ABSTRACT

Title of dissertation: STATE SPACE APPROACHES FOR MODELING ACTIVITIES IN VIDEO STREAMS

Naresh P. Cuntoor
Doctor of Philosophy, 2006

Dissertation directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

The objective is to discern events and behavior in activities using video sequences, which conform to common human experience. It has several applications such as recognition, temporal segmentation, video indexing and anomaly detection. Activity modeling offers compelling challenges to computational vision systems at several levels ranging from low-level vision tasks for detection and segmentation to high-level models for extracting perceptually salient information. With a focus on the latter, the following approaches are presented: event detection in discrete state space, epitomic representation in continuous state space, temporal segmentation using mixed state models, key frame detection using antieigenvalues and spatio-temporal activity volumes.

Significant changes in motion properties are said to be events. We present an event probability sequence representation in which the probability of event occurrence is computed using stable changes at the state level of the discrete state hidden Markov model that generates the observed trajectories. Reliance on a trained

model however, can be a limitation. A data-driven antieigenvalue-based approach is proposed for detecting changes. Antieigenvalues are sensitive to turnings whereas eigenvalues capture directions of maximum variance in the data. In both these approaches, events are assumed to be instantaneous quantities. This is relaxed using an epitomic representation in continuous state space.

Video sequences are segmented using a sliding window within which the dynamics of each object is assumed to be linear. The system matrix, initial state value and the input signal statistics are said to form an epitome. The system matrices are decomposed using the Iwasawa matrix decomposition to isolate the effect of rotation, scaling and projection of the state vector. It is used to compute physically meaningful distances between epitomes. Epitomes reveal dominant primitives of activities that have an abstracted interpretation. A mixed state approach for activities is presented in which higher-level primitives of behavior is encoded in the discrete state component and observed dynamics in the continuous state component. The effectiveness of mixed state models is demonstrated using temporal segmentation. In addition to motion trajectories, the volume carved out in an xyt cube by a moving object is characterized using Morse functions.

STATE SPACE APPROACHES FOR MODELING ACTIVITIES IN VIDEO STREAMS

by

Naresh P. Cuntoor

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
2006

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Professor P. S. Krishnaprasad
Professor Carol Espy-Wilson
Professor Min Wu
Professor Larry S. Davis

Dedication

To my parents.

Acknowledgements

I have been extremely fortunate in having Professor Rama Chellappa as my advisor. I am deeply grateful for his continued support and guidance.

I am especially appreciative to my committee members Professors P. S. Krishnaprasad, Carol Espy-Wilson, Min Wu and Larry Davis for their suggestions and comments on my research. I thank my teachers and mentors at different institutions for their generosity and encouragement that have contributed in large measure to my journey in graduate school. In particular, I am grateful to Professors B. Yegnanarayana, A. N. Rajagopalan, David Jacobs and Richard Schwartz for introducing me to new research areas and spending time in technical discussions.

My warm appreciation goes out to all my fellow students at CfAR and the ECE department for their friendship and support. It is impossible to acknowledge all my friends individually, but I would like to thank Dr. Namrata Vaswani, Dr. Amit Kale, Dr. Volker Krueger, Dr. Kevin Zhou, Feng Guo, Seong-wook Joo, Pavan Turaga, Gaurav Aggarwal, Ashok Veeraraghavan, Narayanan Ramanathan, Aravind Sundaresan, James Sherman Jr., Aswin Sankaranarayanan, Mahesh Ramachandran, Brinda Ganesh and Ninad Jog.

I am indebted to my parents for their enduring love, confidence, patience and inspiration that has made this possible. I am most grateful to my brother for his affection.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The objective of human activity modeling is to develop a system that can discern events and behaviors using video sequences. Learning activity models is useful for several applications including surveillance and monitoring in urban environments, detection of anomalous activities and designing assistive technology for improving the quality of life of elderly citizens. The urgency for effective activity modeling continues to grow with the increasing amount of video data that is available. There are three main sources of video streams: surveillance scenarios (buildings, commercial establishments, transportation centers, city streets), industrial sources (manufacturing processes, rail track monitoring) and home-videos (as evidenced by the phenomenal popularity of youtube.com). All these applications may be said to be *passive* in which the system recognizes, indexes or raises an alarm after processing the input video stream. In contrast, learning activity models can be useful for *active* applications in which the system generates new activities using learnt models. For example, models can be used for motion planning and robotic navigation.

There are several challenges in developing effective activity models. Low-level video processing challenges limit the amount of information that can be extracted from video streams. For instance, tracking objects in surveillance scenarios contin-

ues to be encumbered by poor contrast between foreground and background, and noise. As a result, it may not be possible to reliably obtain trajectories in far field surveillance scenarios. It is desirable to build systems that are robust to low level limitations. Many high-level challenges stem from semantic equivalence, i.e., an activity could produce trajectories whose appearance varies drastically. We as humans, may recognize the activity in spite of drastically varying conditions. It is difficult to automate this task without using domain knowledge during modeling.

Domain knowledge can be readily incorporated in semantic and ontological approaches to activity modeling. Ontology involves manually specifying concepts related to an activity. This intervention for each activity however, is tedious and does not lend itself to easy generalization. Moreover, it relies on the expert's ability to predict all possible scenarios in an activity. Our work focuses on developing statistical models that detect events and behaviors.

At the outset, it may be worth clarifying terms such as actions, activities and events. Many existing approaches distinguish between actions and activities depending on the scale of representation ([1],[2]); i.e., individual parts of the body are said to perform actions such as picking up and putting down objects, whereas human interaction with the environment is said to constitute activities such as those in surveillance scenarios. We do not make this distinction and use the term activities to denote both these cases. The distinction is not crucial since we are interested in applications such as recognition and anomaly detection irrespective of whether they are actions or activities. In [3], activities are decomposed as a sequence of primitive segments. Primitives can be considered as building blocks of activities

that are spread over an interval of time [4]. This is analogous to decomposing speech into phonemes, which are the building blocks of the speech signal. We use the term events to denote instantaneous entities that can be thought of as boundaries between primitive segments. Perceptually significant changes in motion properties are said to represent events. Events are assumed to be instantaneous quantities unlike behaviors that can persist over a time interval.

1.1 Overview of the dissertation

An overview of the dissertation and its organization is described next. This is followed a brief discussion of related work.

1.1.1 Events in a discrete state space setting

Many commonly performed human activities can be regarded as a sequence of certain important events. If the events occur in a particular sequence, then the activity is said to have taken place. We model the event detection problem in a discrete state space setting using motion trajectories of objects as the observed data sequence. An event representation called the event probability sequence is presented in chapter 2 using the hidden Markov model (HMM) framework.

The choice of an appropriate scale of event representation may not be obvious in a given scenario. Constructing a coarse to fine hierarchy of events is a powerful way to address this challenge. We describe goodness criteria and quantitative measures to judge the efficacy of a coarse to fine hierarchical event representation.

1.1.2 Epitomic representation in continuous state space

In chapter 3, an epitomic representation of activities is presented in a continuous state space setting. Motion trajectories of objects within video segments are modeled using linear dynamical systems. The estimated system matrix, initial value of the state and the input signal statistics (mean and covariance) are said to form an epitome. The epitomes are analyzed using the Iwasawa matrix decomposition that allows us to compute geodesic distances between activities that are perceptually informative. The Iwasawa decomposition gives three components that represent rotation, scaling and projective action of the state vector. Besides their application for activity recognition, the usefulness of the decomposition is illustrated for applications such as clustering and key frame detection.

1.1.3 Mixed state model

Quantities such as velocity and acceleration of objects in the scene are continuous valued, whereas behavior (start, stop, etc.) and instantaneous events (grasp, turn, etc.) are discrete-valued. This motivates the need for mixed state systems that can handle both continuous and discrete valued states. In chapter 4, a mixed state model along with its application for video indexing is described. It can characterize multiple moving objects in the scene and adapt to changing behavior. A basis of behaviors is introduced so that domain knowledge can be used in developing activity models.

1.1.4 Key frames using antieigenvalues

In chapter 2 the space of allowable events was confined to a combination of a pair of (unequal) states. This limits the number and nature of events that can be represented. In chapter 5, we present a way to overcome this limitation using antieigenvalues. Antieigenvalues measure changes in the data unlike eigenvalues that extracts dominant characteristics in the data.

1.1.5 Modeling spatio-temporal volumes

The previous methods have focussed on modeling motion trajectories. Alternatively, activities can be represented using spatiotemporal volumes or xyt cubes that are formed by stacking frames in a video sequence. Objects moving in the scene carve out patterns in the volume. In chapter 6 an algebraic way of characterizing the topology of spatio-temporal volumes using Morse functions is presented. We propose a Morse function based on dynamics of moving objects and use the critical points and the associated critical values as a signature of the activity.

1.2 Related Work

In recent years research in activity modeling has grown from modeling simple activities to complex ones involving semantics. A few related works are mentioned below. A more detailed review may be found in [5].

1.2.1 Semantic approaches and event modeling

Event recognition can be traced back to an early work in the field of artificial intelligence where Tsuji *et al.* [6] used simple cartoon films for representation. Neumann and Novak [7] proposed a hierarchical representation of event models. At each level, a template of a verb of locomotion is matched to the observed data. More recently, Kojima *et al.* [8] proposed a natural language representation of human actions based on extracted positions of the hand, head and body. A hierarchy of verbs was built using case frames. Vu *et al.* [4] defined a description of activity consisting of actors, logical predicates and temporal relations between sub-events. Zelnik-Manor and Irani [9] considered events as long-term temporal objects at multiple temporal scales. The chi-squared distance between empirical distributions is used to compare these event sequences. Chan *et al.* [10] extract spatial semantic primitives from the observed data. This procedure uses domain knowledge and a pre-specified set of primitives.

Medioni *et al.* [11] developed a system that includes techniques for object detection, tracking and recognition of multi-agent events. Pre-specified features such as distance, direction of heading and speed were defined for various scenarios. Francois *et al.* [12] described a framework for video event representation and annotation. The Video Event Representation Language (VERL) and the Video Event Markup Language (VEML) were introduced to describe ontologies of activities, and manually annotate events. The user has the freedom to define events and choose the desired level of granularity in representation. In other words, their framework uses

a designed semantic structure to describe activities.

The large amount of data generated by the escalating ubiquity of video cameras poses a significant challenge to semantic approaches. Unlike statistical approaches, semantic ones are limited in their ability to generalize to new scenarios. This is one of the reasons for the increasing popularity of statistical techniques.

1.2.2 Statistical approaches

Johnson and Hogg [13] presented a neural network approach to learn the distribution of motion trajectories. They used vector quantization to cluster trajectories into a known number of classes. The cluster centers were updated by annealing. This approach was developed further in [14] for robust tracking and anomaly detection using a fast fuzzy k-means algorithm. As the number of trajectories and activities increases, comparing raw trajectories may be computationally expensive. Also, it ignores the structure inherent in activities.

Stauffer [15] proposed a factored learning approach to classifying surveillance-type data. These techniques accumulate motion trajectories in the scene to produce an intuitively appealing map of trajectories. Similar to [15], we do not attempt to track objects over the entire video sequence. Rather motion trajectories are extracted for video segments.

Motion features can be selected depending on activities of interest. For example, Parameswaran and Chellappa [16] computed view invariant representations for human actions in both 2D and 3D. In 3D, actions were represented as curves in an

invariance space and the cross ratio is used to find the invariants. Rao *et al.* represented actions as a sequence of *dynamic instants* [1] that are points of maximum curvature along the motion trajectory. Curvature was chosen as a feature to derive view-invariant representations. Similarly, in [17], changes in velocity curve profiles of actions have been used to segment actions in videos streams. Syeda-Mahmood *et al.* [18] used generalized cylinders to represent actions. Assuming that the start and end points are known, they formulated the task as a joint action recognition and fundamental matrix recovery problem. Reliance on features such as curvature can limit the domain of applicability. For example, in an airport tarmac surveillance scenario, trajectories formed by passengers walking from gate to aircraft can follow a straight-line path. So this approach is not entirely applicable for our case. Another limitation is that trajectories may contain only a few points of high curvature making it possible to encode only a few activities.

An unsupervised system for classification of activities was developed by Stauffer and Grimson [19]. Motion trajectories collected over a long period of time were quantized into a set of prototypes representing the location, velocity, and size of the objects. Assuming that the sequence of prototypes in each trajectory consist of a single activity, a co-occurrence matrix representing the probability that a pair of prototypes both occur in the same sequence was estimated and used for classification.

Vaswani *et al.* [20] regarded a sequence of moving points engaged in an activity as a shape using Kendall's shape space theory. Shape is defined as the geometrical information that remains after filtering out the effects of translation, rotation and

scale. Procrustes distance between two activities was used to check for anomalous trajectories. Dynamics are modeled in the shape's tangent space using a first order Gauss-Markov model. This can only model small changes about the mean activity shape. It may be necessary to preserve distinctions arising from factors such as rotation to distinguish between activities. For example, shape representation fails to distinguish between trajectories of persons embarking an aircraft from those of persons disembarking the aircraft.

Factorization approach, originally proposed by [21] and extended to non-rigid shapes by [22], was used in [23] to model activities. They used rank constraint in the factorization theorem to distinguish between trained models. An activity is said to be anomalous if the projections on the basis shapes do not cluster as expected. They demonstrated the effectiveness of the approach for repeated activities such as passengers embarking an aircraft. As the authors note, however, it cannot be used to recognize ground crew movement because of drastic variations across samples.

Caspi *et al.* [24] presented a sequence matching approach that incorporates shape information instead of relying solely on interest points. Based on the extracted motion trajectories, they compute several motion properties that depend on activities of interest. The features are used to match the given unsynchronized video sequences. Our method is similar in principle, in that it computes a mid-level representation based on space-time trajectories. A major difference, however, is that features in [24], which form the mid-level representation, have to be manually specified depending on activities of interest. We attempt to recover structure that is inherent in activities by analyzing the geometrical structure of epitomes.

1.2.3 Graphical models

HMMs and dynamic Bayesian networks (DBNs) have been widely used to model activities. HMMs have been successfully applied in many vision and speech applications ([25],[26],[27]). Usually, the HMM is used as a statistical representation, without an explicit physical interpretation of its parameters ([2],[25], [28], [29]).

In DBNs, states are specified using domain knowledge ([30], [31]) and the graph structure is manually designed. These models clearly reflect semantics, but building (or altering) them can be tedious. Alternatively, the state space may be augmented ([32], [33]). The augmentation generally makes model estimation intractable. The main difference of the proposed approach is that, instead of attaching a semantic significance to states themselves, we use stable transitions between states of an HMM to represent significant changes in motion that may be interpreted as events. Tractability in parameter estimation and event detection is retained. Also, events are not chosen manually. Some related state space approaches are described below.

Brand *et al.* [32] used coupled HMMs to model actions involving body parts such as hands and head. The states of the HMMs representing the motion in different parts of the body are forced to evolve synchronously. In a general activity model, deciding which of the HMM states have to be coupled may not be obvious. Also, an increasing number of parts presents a computational challenge. Ivanov and Bobick [2] consider activities that are semantically defined, i.e., activities with structurally defined relationships of primitives. Their approach used a two-step process that involves an HMM stage to extract probabilities associated with primitives,

and a stochastic grammar parsing stage that recognizes the structure of the activity. The primitives are obtained using sub-HMMs that are trained using previously segmented data.

Koller and Lerner [34] described a sampling approach for learning parameters of a DBN. Hamid *et al.* [30] presented a DBN-framework for recognizing complex multi-agent activities. They used a feature-based particle tracker and a DBN for recognition. In general, if the structure of the activity changes, DBNs have to be completely re-estimated.

Brand and Kettnaker [35] recognized that the states of an HMM need not represent meaningful entities. They interpreted states as events by training the HMMs using entropy minimization rather than the Baum-Welch algorithm. The HMMs were designated as entropic HMMs. Oliver *et al.* [36] presented a layered HMM model for representing activities at multiple levels of temporal granularity. Shi and Bobick [31] presented a propagation-net (P-Net) model to incorporate duration modeling in a DBN (or HMM)-like framework.

Most existing activity modeling approaches use motion trajectories as features of interest ([1], [2], [20]). Alternatively, techniques in video summarization have used frame-based features such as color histograms ([37], [38], [39]). Motion trajectories offer the following main advantages. They contain temporally dense representation of motion that can be used to infer behavior. As shown in [1] and [16], it is possible to compute view invariant signatures of activities using trajectories.

Graphical models may be considered to be part of a broader category consisting of state space approaches. Some related works are described next.

1.2.4 State space approaches

Activity modeling can be viewed as a problem of inferring behaviors and interactions between moving objects using the observed motion trajectories (or extracted features). In [3], activities were modeled as a sequence of linear dynamical systems, as are the proposed epitomic representation. Unlike [3], we analyze patterns in epitomes that change as an activity progresses to find distance measures between them.

Mixed state models have been used for several applications including activity modeling, air traffic management, smart highway system, etc. ([40], [41], [42], [43]). In some of these applications such as [42], [43], the focus is on analyzing the mixed state systems where the model parameters are known (by design). On the other hand, we are interested in learning parameters of mixed state models, which is addressed in [40] and [41] as well. Unlike HMMs, parameter estimation in mixed state models is intractable. Isard and Blake presented a sampling technique for estimating a mixed-state model [40]. They assumed that the structure of the activities is known, and that the parameters are stationary. Ghahramani and Hinton described a varational method for learning [41].

1.2.5 Distance based on dynamics

The notion of distance between dynamics can be traced back to atleast the nineteenth century, when Jordan proposed principal angles between linear vector spaces [44]. Hotelling gave a statistical formulation using canonical correlations

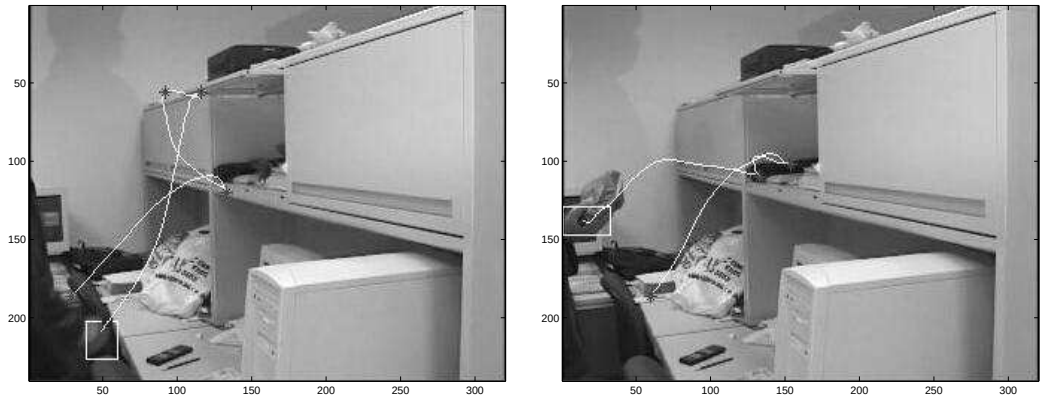
[45]. Based on principal angles, subspace angles have been defined to compare linear systems and ARMA models. De Cock and De Moor described various ways of computing subspace angles between linear systems [46]. It is defined as the principle angle between the column spaces generated by the observability matrices of two models. The Martin distance [47] uses principal angles to compare AR models. In vision literature, subspace angles have been used to measure similarity between dynamical models in [48] for recognizing humans based on gait. In [49], subspace angles between measurement matrices are used to find the distance between actions. We develop a distance metric for comparing activities using ideas, in part, from [47] and [46].

1.3 Datasets

We give a brief description of the datasets used to demonstrate our approaches to activity modeling. The datasets are chosen to illustrate the effectiveness of the approaches in both indoor and outdoor video sequences.

1.3.1 UCF human action dataset

The UCF dataset consists of 60 trajectories of common activities. We divide these into 7 classes: open door, pick up, put down, close door,erase board, pour water into cup and pick up object and put down elsewhere. The hand trajectories are obtained in a two step process in which the hand is first detected and then tracked. A skin detector is used to initialize the position of the hand in the first frame [50].



(a) (b)

Figure 1.1: Sample images from the UCF dataset. (a) Open cabinet door, (b) Pick up an Object. The stars indicate the *dynamic instants* detected by [1].

The hand is approximated as a point object using the centroid of the bounding box on the hand. A mean shift tracker is used to obtain the trajectories. The resulting trajectories are smoothed out using anisotropic diffusion [51] so that corners and sharp changes are retained. A detailed description is available in [1]. Figure 1.1 shows sample images from the dataset along with the trajectories. Typically, most of the activities in the dataset last for a few seconds i.e., of the order of 100 frames.

1.3.2 TSA airport tarmac surveillance dataset

The TSA dataset consists of surveillance video captured at an airport tarmac [20]. The stationary camera operates at approximately 30 frames per second and the frame size is 320×240. Though it is approximately 120 minutes long, a large portion of the video does not contain any activities. We divide the entire data into 23 blocks of about $10,000$ frames each. Here onwards, we refer to such sets of $10,000$ frames

Figure 1.2: Sample images from the TSA airport tarmac surveillance dataset.

as blocks. Moving objects are detected and tracked as described in section 4.2.1. The background at each pixel was modeled using a Gaussian distribution. The parameters are reinitialized every hundred frames. Each frame is compared with the background and the moving objects are detected. A bounding box is drawn on the detected blobs. The KLT algorithm is allowed to choose feature points for tracking within the bounding box. The average trajectory of feature points within the bounding box is regarded as the motion trajectory of the objects. Since the video sequence is long, it is impractical to obtain ground truth of trajectories. The activity model needs to be robust to imperfections in tracking. The ground truth for temporal segmentation was extracted manually, i.e., by direct inspection of the video sequences. Figure 1.2 shows sample images from the dataset along with the trajectories extracted.

1.3.3 Bank surveillance and monitoring dataset

The bank dataset consists of staged videos collected at a bank [52]. There are four sequences, each approximately 15-20 seconds ($\sim$ 400 frames) long. Figure

(a) (b)

Figure 1.3: Sample images from the bank surveillance and monitoring dataset.

1.3 show sample images from the dataset. Moving objects are detected and tracked using the same procedure as in section 1.3.2.

1.3.4 Motion Capture (MOCAP) dataset

The MOCAP dataset available from Credo Interactive Inc. and CMU consists of motion capture data of subjects performing different activities including different kinds of walking, jogging, sitting, and crawling. The system tracks 53 joint locations during an instantiation of the activity and the tracks are stored in bvh format. Since not all the 53 points are relevant to the types of activity that we are interested in, we use only a few of the trajectories. For example, trajectories of the different fingers and toes may not be as informative as the location of the arms, legs or hip for activities such as walking or sitting. We choose five such regions to demonstrate activity classification: head, neck, shoulders, hands and feet. Another practical reason for choosing a subset of the available trajectories is the limited number of

observations per activity.

1.4 Contributions of the dissertation

1. We propose event probability sequences to model activities as a sequence of important instantaneous events. Its effectiveness is demonstrated using activity recognition and anomaly detection.

2. We introduce an epitomic representation using linear dynamical systems as building blocks. The tuple consisting of the system matrix, initial state value and input signal statistics is said to be an epitome. We presente star diagrams as a way of visualizing the epitomes. Star diagrams reflect similarities between motion trajectories of activities, which may not be seen at the motion trajectory level because of semantic ambiguity.

3. We propose using the Iwasawa matrix decomposition to decompose the effect of epitomes on the state vector. It is used to isolate the effects of rotation, scaling and projective action on the moving object. The efficacy of the components is illustrated using clustering and key frame detection.

