents or collisions). Each state of a $(car_p, person_q)$ FSM is of the type $(\Delta X, \Delta Y, status)$ where $status$ is one of $\{split, merge, separate\}$. Each state of a car_p or $person_q$ FSM is of the type (X, Y). For each FSM corresponding to an instance of an object or inter-object interactions, we also maintain the following data structures. For M possible states, $transMx$ and $timeMx$ are $M \times M$ matrices and $time\_states$ is an array of size M. $transMx[i][j]$ maintains the count of the number of transitions of the object from state i to state j. $timeMx[i][j]$ maintains the mean and the variance of the time taken for the transition. $time\_state[i]$ maintains the mean and the variance of the time spent by the object in state i. Finally, at the end of the training phase, we aggregate all the statistics obtained from individual examples in to generic class specific FSM's. For example, in the parking lot scenario, all state transition statistics from individual FSM's like car_p, $person_q$ and $(car_p, person_q)$ are aggregated in to three generic FSM's called $cars$, $people$ and $(car, people)$ respectively, representing generic movement patterns of cars, people and their pair-wise interactions. In the generic FSM's we also assign a weight P_{ij} to each edge (i, j), where P_{ij} is the probability that given the object is in state i it makes a transition to state j. Clearly, $P_{ij} = \frac{transMx[i][j]}{\sum_j transMx[i][j]}$.

4.2 The logical layer

As we have mentioned above, we consider the high probability sequences of the physical layer FSM's to represent logical events. We want to extract all maximal sequences $S_n = (i, j, \ldots, n)$ such that $P(S_n) = P_{ij}P_{jk}\ldots P.n > T_{seq}$ and $P_{lm} > T_{edge}$ for every consecutive pairs of states (l, m) in S_n, where T_{seq} and T_{edge} are the probability thresholds for a sequence and an edge respectively. We call such sequences *HPL* sequences (High Probability Logical sequences). Starting from any edge (i, j), the Algorithm 5 finds all sequences S_n with (i, j) as an edge by examining sub-sequences in both forward and backward directions. We ignore the backward branches of a sequence generated by all forward expansions in order to avoid reaching the same sequence from more than one path, as shown in Figure 4.2.

Every *HPL* sequence in the physical layer represents a state in the logical layer. The system is in a logical state i if the system has most recently gone through a sequence of physical states corresponding to the logical state i. The state transitions at the logical layer happens when the physical state shifts from one *HPL* sequence to another.

4.3 Event layer

For determining complex events which are high probability transitions in the logical layer, we employ an identical algorithm at the logical layer. In our current experiments, we have restricted our framework to three levels of hierarchy.

4.4 Detection of unusual events

We maintain a counter $failcount$ with each *HPL* sequence, which tracks the number of times the FSM corresponding to a *HPL* sequence fails to make a valid transition. A global counter *totalcount* maintains a count of the time

Algorithm 5 **Finding *HPL* sequences**

```
S = {i, j}
P = P_ij
seq = φ
forward = false
find_sequence(S, P, forward) {
  i= first_state(S)
  j= last_state(S)
  in_i = incoming_edges(i)
  out_j = outgoing_edges(j)
  for all e ∈ out_j do
    if P_je > T_edge ∧ P ∗ P_je > T_seq then
      find_sequence(S ∪ {e}, P ∗ P_je, true)
    end if
  end for
  for all e ∈ in_i do
    if P_ei > T_edge ∧ P ∗ P_ei > T_seq then
      if forward==false then
        find_sequence({e} ∪ S, P ∗ P_ei, false)
      end if
    end if
  end for
  if no edge qualifies above threshold conditions then
    seq = seq ∪ S
  end if
}
```

from which any logical transition has occurred. If *totalcount* exceeds a certain threshold, then each *failcount* is checked. If all the *failcount*'s are over a threshold, then the system is in an undefined logical state and an unusual event is flagged. If any transition in a *HPL* sequence is faster or slower than than that predicted by the learning phase statistics, then an *unusual time* alarm is flagged. This, for example, can indicate running when the usual behavioural pattern is to walk.