4. We develop a new distance metric to compare activities using the Finsler-Minkowski metric as the infinitesimal metric. The distance is computed component-wise using the factors in the Iwasawa matrix decomposition.

5. We propose the use of antieigenvalues for detecting key frames in activities. Antieigenvectors are sensitive to turnings in the data, whereas eigenvalues

capture the direction of maximum variance.

6. The volume carved out in space and time by moving objects is considered as a feature to model activities. We present Morse functions as a way of characterizing the topology of the spatio-temporal volumes.

7. Scale Invariant Feature Transform (SIFT) features are shown to be equivalent to a special choice of Morse functions on the space of image volumes in which images $(x, y, I(x, y))$ are considered as 3D objects, where $I(x, y)$ is the intensity value at pixel (x, y).

Chapter 2

Event Probability Sequences

2.1 Introduction

Human activities can be decomposed into a sequence of events that have a natural physical interpretation. This can be accomplished using semantic approaches ([2], [4], [8], [11], [12]) in which events are pre-specified. Alternatively, they can be modeled using statistical approaches ([25], [28], [31], [32], [34], [53]), in which modeling is viewed as a problem of inferring (hidden) events from observed features (e.g., motion trajectories). We present a statistical approach for modeling activities as a sequence of events.

Events can be defined based on certain dominant and persistent characteristics of the data. For example, events can be associated with key frames or exemplars. On the other hand, they can be defined using significant changes in data. Change-based events, in addition to providing an effective way of representing events are naturally suited to anomaly detection. By nature, events based on changes are associated with both past and future states of an object, and thus provide a more complete local representation of activities. Change can be defined in several ways – changes in direction of motion, appearance, etc. The goal of activity modeling is to find stable changes from observed data that can be interpreted as events.

We propose an event detection technique using the HMM framework that

focusses on stable state transitions under the hypothesis that certain state-level transitions denote events. In most of the existing approaches that use HMMs, the focus is either on finding a compact model to represent time series data ([25], [26]) or finding a representation in which states may be interpreted as meaningful entities ([2], [29], [35], [31]). Both these types of approaches emphasize the optimal state sequence, which maximizes the likelihood of generating the given data. It is not necessary, however, for the optimal state sequence to contain physically meaningful information for representing events.

The proposed method differs from the above-mentioned methods in that we attempt to interpret certain stable transitions at the state level as events. So the optimal state sequence is less important for event detection. During event detection, several state sequences of the HMM are explored as follows. Consider an observed data sequence (e.g., motion trajectory) of length T. In an ergodic HMM with N states, there are N^T possible state sequences, each of which can generate the observed data with some probability. The optimal state sequence which is one among these state sequences, maximizes this likelihood. Given an observation sequence and a learned HMM, an event probability sequence $\{e_t^p, t = 1, \ldots, T, p = 2, \ldots, P\}$ is computed in which a local maximum denotes a stable transition at the state level. Here p denotes the scale parameter. An efficient way of exploring state sequences to compute $\{e_t^p\}$ is described.

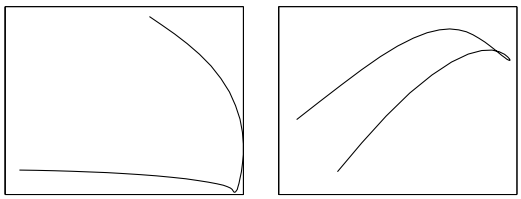
It is desirable to find an event representation of activities that are invariant to changes in viewing conditions. The HMM, however, is view-dependent since it is trained using apparent 2D motion trajectories. So multiple HMMs are required if

the appearance of motion trajectories changes significantly. The usefulness of event probability sequences is illustrated using activity recognition and anomaly detection in both indoor and outdoor scenarios. In many of these scenarios, it is common to have several samples of normal activities, but very few anomalous cases. It is not practical to model all possible anomalous activities, some of which can arise due to subtle, statistically insignificant deviations from normal cases. Events in the proposed methods can be used to detect such anomalies since they are a result of local changes, and is largely unaffected by rest of the activity.

The rest of the chapter is organized as follows. Section 2.2 motivates activity representation as a sequence of instantaneous events and section 2.3 describes the proposed event probability sequences. An efficient way of computing event probabilities is described in section 2.4. Section 2.5 discusses view invariance of events. Section 2.6 demonstrates the usefulness of the proposed approach for recognition and anomaly detection using the UCF human action dataset, the CMU motion capture dataset and the TSA airport surveillance dataset.

2.2 Motivation

In this section, typical activities are described to motivate a representation of activities as a sequence of events. Consider an office environment, in which activities as opening and closing doors, picking up and putting down objects, and sitting and standing are commonly observed. Figure 1.1 shows sample images depicting such activities, along with automatically extracted motion trajectories of



(a) (b)

Figure 2.1: (a) Motion trajectory of picking up an object from the desk and (b) picking up an object in the cabinet.

the hand. Figure 2.1 shows trajectories of picking up an object. Though the two trajectories represent the same activity, the variation in appearance is significant. The high degree of intra-class variability poses a challenge to existing approaches like the HMM. Moreover, for the activity *pick up an object* to occur, the time instant associated with the picking up the object is more important than the rest of the trajectory. This suggests a representation that can highlight important time instants when events such as *start*, *pick up* and *stop* occur. At a finer resolution, we may say that the sequence of events *extend hand - make contact with object - pick up - withdraw hand* forms the activity. Each of these events represent a significant change in the motion trajectory that has a semantic interpretation. These changes are highlighted as peaks in the event probability sequence in the proposed model. Similarly, a typical activity in a parking lot may be represented as a sequence of events: *exit building - enter parking lot - enter car - exit parking lot.*

In an airport tarmac surveillance scenario, common activities include passengers embarking and disembarking an airplane. One may be interested in not only recognizing the activity, but also in detecting anamalous activities. Anomalies can

be of various types: spatial, temporal, contextual, etc. It is not practical to enumerate all possible anomalies, and check for them. An alternative approach is to model normal activities and declare deviations from the normal model as anomalies. This too, has limited applicability since anomalies can result from subtle deviations from the normal trajectories that need not be statistically significant. Also, anomalies can be localized in time and space so that a major portion of the activity would appear normal. The anomalous part can be confined to a small region (in time and space) of the activity.

Activities can be represented as a sequence of events that are localized in time and space, as illustrated in the examples above. We propose a representation called event probability sequences for this task. Events are detected based on stable, local changes in motion properties that can be semantically interpreted.

2.3 Event probability sequence

In this section, we describe a probabilistic event representation that quantifies the notion of important characteristics of activities. Every motion trajectory in the scene is associated with an event probability sequence that is computed using a two-step process: an HMM is learnt using the given motion trajectories, and event probability sequences are computed using the learnt HMM and given motion trajectories. One of the main advantages of retaining the HMM learning step is that it allows for easy generalization, i.e., the structure of the model need not be manually specified for different activities. Using the learnt HMM, we explore a subset of the

state sequences to detect events. The hypothesis is that significant changes in the video sequence are reflected as events; and a sequence of events forms an activity.

Let $O = \{o_1, o_2, \ldots, o_T\}$ represent the motion trajectory (observed sequence) of an object for T frames, where o_t is a 2-vector of the pixel location at frame t. The observed sequence O is assumed to be generated by an HMM whose hidden state sequence is denoted by $Q = \{q_1, q_2, \ldots, q_T\}$. Here $q_t \in \{1, \ldots, N\}$, where N is the number of states. For every $t = 1, \ldots, T$, q_t can take any of N discrete values. So there are N^T possible state sequences that can generate the observed motion trajectory O. Among these N^T state sequences, the optimal state sequence maximizes the probability that the given motion trajectory is observed. Instead of the optimal state sequence alone, we explore other state sequences to detect events.

The key idea is that stable transitions at the state level reflect significant changes in motion properties that are denoted as events. State-level transitions provide a robust representation of change compared to those defined at the data-level. Moreover, the number of distinct changes at the state level at any given time is finite (and equal to $N^2 - N$), and its probability of occurrence can be computed efficiently. The simplest change at the state level is passing from state i at time t to state j at time $t + 1$ (denoted by $i \rightarrow j$). In the example of picking up an object (figure 2.1), there is a significant change associated with the *pick up* event when the hand, after having made contact with the object reverses direction and withdraws. The *pick up* event can be said to be a semantic change in state from state i before picking up the object to state j after picking up the object. The goal of activity modeling is to detect these changes in motion trajectories extracted from

video sequences without an explicit semantic model.

The probability of state change $i \rightarrow j$, conditioned on the observed trajectory, can be expressed [26] as:

$$\xi_t(i,j) \quad = \quad P(q_t = i, q_{t+1} = j | O, \lambda) \tag{2.1}$$

for states $i, j \in \{1, 2, \ldots, N\}$. $\xi_t(i,j)$ is used in the Baum-Welch algorithm to efficiently compute the HMM parameters [26].

The change in (2.1) is based on one time-step before and after an event's occurrence. So it accounts for the hand's state one frame before and after the *pick up* event. For stable transitions at the state level, instead of transitions of the form $i \rightarrow j$ in (2.1) is modified so that the probability of change associated with the following state transitions $i \rightarrow i \rightarrow j \rightarrow j$ is used, where the inner $i \rightarrow j$ transition occurs at time t [54]. In other words, given the observed trajectory, the probability of its generation by the following state sequence is computed: persistence in state i for two consecutive frames, then a transition to a state j and persistence in state j for two consecutive frames. This uses the following sequence of states to detect an event at time t:

$$\underbrace{i \rightarrow i}_{\text{2 frames}} \quad \rightarrow \underbrace{j \rightarrow j}_{\text{2 frames}} . \tag{2.2}$$

The event variable $\eta_t^2(i,j)$ is defined as follows (so that $\eta_t^1(i,j) \stackrel{def}{=} \xi_t(i,j)$):

$$\eta_t^2(i,j) \quad = \quad P(q_{t-1} = i, q_t = i, q_{t+1} = j, q_{t+2} = j | O, \lambda) \tag{2.3}$$

More generally, the region of support of an event can be extended from two

frames to p frames by considering a subset of state sequences that are constrained for $2p$ frames. Instead of the sequence of states in (2.2), $\eta_t^p(i,j)$ uses the following sequence of states:

$$\underbrace{i \rightarrow i \rightarrow \ldots \rightarrow i}_{p \text{ frames}} \rightarrow \underbrace{j \rightarrow j \rightarrow \ldots \rightarrow j}_{p \text{ frames}} \tag{2.4}$$

Event variable $\eta_t^p(i,j)$ for $p = 2, 3, \ldots, P$ is defined as follows.

$$\eta_t^p(i,j) = P(q_{t-p} = i, q_{t-p+1} = i, \ldots, q_t = i, q_{t+1} = j, q_{t+2} = j, \ldots, q_{t+p+1} = j | O, \lambda), \tag{2.5}$$

where $P \in \mathbb{N}$. At every time instant, there are N^2 transitions between pairs of states (i,j). Of these, there are $N(N-1)$ transitions between distinct (i,j). A transition from state i to j, where $j \neq i$ represents change. The most likely change among the $N(N-1)$ transitions may be interpreted as an event. The event probability, parameterized by scale parameter p, is defined as follows:

$$e_t^p(k,l) = \max_{i \neq j} \eta_t^p(i,j), \tag{2.6}$$

where $(k,l) = \arg\max_{i \neq j} \eta_t^p(i,j)$. As p increases, the region of support for computing an event increases. The *before* and *after* states k and l are said to characterize the type of the event.

There are several reasons for choosing events based on transitions. First, it is a simple and robust way of representing change. The event probability can be efficiently computed using the variables in the Baum-Welch algorithm (section 2.4.2). Though stable transitions at the state level yield events, the representation is tied to the underlying HMM. Since the HMMs are trained using 2-D motion

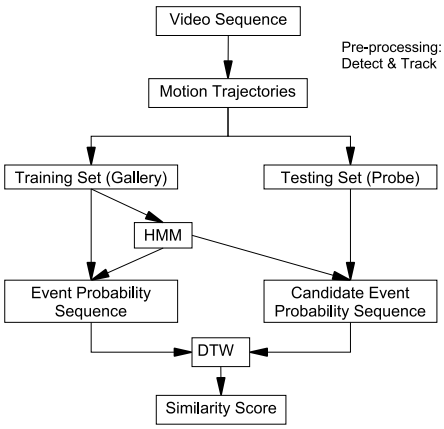


Figure 2.2: An outline of the event probability sequences for activity recognition

trajectories, their parameters are view-dependent. This, however, is not a significant limitation provided viewing conditions do not change drastically. Under certain assumptions, the event probability sequence is quasi invariant to changes in viewing conditions(section 2.5).

2.4 Approach

Figure 2.2 shows an overview of the proposed method. The training phase cconsits of three steps: pre-processing, HMM estimation and computing event probability sequences. Motion trajectories are extracted and used to train an HMM. Using the trained HMM, an event probability sequence is computed for every trajectory of the activity. Given a new trajectory (not used for training) in the testing phase, candidate event probability sequences are computed using each of the learnt HMMs so that there are as many candidate event probability sequences as the number of trained HMMs. Every candidate event probability sequence is compared with those

computed during the training phase using dynamic time warping [55] to handle slight changes in time alignment. These steps are described in the remaining part of this section.

2.4.1 Pre-processing

Given a video sequence, moving objects are detected and motion trajectories are automatically extracted. Motion trajectories of different objects in the scene are separately modeled. The details are described in section 2.6.

2.4.2 Efficient computation of event probability sequences

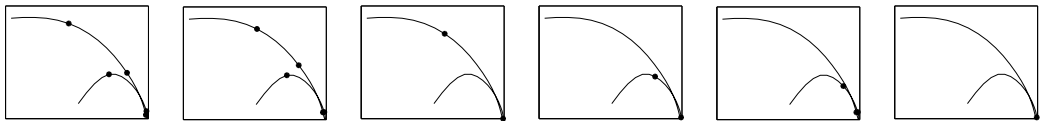
Let $\lambda = (A, B, \Pi)$ represent the HMM [26]. In our notation, we have tried to maintain consistency with the commonly-used HMM notation, e.g. as in [56].

- $A = [a_{ij}]$ is the probability transition matrix of size $N \times N$. The quantity $a_{ij} = P(q_t = j | q_{t-1} = i)$ denotes the one-step transition probability from state i to state j.

- B represents the probability of observing a given data vector conditioned on the current state. In our experiments, the (stationary) output distribution $b_j(o_t) = P(o_t | q_t = j)$ is assumed to be Gaussian, i.e., $b_j(o_t) \sim \mathcal{N}(o_t; \mu_j, \Sigma_j)$.

- Initial probability of states is given by $\Pi = [\pi_1, \ldots, \pi_N]$, where $\pi_i = P(q_0 = i | \lambda)$.

The parameters of the HMM are computed using the Baum-Welch algorithm (Appendix A, [26]). Using the learnt HMM, an event probability sequence is com-



(a) (b) (c) (d) (e) (f)

Figure 2.3: Events detected at different scales for *picking up* an object. The scale parameter p varies from (a) $p = 3$ to (f) $p = 8$.

puted for every motion trajectory.

An efficient algorithm for computing event probability η_t^2 is described below. From (2.3), we have

$$\eta_t^2(i,j) = \frac{P(q_{t-1}=i, q_t=i, q_{t+1}=j, q_{t+2}=j, O|\lambda)}{P(O|\lambda)} \tag{2.7}$$

$$= \frac{P(q_{t-1}=i, o_1^{t-1}, q_t=i, q_{t+1}=j, q_{t+2}=j, o_t^T|\lambda)}{P(O|\lambda)} \tag{2.8}$$

It is easy to show that (2.8) simplifies to

$$\eta_t^2(i,j) = \frac{\alpha_{t-1}(i)a_{ii}b_i(o_t)a_{ij}b_j(o_{t+1})a_{jj}b_j(o_{t+2})\beta_{t+2}(j)}{P(O|\lambda)}, \tag{2.9}$$

where $\alpha_t(i)$ and $\beta_t(j)$ are forward and backward variables defined as follows [26]:

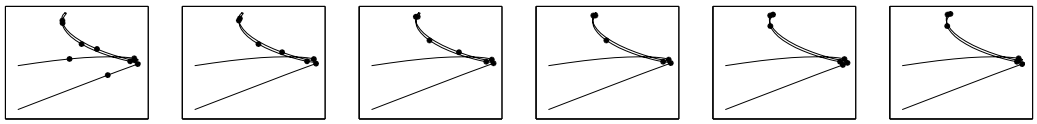
$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = i|\lambda) \tag{2.10}$$

$$\beta_t(j) = P(o_{t+1}, o_{t+2}, \ldots, o_T|q_t = j, \lambda) \tag{2.11}$$

Similarly, for $p \geq 1$,

$$\eta_t^p(i,j) = \alpha_{t-p}(i)a_{ii}^p b_i(o_{t-p+1})b_i(o_{t-p+2})\ldots b_i(o_t)a_{ij} \times$$

$$b_j(o_{t+1})b_j(o_{t+2})\ldots b_j(o_{t+p+1})a_{jj}^p b_j(o_{t+2})\beta_{t+p+1}(j)/P(O|\lambda) \tag{2.12}$$



(a) (b) (c) (d) (e) (f)

Figure 2.4: Events detected at different scales for *opening* the cabinet door. The scale parameter p varies from (a)$p = 3$ to (f) $p = 8$.

Events detected for different activities are illustrated using examples. Consider the activity of picking up an object. We may expect the relevant event to occur at the instant when the object is picked up. At a finer scale, we may think of events marking the start of the hand, the hand about to make contact, the hand making contact, the hand starting to withdraw along with the object and the withdrawal of the hand from the scene. The scale parameter p provides a control over the level of detail that we may be interested in. For example, when $p = 3$, *picking up* an object is characterized by the longer sequence of events described above, whereas for $p = 8$, the activity has one event that corresponds to the instant the object is picked up. Figure 2.3 shows the trajectories and the sequence of events detected at various scales. The events are represented by dots. Similarly, for *opening* the cabinet door, the number of events is related to the choice of scale. Figure 2.4 shows the trajectories and the sequence of events detected.

2.4.3 Parameter Selection

In this section, we discuss the choice of HMM model order N and the scale parameter p. Researchers have proposed several criteria to estimate the optimal

model order N^* for HMMs. We use the Bayesian Information Criterion (BIC) in our experiments since it has been shown to be a strongly consistent Markov order estimator [57]. The optimal value N^* is given by

$$N^* = \arg\min_{N \in \mathcal{N}} \log l(N) + \frac{k_N}{2} \log T \tag{2.13}$$

where $\log l$ is the negative log likelihood of the observed motion trajectory of length T and k_N is the degrees of freedom associated with the N^{th} order model. We describe two choices of scale parameter p: conditionally optimal p^* (conditioned on optimal N^*) and jointly optimal $(N, p)^*$.

2.4.3.1 Conditionally optimal scale parameter p^*

Using BIC, the scale parameter can be chosen as follows:

$$p^* = \arg\min_{p \in \mathcal{P}} (\Sigma_{t=1}^{t=T} e_t^p(k, l) + \frac{p}{2} \log T). \tag{2.14}$$

These two criteria for optimality, i.e., $N = N^*$ and $p = p^*$ imply that both the representation of trajectory and the sequence of events are optimal. This is a stronger requirement than finding the optimal scale of event representation. Instead of the optimal pair (N^*, p^*). there may be a pair of values $(N, p)^* = (N_1, p_1)$ that gives an optimal representation of event probabilities, where N_1 may be a sub-optimal order. This suggests a way to modify the optimality criterion to focus on the event probability sequences.

2.4.3.2 Jointly optimal parameters $(N, p)^*$

Events at certain key frames are said to represent the activity. This does not require the entire trajectory to be modeled optimally. So we can confine the penalized likelihood calculations to the key frames, or equivalently to the sequence of event probabilities $\{e_t^p, t \in [1, T]\}$. If the underlying HMM is ergodic, each event can be one of $N^2 - N$ possible types as given by (2.6). If a left-to-right model is assumed for the HMM, then the degrees of freedom for the possible types of events is halved. The overall penalty is $\alpha(N^2 - N) + p$, where $0 < \alpha \leq 1$ with equality for ergodic HMM. For a left-to-right model, $\alpha = \frac{1}{2}$. The event likelihood term associated with the pair (N, p) is $\sum_t e_t^p$. We discuss the behavior of event probability sequences for different values of N and p.

Case 1: Low value of N: Since only a few events are allowed to occur, we may not be able to obtain a sufficiently rich sequence of events to represent the activity. For example, with a 2-state HMM, only two types of events are allowed: $(1, 2)$ and $(2, 1)$, where 1 and 2 are state indices. Even at moderate p values, many activities may resemble each other leading to poor recognition performance.

Case 2: High value of N: This permits the occurrence of several types of events. Besides the number of observation sequences required to estimate a higher order HMM, there is the issue of over-modeling of events. The maximum value of scale parameter P has to smaller than that of *Case 1* to obtain a reasonable number of events since the required transitions in (2.6) may not occur for larger values of p.

2.4.4 Matching event probability sequences

Given a test sequence O_0, we extract the motion trajectory as before. Suppose there are R trajectories in the training set. Then there are R sets of event probability sequences that are computed using the respective HMMs for different values of p. In general, the number of distinct HMMs is less than the number of training trajectories. Multiple trajectories may be associated with the same HMM, but each of the trajectories is associated with an event probability sequence. We compute R candidate event probability sequences using O_0 as the observation sequence in (2.5) and (2.6).

The candidate event probability sequences are compared with the event probability sequences that were obtained during the training phase (at the same scale) to compute a similarity score. If there are P candidate event probability sequences computed for different scale parameters, we obtain P similarity scores. A direct frame-to-frame matching of event probability sequences is not realistic since the events need not occur at exactly the same time instants during different realizations of an activity. There have to be allowances for missed or spurious events as well. We use dynamic time warping (DTW) for computing the similarity score since it allows for non-linear time normalization [55].

A brief outline of the DTW algorithm is described here. The objective of the algorithm is to align the test sequence $\{x(t), t = 1, \ldots, T_1\}$ with the reference sequence $\{y(t), t = 1, \ldots, T_2\}$ so that the distance between the two sequences along the warping path $\{C(t), t = 1, \ldots, T_1\}$ is minimized. Both the signals $x(t)$ and

$y(t)$ can be vector-valued. A band of size $2W$ is defined along the diagonal, which contrains the warping region. Roughly speaking, this means that a test vector $x(t)$ can be matched with a one of the reference vectors in the sequence $\{y(t-W), \ldots, y(t+W)\}$. For every point within the warping region, the distance between the test and reference vectors are computed using a suitable norm (Euclidean, in our case). The warping path is found by the backtracking step.

2.5 View invariance in representation

In this section, we discuss the effect of changing viewing conditions on event probability sequences. The changes could be due to changes in position of the person performing an activity while the camera and rest of the objects in the scene are held fixed. Or, it could be due to camera motion. These two cases are discussed separately. The event probability sequence is shown to be a quasi view-invariant representation, and sufficient conditions on the structure of event generating HMMs are presented.

If the camera and objects in the scene are fixed, motion trajectories can appear different due to difference in style between persons performing the activity. The differences, however, are minimal at the time of occurrence of events. For instance, consider the example of activity of picking up an object performed by two persons. Due to changes in style, motion trajectories of the hands in the two cases may require different HMMs. The changes near *pick up* event, however, is similar in both the cases. As another example, consider an airport tarmac surveillance scenario. One

of the common activities is that of a luggage cart approaching a plane after arrival (and before departure). The luggage cart can enter scene in a wide area, but the event of interest occurs when it stops near the plane. This location is relatively unchanging though the rest of the trajectory of the luggage cart can vary drastically. So the event probability sequences can be expected to resemble each other (for an appropriate scale parameter) in spite of differences in motion trajectories in other parts of the activity. In other words, though the HMMs may differ for the two persons performing the same activity, they give rise to similar stable transitions that mark events of interest.

Changes in motion trajectories can occur due to camera motion, in which case the position of objects in the scene also changes unlike the former case. The parameters of HMMs depend on the viewing conditions. This is not a serious limitation as far as the comparing event probability sequences are concerned, which form a relative invariant. Fels and Olver [58] give formal definitions of absolute and relative invariants as follows. An invariant or absolute invariant of a transformation group G acting on a manifold M is a real valued function $I : M \to \mathbb{R}$, which is unaffected by group transformations, i.e., $I(g \cdot x) = I(x)$ for all $g \in G$ and all x in the domain of definition of I. A relative invariant of a weight μ is a function $R : M \to U$ which satisfies $R(g \cdot x) = \mu(g, x)R(x)$ for all $x \in M$, $g \in G$ where defined, U is a finite-dimensional vector space and μ is a smooth mapping. The HMM representation of the trajectories may be regarded as the weight function μ. The event computation plays the role of $R(x)$. We describe a sufficient condition on the structure of HMMs corresponding to different viewing directions to ensure that the event probability

sequence are view-invariant.

Assume that an affine camera model is used to relate the 3D trajectories to the 2D trajectories viewed from two different viewing points. The state descriptions represent the spatial context in the image plane. The transition matrix encodes the temporal evolution of the activity. It is reasonable to expect equivalence relations between the models corresponding to two different views. The idea is that such relationships can be captured by using the event probability sequence.

Definition 1: A pair of HMMs are said to be *conforming* if there exists a homeomorphism between the set of states of the two HMMs.

Proposition 1 (Sufficient condition for event sequence to be view invariant): For the event probability sequences to be invariant under changing viewing conditions, the associated HMMs must be conforming.

Proof: In appendix.

2.5.1 An example of view-invariance

It may be worth seeing the implications of the condition in the proposition through an example. In particular, it shows the effect of the condition being invalidated.

To illustrate view-invariance using the event probability sequence, consider a set of straight-line trajectories generated by a person walking from one part of the scene to some other fixed part. Assume that the camera axis (z-axis) is perpendicular to the 3D trajectory. Let the HMM contain two states, and be a left-to-right

model with a single Gaussian distribution per state. From (2.5), we observe that a sharp peak in the event probability sequence occurs at the instant of switching from state 1 to 2. Clearly, camera translation is not an issue. Rotation by an angle θ about the z-axis causes the 2-D trajectory to rotate by θ and the HMM parameters change correspondingly. In particular, the mean values of the two states also rotate by the same angle. The homeomorphism between the two trajectory models across viewing directions ensures that there exists a corresponding pairs of states (k, l) and $(\tilde{k}, \tilde{l})$ such that $\eta_t^p(i, j)$ and $\eta_t^p(\tilde{i}, \tilde{j})$ respectively are maximized. In other words, the event probability sequence is preserved. As the angle changes beyond 90^o, the roles of the two states are reversed.

Rotation with respect to the x-axis (tilt) does not change the 2-D trajectory. Camera rotation with respect to the y-axis (pan) by an angle α causes a foreshortening of the 2D trajectory by a factor $\cos\alpha$. Correspondingly the mean values of the two states move closer. As long as the two states are physically separated in the 2D image, the two states remain distinct. This ensures that O and $\tilde{O}$, as described in the proposition, are of the same dimension. Practically speaking, this means that straight-line trajectories in one viewing direction do not collapse to a point when viewed from a different direction.

2.6 Experiments

We demonstrate our approach to activity recognition and anomalous trajectory detection using the UCF, the Credo Interactive Inc./CMU MOCAP and the TSA

airport surveillance datasets.

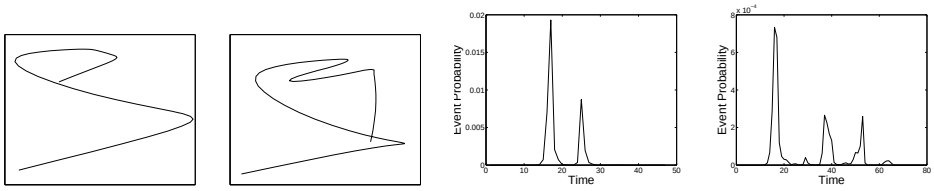
2.6.1 Activity recognition

2.6.1.1 UCF human actions dataset

A description of the dataset is presented in section 1.3.1 along with a summary of the tracking algorithm used to extract trajectories of the hand as different actions such as picking up, opening a door and erasing the white-board, are performed.

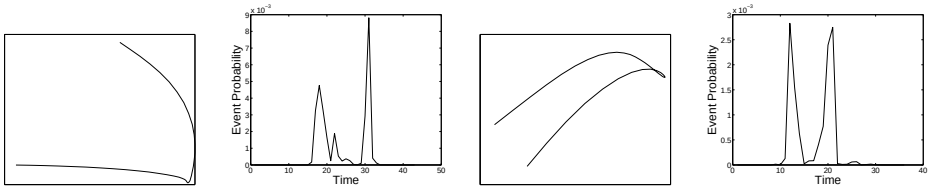
Training conforming HMMs: We build a view-invariant representation of the activity trajectories using the event probability sequence. Given trajectories of an activity, we compute the parameters of an HMM with 4-states using the Baum-Welch algorithm. Each state is modeled using a single Gaussian. The state transition matrix is assumed to be a left-to-right model. Generally, if the variations in viewing direction are reasonably small, multiple HMMs are not required since the distribution of the states of the HMM provides sufficient generalization across these views. As the variation becomes large, the states of the HMM become too diffused to model the data reliably. In our experiments, we found that each of the following activities needed 2 HMMs: *open the cabinet door*, *pick up an object*, *put down an object*, whereas *close the cabinet door*, *erase white board*, *pour water into a cup*, *pick up an object and put it down elsewhere* all had one HMM per activity.

Event Probability Sequences for training data: Using the trained HMM, we compute the event probability sequence using the relations for $\eta_t^p(i,j)$ and $e_t^p(k,l)$ given in (2.5) and (2.6). The event probability sequence, along with the HMM, is



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

Figure 2.5: *Invariance to spurious events:* (a) Hand trajectory for *closing the door*, (c) its event probability sequence. (b) *closing the door along with random motion.* This generates a spurious peak in the event probability sequence, shown in (d).



(a) (b)

Figure 2.6: (a) Hand trajectory and (b) its event probability sequences for *picking up an object from the desk* with $p = 5$; (c) and (d) for *picking up an umbrella from the cabinet.* Both instances of *pick up* activity have two dominant events.

stored as the signature of the activity.

Matching: Given an new (test) trajectory x, we compute a set of candidate event probability sequences $e_t^{x,g}(k,l)$ for every model in the database using HMM λ_g for $g = 1, \ldots, G$, where G is the number of trajectories in the database. To compare the candidate event sequences with those obtained during training, we use the DTW algorithm. The best match for each event is found by searching in a local neighborhood. The globally optimal path for matching two event sequences is found by backtracking. It allows for correct recognition even if there are spurious or missed

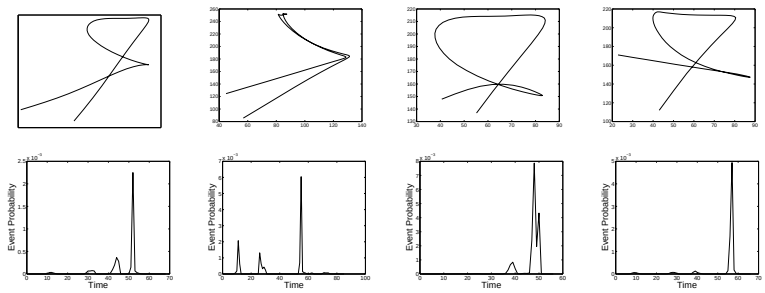
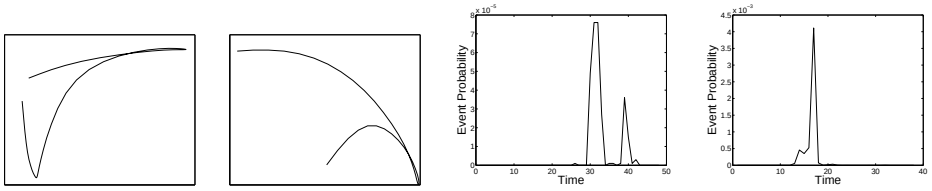


Figure 2.7: *Quasi-view Invariance:* Different samples of *open cabinet door*. The appearance of the trajectory depends on the location of the person performing the activity. The top row shows the hand trajectories, and bottom row the corresponding event probabilities.

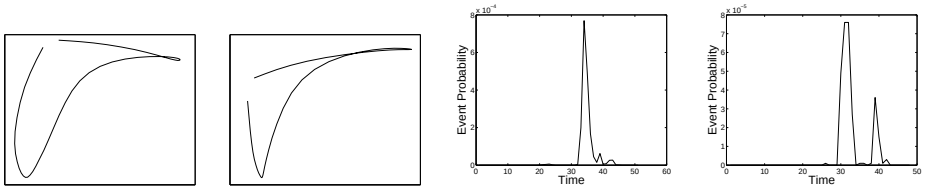
events. For example, in figure 2.5, the activity is correctly recognized in spite of spurious events. We compute the candidate event probability sequences at 6 scales for scale parameter values ranging from $p = 3$ to $p = 8$. These are matched with the event probability sequences in the database, at the appropriate scale. This yields a 6-D vector of similarity scores. To calculate the overall similarity score, we perform a coarse-to-fine matching as described in section 2.4.4. Figure 2.10 summarizes the recognition results using cumulative match scores (CMS) as the performance measure. The CMS is computed by accumulating recognition rates from rank 1 onwards. For instance, a CMS of 90% at rank 5 means that on an average, within the top 5 matches, the action is correctly recognized 90% of the time. The first two columns are the CMS scores at rank 1 and 5 respectively, obtained using the proposed method. The third column shows the recognition rate obtained in [1].

Analysis of results: We illustrate properties of the recognition scheme with a few examples:



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

Figure 2.8: Recognizing sub-activities: (a) Hand trajectory for *picking up an object from the cabinet shelf and putting it on the desk*,(c) its event probability. (b) Hand trajectory for *picking up an object* and (d), the event sequence for the top match using (a) as test sequence.



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

Figure 2.9: *Recognizing composite activities:* (a) Hand trajectory for *picking up an object from the desk and putting it back on the desk*, (c) corresponding event probability. (b) is another sample of the same composite activity. It was the top match when (a) as test sequence. (d) shows the event probability sequence computed during the testing phase.

- *Context insensitive:* Figure 2.6 shows trajectories and event probability sequences associated with *pick up* activities that differ in context. Though their HMMs - for picking up object from the desk and picking up an umbrella from cabinet - are different, both event sequences show two peaks surrounding the instant when the *pick up* event occurs.

- *Quasi view invariance:* Unless the viewing directions cause singularities, i.e., straight lines in one view collapsing to a point in the other, etc., the events detected remain insensitive to the viewing direction. Figure 2.7 shows the differences in appearance due to differing position of the person opening the cabinet. All the four trajectories were correctly recognized.

- *Spurious events:* Stylistic variations or errors in tracking may cause spurious events. For example, in figure 2.5, the first figure shows a normal activity of closing the door. In the second figure, the hand performs some random motion after closing the door creating spurious events. In spite of the spurious peaks, the activity was correctly categorized.

- *Sub-activities:* Composite activities may have sub-activities embedded within them. The sub-activities may be regarded as activities themselves, and we may wish to recognize them individually. On the other hand, we may be interested in recognizing the composite activity in its entirety. Figure 2.8 illustrate the former. The top 3 matches were sub-activities and the 4^{th} match was the correct composite activity. In figure 2.9 the composite activity is recognized correctly in the top 1^{st} and 2^{nd} matches. The next 3 matches are the sub-

activities.

Comparison with the UCF method[1]: Rao *et al.* treat activities as a sequence of *dynamic instants* [1], which are points of maximum curvature along the trajectory. They may be considered as a subset of our event probabilities e_t. Events in e_t do not require sharp curvatures in the trajectory unlike dynamic instants. For instance, the action *pick up umbrella while twisting hand* is not recognized as a pick up action in [1] because of numerous peaks in curvature caused by twisting. The event probability sequences, however, are correctly matched. Further, collinearity in the trajectory is an issue in [1] and causes incorrect matches. In our case, the changes in direction of motion of the hand is sufficient to produce event probabilities. For similar reasons, recognition using dynamic instants may not be able to deal with composite activities that have sub-activities. A comparison of recognition rates is given in figure 2.10. The improved recognition performance of the event probability sequences may be attributed to two reasons. An event probability sequence is defined using variations at the HMM state level. This is more robust than defining events at the data level. The set of event probability sequences indexed by the scale parameter is a richer representation of the activity compared to the dynamic instants in [1] that are defined at a scale corresponding to the observed data level. We match event sequences at multiple scales of the activity. The experiments demonstrate that a jointly optimal choice of parameters (N, p) yields better performance compared to the conditionally optimal choice.

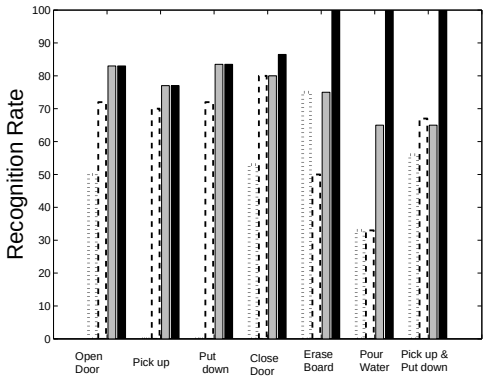


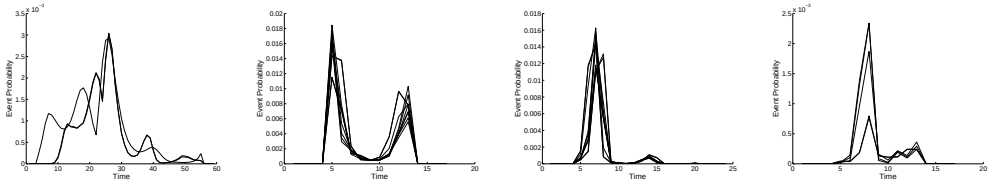
Figure 2.10: Recognition rate for UCF database. (a) Dotted line: UCF results [1], (b) Dashed line: Overall similarity obtained by integrating P similarity scores from coarse-to-fine scales, (c) Gray bar: Conditionally optimal p^*, (d) Black bar: Jointly optimal $(N, p)^*$

Figure 2.11: TSA dataset. Dominant activities are passengers embarking and disembarking.

2.6.1.2 Motion Capture (MOCAP) dataset

We use the MOCAP dataset (section 1.3.4) to demonstrate recognition of commonly performed human actions such as walking, running and sweeping the floor. In the confusion matrix obtained by comparing all the activities to each other, we may expect activities such as *normal walk*, *blind walk*, *jogging* to be clustered more closely compared to *sit* or *neutral*. There are 9 activities in the dataset and approximately 75 sets of observation overall. The tracks for an activity, say walking, consists of multiple cycles of the activity. We divide the sequence into individual walking cycles and treat each half-cycle as an observation. Half-cycle refers to the part of the walking cycle starting from the standing pose, right (or left) leg forward, reaching the swing pose, and withdrawing the right (or left) leg to the standing pose. This assumes that the right-leg forward and left-leg forward half-cycles are symmetric. We increase the number of observations to 365 by treating similar trajectories of nearby locations as multiple samples, i.e., 2 locations near the abdomen are treated as multiple samples of the same location. To ensure that there is no bias due to the displacement between similar locations, we use mean-subtracted trajectories for all locations. This gives us more samples per location but loses the physical context by forcing the mean to be zero.

Training event probability sequences: We divide the dataset into two parts: the training and testing set. Using the training set trajectories obtained by the MOCAP system, we build one HMM per activity. We assume a left-to-right 4-state, single Gaussian model. For each trajectory, we compute the event probability

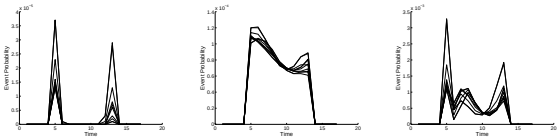


(a) (b) (c) (d)

Figure 2.12: MOCAP dataset: Examples of event probability sequences. Each figure has multiple event sequences corresponding to multiple observations. (a) Sit, (b) Blind walk, (c) and (d) Two instances of normal walk.

sequence as described in section 2.3 using the HMM for that activity. This forms the training set of event probability sequences. We observe that the different forms of walking, i.e., *normal walk*, *blind walk*, *prowl walk*, usually exhibit two events in the event probability sequence at approximately similar time instants during the walk cycles. The strength of the two events may vary based on the type of walking. Figure 2.12 shows the event probability sequences for some of the activities.

Comparing test data: Given a test observation sequence, we compute as many candidate event probability sequences as the number of HMMs in the database. Throughout the experiment, we fixed the scale parameter at $p = 3$. In contrast, in the UCF dataset, we matched event probability sequences at multiple scales. The intra-class variation in the MOCAP dataset activities is much less compared to that in the UCF dataset. This could be one of the reasons that we are able to correctly identify all activities in the MOCAP dataset by considering the event probability sequences at one value of p. We compare each of these event sequences with the ones in the training set using DTW. All the activities were correctly recognized.



(a) (b) (c)

Figure 2.13: MOCAP dataset: Illustrating relative invariance of event probability sequences. In all three cases, the test sequences are blind walk sequences. The HMM used to generate the event sequences are (a) Normal walk, (b) Jog, (c) Exaggerated walk. In the confusion matrix across activities, these three activities resembled blind walk.

Table 2.1 summarizes the activities that were the closest matches following the top match. We observe that the different types of walking resemble each other whereas the similarity scores corresponding to *sitting*, *sweeping with a broom* are significantly larger. Figure 2.13 illustrates the relative invariance property of event probability sequences. The figure shows the event probability sequences obtained by using *blind walk* tracks as the test sequence and *normal walk*, *jog* and *exaggerated walk* as the training HMM. These were the closest matches to the *blind walk* activity.

View invariance: We tested view invariance of event probability sequences for two activities across four viewing directions within a 120^o hemisphere (60^o on either side of reference direction). We chose *walking* and *sweeping with a broom* as the two activities. Using the 3-D motion capture data, several 2-D views were synthesized. The classification rates are summarized in table 2.2. The same training data was used in all four test viewing directions. We compared the time of occurrence of events and not their type or strength in computing the similarity score. The event

Table 2.1: MOCAP dataset: Closest-matching activities based on comparing event probability sequences. All activities were correctly recognized. Table shows the matches following the top match.

Test activity	Match #1	Match #2	Match #3
Blind-walk	Exaggerated walk	Normal walk	Jog
Prowl-walk	Normal walk	Exaggerated walk	Jog
Broom	Broom2	Exaggerated walk	Normal walk
Crawl	Prowl walk	Jog	Normal walk
Exaggerated walk	Normal walk	Jog	Blind walk
Jog	Normal walk	Prowl walk	Exaggerated walk
Sit	Sit1	Jog	Prowl walk
Normal walk	Sad Walk	Jog	Prowl walk
Sad walk	Normal walk	Jog	Jog

probability sequences for the test sequences are generated using the trained HMM for the reference direction.

2.6.2 Anomalous trajectory detection

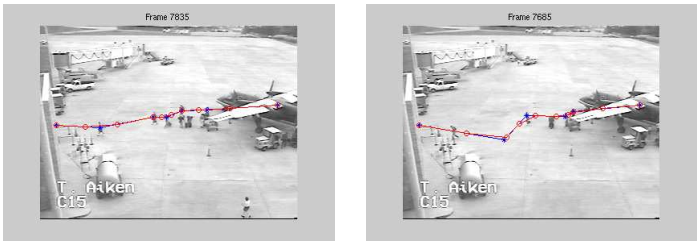
Anomalies can be subtle deviations from normal activities. This makes it difficult to use measures based on model approximation error (or likelihood) to detect anomalies. On the other hand, event probability sequences can be used to detect anomalies as demonstrated in this section.

Table 2.2: MOCAP dataset: Effect of changing viewing direction: Classification rate for distinguishing between walking and sweeping activities. T.V.D.=Test data Viewing Direction. All the four cases are compared with the same reference viewing direction.

	T.V.D. #1	T.V.D. #2	T.V.D. #3	T.V.D. #4
Walk	100	94	91	80
Sweep	100	98	95	95

2.6.2.1 TSA airport surveillance dataset

Figure 2.11 shows extracted motion trajectories from the TSA dataset (section 1.3.2). We trained HMMs for three activities that occur around the plane: passengers embarking, passengers disembarking and luggage cart leaving the plane. During the training phase, the HMM parameters are estimated using the Baum-Welch algorithm and event probability sequences are computed using the learnt HMMs. Figure 2.16 shows the event probability sequences for passengers disembarking and proceeding to the gate. We observe that fewer number of events are detected at coarser scales as expected. At a particular scale, we may think of the events as reflecting the progress of a passenger as he/she walks from the airplane to the gate. The events partition the path into regions. If we use a left-to-right HMM model, these regions roughly correspond to the states of the HMM at a sufficiently fine scale. In other words, at the appropriate scale $\tilde{p}$, we may expect $N-1$ events, where N is the number of states at regularly spaced intervals.



(a) (b)

Figure 2.14: (a) Snapshot of TSA dataset (b) Simulated anomaly - person deviates from virtual path between plane and gate, and walks toward the fuel truck.

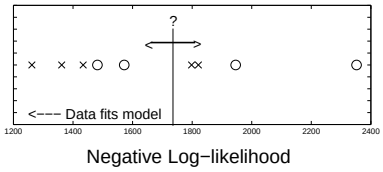
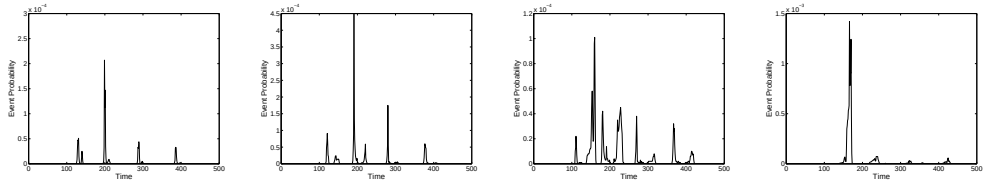


Figure 2.15: Negative log likelihood for normal ('x') and anomalous ('o') instances of people walking to the gate after disembarking. Values on the left end of the scale are more likely to be generated by the learnt HMM.

Given a test trajectory, we compute the event probability sequence using each of the learnt HMMs (for the three activities) to generate three candidate event probability sequences. These candidates are compared with trained event sequences using DTW. All activities were correctly recognized. Not surprisingly, in this limited setting, the HMM itself (without reference to event probability sequences) successfully classifies the different activities, i.e., the log-likelihood obtained using the Viterbi algorithm is able to distinguish among the three activities.