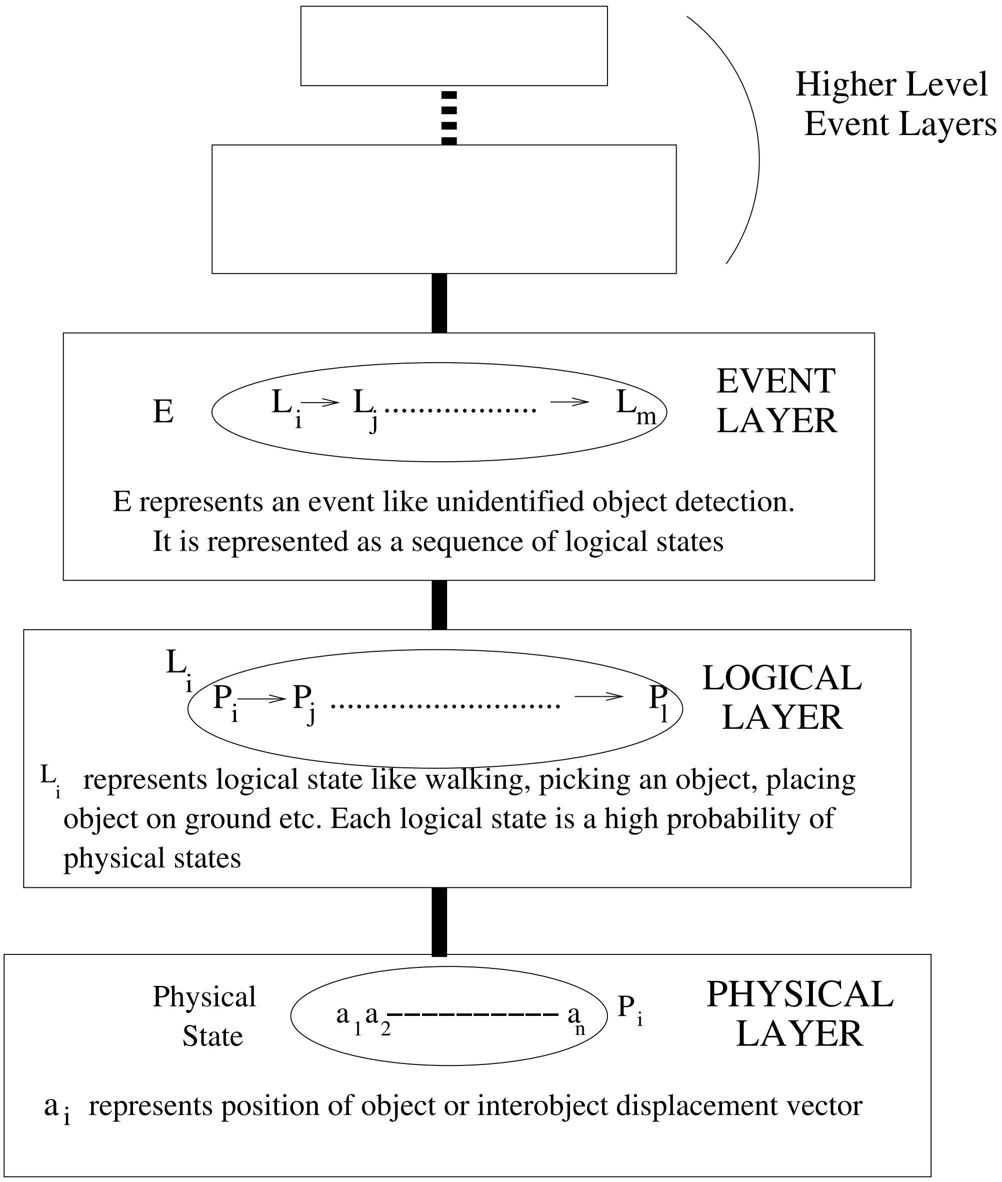

Figure 4.1: Multi-layered FSM model

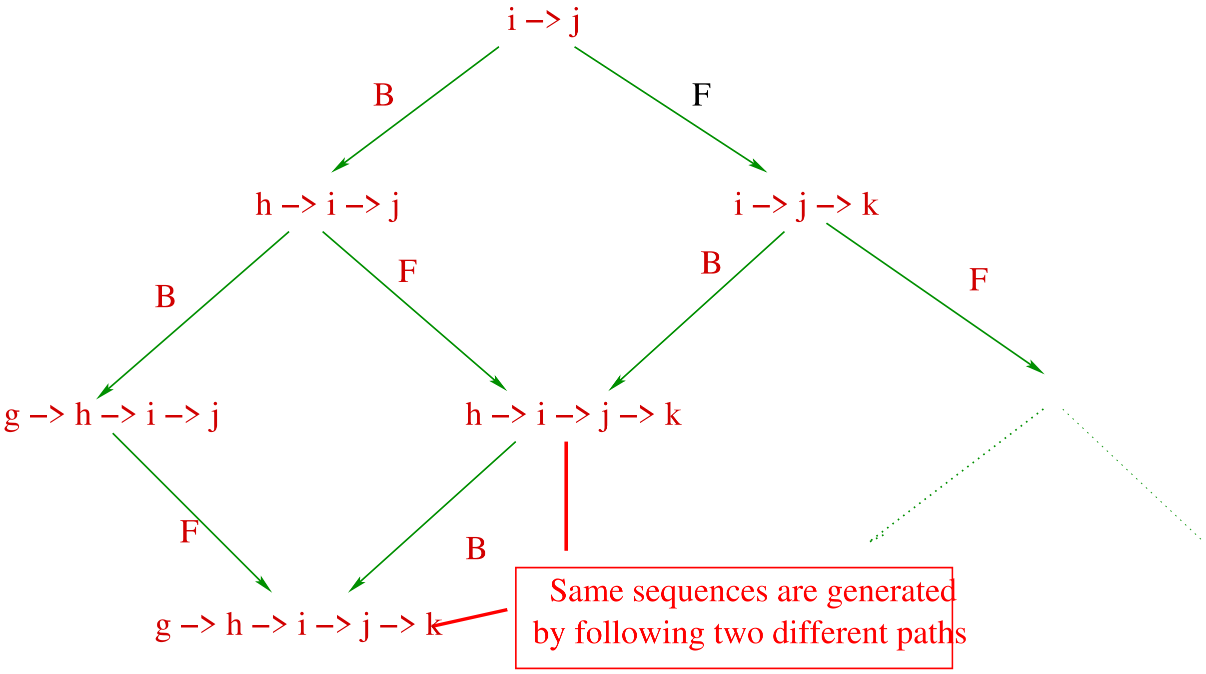

Figure 4.2: Example of multiple paths for the same sequence

Chapter 5

Results

In what follows we present some results on activity recognition. Please visit `http://www.cse.iitd.ernet.in/vglab/research/research_group2/1/activmon/index.shtml` to see the videos.

In Figure 5.1 we show results of recognition of usual walking patterns. The system was trained (unsupervised) with sequences in which people walk in the corridor in either directions. The green box (upper right corner) indicates that some logical sequence has been completed. In Figure 5.2 we show the corresponding back-ground subtracted frames and the extracted ground positions. In Figure 5.3 we show a sequence in which the person tries to cross a barrier and jump in to thin air. The red signal (upper right corner) indicates that unusual sequence transitions have happened.

In Figure 5.4 we show results in a parking lot scenario. The system was trained with sequences in which different cars move in to the parking slots. The slot in front of the gate (fourth from the right) is actually marked as 'No Parking'. The white car is recognized to have parked correctly, indicated by the small green square on the car. The parking of the red car in to the illegal area is flagged as an unusual event, indicated by a red square on the car. The training was unsupervised.

Both of the above examples are based on learning of transitions of absolute positions of individual objects. In the example of Figure 5.5 we show results of an FSM representing interaction of two objects - a person and a car. The states of the FSM represent distance vectors between a person and a car, and events such as splitting and merging. The system is trained with sequences in which people walk toward cars, open the door and get in. The figure on the left shows usual recognition - the green box indicates that some logical sequence has been completed. The figure on the right shows a frame of the sequence in which the time spent in the state is more than usual - the blue signal indicates a time alarm, i.e., system is in a particular state for more than the expected time.

Our final result, in Figure 5.6, corresponds to explicit programming of stealing. We show one frame of a stealing sequence, and the corresponding state as detected during a run of the FSM.

Figure 5.1: The walking example, recognition of a usual event.