Anomaly detection using event probability sequences: Consider normal trajectories of people disembarking from the plane and walking toward the plane

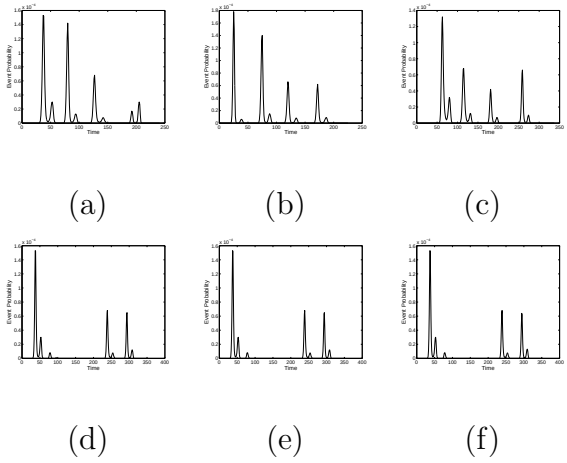


(a) (b) (c) (d)

Figure 2.16: Event probability sequence at different scales for normal trajectories of people deplaning. At coarse scales, fewer events are detected. The scale parameter increases from (a) through (d).

(figure 2.14(a)). Deviations from the normal path taken by passengers may be considered anomalies. The extent of deviation need not be large. Since there are no anomalous cases in the dataset, we simulate one by introducing a spatial anomaly. A person is assumed to deviate from the normal path (as illustrated in figure 2.14(b)). The hypothetical person violates the normal path, and walks toward the fuel truck (lower left corner of figure 2.14(b)). The extent of deviation is controlled by a parameter σ. As the value of σ increases, the V-shape of the anomalous trajectory in figure 2.14(b) "deepens", and the deviation from the normal path increases.

Using the set of normal trajectories, a 5-state left-to-right HMM is trained. Figure 2.15 shows a plot of negative log-likelihood for different normal (marked with a cross) and anomalous (marked with a circle) cases. As the value increases along the x-axis, it indicates that the trajectories are less likely to be generated by the trained HMM. It may be tempting to find a threshold above which trajectories are declared anomalous (i.e., outlier detection). This is not a feasible solution as illustrated in figure 2.15. In some instances, the anomalous trajectory has a higher



(a) (b) (c)

(d) (e) (f)

Figure 2.17: The top row shows event probabilities for 3 normal trajectories of people deplaning and walking toward the terminal. There are four dominant events in the normal trajectories, irrespective of the exact paths that people follow. The bottom row shows trajectories with increasing extents of spatial deviation $\sigma^2 = 2, 4, 16$ respectively.

probability (more left on the scale) of being generated by the trained HMM compared to a normal one. The HMM likelihood is insensitive to anomalies that are subtle deviations from normal trajectories. Also, the deviations can be confined to a part of the activity.

Given a new trajectory, we use the trained HMM to compute event probabilities. Any anomalies present are reflected in the sequence of events detected, even if the anomaly is present in a part of the trajectory. The method does not accumulate errors at all time instants, but only based on the times when events occur or when an event was expected to occur as seen in the training data. To declare the presence of an anomaly, we use both the number of events detected (spurious and missing events are both anomalies) as well the location of the detected events. Figure 2.17 (d)-(f) shows the event probability sequences for a person who deviates from the normal path and later rejoins the virtual path. Figures 2.17 (a)-(c) show three such sequences. We observe that the latter two dominant peaks in the anomalous trajectories resemble the latter half of the normal event sequences, whereas a missing event in the first half indicates an anomaly.

Measuring relative speed using event probability sequence: The relative strength of the events is a measure of walking speed of the passenger. Consider the event probability sequence at scale $\tilde{p}$. The event variable $\eta_t^{\tilde{p}}(i,j)$ measures the probability of persisting in state i for $\tilde{p}$ frames, transitioning to state j at time t and persisting in state j for $\tilde{p}$ frames conditioned on the observed data. A faster moving person is likely to persist to a lesser extent in either state i or j. Consequently, the strength of the event is likely to be less compared to a passenger walking at a slower

speed. Whereas a passenger who stops at an intermediate point will not produce an event since the necessary state transition does not occur. The relative strength of the events in the event probability sequence bears an inverse relation to speed. In figures 2.16(a) and (b) we see that the second event is larger than the rest. This means that initially, as the passenger started walking from the plane, he/she was walking slowly. Gradually, as the passenger approached the terminal, he/she picks up speed as the latter event probabilities show.

2.7 Coarse to fine event hierarchy

Human activities can be modeled at various scales, ranging from a coarse scale that captures aggregate information to a fine scale that reflects detailed characteristics. Coarse-to-fine hierarchical structures have been successfully used to represent images and objects ([59], [60], [61]). Hierarchical structures are attractive because of their modularity and efficiency. Activities can be divided into smaller parts such as sub-activities and events that are localized in time and space. The division proceeds till a level of detail that is suitable to the intended application is attained. At the same time, the model has to effectively address limitations of low video quality and computation time.

There are several ways of decomposing activities to construct a hierarchical structure. It may be useful to have a set of guiding principles to analyze the effectiveness of a hierarchy. We describe criteria based on existing work in 3D shape and action modeling [16], [62]. The usefulness of coarse-to-fine event representation

is demonstrated for efficient indexing and browsing. This addresses the effect of degrading video quality (reduced frame rate and low resolution) on event detection.

2.8 Analysis of coarse to fine event models

In this section, we describe five criteria that can be used to determine the effectiveness of a coarse-to-fine event representation. The criteria are *minimalism*, *stability*, *consistency*, *accessibility* and *applicability*. They are based on those developed in and [62] and [63] for 3D shape modeling, and extended by [16] to incorporate the effect of covariates in modeling actions. Activities can be viewed as objects in spatio-temporal domain. Therefore, many criteria that are relevant in 3D shape modeling can be readily extended to activity modeling. We propose quantitative measures to evaluate the criteria.

2.8.1 Minimalism

An event representation should use a minimal number of model parameters required for an application. Generally, the number of parameters needed increases from coarse-to-fine scale. In statistical event models, minimalism can be quantified using a suitable information theoretic criterion such as Akaike information criterion (AIC), bayesian information criterion (BIC) or minimum length description (MDL) [64].

2.8.2 Stability

Events should be robust to small changes that are caused by imperfections in low-level processing techniques and noise. On the other hand, sensitivity may be desirable at fine scales in order to distinguish between activities with subtle differences.

In event probability sequences, as the scale parameter p increases, stable events are retained. The change in total probability of event probability sequence with change in scale parameter can be used as a measure of stability. If the coarse-to-fine structure is a stable representation, the sum of event probabilities is expected to decrease with increasing value of scale parameter. So, the negative log probability of the sum of event probablities increases. Let L_{p_i} denote the negative log probability at scale p_i. An event representation is said to be stable if the following conditions are satisfied:

1. $L_{p_i} > L_{p_j}$ for all $p_i < p_j$.

2. $|L_{p_i} - L_{p_j}| > |L_{p_j} - L_{p_k}|$ for all $p_i < p_j < p_k$.

3. $L_{p_M} \rightarrow L$ for sufficiently large p_M, and a constant L.

If the above statements hold for all $p < p_i < p_j < p_k$, where $p > 1$, then the event representation is said to be marginally stable.

2.8.3 Consistency

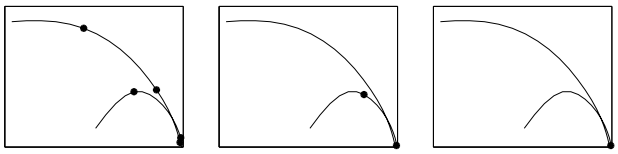
Events should be consistent at every scale, i.e., the number of events and the location of each event should converge in probability to the true value as the number of samples grows. One of the advantages of consistent event representation is that it places less emphasis on obtaining perfect tracking results. Even if some parts of the trajectory are incorrect, its effect would be mitigated at a sufficiently coarse scale. Consistency can be quantified using a statistical test or a coefficient of consistency such as Pearson's test and Cronbach's correlation coefficient. Pearson's test assumes that the underlying data is normally distributed, which need not be valid in real data. So we use the Cronbach's alpha coefficient, which is defined as follows [65]:

$$\alpha = \frac{K}{K-1}(1 - \frac{\Sigma\sigma^2_{X_i}}{\sigma^2_Y}), \tag{2.15}$$

where K is the total number of components, $\sigma^2_{X_i}$ is the variance in the i^{th} component and σ^2_Y is the total variance found by summing across components. Ideally, α should be equal to 1, its maximum value. In the case of event probability sequences, the components are the number of peaks in the event probability and the location of each event. So the number of components need not be constant across activities or across scales in an activity. Each component has to be normalized (mean-subtracted) before computing α.

2.8.4 Accessibility

The event representation should be constructed such that fundamental limitations are taken into account. For instance, it may not be possible to determine fine



(a) (b) (c)

Figure 2.18: Events detected at different scales for trajectory of *picking up* an object. The scale parameter p varies from (a) $p = 3$, (b) $p = 6$ (c) $p = 8$. Original video resolution is used in event detection.

details of an activity because of low video resolution. Also, computationally feasible model estimation algorithms should be available.

2.8.5 Applicability

Ultimately, application-specific performance measures decide the utility of a model. The relative importance of above criteria may be tuned to a specific application. For instance, stability of coarse-to-fine structure may be more important for activity recognition rather than anomaly detection. Applicability can be quantified using performance measures such as recognition rate, ROC curves and delay time in event detection.

2.9 Experiments

In this section we highlight the advantages of coarse-to-fine hierarchical activity representations for quick video browsing. Hierarchical structures readily lend themeselves to efficient browsing. When browsing, activities can be compared at a

coarse scale. Subsequent comparison can be restricted to only those activities that are matched at the coarse scale. Another way to reduce computational time is by reducing the frame rate of test video sequences. These are illustrated using the UCF and TSA datasets.

2.9.1 Consistency in event detection: effect of reduced frame rate

The UCF human action dataset is used to demonstrate consistency in event detection along with the effect of video quality. Several motion trajectories of each activity are used to train a 4-state left-to-right HMM. Event probability sequences are computed for every motion trajectory as described in section 2.3. Events detected for *picking up an object* are shown in figure 2.18. The dots represent events detected along the trajectory traced out by the hand while picking up an object lying on the desk. As expected, fewer events are observed at a coarse scale (larger value of p) compared to those detected in a fine scale.

Results: The Cronbach's alpha co-efficient is used to quantify consistency in the event representation across multiple observations at a particular scale. The results are summarized in table 2.3. The effect of reduced frame rate on consistency of events detected is also shown. It is observed that the events detected remain reasonably consistent at a reduced frame rate as well. This suggests that computation of event probability sequences can be speeded up by processing frames at a reduced rate without adversely impacting event detection. If the frame rate is further halved, however, there is a precipitous drop in performance. Also, many quantities in the

event computation (section 2.3) become ill-conditioned because of fewer available data points.

2.9.2 Coarse-to-fine structure for efficient browsing: effect of reduced spatial resolution

Surveillance videos are usually recorded over long periods of time making it necessary to develop methods to enable quick and efficient browsing. The user may be interested in producing reports that broadly summarize activities in the scene. If necessary, relevant parts of the video sequence can be analyzed to extract fine details of activities. Application of coarse-to-fine structure for video browsing is illustrated using the TSA airport tarmac surveillance dataset.

Activities were classified into four classes of movement: aircraft, passengers, luggage cart and ground crew. HMMs for each class were trained using multiple trajectories at the fine scale, and event probability sequences computed. A 4-state left to right HMM is used. At a coarse scale (spatial resolution is halved), motion histograms are computed and used to train an HMM and compute event probability sequences as before. Using a departure scenario, several synthetic motion trajectories were generated and used to train an HMM. This was necessary since the dataset consists of only three departure scenarios that occur in entireity.

The testing phase consists of the following steps. Events are compared at the coarse scale using event probability sequences of motion histograms. If a match is found, event probability sequences are computed at high resolution (original video

resolution) using motion trajectories. Detected activities are classified into one of the four classes of movement (defined in the training phase) at the fine scale.

Results: Recognition rates for the aircraft departure activity are presented below. The dataset contains three such scenarios (one of whose trajectories were perturbed and used for training).

(i) At the coarse level, four segments were detected as aircraft departure. The three expected segments were correctly identified. The fourth segment was a false alarm that contained aircraft arrival.

(ii) The segments detected at the coarse level were analyzed further to identify detailed activities. Recognition rates obtained for different activities were as follows. Movement of ground crew: 92%; movement of luggage cart to plane: 100%; passengers embarking: 78%; plane departure: 100%. Some of the passengers were not correctly identified because of truncated trajectories after loss of tracking. In the fourth segment (false alarm), movement of passengers and aircraft were not detected so that matching at the coarse level can be rejected.

2.10 Summary

In this chapter, we have presented a coarse to fine hierarchical event characterization that explores several state sequences of the HMM that generates the observed motion trajectories. The main hypothesis is that significant changes in motion can be interpreted as semantically salient events. With this hypothesis, a computationally efficient event detection procedure was developed using the HMM

Table 2.3: Consistency of hierarchical event representation in the UCF human action dataset measured using Cronbach's alpha coefficient. The maximum (and best) value of the coefficient is unity. Results for different activities under two temporal resolutions demonstrate the trade-off between video quality and reliability of event representation.

	Original video	Reduced frame rate (by a factor of 2)
Open door	0.82	0.81
Pick up object	0.94	0.95
Put down object	0.81	0.76
Close door	0.65	0.67
Pick up object and put down elsewhere	0.68	0.70
Pour water	0.77	0.71
Erase board	0.91	0.86

forward and backward variables. The probabilistic event model is called an event probability sequence that measures the probability that an event occurs at every time instant.

Properties such as minimalism, stability and consistency are presented for evaluating the goodness of hierarchical event representations. The effectiveness of event probability sequences for activity recognition and anomaly detection was demonstrated using both indoor and outdoor video sequences.

Chapter 3

Epitomic Representation of Activities

3.1 Overview

Our aim is to infer structure inherent in activities for activity recognition and video indexing. The tasks involved can be divided into three phases as shown in figure 3.1: low-level, mid-level and high-level phases.

The low-level processing phase accepts video sequences as input and produces motion trajectories as output. It has modules for modeling the background, detecting moving objects and obtaining motion trajectories of the detected objects. The operations are summarized in section 3.2.

The mid-level processing phase accepts trajectories as input and estimates parameters of epitomes, which form a mid-level activity representation. The kinematics of motion is assumed to be linear within a video segment so that it can be characterized using a system matrix and input signal u_k as described in (3.2). The tuple consisting of the system matrix, the initial value of the state vector and the input signal statistics is said to form an epitome. Each moving object is modeled as a sequence of epitomes. The system matrices are decomposed into three components that have a physical interpretation using the Iwasawa matrix decomposition. The three components represent rotation, scaling and translation of the state vector as the motion of the object changes across epitomes. The estimation of epitomic

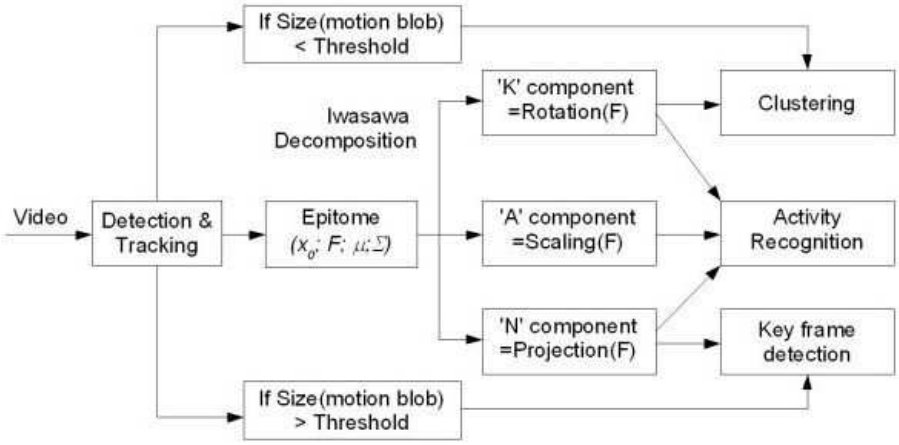


Figure 3.1: Overview of the epitomic model for activities.

parameters is described in sectoin 3.3.1 and the Iwasawa matrix decomposition is presented in section 3.4.

The separation between high-level and mid-level processing phases is not as clear as that between the mid and low-level phases. The high-level phase involves distance computation between epitomes (and in turn, activities) for applications such as activity recognition and video indexing. In section 3.5, we discuss the problem of selecting a distance metric that is not only cognizant of distances between matrices but also lends itself to physical interpretation. The physical interpretation is necessary to introduce high-level domain knowledge when comparing activities.

3.2 Low level video processing

In our experiments we have used existing algorithms for object detection and tracking with slight modifications. They are briefly summarized in this section. Tracking is challenging in surveillance scenarios because of low video resolution,

poor contrast and noise. Instead of attempting to track objects across the entire video sequence, we periodically reinitialize the tracker. The low-level tasks may be divided into two components: moving object detection and tracking. The detection component uses background subtraction to isolate the moving blobs. We use a procedure based on [66] and [67], which is summarized here. The background in each RGB color channel is modeled using single independent Gaussian distributions at every pixel using ten consecutive frames. If the normalized Euclidean distance between the background model and observed pixel value in a frame exceeds a certain threshold, then the pixel is labeled as belonging to a moving object. A static background is insufficient to model a long video sequence because of changing lighting conditions, shadows and cumulative effects of noise. So the background is reinitialized at regular intervals.

Motion trajectories are obtained using the KLT algorithm [68] whose feature points are initialized at detected locations of motion blobs. The KLT algorithm selects features with high intensity variation and keeps track of these features. It defines a measure of dissimilarity to quantify the change in appearance between frames, allowing for affine image changes. Parameters control the maximum allowable interframe displacement and proximity of feature points to be tracked. The trajectories from the KLT tracker are smoothed using a median filter. The effect of tracking errors is discussed in section 3.7. Of the three datasets used in the experiments, tracking was accurate and reliable in the indoor bank monitoring dataset and the UCF human action dataset. On the other hand, there were a few tracking errors in the TSA airport tarmac surveillance dataset that caused errors in temporal

segmentation.

3.3 Epitomes

A video sequence is divided into segments of length T. Moving objects are detected and their short-time motion trajectories are obtained as described in the previous section. The kinematics of motion within segments is assumed to be linear so that linear systems can be used in modeling (3.2). The estimated tuples $(x_0; F; (\mu, \Sigma))$ are called activity epitomes, where x_0 represents the initial state, $F \in GL(n, \mathbb{R})$ is an invertible $n \times n$ system matrix and (μ, Σ) represents the statistics of input signal $u_k, k \in [0, T]$ that is assumed to be i.i.d. Gassian distribution.

Assuming full state output, kinematics of motion in each video segment can be written as:

$$x_{k+1} = Fx_k + u_k, \tag{3.1}$$

$$y_k = x_k, \tag{3.2}$$

where $x_k \in \mathbb{R}^4$ is the state vector with initial value x_0, F is the system matrix of size 4×4 and u_k is the input signal. The state vector represents the 2-D position and velocity. The model in (3.2) is a state-space representation of the familiar Newton's laws in mechanics given by $m\ddot{q}(t) + k_1\dot{q}(t) + k_2 q(t) = u(t)$, which describes the motion of bodies subjected to conservative forces. Forces in mechanics could be due to gravity, energy in springs, and so on. In activity modeling we are not trying to infer forces that cause humans or vehicles to move, although that could be

useful in building realistic models for synthesizing trajectories. Instead, the goal is to build statistical models that can capture kinematics from video streams. In [3], the control input u_k is assumed to be a constant unit signal scaled by a parameter. The signal u_k is typically treated as Gaussian noise [48].

The signal u_k can have a useful role in modeling activities, besides providing a way to deal with uncertainty. It can be used to synthesize parts of the activity for handling occlusions. Also it provides a way of visualizing the data. We introduce *star diagrams* as a means of visualizing epitomes in section 3.3.2 after describing the computation of system matrices in the next section.

3.3.1 Estimation of epitomes

There are several ways of estimating parameters of linear dynamical systems. We illustrate a straightforward estimation procedure that can accommodate a desired structure in F. Consider the state evolution given by (3.1). The state x_k is said to be the instantaneous position and velocity. Let $F : H \rightarrow H$ be a matrix that relates the previous state x_{k-1} to the current state x_k, where $H \subset GL(n, \mathbb{R})$ (The Lie matrix groups are defined in appendix C). For robust estimation, we assume that the state of the system remains constant for a short interval of time W. The value of W depends on the type of data. For instance, it may be reasonable to assume $W = 100$ or 4 seconds in far field surveillance data. On the other hand, we may assume $W = 15$ or 1 second for short-term human actions (e.g. opening the door, picking up an object, etc.) performed in an office environment. Of course, there is

no reason for W to be a fixed number and we can use the temporal segmentation to find best value of W for different parts along the trajectory. For simplicity, we assume a fixed W. The least squares estimate of F that relates the past and future values of state is computed as follows.

For two vectors $x, b \in \mathbb{R}^n$, let $Fx = b$, where $F = [f_{ij}]$, $i, j = 1, 2, \ldots, n$. This can be rewritten as $Xf = b$, where

$$f = (f_{11}, f_{12}, \ldots, f_{1n}, f_{21}, \ldots, f_{2n}, \ldots, f_{n1} \ldots, f_{nn})$$

and X is a matrix that consists of rows of the form $(0, 0, x_1, x_2, \ldots, x_n, 0, 0, \ldots, 0)$. The entries in F can be selectively set to zero to impose prior knowledge of the dynamics. For example, the dynamics may be modeled using a decoupled second-order system, which creates a block diagonal matrix F. Suppose $Fx = b$ holds for W vector pairs $(x_1, b_1), \ldots, (x_W, b_W)$, we can write

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_W \end{pmatrix} f = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_W \end{pmatrix} \tag{3.3}$$

We use least squares technique to solve for f in (3.3) and recompose the vector f into the matrix F.

3.3.2 Star diagram

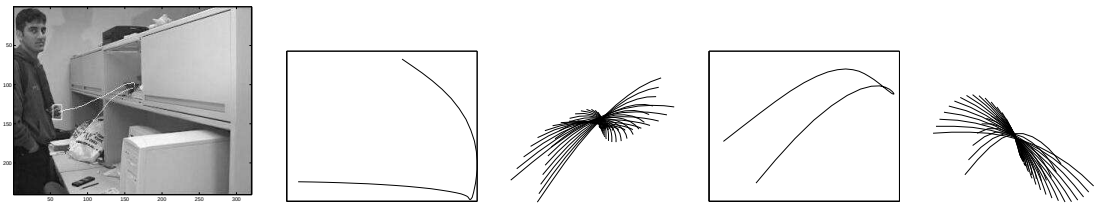
In this section, we present a way to visualize the learnt epitomes, which is referred to as *star diagrams*. Star diagrams provide a snapshot of an activity and

demonstrate similarities between motion trajectories of the activity that may not be observed at the level of motion trajectories because of semantic ambiguity and context dependency. Semantic ambiguity refers to incomplete textual description of an activity whose motion trajectories can vary significantly across samples [2]. Spatial and temporal context dependency can produce similar effects. The variations in appearance of motion trajectories are illustrated using *picking up an object* as an example (figure 3.2). It depends on several factors including position of the object relative to the initial position of the hand, viewing direction and location of the person performing the action. Star diagrams reflect these variations and capture the activity of interest.

Trajectories of activity are extracted and segmented into intervals of equal length. For each segment, parameters of epitomes, i.e., the system matrices and the input signal statistics (mean and covariance) are calculated. The trajectory is thus represented as a sequence of epitomes. Using the learnt epitomes, trajectories are sythesized using different values of the signal sampled from the learnt distribution. Temporally overlaid patterns of synthesized trajectories are referred to as star diagrams.