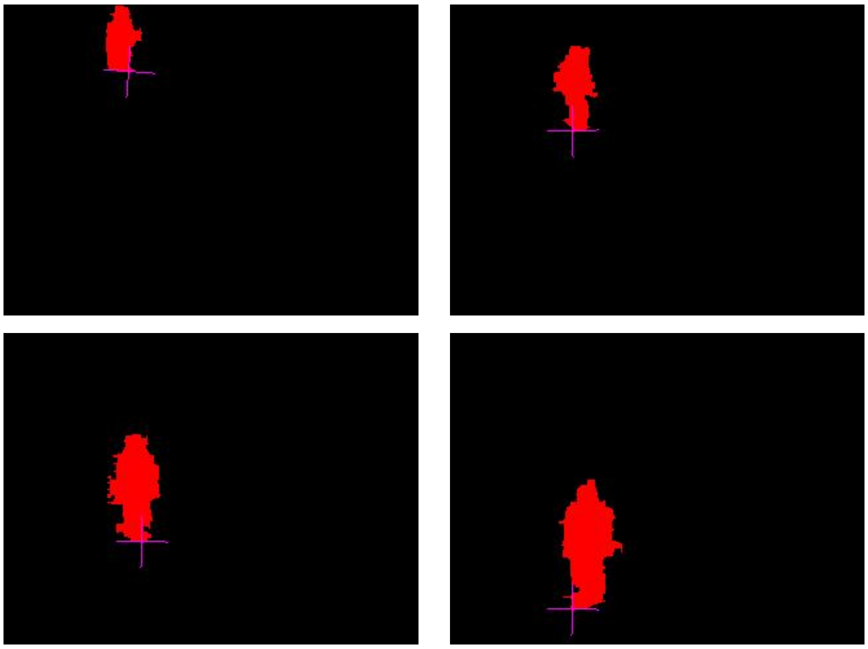

Figure 5.2: The walking example, back-ground subtraction and tracking.

Figure 5.3: The walking example, detection of an unusual event.

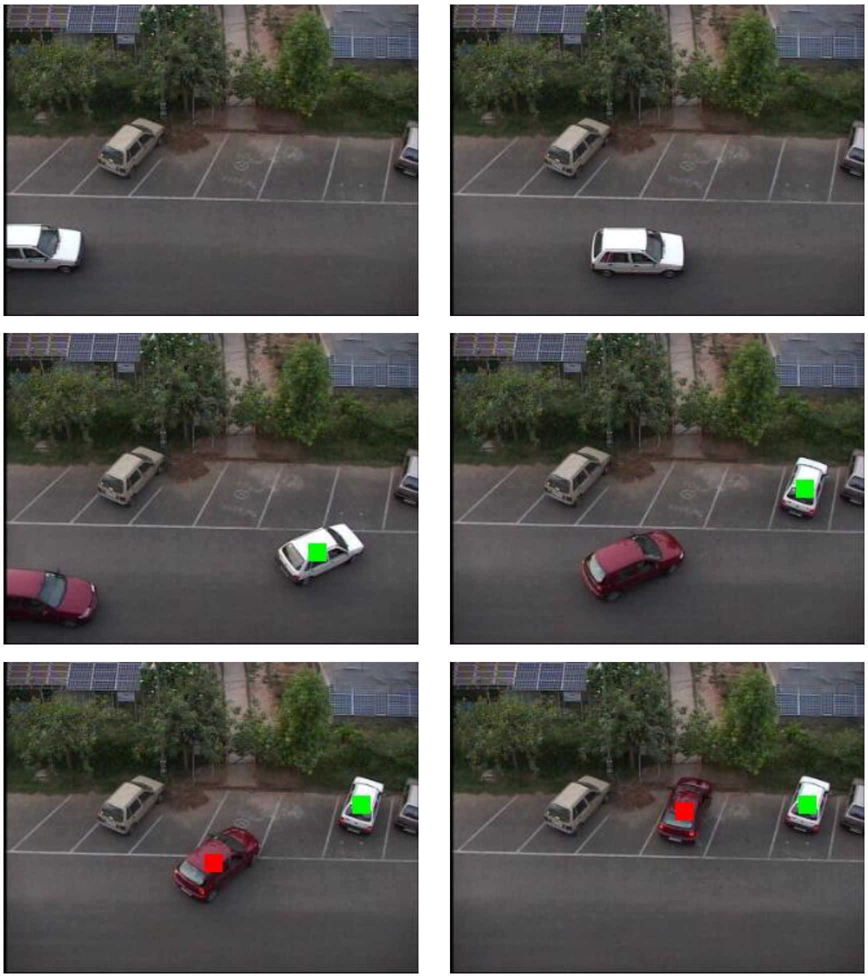

Figure 5.4: Frames from a car-parking example.

Figure 5.5: An example of detection of possible car theft.

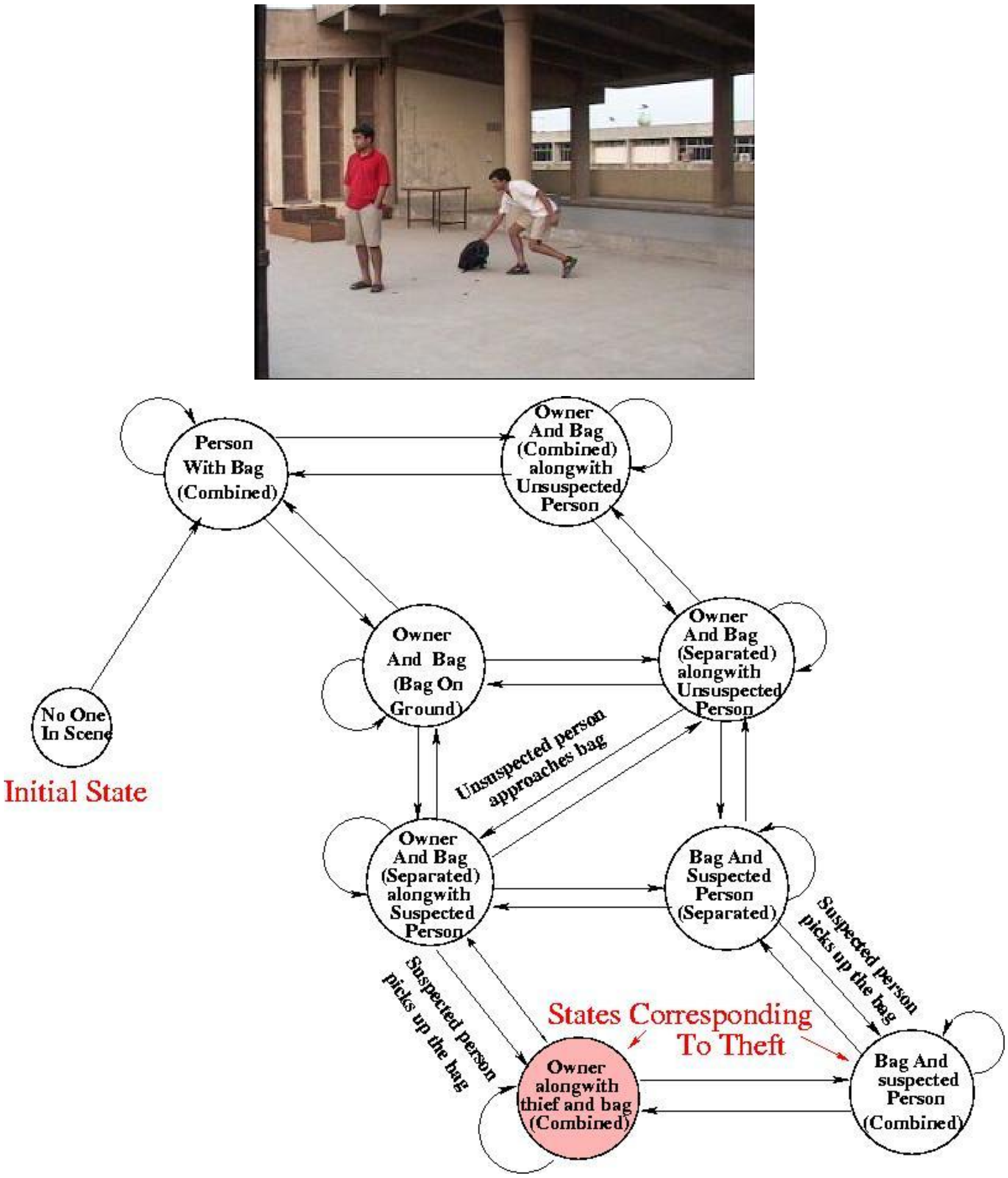

Figure 5.6: Explicit programming of the stealing example.

Chapter 6

Conclusion

We have presented a new framework for unsupervised learning of usual activity patterns and detection of unusual activities based on a network of finite state machines. The FSM framework is simple yet powerful and can reliably capture complex and concurrent inter-object interactions. Our preliminary results demonstrate the potential of the approach. In our present implementation, we do not handle the cases of complex image processing and tracking under occlusion. In future we plan to include such cases and test the framework rigorously under dense situations with many activities happening simultaneously.