Figure 3.2 shows star diagrams for the pick up activity in figure 3.2. The star diagrams in the two cases of picking up objects (figure 3.2(a) and (b)) differ by a rotation angle reflecting the differences in position of the object that creates differences in the appearance of trajectories. The similarity in relative structure of epitomes within each diagram is exploited during activity recognition.

Figure 3.3 shows star diagrams for surveillance scenarios at an airport tarmac.



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

Figure 3.2: UCF Dataset: (a) Sample image along with the extracted hand trajectory, (b) Motion trajectory for *picking up an object from the desk* and (c) its star diagram; (d)Motion trajectory for *picking up an umbrella from the cabinet* and (e) its star diagram.

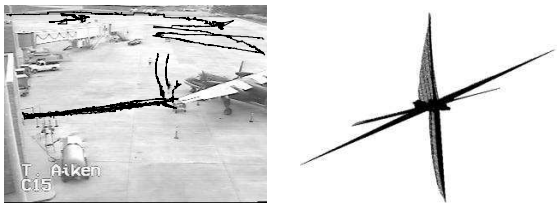


Figure 3.3: TSA airport surveillance dataset. (a) Motion trajectories of passengers embarking; (b) its star diagram

Figure 3.3(a) depicts a scene in which passengers embarking an airplane is the dominant activity. It also contains movement of ground crew and a truck. The star diagram (figure 3.3(b)) shows a compact representation using activity epitomes that were learnt using motion trajectories in figure 3.3(a).

3.4 Iwasawa decomposition

Star diagrams provide a visual representation of epitomes. Motivated by the structure inherent in star diagrams, we propose a decomposition of epitomes into three components using the Iwasawa matrix decomposition. The three components are rotation, scaling and projection of the state vector.

Definition 1 *Let $F \in GL(n, \mathbb{R})$. Then there exist unique matrices K, A, N, such that $F = KAN$, where*

- *K is an orthogonal matrix.*
- *A is a diagonal matrix with positive diagonal entries*
- *N is a unit upper triangular matrix, i.e., all the diagonal elements are unity.*

This is called the Iwasawa matrix decomposition [69].

It may be worth noting that the Iwasawa decomposition applies globally to the entire group manifold [70]. If $F \in SL(n, \mathbb{R})$ (i.e, det $F = +1$), the matrices take the following form [71] for $n = 2$:

$$K = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}, \quad A = \begin{pmatrix} \sqrt{a_1} & 0 \\ 0 & \frac{1}{\sqrt{a_1}} \end{pmatrix} \text{ and } \quad N = \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix}, \quad (3.4)$$

where $\beta \in \mathbb{R}$, $a_1 \in \mathbb{R}^+$ and $\theta \in \mathbb{R}/\pi\mathcal{Z}$. Each of the three components are a one-parameter family that decouple the effect of transformation F on the state of the moving object. This is used to define distances between F's in the next section.

3.4.1 Special case: $n = 2$

Consider $F \in SL(2, \mathbb{R})$

$$F = \begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{pmatrix},$$

where $f_{ij} \in \mathbb{R}$, $i, j \in \{1, 2\}$. By hypothesis, $f_{11}f_{22} - f_{12}f_{21} = 1$. From (3.4), the components of the decomposition can be calculated as follows:

$$\begin{aligned} a_1 &= f_{11}^2 + f_{21}^2 \\ \cos\theta &= \frac{f_{11}}{\sqrt{f_{11}^2 + f_{21}^2}} \\ \sin\theta &= \frac{-f_{21}}{\sqrt{f_{11}^2 + f_{21}^2}} \\ a_1 &= \frac{1}{f_{11}}\left(\frac{f_{21} + f_{12}(f_{11}^2 + f_{21}^2)}{f_{11}^2 + f_{21}^2}\right) \\ &= \frac{1}{f_{11}}\left(\frac{f_{21}(f_{11}f_{22} - f_{12}f_{21}) + f_{12}(f_{11}^2 + f_{21}^2)}{f_{11}^2 + f_{21}^2}\right) \\ &= \frac{f_{11}f_{12} + f_{21}f_{22}}{f_{11}^2 + f_{21}^2} \end{aligned} \quad (3.5)$$

Remark 1: The Iwasawa decomposition $F = KAN$ is unique upto a factor α that can occur in K along the secondary diagonal, i.e., the decomposition in

definition 1 can be replaced by the following:

$$K = \begin{pmatrix} \cos\theta & \alpha\sin\theta \\ -\alpha^{-1}\sin\theta & \cos\theta \end{pmatrix}, \quad A = \begin{pmatrix} \sqrt{a_1} & 0 \\ 0 & \frac{1}{\sqrt{a_1}} \end{pmatrix} \text{ and } \quad N = \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix},$$

where $\beta \in \mathbb{R}$, $a_1 \in \mathbb{R}^+$, $\alpha \neq 0$ and $\theta \in \mathbb{R}/\pi\mathcal{Z}$.

3.4.2 Special case: Symplectic group

For $n \geq 2$, $F \in Sp(2n, \mathbb{R}) \subset SL(2n, \mathbb{R})$ it has been shown that the decomposition has a block structure [72]:

$$K = \begin{pmatrix} \tilde{K_1} & \tilde{K_2} \\ -\tilde{K_2} & \tilde{K_1} \end{pmatrix}, \quad A = \begin{pmatrix} a_1 & 0 & \dots & & 0 \\ 0 & a_2 & 0 & \dots & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & a_m & 0 & \\ 0 & \dots & 0 & a_1^{-1} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & & & a_m^{-1} \end{pmatrix} \quad N = \begin{pmatrix} \hat{N_1} & \hat{N_2} \\ 0 & (\hat{N_1}^{-1})^T \end{pmatrix} \tag{3.6}$$

where $K \in SO(n)$, $a_i > 0$ for $i = 1, \dots, m$, $m = n/2$ and $\hat{N_1}\hat{N_2}^T = \hat{N_2}\hat{N_1}^T$. $\tilde{K_1}, \tilde{K_2}, \tilde{N_1}, \tilde{N_2}$ are all block matrices of size $m \times m$.

Theorem 1 *Let $P \in Sp(2n, \mathbb{R}) \subset SL(2n, \mathbb{R})$, $n > 1$ such that*

$$P = \begin{pmatrix} P_1 & P_2 \\ P_3 & P_4 \end{pmatrix}$$

for $P_i \in \mathbb{R}^{n\times n}$, $i = 1, \ldots, 4$. *Let*

$$R = P^T P = \begin{pmatrix} R_1 & R_2 \\ R_2 & R_4 \end{pmatrix} \in Sp(2n, \mathbb{R}),$$

where R_1 *is positive definite. Then Iwasawa decomposition of* $P = KAN$ *is given by*

$$A = \begin{pmatrix} D^{1/2} & 0 \\ 0 & D^{-1/2} \end{pmatrix}, N = \begin{pmatrix} Q & QR_1^{-1}R_2 \\ 0 & (Q^{-1})^T \end{pmatrix}$$

and $K = PN^{-1}A^{-1}$.

Proof: In appendix.

3.4.3 General case

Let $F \in GL(n, \mathbb{R})$ be a non-singular matrix with real entries. Consider

$$M = F^T F, \tag{3.7}$$

where M is symmetric and positive definite. (3.7) can be written as

$$M = (KAN)^T (KAN) \tag{3.8}$$

$$= N^T A^T K^T K A N \tag{3.9}$$

$$= N^T A^T A N, \tag{3.10}$$

using $F = KAN$ with K being orthogonal, i.e.,$K^T K = I$. Let

$$R \stackrel{def}{=} AN \tag{3.11}$$

Clearly, R is an upper triangular matrix. (3.10) can be written as

$$M = R^T R \tag{3.12}$$

Table 3.1: Computing the Iwasawa matrix decomposition of an invertible matrix F

Let $M = F^T F$
Compute the Cholesky decomposition $M = R^T R$
Form the diagonal matrix $A = diag(R)$
Compute the unit upper triangular matrix $N = A^{-1}R$
Compute $K = FR^{-1}$

The factorization in (3.12) is computed using the Cholesky decomposition that factorizes any symmetric, positive definite matrix into the product of a lower triangular matrix and its transpose, i.e., $M = R^T R$.

From definition 1, we know that N is a unit upper triangular matrix. The diagonal matrix A is formed by extracting the diagonal elements of R, i.e., $A = diag(R)$ so that

$$N = A^{-1}R \tag{3.13}$$

. Since $F = KAN$ (definition 1), the K component can be obtained by

$$\begin{aligned} K &= FN^{-1}A^{-1}, \\ &= FR^{-1}. \end{aligned} \tag{3.14}$$

The steps involved in computing the Iwasawa matrix decomposition is summarized in table 3.1.

Remark 2: The Iwasawa decomposition can also be written as $\tilde{N}\tilde{A}\tilde{K}$, such

that

$$\tilde{N} = \begin{pmatrix} 1 & 0 \\ \tilde{\beta} & 1 \end{pmatrix}, \quad \tilde{A} = \begin{pmatrix} \sqrt{\tilde{a_1}} & 0 \\ 0 & \frac{1}{\sqrt{\tilde{a_1}}} \end{pmatrix} \text{ and } \quad \tilde{K} = \begin{pmatrix} \cos\tilde{\theta} & \sin\tilde{\theta} \\ -\sin\tilde{\theta} & \cos\tilde{\theta} \end{pmatrix},$$

where $\tilde{\beta} \in \mathbb{R}$, $\tilde{a_1} \in \mathbb{R}^+$ and $\tilde{\theta} \in \mathbb{R}/\pi\mathcal{Z}$.

Example 1 *The following example illustrates the Iwasawa decomposition. Consider*

$$F = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}, \tag{3.15}$$

which represents a second-order decoupled motion model (i.e., the x and y components of kinematics are decoupled). Its Iwasawa decomposition gives:

$$K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}, \quad N = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \frac{-1}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

3.4.4 Geometric interpretation and Singular Value Decomposition

The Iwasawa decomposition yields $F = KAN$ as described in definition 1. The matrix K contains a set of orthonormal basis vectors for F with $K^TK = I$. It is used to compute the principal angles between matrices. As seen from (3.4), K represents the effect of rotation component on the state vector that is parameterized

by a single parameter θ. The diagonal matrix A of size contains positive real entries representing a scaling of the state vector along each dimension. If the transformation F of size $2n \times 2n$ belongs to the special linear group (i.e., $\det F = 1$), then the A component has half as many degrees of freedom as F. The scaling in the first n dimensions of the state vector determines the extent of scaling in the remaining dimensions. The N component introduces a lens action or a projective action. It can also be interpreted as a translation in certain cases. For instance, consider the case $n = 3$.

$$N = \begin{pmatrix} 1 & \beta_1 & t \\ 0 & 1 & \beta_2 \\ 0 & 0 & 1 \end{pmatrix},$$

where $\beta_1, \beta_2, t \in \mathbb{R}$. Or equivalently, the matrix N can be represented by the tuple (β_1, β_2, t). Each point (β_1, β_2, t) can be viewed as a translation from 0 to this point $(\beta_1, \beta_2, t) \cdot (0, 0, 0) = (\beta_1, \beta_2, t)$. The 0 point corresponds to the identity matrix. In the 2×2 case, N has one free variable $\beta \in \mathbb{R}$ as seen in (3.4).

Suppose $F \in GL(n, \mathbb{R})$ is a full-rank, square matrix with real entries. Using SVD, it can be factorized as $F = U\Sigma V^T$, where U and V are orthonormal matrices, and Σ is a diagonal matrix of singular values. The Iwasawa decomposition also has three components in its factorization $F = KAN$. It may be interesting to see the connection between the two decompositions.

Consider the case $n = 2$, i.e., when F is a 2×2 invertible matrix. Appplying SVD, we obtain a pair of left eigenvectors in the matrix U and a pair of right eigenvectors in V. This specifies a co-ordinate basis for F. The diagonal matrix Σ

has singular values (σ_1, σ_2), which are both positive. A comparison between the two decompositions is given below:

- The K component of $F = KAN$ decomposition reflects the amount of rotation in F. The U and V components in $F = U\Sigma V^T$ specify a coordinate basis.
- Both the diagonal matrices Σ and A in the two decompositions contain positive real entries.
- The N component in KAN decomposition, which is in reduced row-echelon form captures a projective (lens) action of F. It contains the translative part of F as described above.

3.5 Distance between activity epitomes

It is necessary to define a notion of distance between activity epitomes for many applications including activity recognition and clustering. Euclidean distance between $F_1 \in \mathbb{R}^{n\times n}$ and $F_2 \in \mathbb{R}^{n\times n}$ may be defined, in which F_1 and F_2 are thought of as vectors in $\mathbb{R}^{n^2}$. Alternatively, Frobenius distance between F_1, F_2 can be used, which is defined as follows:

$$d_f(F_1, F_2) \quad = \quad ||F_1 - F_2||_F^2 = tr((F_1 - F_2)(F_1 - F_2)^T). \tag{3.16}$$

This does not take the geometry of the space into account and it ignores the group structure of matrices. In this section, we define a Riemannian metric to compare epitomes. The Iwasawa matrix decomposition simplifies the calculation of the metric and provides a physical insight into quantities being compared.

In the familiar Euclidean setting, distance between points $p_1, p_2 \in \mathbb{R}^n$ in the familiar Euclidean setting can be computed by integrating infinitesimal metrics $dx_1, dx_2, \ldots, dx_n$ from p_1 to p_2. This gives straightline distances between p_1 and p_2. Similarly, distance between points on a manifold can be computed by integrating with respect to a Riemannian metric. Unlike dx and dy in the Euclidean case, however, the choice of metrics in the case of manifolds may not be obvious. There are broadly, two approaches to overcome this problem as summarized below.

A Riemannian metric can be chosen (or learnt from the data) so that the length minimizing geodesic distance can be computed by evaluating the integral along a parametrized curve. The Euclidean metric dx is replaced by a Riemannian metric Δ. The geometry of space $GL(n, \mathbb{R})$ in which dynamical matrices lie is well-studied and several metrics have been proposed [73], [74]. So, in principle, one of the metrics can be chosen and used to evaluate the geodesic distance. A few such metrics are described next.

Riemannian metrics have been defined on the space of linear systems using traces [74]. Let

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k, \\ y_k &= Cx_k + Dv_k \end{aligned} \tag{3.17}$$

be a linear system with state $x_k \in \mathbb{R}^n$, control input $u_k \in \mathbb{R}^m$, output[1] $y_k \in \mathbb{R}^p$, measurement noise $v_k \in \mathbb{R}^m$ and matrices[2] $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}, C \in \mathbb{R}^{p \times n}, D \in$

[1]The notation x for state and y for system output is not related to the co-ordinate position on the xy plane. Here, the definition of the state vector reflects physical parameters of the system.

[2]The notation A for the system matrix is not meant to invoke the A component of the KAN

$\mathbb{R}^{p\times m}$. The following inner product is used to define the norm of the system:

$$\|(A,B,C,D)\|^2 \;=\; tr(CL(BB^T)C^T + DD^T), \tag{3.18}$$

where $L(K)$ is the solution to the discrete time Lyapunov equation $L - ALA^T = K$. The inner product in (3.18) induces a Riemannian metric on the manifold. Analytical computation of the Riemannian metric tensor becomes rather difficult even for $n = 3$ case as shown in [74] requiring numerical computation. Nevertheless, using the Riemannian metric induced by the inner product can be an effective strategy. The metrics computed using traces can be used for comparing epitomes in activity recognition. The choice of a suitable trace may depend on the application of interest.

Distance measures based on subspace angles (or principal angles) are widely accepted for comparing matrices that represent dynamics. A commonly used distance (e.g., in [48]) is the Martin distance d_M^2 defined as [47]:

$$d_M^2 \;=\; -\log \prod_{i=1}^{n} \cos^2(\theta_i), \tag{3.19}$$

where $\theta_i, i = 1,\ldots,n$ are subspace angles. Subspace angles capture differences that are caused by rotation of subspaces while ignoring changes in other factors such as scaling and translation. Finsler distance [75] uses the largest principal angle between the subspaces unlike the Martin distance that uses all n values in (3.19). This is best suited when the signal is scalar-valued and generated by a strictly second-order stationary process. In our case, the number of outputs of the system is 4 (2D

decomposition.

position and velocity). When the number of inputs and outputs associated with the system is more than one, the distance is not guaranteed to be non-negative.

Alternatively, geodesic distance can be approximated by a sequence of small Euclidean distances that converge to the geodesic in the limit. For example, Klassen et al. [76] used a shooting approach to compute geodesic distances between shapes. The algorithm involves taking small increments in the tangent space at a point, and then re-projecting onto the manifold. Since the tangent space is a vector space, the Euclidean metric is used to compute the small distances at every step. The shooting approach requires a measure of "energy" along which to shoot at each point. Also, distances computed by shooting are sensitive to local minima. Given a suitable energy function, the shooting approach is attractive not only for its simplicity but also for providing well-defined statistical measure. The approximate geodesic distance need not provide physical insight into the inherent structure that is reflected in star diagrams (section 3.3.2).

We propose a way of computing distance between epitomes by choosing a metric motivated by geometry of the underlying space as in the former approach. At the same time, analytical tractability is retained by computing distances along each of the components following the Iwasawa matrix decomposition. The space underlying dynamics is called Finsler space [73]. Finsler geometry is concerned with integrals of the form

$$\int_a^b f(q^1, \ldots, q^n; \frac{dq^1}{dt}, \ldots, \frac{dq^n}{dt})dt.$$

The function $f(q^1, \ldots, q^n; \frac{dq^1}{dt}, \ldots, \frac{dq^n}{dt})$ is positive unless all $\frac{dq^i}{dt}$ are zero. Here q

stands for position and $\frac{dq}{dt}$ for velocity so that F denotes speed of the moving object. The integral measures the total distance traveled. Finsler spaces also appear in optics where the amount of time it takes for light to travel is captured by $f(\cdot)$. Another example of Finsler spaces occurs in applied mechanics where the extent of elastic-plastic deformation of bodies is measured. The natural metric in Finsler space is the Finsler-Minkowski metric.

3.5.1 Finsler-Minkowsi metric

Let $F \in G$ and $\Delta : TG \to \mathbb{R}$ be a continuous mapping such that for any $p \in G$, $\alpha \in \mathbb{R}$ and $\xi \in T_pG$

$$\Delta_p(\alpha\xi) \quad = \quad |\alpha|\Delta_p(\xi). \tag{3.20}$$

Then for any piecewise C^1 curve $\eta : [p_1, p_2] \to G$ its length can be defined by

$$l(\eta) = \int_{p_1}^{p_2} \Delta_{\eta(t)}(\dot{\eta}(t))dt,$$

which gives rise to a non-oriented distance on G. Instead of the non-oriented distance in (3.20), consider an oriented version obtained by an additional constraint of positive homogeity, i.e.,

$$\Delta_p(\alpha\xi) \quad = \quad \alpha\Delta_p(\xi). \tag{3.21}$$

A special case of (3.21) is the Finsler-Minskowsi metric $\Delta(F, \dot{F}) = \Delta(F^{-1}\dot{F})$ [77].

The geodesic distance $D : G \times G \to \mathbb{R}$ between matrices F_0 and F_1 can by

calculated using the norm induced by the inner product Δ as follows:

$$D(F_0, F_1) \;=\; \min\{I_\Delta(F(\cdot)) : F \in C^1([0,1], G), F(0) = F_0, F(1) = F_1\}, \quad (3.22)$$

with $I_\Delta(F(\cdot)) = \int_0^1 \Delta(F, \dot{F})dt$. More precisely, the minimum in (3.22) should be replaced by an infimum and existence of geodesics has to be established. It is known that geodesics exist as long as the manifold is complete ([73], [78]).

We illustrate the computation of geodesic distances using examples.

Example 2 *Consider* $A = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \end{pmatrix}$. *A simple calculation shows that the norm* $\Delta(A)$ *is*

$$\Delta(A^{-1}\dot{A}) \;=\; \frac{|\dot{a_1}|}{a_1} + \frac{|\dot{a_2}|}{a_2} \quad (3.23)$$

The shortest path distance from the identity matrix I *to* A *is* $D(I, A) = |\log a_1| + |\log a_2|$.

Example 3 *Consider* $N \in Aff(1)$ *(also known as the "*$ax + b$*" group), i.e.,* $N = \begin{pmatrix} \rho & \beta \\ 0 & 1 \end{pmatrix}$. *The Finsler-Minkowski metric becomes* $\Delta(N^{-1}\dot{N}) = \frac{|\dot{\rho}|+|\dot{\beta}|}{\rho}$. *For a unit upper triangular matrix,* $\alpha = 1$ *so that the shortest path distance minimizes* $\int_0^1 \dot{\beta}d\beta$.

Example 4 *For* $K = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$, *the Martin distance in (3.19) can be used.*

3.6 Applications

We illustrate the usefulness of epitomic representation of activities and the Iwasawa matrix decomposition using activity recognition, clustering and temporal

segmentation.

3.6.1 Activity Recognition

Epitomic representation allows us to recognize activities based on motion properties within video segments. Since it does not rely on a global model, it provides a degree of robustness to incomplete trajectories. Some common causes for missing parts of trajectories are occlusion and tracking failure. The steps involved in activity recognition using epitomes is outlined below.

Moving objects are detected and their motion trajectories are extracted as described in section 3.2. Epitomes for segments of each trajectory are then computed. The Iwasawa matrix decomposition is used to factorize the system matrices into the three components K, A, N that represent rotation, scaling and translation. The components are compared their counterparts obtained using the reference set. Distances between components are calculated using the Finsler-Minkowski metric (section 3.5).

Apart from their usefulness for activity recognition, the geometrical significance of the Iwasawa components can be exploited by analyzing their effect separately. In particular, the N and K components provide perceptually useful cues as described in the remaining part of this section.

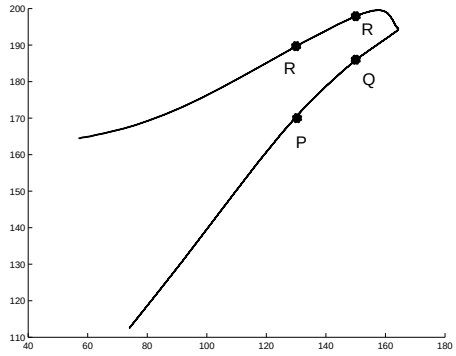


Figure 3.4: Illustrating the effectiveness of the N component of the Iwasawa decomposition using a sample trajectory from the UCF dataset.

3.6.2 Key Frame Detection using the N Component

In the proposed method, epitomes are estimated using video segments with uniform boundaries that are pre-specified (e.g., a video segment may be defined as a sequence of 100 frames). Though such segmentation is computationally attractive, it may not provide perceptually informative cues for video indexing. The N component of the Iwasawa decomposition provides a way to overcome the limitation to some extent. From (3.4), we know that the N component has a unit upper triangular structure. In particular, for $F \in GL(2, \mathbb{R})$, the N component has one free element $\beta \in \mathbb{R}$. We propose a temporal segmentation based on sign changes of β since it reflects abrupt changes in direction of motion. The following example is used to show the usefulness of $sign(\beta)$.

Let $x_k = (x_k^{(x)}, x_{k-1}^{(x)}, x_{k-1}^{(y)}, x_k^{(y)})$ represent the state, where $(x_k^{(x)}, x_k^{(y)})$ denotes the position of the object at time k. Figure 3.4 shows a trajectory from the UCF dataset for picking up an object. The discrete time second-order decoupled system

can be written as

$$x_{k+1} = \begin{pmatrix} f_1^{(x)} & f_2^{(x)} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & f_1^{(y)} & f_2^{(y)} \end{pmatrix} x_k + \begin{pmatrix} u_k^{(x)} \\ 0 \\ 0 \\ u_k^{(y)} \end{pmatrix} \tag{3.24}$$

For simplicity, consider the x component of motion, i.e., the top left 2×2 block of the system matrix in (3.24). The variation of the N component is discussed as the object traces the path $PQRS$ in three consecutive epitomes. The x and y co-ordinates of the points are denoted using superscripts, e.g., $P = (P^{(x)}, P^{(y)})$. The effect of N component is given by

$$x_{k+1}^{(x)} = \begin{pmatrix} 1 & \beta^{(x)} \\ 0 & 1 \end{pmatrix} x_k^{(x)} + \begin{pmatrix} u_k^{(x)} \\ 0 \end{pmatrix}, \tag{3.25}$$

where the superscript on the truncated state vector denotes motion of the x component. Using (3.25),

Case 1 *Point P to Q: As the object moves from P to Q, since $P^{(x)} < Q^{(x)}$,* $\beta^{(x)} > 0$.

Case 2 *Point Q to R: As the object moves from Q to R, since $P^{(x)} = Q^{(x)}$,* $\beta^{(x)} = 0$.

Case 3 *Point R to S: As the object moves from R to S, since $P^{(x)} > Q^{(x)}$,* $\beta^{(x)} < 0$.

So the zero crossings of $\beta^{(x)}$ (and $\beta^{(y)}$) denote perceptually significant changes. The frames corresponding to the zero crossings are said to be key frames.

3.6.3 Clustering using the K component

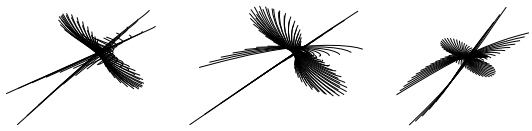
As shown in the star diagrams (figure 3.2), epitomes tend to cluster to produce certain dominant patterns in motion trajectories. In particular, similarities in the shape of trajectory segments are captured by star diagrams. This suggests that the K component of the Iwasawa decomposition can be used to cluster epitomes, since it indicates direction of eigenvectors of the system matrix. We use the k-means algorithm to cluster trajectories based on parameter θ of the K component. The value of the number of clusters k is manually initialized.

3.7 Experiments

We demonstrate the usefulness of epitomic representation of activities and the Iwasawa matrix decomposition using the UCF human action dataset and the TSA airport surveillance dataset.

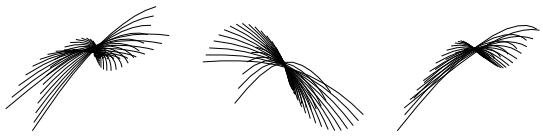
3.7.1 UCF human actions dataset

A brief description of the dataset along with detection and tracking algorithms is prresented in section 1.3.1. Figure 3.5 shows star diagrams for different cases of *opening the cabinet door*. The star diagrams for several instances of *picking up an object* is shown in figure 3.6.



(a) (b) (c)

Figure 3.5: UCF Dataset: Star diagram for *opening the door*.



(a) (b) (c)

Figure 3.6: UCF Dataset: Star diagram for *picking up an object*.

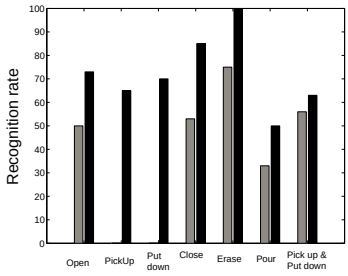


Figure 3.7: Cumulative match score (CMS) percentages for activity recognition at rank 1 (recognition rate) and rank 5. Recognition rates in column 2 for proposed method is compared with those reported in [1] in column 4)

3.7.1.1 Activity recognition:

We follow Rao et al.'s ([1]) division of the dataset into gallery and probe sets to ensure that the results remain comparable. Trajectories in the gallery set are labeled with the activity, whereas those in the probe set are matched with trajectories in the gallery set. For every probe trajectory in the dataset of 60 trajectories, the remaining 59 form the gallery.

An extracted motion trajectory from the gallery set is divided into segments of 20 frames with overlap. The state vector is formed using the instantaneous position along the trajectory. An epitome $(x_0; F; \mu_u, \Sigma_u)$ is estimated for every segment as described in section 3.2. All 16 elements of $F \in GL(4, \mathbb{R})$ are estimated using least squares (section 3.2). The Iwasawa decomposition is used to find the K, A, N components of the estimated F matrices.

The distance from a test video sequence to those in the gallery is computed as follows. Epitomes of the test video sequence as before and its K, A, N components are computed. The distance between components are computed separately using the metrics as described in examples 2, 3 and 4 (section 3.5). The recognition rates are summarized in table 3.7 along with comparable rates in [1].

3.7.1.2 Key frame detection

Activities such as picking up objects and opening a cabinet door have distinctive points along the trajectory that contain a sharp change in motion. These points denote perceptually siginificant time instants when the object is picked up or when

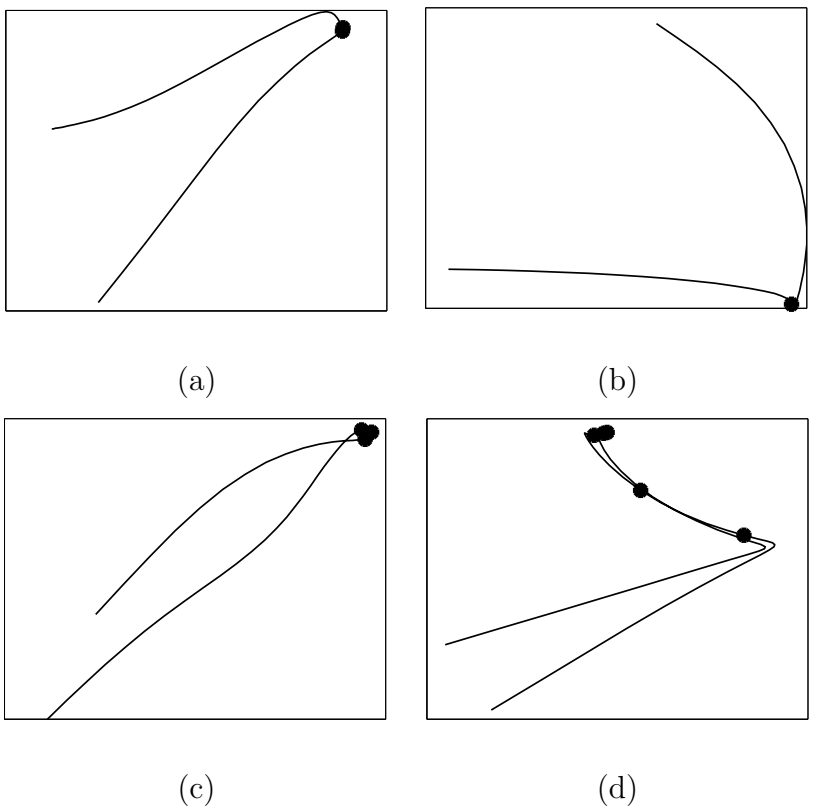


Figure 3.8: UCF dataset: Temporal segmentation of trajectories. The dots represent the segment boundaries detected using zero crossings of the N component.

the door is opened. The N component can be used to detect these time instants.

Motion trajectories are modeled using a decoupled second-order system (3.24) so that parameters $\beta^{(x)}$ and $\beta^{(y)}$ can be used to find key frames (section 3.6.2). The system matrices are factorized using the Iwasawa decomposition. The N component of $F \in GL(4, \mathbb{R})$ has two real-valued parameters $\beta^{(x)}$ and $\beta^{(y)}$ as indicated in (3.24) and (3.25). A key frame is said to be detected when a zero crossing is detected in either $\beta^{(x)}$ or $\beta^{(y)}$ sequence.

Figure 3.8 shows the detected segment boundaries for a few activities. In figures 3.8(a)-(c), trajectories for picking up an object are shown along with the

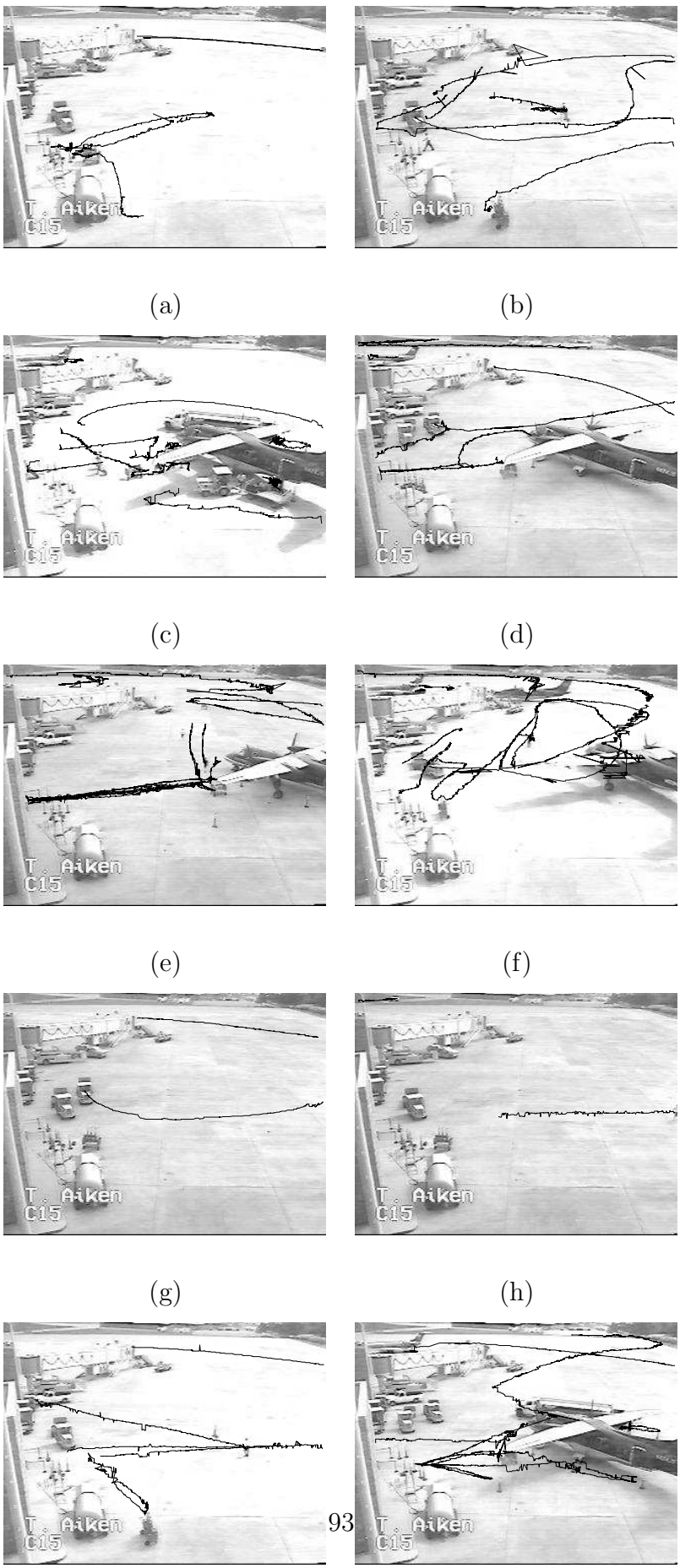
Table 3.2: Accuracy of key frame detection using the UCF dataset. The table shows the averge difference between the accuracy and the number of key frames detected in the proposed method and those reported in [1].

	Avg. difference in # frames	Avg. difference in location of key frames
Open door	1.2	7.2
Pick up object	1.1	3.9
Put down object	1.1	5.9
Close door	2.7	8.1
Pick up object and put down elsewhere	2.1	6.1
Pour water	1.8	3.3
Erase board	4.2	5.1

detected key frames. Table 3.2 shows the accuracy of the detected key frames in which the results from the zero crossings of $\beta^{(x)}, \beta^{(y)}$ are compared with the location of key frames obtained manually.

3.7.2 TSA airport tarmac surveillance dataset

The TSA airport dataset is summarized in section 1.3.2. A video segment that is ten thousand frames (7 minutes) long is referred to as a block for convenience. A snapshot from a block may not be sufficient to describe the activity within that block. For each of the blocks in figure 3.9, dominant activities for each block are



(a) (b)

(c) (d)

(e) (f)

(g) (h)

(i) (j)

Figure 3.9: Motion trajectories for ten blocks of TSA data.

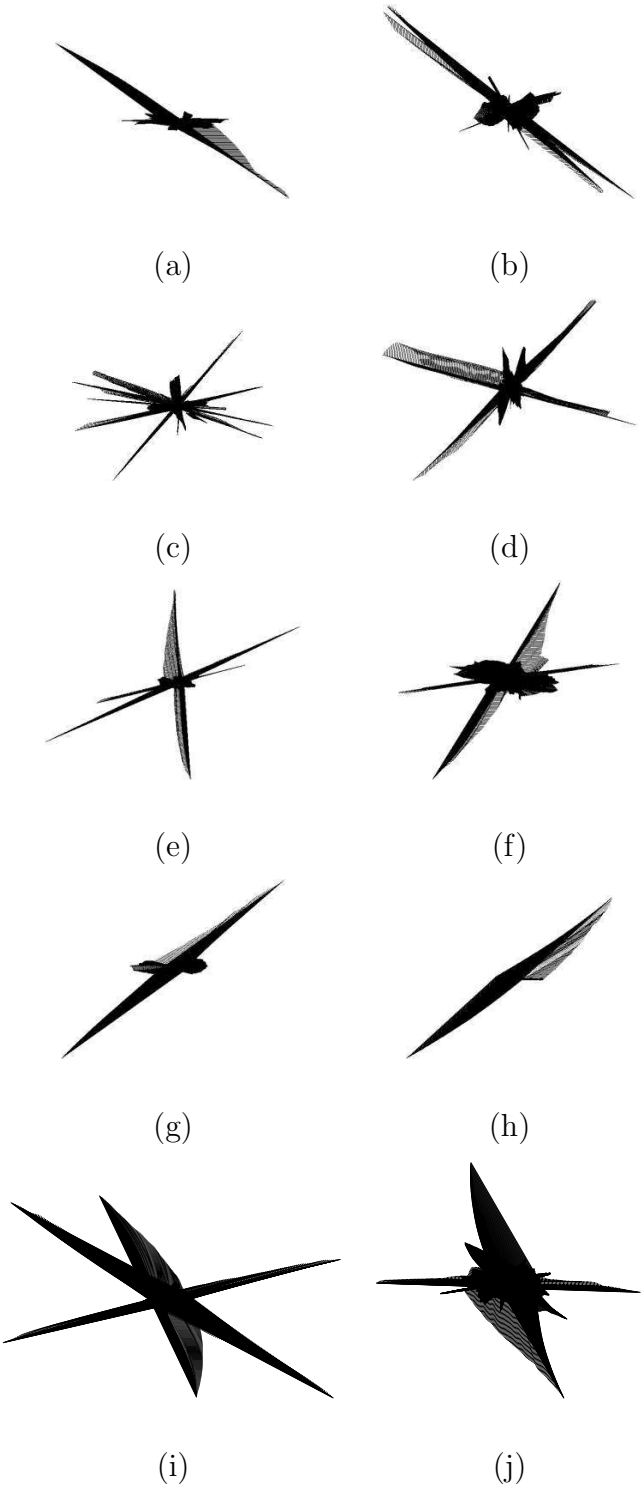


Figure 3.10: TSA dataset: Star diagrams for activities in blocks of video sequences. The indices (a)-(j) correspond to those in figure 3.9.

summarized below:

- Block (a): Ground crew walk from gate to center of the scene and returns; another ground crew drives away in a truck.
- Block (b): Ground crew walk back and forth from the gate; a ground crew drives away in a luggage cart.
- Block (c): Luggage cart activity around plane and leaves; ground crew walks around plane; and to the gate.
- Block (d): Luggage cart and ground crew activity around the plane; luggage cart leaves; truck crosses the scene.
- Block (e): Plane enters; luggage cart goes to plane; passengers embark.
- Block (f): Luggage cart goes to plane; ground crew from gate to plane and back. Luggage cart leaves.
- Block (g): Ground crew walk from gate to truck and drive away; truck crosses scene at the top.
- Block (h): Ground crew walk across scene.
- Block (i): Ground crew walk across scene and back to gate; truck crosses scene at the top.
- Block (j): Ground crew walk from gate to plane and back; walks across the scene and back to gate.

3.7.2.1 Activity recognition:

Each motion trajectory is modeled separately so that at any given time there are as many epitomes as the number of objects in the scene. We test the proposed method for activity recognition. In figure 3.9, the right column shows the closest match for blocks of activity in the left column. The corresponding segmentwise trajectories (temporally overlaid) are shown in figure 3.10 in the same order, i.e., figure 3.10(a) shows segments of trajectories for activity block in figure 3.9(a); similarly for (b),(c), etc. From the list of activities in each block given above, the top matches were correct, except for block (g). Since we have approximated moving objects as points, a truck driving away slowly was confused with a human (ground crew) walking along a similar path.

3.7.2.2 Key frame detection using the N component

Trajectories are extracted as described in section 3.2 and epitomes are computed for segments of trajectories assuming a decoupled second-order model (section 3.6.2). This is the same model that was used to identify key frames in the UCF dataset (section 3.7.1). We test the efficacy of the N component for identifying the completion of luggage transfer between an aircraft and luggage cart. In the two hour long sequence, there are eighteen instances showing presence of luggage cart in the scene. Of these four instances correspond to movement across the scene without meeting an aircraft. In the remaining fourteen instances (or seven roundtrips to the aircraft), the luggage cart goes to the aircraft to transfer luggage. The objective is

to identify trajectories that correspond to these exchanges and the time instant at which the exchange occurs.

Motion blobs are detected and tracked as described in section 3.2. Since we are interested in activities involving luggage carts, we track motion blobs whose size exceeds a certain threshold. The smaller blobs (which correspond to humans) are discussed in the next section. The appearance-based thresholding greatly reduces the number of errors in tracking that are caused when humans and luggage cart merge and split. Of the fourteen cases in which luggage carts were present, twelve were correctly identified. The average error in localizing the time at which the transfer occurs was eighteen frames. The following objects caused three false alarms:

- Movement of fuel truck
- Two cases of truck moving to the terminal, stopping briefly before exiting the scene.

3.7.2.3 Clustering using the K component

There are two classes of humans in the dataset: passengers and ground crew persoonel. It is necessary to distinguish between these two classes for applications such as anomaly detection. Unexpected motion patterns of passengers may be considered anomalous unlike the same motion pattern involving ground crew personnel. We use the K component of the epitome to distinguish between these two classes since the motion of passengers is tightly clustered in this space. The K component represents rotation of the state vector as shown in (3.4). As shown in figure 3.9,

the trajectories of passengers are tightly clustered as they walk from the gate to the aircraft or vice-versa. This creates two values of θ that parameterizes the x component in the epitomic representation.

In this experiment, we retain motion blobs whose size was below the threshold in the previous experiment (section 3.7.2.2). The motion trajectories were modeled using a decoupled second-order system (section 3.6.2). The K components of system matrices are computed using the Iwasawa decomposition. The K matrices are characterized by a single parameter θ as given by (3.4). The θ values are computed and used for clustering using a k-means procedure with $k = 2$. Two cluster centers are chosen since we are interested in modeling passengers as they embark an aircraft or disembark from it.

There were nineteen trajectories of passengers in the dataset. All of these were correctly clustered in the one of the two passenger clusters. There were twelve thirty six trajectories corresponding to ground crew personnel. Of thirteen were wrongly classified as passengers since they followed the same path as the passengers. It is difficult to reduce these errors without using domain knowledge (e.g., assuming that the passengers appear within a given time interval).

3.8 Summary

An epitomic representation for modeling activities was described in this chapter. The kinematics of motion is assumed to be linear within video segments. The system matrix that represents the linear kinematics along with the statistics of the

input signal and initial value of the state is said to be an epitome. Star diagrams are introduced as a new way of visualizing epitomes.

The physical significance of eptitomic representation is highlighted using the Iwasawa matrix decomposition that isolates th eeffect of rotation, scaling and projective action of the state vector as the object moves in the scene. Gedesic distance was introduced as a measure of similarity between epitomes that is both physically meaningful and mathematically well-defined. The Iwasawa decompotion factors was shown to be useful in computing the distance along each component.

Chapter 4

Mixed State Space Models

4.1 Introduction

Modeling complex activities involves extracting spatio-temporal descriptors associated with objects moving in a scene. It is natural to think of activities as a sequence of segments in which each segment possesses coherent motion properties. There exists a hierarchical relationship extending from observed features to higher-level behaviors of moving objects. Features such as motion trajectories and optical flow are continuous-valued variables, whereas behaviors such as start/stop, split/merge and move along a straight-line are discrete-valued. Mixed state models provide a way to encapsulate both continuous and discrete valued states.

In general, the activity structure, i.e., the number of behaviors and their sequence, may not be known a priori. It requires an activity model that can not only adapt to changing behaviors but also one that can learn incrementally and 'on the fly'. Many existing approaches assume that the structure of activities is known; and a fixed number of free parameters is determined based on experience or by estimating the model order. The structure then remains fixed. This may be a reasonable assumption for activities such as walking and running, but becomes a serious limitation when modeling complex activities in surveillance and other scenarios. We are interested in these classes of activities. Instead of assuming a fixed global model

order, local complexity is constrained using dynamical primitives within short-time segments. We choose a basis of behaviors that reflects generic motion properties to model these primitives. For example, the basis elements represent motion with constant velocity along a straight line, curved motion, etc. Using the basis of behaviors, we present two Behavior-Driven Mixed State (BMS) models to represent activities: offline and online BMS models. The models are capable of handling multiple objects, and the number of objects in the scene may vary with time. The basis elements are not specific to a particular video sequence, and can be used to model similar scenarios.

We present a Viterbi-based algorithm to estimate the switching times between behaviors and demonstrate the usefulness of the proposed models for temporal segmentation and anomaly detection. Temporal segmentation is useful for indexing and easy storage of video sequences, especially in surveillance videos where a large amount of data is available. Besides the inherent interest in detecting anomalies in video sequences, anomaly detection may also provide cues about important information contained in activities.

The rest of the chapter is organised as follows. Section 4.2 describes low-level processing methods for detecting and tracking moving objects. The kinematics of extracted trajectories is modeled using linear systems. Section 4.3 describes offline and online BMS models. Section 4.4 describes a basis for representing segments of video sequences, and a Viterbi-based algorithm for segmentation. Section 4.5 illustrates the usefulness of the proposed method using temporal segmentation and anomaly detection. The airport surveillance TSA dataset, the bank surveillance

dataset and the UCF database of human actions are used.

It may be useful to compare the proposed models with the HMM approach and other mixed state models in order to place our work in context. The HMM topology, i.e., the number of states and the structure of the transition matrix is assumed to be known. The state transitions are assumed to be Markovian. The observed data is assumed to be conditionally independent of its past given the current hidden state. Also, the output distribution is assumed to be stationary. This makes the estimation procedure tractable. The Viterbi algorithm is then used to find the optimal state sequence efficiently.

We address some of these issues in the proposed activity model. In particular, the evolution of hidden (discrete) states is allowed to depend on the continuous state, which relaxes the Markov assumption. This causes the computational complexity of the parameter estimation process to grow exponentially [41]. To overcome this problem, we introduce a basis of behaviors motivated by motion properties of typical activities of humans and vehicles within a short-time window. A basis can be chosen so that it applies to similar scenarios across datasets. In our experiments, the same basis of behaviors is used in both the TSA airport surveillance dataset and the bank monitoring dataset. Further, we present a cost-based Viterbi algorithm instead of the usual probability based one, since it is not easy to compute the normalization terms of the probability distribution.