Bibliography

[1] J. K. Aggarwal and Q. Cai. Human Motion Analysis: A Review. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 21(11):123 – 154, 1999.

[2] H. Buxton and S. Gong. Advanced visual surveillance using bayesian networks. In *Proc. Intl. Conf. Computer Vision*, pages 111 – 123, 1995.

[3] A. R. Chowdhury and R. Chellappa. A Factorization Approach for Activity Recognition. In *CVPR Workshop on Event Mining*, 2003.

[4] I. Cohen and G. Medioni. Detecting and Tracking Moving Objects for Video Surveillance. In *Proc. Computer Vision and Pattern Recognition*, 1999.

[5] R. T. Collins and et. al. A system for video surveillance and monitoring. Technical report, CMU-RI-TR-00-12, 2000. Also in Special section on video surveillance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), August 2000.

[6] D. Gibbins, G. N. Newsam, and M. Brooks. Detecting Suspicious Background Changes in Video Surveillanceof Busy Scenes. In *Proc. of IEEE Workshop onApplications in Computer Vision*, pages 22 – 26, 1996.

[7] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge University Press, 2000.

[8] S. Hongeng, F. Bremond, and R. Nevatia. Representation and Optimal Recognition of Human Activities. In *Proc. Computer Vision and Pattern Recognition*, pages 1818 – 1825, 2000.

[9] S. Hongeng and R. Nevatia. Multi-agent Event Recognition. In *Proc. Intl. Conf. Computer Vision*, pages 84 – 93, 2001.

[10] M. V. Hua Zhong, Jianbo Shi. Detecting Unusual Activity in Video. In *Proc. Computer Vision and Pattern Recognition*, page (to appear), 2004.

[11] Y. Ivanov and A. Bobick. Recognition of Visual Activities and Interactions by Stochastic Parsing. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 22(8):852 – 872, August 2000.

[12] V. Kettnaker. Time-dependent HMMs for Visual Intrusion Detection. In *IEEE Workshop on Event Mining: Detection and Recognition of Events in Video*, June 2003.

[13] A. Madabhushi and J. Aggarwal. A Bayesian Approach to Human Activity Recognition. In *Proc. of the 2nd International Workshop on Visual Surveillance*, pages 25 – 30, 1999.

[14] G. Medioni, I. Cohen, F. Bremond, S. . Hongeng, and R. Nevatia. Event Detection and Analysis from Video Stream. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 23(8):873 – 889, 2001.

[15] D. Moore, I. Essa, and M. Hayes. Exploiting Human Actions and Object Context for Recognition Tasks. In *Proc. Intl. Conf. Computer Vision*, pages 80 – 86, 1999.

[16] N. Oliver, E. Horvitz, and A. Garg. Layered Representations for Human Activity Recognition. In *Fourth IEEE Int. Conf. on Multimodal Interfaces*, pages 3 – 8, 2002.

[17] N. Oliver, B. Rosario, and A. Pentland. A Bayesian Computer Vision System for Modeling Human Interactions. In *Proc. of ICVS*, pages 255 – 272, 1999.

[18] Y. Shi and A. Bobick. Representation and Recognition of Activity using Propagation Nets. In *Proc. of 16th Intl. Conf. on Vision Interface*, 2003.

[19] T. Starner and A. Pentland. Visual Recognition of American Sign Language using Hidden Markov Models. In *Intl. Workshop on Automatic Face- and Gesture-Recognition*, 1995.

[20] C. Stauffer and W. Grimson. Learning Patterns of Activity using Real-time Tracking. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 22(8):747 – 757, 2000.

[21] N. Vaswani, A. R. Chowdhury, and R. Chellappa. Activity Recognition Using the Dynamics of the Configuration of Interacting Objects. In *Proc. Computer Vision and Pattern Recognition*, pages 633 – 640, 2003.

[22] J. Yamato, J. Ohya, and K. Ishii. Recognizing Human Action in Time-Sequential Images using Hidden Markov Model. In *Proc. Computer Vision and Pattern Recognition*, pages 379 – 385, 1992.

[23] L. Zelnik-Manor and M. Irani. Event-based video analysis. In *Proc. Computer Vision and Pattern Recognition*, 2001.