Remark on notation and terminology: We use the term non-stationary activities to suggest that parameters of behavior can change with time. The term has been used in similar contexts in both speech [79] and activity recognition [20].

Throughout the chapter, we use $x(t) \in R^n$ to represent a continuous valued variable and $q(t) \in \{1, 2, \ldots, N\}$, a discrete valued variable. We use the notation $x_{t_1}^{t_2}$ to denote the sequence $\{x(t_1), x(t_1 + 1), \ldots, x(t_2)\}$.

4.2 Low-level video processing

The types of activity of interest may be illustrated using the following example. In video sequences of an airport tarmac surveillance scenario, we may observe segments of activities such as movement of ground crew personnel, arrival and departure of planes, movement of luggage carts to and from the plane and embarkation and disembarkation of passengers. The video sequences are usually long. It would be useful to segment and recognize activities for convenient storage and browsing. Viewed as an inference problem, activity modeling involves learning parameters of behaviors using motion trajectories extracted from video sequences.

4.2.1 Detection and tracking

Moving objects are detected and their motion trajectories are extracted as described in section 3.2. The effect of errors in low-level processing on activity modeling is discussed in section 4.5. Of the three datasets used in the experiments, tracking was accurate and reliable in the indoor bank monitoring dataset and the UCF human action dataset. On the other hand, there were a few tracking errors in the TSA airport tarmac surveillance dataset that caused errors in temporal segmentation.

In the case of a single object moving in the scene, its motion trajectory and velocity (computed using finite differences) forms the continuous valued state $\{x(t), t \in [0, T]\}$, where, $x(t) \in \mathcal{R}^4$. When several objects are present in the scene, this can be extended in a relatively straightforward manner if the number of objects remains constant. If the number of objects varies with time, there are several ways of defining the continuous state as described in the next section

4.2.2 Handling multiple objects

Let $m(t)$ be the number of objects present in the scene at time t. Let $X_c(t) \in \mathcal{R}^{4m(t)}$ represent the composite object. We use the notation $\mathbf{X_c}(t)$ to indicate the sequence $\{X_c(1), X_c(2), \ldots, X_c(t)\}$. Each of the m trajectories is associated with the observation sequence with four components representing the 2-D position and velocity. Clearly, the number of objects $m(t)$ need not be constant. This problem of varying dimension can be handled in several ways. For example, $m(t)$ can be suitably augmented to yield a constant number M by creating virtual objects. In [20], motion trajectories are represented using Kendall's shape space. The trajectory is resampled so that the shape is defined by k points. As an illustration, consider the trajectory formed by passengers (treated as point objects) exiting an aircraft on a tarmac and walking toward the gate. The number of passengers in the scene $m(t)$ can vary with time. Irrespective of the value of $m(t)$, a common motion trajectory can be formed by connecting the position of the first passenger to that of the last passenger such that the curve passes through every passenger in the scene. The

common trajectory is resampled at k points creating k virtual passenger positions, and used to represent the shape. This is equivalent to defining an abstracting map from a $4m(t)$-D space to a $4k$-D space. When the objects are not interacting or the nature of interaction is unknown, it is not clear how to place the k virtual objects to obtain a constant cardinality.

Though there may be several objects in the scene, there are only a few types of activities. For instance, in a surveillance scenario, there may be several persons walking on a street. Each person has his/her own dynamics whose parameters can vary. Walking activity, however, is common across persons. This motivates the usefulness of constructing a basis of behavior. In this example, the direction and speed of walking could distinguish different basis elements.

The choice of a basis of behavior depends on the domain of application, but need not be specific to datasets. In our experiments, we use the same basis across two surveillance scenarios, one captured on airport tarmac and the other inside a bank. If there is insufficient domain knowledge to guide the selection of a basis, a generic basis based on eigenvalues of the system matrix can be used to distinguish between basis elements (section 4.3.3).

The dynamics of objects in the scene is modeled individually using the most likely basis element. The number of objects $m(t)$ is allowed to vary at discrete time intevals so that $m(t)$ is constant over a short video segment. The change in the value of $m(t)$ is modeled as a one-step random walk. The conditional probability distribution function (pdf) for a segment s can be written as $f(\mathbf{X_c}(t), m(t)|S = s) = b_{s,m}(\mathbf{X_c}(t))P(m(t) = m|S = s)$. A behavior segment $s \in \mathcal{S}$ is characterized by the

distribution of the number of objects in the scene $P(m|s)$ and a family of distributions $b_{s,m}(\mathbf{X_c}(t))$ that describes the segment. The pdf $b_{s,m}(\mathbf{X_c}(t))$ is calculated using a basis of behaviors. This value is used for temporal segmentation (section 4.4.1). To place this definition in context, consider an HMM. In this case, the probability of the segment is written as the product $b_{s,m}(\mathbf{X_c}(t)) = \prod_{i=1}^{t} f(X_c(i)|s)$ and the HMM persists in this state with a geometric distribution.

4.3 Mixed State Models

Let the sequence of discrete states be $\{q(1), q(2), \ldots, q(T)\}$, where $q(i) \in \{1, 2, \ldots, N\}$ indexes the discrete valued behavior. The objects may transit through M behaviors, switching at time instants $\tau = \{\tau_0, \tau_1, \ldots, \tau_M\}$, where $\tau_0 = 0, \tau_M = T$. In general, the number of behaviors M and switching instants τ_i's are unknown. We present two BMS models to represent the behavior within such segments: offline and online BMS models respectively.

Consider the general state equations of continuous and discrete variables.

$$\dot{x}(t) = h_{q(t)}(x(t), u(t)), x(0) = x_0 \tag{4.1}$$

$$q^{+}(t) = g(q_1^{t-1}, x_1^{t-1}, n(t)) \tag{4.2}$$

The continuous state dynamics $h_{q(t)}$ depends on the discrete state $q(t)$. It captures the notion that a higher-level behavior evolves in time and generates correlated, continuous-valued states $x(t)$. The continuous state dynamics within each segment is limited by the form of $h_{q(t)}$. The discrete state $q(t)$ evolves according to $g(\cdot)$ and depends not only on the previous discrete state, but also on past values of the

observed data x_1^{t-1}. $u(t)$ and $n(t)$ represent noise. This makes the evolution of discrete state non-Markovian.We make the following assumptions.

A1: The number of discrete state switching times is finite.

A2: Discrete state transitions occur at discrete time instants, i.e., $\tau_i = k\alpha$ for $i = 1, \ldots, M-1$, where k, α are integers.

A3: Between consecutive switching instants $\tau_i, \tau_{i+1}, i = 1, \ldots, M$, the parameters of the continuous dynamical model do not change.

(A1) ensures that we do not run into pathological conditions such as Zeno behavior[1]. (A2) and (A3) are the practical conditions required for robust estimation of parameters of each segment. We arrive at the offline and online BMS models by making certain additional assumptions in (4.1) and (4.2) as explained in sections 4.3.2 and 4.3.3.

4.3.1 Special case: AR-HMM

Before describing the proposed mixed state models, we review the auto-regressive (AR) HMM, which is a special case of (4.1) and (4.2). The AR-HMM was introduced in [80] using a cross entropy setting. In addition to (A1)-(A3), the AR-HMM requires the following assumptions:

A4: The number of discrete states N is known.

[1]Roughly speaking, an execution of a mixed system is called Zeno, if it takes infinitely many discrete transitions in a finite time interval.

A5: The processes are stationary and the model parameters do not depend on time.

Similar to the HMM, the hidden state in the AR-HMM follows the Markov dynamics.

$$P(q(t)|q_1^{t-1}, x_1^{t-1}) \quad = \quad P(q(t)|q(t-1)) \tag{4.3}$$

The joint distribution of the continuous and discrete states can be written as follows.

$$\begin{aligned} f(x(t), q(t)|x_1^{t-1}, \quad q_1^{t-1}) \quad &= \\ f(x(t) \quad &, q(t)|q(t-1), x_{t-\alpha-1}^{t-1}) \end{aligned} \tag{4.4}$$

This is useful for obtaining the optimal state sequence using the Viterbi algorithm. Using (4.3) and (4.4), we have

$$\begin{aligned} f(x(t), q(t)|q(t-1), x_{t-\alpha-1}^{t-1}) \quad &= \quad f(x(t)|q(t), x_{t-\alpha-1}^{t-1}) \\ &\times P(q(t)|q(t-1)) \end{aligned} \tag{4.5}$$

The distribution $f(x|\cdot,\cdot)$ is assumed to be normal. The mean and variance depends on the discrete state. The parameters can be estimated using these hypotheses in an EM setting [26].

4.3.2 Offline BMS model

The Markov assumption of discrete state evolution in (4.3) means that the behavior parameters change without a direct dependence on the observed data. It would be more reasonable to allow past values of observed data to influence changes in behavior. So we present an offline BMS model whose discrete state transition is given by the following:

$$f(q(t)|q_1^{t-1}, x_1^{t-1}) \quad = \quad f(q(t)|q(t-1), x_{t-\beta}^{t-\alpha}), \tag{4.6}$$

where $q(t) \in \{1, \ldots, N\}$ for some known number of states N and $\beta = k\alpha$ for some integer k. Let the effective state be $r(t) = (q(t), x_{t-\beta}^{t-\alpha})$ so that (4.6) can be rewritten. The state evolution of $r(t)$ is Markov and the parameters and switching times can be computed, in principle, using algorithms similar to the AR-HMM case. The computation of the parameters, however, is not as elegant as the classical HMM and it is difficult to construct a recursive estimation procedure like the EM algorithm (briefly described in section 4.4). Also, the transition probability $P(r(t)|r(t-1))$ depends on the observed data and violates assumption (A5). The transition probability of the effective state can be written as follows:

$$\begin{aligned} f(r(t)|r(t-1)) &= f(x_{t-\beta}^{t-\alpha}|q(t), q(t-1), x_{t-\beta}^{t-\alpha}) \\ &\quad f(q(t)|q(t-1), x_{t-\beta}^{t-\alpha}) \qquad (4.7) \\ &= \frac{f(r(t-1)|q(t), q(t-1), x_{t-\beta}^{t-\alpha})}{f(x_{t-\beta}^{t-\alpha}|q(t-1))} \\ &\quad \times (f(x_{t-\beta}^{t-\alpha}|q(t), q(t-1)) \\ &\quad \times f(q(t)|q(t-1))) \qquad (4.8) \end{aligned}$$

The probability in (4.7) is difficult to compute due to two main reasons. Unlike (4.3), (4.7) depends on $x_{t-\beta}^{t-\alpha}$. So the transition probability matrix is no longer stationary. For parameter estimation using the EM algorithm, the denominator term in (4.8) cannot be computed. So we turn to the underlying state equation (4.1), and define an offline BMS model as a sequence of linear dynamics. The calculation of probabilities can be replaced with running and switching costs incurred due to the estimated dynamical parameters. In addition to (A1)-(A4), we assume the following:

A6: The segment-wise dynamics are linear, i.e., (4.1) takes the following form:

$$\dot{x}(t) \quad = \quad A_{q(t)}x(t) + b, x(0) = x_0, \tag{4.9}$$

where $A_{q(t)} \in \{A_1, A_2, \ldots, A_N\}$ for some known N, are obtained by training.

The offline BMS model can be used for activity recognition and anomaly detection. Using training data, we can compute the parameters of normal behaviors. This allows us to not only check for anomalies but provides a way to localise anomalous parts of the activity, i.e., the unexpected $A_{q(t)}$ segments.

4.3.3 Online BMS model

If the parameters of behaviors are unknown or time-varying, an activity model that can estimate parameters of the model 'on the fly' is needed. We present an online BMS model for non-stationary behaviors. Assume that (A1)-A(3) and (A6) hold, and relax (A4)-(A5). The number of segments N may be unknown, but (A6) can be used to restrict the complexity of $x(t)$ within a segment. This motivates the construction of a basis of behaviors. The basis elements represent generic primitives of motion depending upon the parameters of $A_{q(t)}$. Specifically, for the segment-wise linear dynamics of surveillance videos, we choose basis elements to model the following types of 2-D motion: straight line with constant velocity, straight line with constant acceleration, curved motion, start and stop.

The eigenvalues of the system matrix A are used to characterize the basis elements. Consider a linear time-invariant system $\dot{x(t)} = Ax(t)$, where A is a real valued square matrix. Fixing the initial state $x(0) = x_0$, we have $x(t) = exp(At)x_0$,

where $exp(At) = \Sigma_{k=0}^{\infty} \frac{t^k}{k!} A^k$ [81]. Depending on the eigenvalues λ_1, λ_2 of A, the equilibrium point exhibits the following types of behavior: curved trajectories (both eigenvalues are non-zero and real), straight line trajectories (one of the eigenvalues is zero), spiral trajectories (complex eigenvalues). These distinctions are syntactic rather than semantic, i.e., these types of motion may be considered as a context-free vocabulary. We use these as the basis to describe behaviors of segments. Though the total number of behaviors may be unknown a priori, we can specify a basis of behaviors by partitioning the space of dynamics using the location of eigenvalues, i.e., region in the space of allowable eigenvalues.

4.4 Approach

The estimation task in either offline or online BMS model consists of two main septs: computing the parameters of the behaviors, and identifying switching times between segments. It may be tempting to use the EM algorithm in this case [82]. The EM algorithm involves an iteration over the E-step to choose an optimal distribution over a fixed number of hidden states and the M-step to find the parameters of the distribution that maximize the data likelihood [82]. Unlike the classical HMM, however, the E-step is not tractable in switched state space models [41]. To work around this, [41] presents a variational approach for estimating the parameters of switched state space models, whereas [40] presents a sampling approach. Either of these approaches is applicable in the offline BMS case, but neither is suitable for the online BMS model. We propose an algorithm that has two main components:

a basis of behaviors for approximating behaviors within segments and the Viterbi-based algorithm.

The parameters of each segment is chosen so that the approximation error $R(\tau, t_0, q)$ defined below is minimized.

$$R(\tau, t_0, q) = \frac{1}{\tau - t_0} \int_{t_0}^{\tau} (x - \hat{x}_q)^T (x - \hat{x}_q) dt, \tag{4.10}$$

with $\hat{\dot{x}}_q(t)$ is a solution to (4.9).

$R(\tau, t_0, q)$ is the accumulated cost of using the q^{th} family of behaviors to approximate the current segment. For linear dynamics, the least square estimate minimizes this error. This is consistent with the probability density estimates under normal assumption for AR-HMM.

4.4.1 Viterbi-based algorithm

The Viterbi algorithm is used to find the optimal state sequence $Q = \{q(1), q(2), \ldots, q(T)\}$, for the given observation sequence $X = \{x(1), x(2), \ldots, x(t)\}$ such that the joint probability of states and observation is maximized. To place the proposed Viterbi-based algorithm in context, we trace the modifications starting with the Viterbi algorithm for the classical HMM approach. The quantity $\delta(t, i)$ is defined as follows [26]:

$$\delta(t, i) = \max_{q_1^{t-1}} f(q_1^{t-1}, q(t) = i, x_1^t | \lambda), \tag{4.11}$$

where λ is the given HMM or AR-HMM. In the classical HMM case, we assume a Markov state process $P(q(t)|q_1^T) = P(q(t)|q(t-1))$ and that the observations are

conditionally independent of the past given the current state, i.e.,

$$f(x(t)|x_1^T, q_q^T) \quad = \quad f(x(t)|q(t)). \tag{4.12}$$

It allows us to express (4.11) recursively as follows:

$$\delta(t,j) \quad = \quad \max_{1\leq i\leq N}[\delta(t-1,i)a_{ij}]f(x(t)|q(t)=j), \tag{4.13}$$

where $A = [a_{ij}]_{1\leq i,j\leq N}$ is the state transition probability matrix. The a_{ij}'s, which are stationary, can be estimated using the Baum-Welch algorithm (Appendix A). The trellis implementation of the Viterbi algorithm is used to compute the optimal state sequence efficiently. The size of the trellis is $N \times T$, where one observation variable $x(t)$ is involved at each stage [83]. In the AR-HMM, the observation probability equation is written as (4.4) instead of (4.12). It is easy to derive the optimal state sequence similar to the previous case. The major difference is that at each stage, the error computation involves a window of observed data $x_{t-\alpha-1}^{t-1}$ instead of one variable $x(t)$ [84].

Compared to AR-HMM, the offline BMS model is more general in that the evolution of state sequence is not Markov, but is allowed to depend on the continuous state (4.6). This makes the computation of joint probabilities for $\delta(t,i)$ difficult, as explained in section 4.3.2. The effective state $r(t) = (q(t), x_{t-2\alpha}^{t-\alpha})$, however, is Markov. We use this to set up a Viterbi-like algorithm based on approximation costs incurred in persisting in each behaviors and switching costs due to transitions among behaviors. If the denominator in (4.6) could be computed, then these costs could be readily turned into probabilities. Also the probability a_{ij} is not stationary anymore, and depends on the previous values of continuous state. The main difference in

implementation is a reduced size of the trellis. By assumption (A2), the size of the trellis reduces from $N \times T$ to $N \times K$, where $K\alpha = T$ and α is the minimum size of each segment. This time axis is further halved due to effective state $r(t)$ being Markov instead of $q(t)$ as shown in (4.7) and (4.8). The recursive equations are given below. The online BMS case presents an additional challenge due to non-stationarity. In this case, the N states represent N basis elements of behaviors.

In (4.13), the basic principle of dynamic programming is used to write the recursive equation using two quantities: observation probability $f(x|q)$ and the state transition probability a_{ij}. The approximation cost $R(\tau, t_0, q)$ is an analog of $f(\cdot|\cdot)$. We define the switching cost to be an analog of a_{ij}. For the BMS model, the transition probability for the effective state is given in (4.7). Using (4.6), we have

$$\begin{aligned} f(q(t) = j \quad | \quad & q(t-1) = i, x_{t-2\alpha}^{t-\alpha}) \\ = \quad & \frac{f(q(t) = j, q(t-1) = i | x_{t-2\alpha}^{t-\alpha})}{f(q(t-1) = i | x_{t-2\alpha}^{t-\alpha})} \end{aligned} \tag{4.14}$$

Using (4.14), the switching cost $S : \partial Inv(i) \times \partial Inv(j) \rightarrow \mathcal{R}^+$ is defined as follows:

Let $t_1 \in [\tau_i, \tau_{i+1})$ be a candidate switching time. Larger the value of the switching function, higher is the error due to switching at t_1, i.e., $\tau_{i+1} = t_1$, when the discrete state changes from m to n. The invariant set $Inv(i)$ denotes the continuous state dynamics for the hidden state i, i.e., as long as $x(t) \in Inv(i)$, we say that the object exhibits the behavior indexed by the index i. The boundary of the invariant set is denoted by $\partial Inv(i)$.

$$S(m, n) \quad = \quad \frac{(1 + R(t_1, \tau_i, m))(1 + R(\tau_{i+1}, t_1, n))}{(1 + R(\tau_{i+1}, \tau_i, m))} \tag{4.15}$$

The 1's are added to ensure that the function is well-defined at all time instants. If t_1 was the true switching time, the approximation error in the numerator will be smaller than that in the denominator.

Let $\delta(k,n)$ denote the cost accumulated in the n^{th} behavior at time k and $\psi(k,n)$ represent the state at time k which has the lowest cost corresponding to the transition to behavior n at time k. The time index k is used instead of t, to denote that switching is assumed to occur at discrete time instants (assumption (A2)).

- Initialization: For $n \in N$, let

$$\begin{aligned} \delta(1,n) &= R(1,1,n) \\ \psi(1,n) &= 0 \end{aligned}$$

- Recursion: For $2 \leq 2k \leq T$ and $1 \leq j \leq N$,

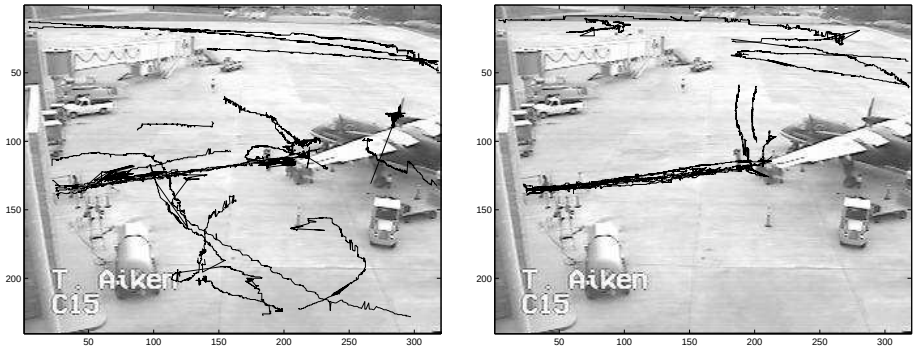
$$\begin{aligned} \delta(k,j) &= \min_{1\leq i\leq N}[\delta(k-1,i) - S(i,j)] - R(k,\tau_{k-1},j) \\ \psi(k,j) &= \arg\min_{1\leq i\leq N}[\delta(k-1,i) - S(i,j)] \end{aligned}$$

- Termination:

$$\begin{aligned} C^* &= \min_{1\leq i\leq N}[\delta(T,i)] \\ q^*(T) &= \arg\min_{1\leq i\leq N}[\delta(T,i)] \end{aligned}$$

- Backtrack: For $k = T-1,\ldots,1$,

$$q^*(k) = \psi(k+1, q^*(k+1))$$

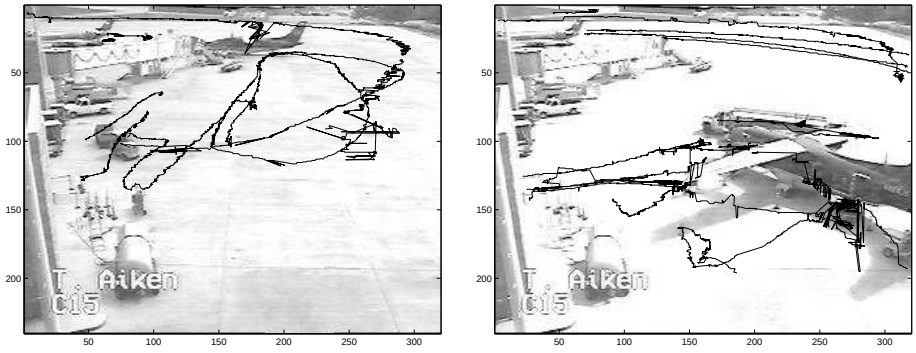


(a) (b)

Figure 4.1: TSA airport tarmac surveillance dataset. Each image represents a block of 10000 frames along with motion trajectories extracted.

4.4.2 Anomaly detection using Offline BMS model

It is common to have several examples of normal activities, and a very few samples of anomalies making it difficult to model anomalies. Therefore, anomaly detection can be formulated as change detection (or outlier detection) from the normal model. Anomalies can be either spatial, temporal or both. Examples of anomalies are path violation, gaining unrestricted access, etc. Offline BMS models are trained using normal video sequences. Given a test (anomalous) video sequence, motion trajectories and observation sequence are extracted as before. The Viterbi-based algorithm is initialized with parameters learnt using training data. If an unexpected state sequence is detected, an anomaly is declared. This assumes that short time dynamics is consistent with the normal activity, but anomaly exists due to an unexpected sequencing. Thus a completely unrelated activity would not be declared an anomaly.



(a) (b)

Figure 4.2: TSA airport tarmac surveillance dataset. Each image represents a block of 10000 frames along with motion trajectories extracted.

4.5 Experiments

We demonstrate the usefulness of the online BMS model for temporal segmentation and the offline BMS model for anomaly detection using the following three datasets: the TSA airport surveillance dataset, bank dataset and the UCF human action dataset.

4.5.1 TSA airport tarmac surveillance dataset

A brief description of the dataset along with detection and tracking algorithms is prresented in section 1.3.2.

In four blocks (a block is a ten thousand frame long video sequence), we observe a significant amount of multi-object activity when planes arrive and depart. The four blocks form the test set. Figures 4.1 and 4.2 shows the motion trajectories for these blocks. The remaining portion of the dataset is used as the training

Table 4.1: TSA dataset: Temporal segmentation of a block of video using the online BMS model. Legend: GCP= ground crew personnel, PAX= passengers, Det.=Segment Detected, TF=Tracking failed.

#	Block in Figure 4.1(a)	Comment
1	2 GCP split, walk away	Det.
2	GCP across tarmac	Det.
3	Truck arrives, GCP	Det.
4	GCP across tarmac	Det.
5	Truck	Det.
6	GCP movement	Det.
7	Plane-I arrives	Det.
8	Luggage cart to plane	Det.
9	Truck crossing scene	TF
	Plane-II arrives	Det.
10	GCP and luggage cart	
	approach plane-I	Det.
11	-	Extra segment
12	PAX disembark	Det., 2 extra segments

Table 4.2: TSA dataset: Temporal segmentation of a block of video using the online BMS model. Legend: GCP= ground crew personnel, PAX= passengers, Det.=Segment Detected, TF=Tracking failed.

#	Block in Figure 4.1(b)	Comment
1	Luggage cart	TF.
2	GCP movement near plane	Det.
3	Luggage cart enter	Det.
4	Plane enters	Det.
5	PAX embark	Det.
6	Luggage cart	TF.
7	PAX embark	Det.

Table 4.3: TSA dataset: Temporal segmentation of a block of video using the online BMS model. Legend: GCP= ground crew personnel, PAX= passengers, Det.=Segment Detected

#	Block in Figure 4.2(a)	Comment
1	Plane exits	Det.
2	GCP movement	Det.
3	Luggage cart to plane-II	Det.
4	Bag falls off luggage cart	Det.
5	Luggage cart to plane-II	Det.

Table 4.4: TSA dataset: Temporal segmentation of a block of video using the online BMS model. Legend: GCP= ground crew personnel, PAX= passengers, Det.=Segment Detected, TF=Tracking failed.

#	Block in Figure 4.2(b)	Comment
1	2 GCP movement	Det.
2	Luggage cart	TF
3	GCP movement	TF
4	Plane-II arrives	Det.
5	GCP movement	TF
6	Luggage cart to plane-II	Det.
7	Plane-III arrives	Det.
8	PAX embark	Det.
9	Truck movement	Det.
10	GCP near plane-II	TF
11	Luggage cart from plane-II	Det.
12	More PAX embark	Det.

set. It may seem large compared to the size of the test set. The activity content, however, is not as dense as in the test set. The paucity of training data makes it unrealistic to train a model in the conventional sense, where parameters of the mixed state model are estimated. Instead we train an online BMS model, which involves finding a basis of behavior. The values of parameters are less important than the region of parameter space they represent. Accordingly, the basis has elements that can produce the following types of motion: constant velocity along a straight line, constant acceleration along a straight line, curved trajectories with constant velocity, start and stop.

We demonstrate temporal segmentation of the four test blocks using the online BMS model. The segmentation results for the four blocks shown in figures 4.1(a)-(b) and 4.2(a)-(b) are summarised in Tables 4.1 - 4.4 respectively. On an average, there were 15% missed detections in segmentation. This was mainly because of tracking errors.

4.5.2 Bank surveillance dataset

The actors in the bank surveillance and monitoring dataset (section 1.3.3) demonstrate two types of scenarios:

- *attack* scenario where a subject coming into the bank forces his way into the restricted area. This is considered as an anomaly.
- *no attack* scenario where subjects enter/exit the bank and conduct normal transactions. This depicts a normal scenario. The normal process of trans-

(a) (b)

Figure 4.3: Bank dataset: Two segments detected in the *no attack* scenario: (a) A subject enters the bank, goes to the area where paper slips are stored. Another subject enters the bank and goes to the counter area, (b) Exit bank.

actions is known a priori and we train an offline BMS model using these trajectories.

4.5.2.1 Temporal segmentation

We retained the same basis of behavior that was used for the TSA dataset in section 4.5.1. Though the TSA data is captured outdoors and the bank data indoors, they are both surveillance videos. They retain similarity at the primitive or behavior-level. For the *no attack* scenario, segmentation using the online BMS model yielded two parts. In the first segment, we see two subjects entering the bank successively. The first person goes to the paper slips area and the second person goes to the counter. In the second segment, the two subjects leave the bank. Figure 4.3 shows sample images from the two segments. We store the parameters of these behavioral segments as the normal activity.

Figure 4.4 shows an example of an *attack* scenario. Here, the online BSM

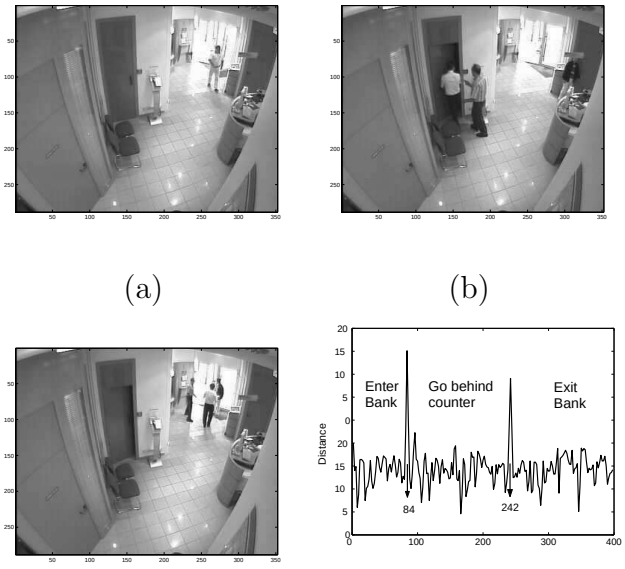


Figure 4.4: Bank dataset: Three segments detected in the *attack* scenario: (a) Enter bank, (b) Gain access to the restricted area behind the counter (c) Exit bank. (d) Shows a plot of the switching function. Peaks in the plot indicate boundaries in temporal segmentation.

model yielded three segments. In the first segment, the person enters the bank and proceeds to the area where the deposit/withdrawal slips are kept. This is similar to the first segment in the *no attack* case. During the second segment, he follows another person into the restricted area behind the counter. The third segment consists of the person leaving the bank.

4.5.2.2 Anomaly detection

The parameters of an offline BMS model are estimated using the *no attack* scenario. To detect the presence of an anomaly, we compute the error accumulated along the optimal state sequence using the test trajectory. It is difficult to assess the performance of this naive scheme since we have very few samples. Alternatively, we use the online BMS model to detect anomalies. If we assume that the *attack* scenarios were normal activities while the *no attack* scenario was an anomaly, we may expect the comparison scores of the different *attack* scenarios to be clustered together. For each of the four scenarios in the dataset, parameters of their online BMS models are computed. We form a similarity matrix of size 4×4 in order to check whether the *attack* scenarios cluster separately. The $L1$ distance between the histograms of parameters of learnt behavior is used as the similarity score. Table 4.5 shows the distance between the different *attack* examples with the *no attack* case. We observe that the *attack* scenarios are more similar to each other compared to the *no attack* scenario.

Table 4.5: Comparing the *no attack* and *attack* scenarios in bank surveillance data. $L1$ distance between histograms of parameters of online BMS model is used as similarity score.

#	*No Attack*	*Attack 1*	*Attack 2*	*Attack 3*
No Attack	0	310	424	362
Attack 1	310	0	218	278
Attack 2	424	218	0	180
Attack 3	362	278	180	0

4.5.2.3 Comparison of results

Georis et al. [52] presented an ontology-based approach for video interpretation in which activities of interest are manually encoded. They demonstrated the effectiveness of ontologies for detecting attacks on a safe in a bank monitoring dataset. Their method requires a detailed description in the form of a set of rules to detect an 'attack' activity. The proposed method, however, is data-driven. The extent of deviation observed in a given video sequence compared to a normal scenario is used as a measure for detecting anomalies. Comparitive results are summarized below.

In [52], the authors report the following results on tracking persons in the bank scene: 88% true positives, 12% false negatives and 2% false positives. There were no errors in tracking in our method.

For anomaly detection (i.e., detecting that the bank safe was attacked), the results reported in [52] are 93.5% of true positives, 6.25% of false negatives and 0%

of false positives. These results correspond to 16 repitions of the attack scenario. We have access to only 3 attack scenarios. On these, we obtained correct anomaly detection in all three scenarios.

4.5.3 UCF Human action dataset

The UCF human action dataset (section 1.1) is used to illustrate the effectiveness of the Viterbi-based segmentation to detect primitive behaviors. We may think of many actions as a sequence of behaviors. For example, picking up an object may be abstracted as *extend the hand toward object - grab object - withdraw the hand*; erasing the black board, as *extend hand - move hand side to side on the board - withdraw hand*; opening the door, as *extend hand - grab knob - withdraw hand*. To generate an action, we may compose a sequence of systems that operate with the appropriate parameters.

We employ the Viterbi-based segmentation described in section 4.4.1 to find the segments of actions. We show some of the segmentation results in figure 4.5. A description of motion trajectories shown in figure 4.5 along with the detected segment boundaries is given below.

- (a) Open cabinet door: *start - reach door handle - open door - withdraw hand*.
- (b) Pick up object: *start - pick up object*.
- (c) Put down: *start - put object in cabinet - reach door handle - close door - withdraw hand* .

- (d) Open cabinet door: *start - reach door handle - open door - extra segment - withdraw hand.*
- (e) Pick up object: *start - pick up object.*
- (f) Put down: *start - put object in cabinet - reach door handle - close door - withdraw hand .*
- (g) Open cabinet door: *start - reach door handle - open door - withdraw hand.*
- (h), (i), (j) Pick up object: *start - pick up object.*
- (k), (l) Open cabinet door: *start - reach door handle - open door - withdraw hand.*
- (m) Pick up and put down elsewhere: *pick up - put down - withdraw hand*
- (n) Open cabinet door (different viewing direction): *start - reach door handle - open door - withdraw hand.*
- (o) Pick up and put down elsewhere: *pick up - put down (late detection) - withdraw hand.*

Table 4.6 shows the mean and variance of the number of segments detected for different activities. The same number of segments were detected consistely across multiple samples of activities *picking up* and *pouring water* an object. On the other hand, variance in the number of segments detected for *picking up an object and putting it down elsewhere* was high. This was because of excessive differences in appearance across samples.

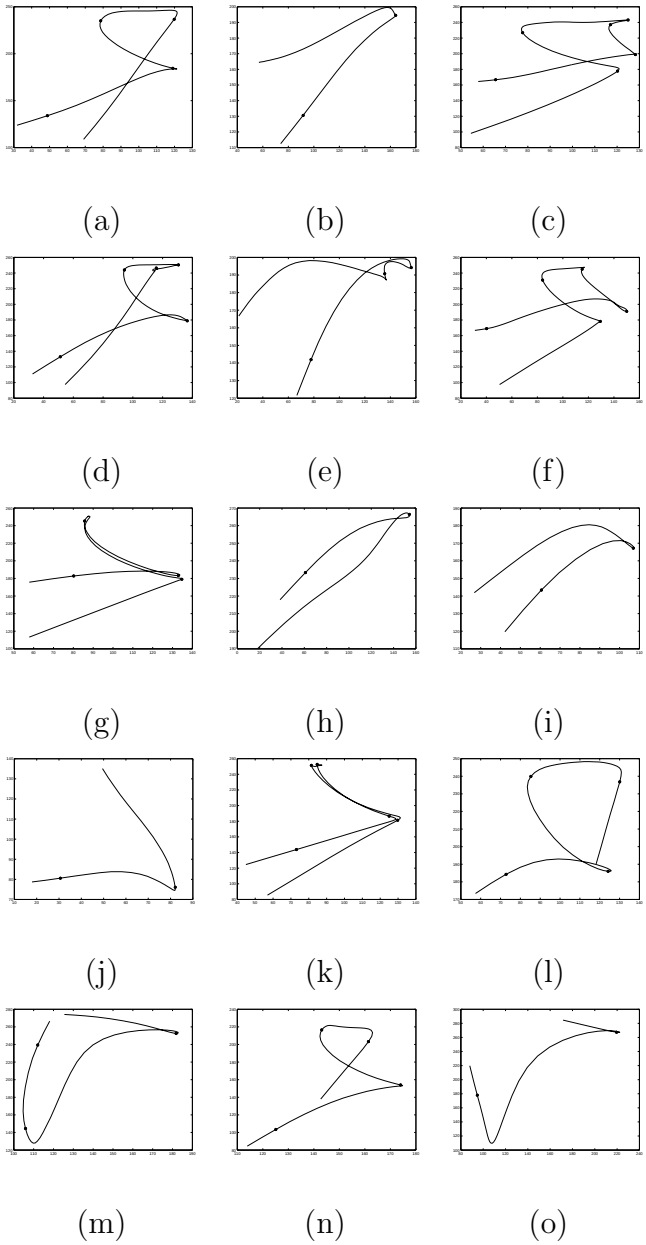


Figure 4.5: Motion trajectories of the hand in the UCF dataset for different actions. The dots along the trajectory denote segment boundaries detected.

4.5.3.1 Comparison of results

We compared the location of detected segment boundaries with the *dynamic instants* described in [1]. Dynamic instants are points of high curvature along the trajectory. They are chosen as the feature of interest to ensure view invariant representation of actions. The first segment switching in the proposed approach occurs after sufficient evidence about the dynamics has been accumulated. This is marked as the start segment, which usually occurs in 10-15 frames. We ignore this boundary point when comparing our results with dynamic instants in [1] since it does not have an explicit *start* instant. Segment boundaries detected using our method, which are marked by dots in figure 4.5 are compared with the dynamic instants in [1]. The average difference between the two values are as follows: 5.4 frames for open door, 4.1 frames for close door, 3.3 frames for pour water, 2.0 frames for erase board and 6.2 frames for pick up object and put down elsewhere.

4.5.4 Home-care applications

Though the proposed approach was demonstrated using video sequences collected in office and airport environments, it can be easily applied to home-care scenarios. De Natale et al. [85] describe fall detection and accessing unauthorized locations as typical applications in home-care scenarios. We highlight the applicability of our method to some home-care applications.

As demonstrated using the bank monitoring dataset, our approach is able to detect if a person gained access to unauthorized places. Similarly, as experiments

Table 4.6: Number of segments detected for activities in the UCF indoor human action dataset.

Activity	Average # segments	Variance
Open door	4.44	0.53
Pick up	2.27	0.21
Put down	2.50	0.94
Close door	4.25	0.25
Erase	6.50	0.33
Pour water	3.00	0.00
Pick up object & put down elsewhere	3.75	3.92

using the UCF dataset demonstrated, human actions were segmented into a sequence of elementary parts. For instance, opening a cabinet door was represented as *start - reach door handle - open door - withdraw hand*. These actions could be used to find the number of times is medicine cabinet is used. If an opening action does not occur at expected times, an alarm could be issued.

4.6 Summary

In this chapter, mixed state models using segments of piecewise linear mixed state models was presented. A sequence of such segments is said to characterize an activity based on a context-independent basis of behavior. Parameters of the segmentwise models and switching times between them were estimated using a Viterbi-based algorithm. Experiments using surveillance video streams in both indoor and outdoor settings demonstrate that the method can be used to analyze activities at different scales. The usefulness of the proposed method is shown using applications such as temporal segmentation and anomaly detection.

Chapter 5

Spectral Method for Event Detection

5.1 Introduction

The theory of antieigenvalues is based on changes in the data. It is sensitive to how much a data vector is turned from a known direction, rather than the direction of persistence [86]. On the other hand, eigenvectors represent the direction of maximum spread of the data and the eigenvalues are proportional to the amount of dilation. We propose an antieigenvalue-based approach for detecting key frames by investigating properties of operators that transform past states to observed future states.

This chapter is organized as follows. Section 5.2 motivates the key-frame based representation for activities. Section 5.3 gives a brief overview of antieigenvalue theory. Section 5.4 describes the proposed approach. Section 5.5 demonstrates the proposed method using two datasets: the MOCAP database and the UCF human action database.

5.2 Key frame representation

As we argued through examples of opening a door, walking, etc., many activities can be represented using key frames instead of the entire video sequence. Generally, there are three ways to decide on what constitutes a key frame. We may use domain knowledge in a top-down fashion. It requires an extensive model for

the activity, which may be tedious. It relies on our ability to detect the key frames across variations in the data that occur due to structural changes and noise [30]. We may hypothesize that the important characteristics of the activity are present in the persistent and dominant frames [37]. This makes it difficult to detect subtle changes, since it may be difficult to distinguish them from noise. We may look for key frames that are a result of certain changes in the data. In other words, changes in the activity may be more useful than the absolute values of a dominant feature in representing the activity. We present an unsupervised approach for detecting key frames based on changes in the data.

Let the past state vector $\mathbf{x}_-$ be transformed by an operator A_t to a future state vector $\mathbf{x}_+$. If motion properties do not change appreciably, then $\mathbf{x}_+$ may be related to $\mathbf{x}_-$ by an identity transformation modulo translation. Such a transformation may be less interesting compared to the case where A_t turns the state $\mathbf{x}_-$. We show how antieigenvalues may be used to detect such changes and to identify the key frames. In contrast, eigenvalues are tuned to detecting identity-like transformations. It is important to point out that these quantities are of intrinsic interest in their own right. As the term denotes, however, it may be easier to gain an insight into antieigenvalues by contrasting with eigenvalues and eigenvectors of the operator.

The motion trajectories are associated with two quantities: the antieigenvalue sequence, which is the sequence of antieigenvalues for the operator A_t for every time t, and the location of the key frames detected using minima in the average antieigenvalue sequence. Both the extent of change as given by the antieigenvalues and the location of key frame are useful for recognition. If viewing conditions change,

we may expect the time instants of occurrence of key frames to be more useful since the extent of change depends on viewing direction. On the other hand, if the viewing direction is fixed, antieigenvalues may be used in comparing two activities. We illustrate both these cases in our experiments.

5.3 Antieigenvalues

We present a brief description of antieigenvalues before discussing its application. A detailed discussion of antieigenvalues may be found in [86] or [87].

For a square matrix A, a non-zero vector $\mathbf{x}$ is said to be an eigenvector if $A\mathbf{x} = \lambda\mathbf{x}$, and λ is called the eigenvalue. Equivalently, we may state the condition as $\cos\theta = 1$, where θ is the angle between $\mathbf{x}$ and $A\mathbf{x}$. Geometrically, we may think of eigenvectors as those that dilate A but do not turn at all. The eigenvalues represent the amount of dilation. On the other hand, antieigenvectors are critical to the turning of A. Instead of seeking $\cos\theta = 1$ or $\theta = 0$, antieigenvectors minimize $\cos\theta$, or equivalently, maximize θ. The first antieigenvalue is defined as [86]

$$\mu_1(A) \;=\; \inf_{A\mathbf{x_n}\neq 0} \frac{\Re\langle A\mathbf{x_n}, \mathbf{x_n}\rangle}{\|A\mathbf{x_n}\|\|\mathbf{x_n}\|}. \tag{5.1}$$

It has been shown [87] that antieigenvectors occur in pairs. The first antieigenvector is the pair

$$\mathbf{x} \;=\; \pm\left(\frac{\lambda_n}{\lambda_1+\lambda_n}\right)^{1/2} e_1 + \left(\frac{\lambda_1}{\lambda_1+\lambda_n}\right)^{1/2} e_n, \tag{5.2}$$

where λ_1 is the smallest eigenvalue with eigenvector e_1 and λ_n is the largest eigen-

value with eigenvector e_2. For example, let

$$A = \begin{pmatrix} 9 & 0 \\ 0 & 16 \end{pmatrix} \tag{5.3}$$

The eigenvalues of A are $\lambda = 9, 16$. Using (5.2), the first antieigenvector is $\mathbf{x_1} = (\frac{-4}{5}, \frac{3}{5})$. The antieigenvalue may be calculated by substituting the value of x_1 in (5.1). The first antieigenvalue is $\mu_1(S) = \frac{\langle A\mathbf{x_1}, \mathbf{x_1}\rangle}{\|A\mathbf{x_1}\|} = 0.96$ The second antieigenvector is $\mathbf{x_2} = (\frac{3}{5}, \frac{4}{5})$ and the corresponding antieigenvalue is 0.97.

The first total antieigenvalue is defined as $|\mu_1(A)| = \inf_{A\mathbf{x}\neq 0} \frac{|\langle A\mathbf{x}, \mathbf{x}\rangle|}{\|A\mathbf{x}\|\|\mathbf{x}\|}$. The higher total antieigenvalues are similarly defined.

The total antieigenvalues for matrices of size greater than 2×2 may be calculated as follows (theorems 2.1 and 2.2 in [86]). Let A be a normal operator with eigenvalues $\lambda_i = \beta_i + j\delta_i$, $i = 1, \ldots, n$ and $j = \sqrt{-1}$. Then the first total antieigenvalue is either 1 or the smallest number in the set of values

$$G = \{\frac{\sqrt{(\beta_i|\lambda_j| + \beta_j|\lambda_i|)^2 + (\delta_i|\lambda_j| + \delta_j|\lambda_i|)^2}}{(|\lambda_i| + |\lambda_j|)\sqrt{|\lambda_i||\lambda_j|}}, \tag{5.4}$$

where $i \neq j, 1 \leq i \leq n, 1 \leq j \leq n\}$. If $|\mu_1(A)| = 1$, then the first total antieigenvector is $\mathbf{z_1} = (z_1, z_2, \ldots, z_n)$ with $|z_j| = 1$ for some j and all other $z_i = 0$. If $|\mu_1(A)|$ is one of the values in G, then the components of $\mathbf{z_1}$ satisfy $|z_i|^2 = \frac{|\lambda_j|}{|\lambda_i|+|\lambda_j|}, |z_j|^2 = \frac{|\lambda_i|}{|\lambda_i|+|\lambda_j|}$, all all other $z_k = 0$. Further, all higher total antieigenvectors take their value from the set G and the corresponding higher total antieigenvectors possess the same component structure as the first total antieigenvector.

5.4 Key frame detection using antieigenvalues

In this section, we describe the proposed antieigenvalue-based key frame detection procedure. The key frames are used to compare two activities.

5.4.1 Feature selection

We obtain trajectories of the moving object and compute its apparent velocities. The tracking procedure for the different datasets is outlined in section 5.5. The state of a moving object is said to be the tuple $(x(t), y(t), \dot{x}(t), \dot{y}(t))$, where $(x(t), y(t))$ represents the instantaneous position. We assume that the state undergoes certain important changes at the key frames. We are interested in detecting these changes, rather than modeling the entire sequence of frames. Let $A_t : H \rightarrow H$ be an operator that relates the past state $\mathbf{x}(t_-)$ into the future state $\mathbf{x}(t_+)$, where H is the Hilbert space domain. There are two estimation tasks here. We need to estimate the past and future states $\mathbf{x}(t_-)$ and $\mathbf{x}(t_+)$. For robust estimation, we assume that the state of the system remains constant for a short interval of time. The other estimation tasks involves optimizing the parameters of the operator A_t. If there is no change in the state from t_- to t_+, we may expect A to be the identity matrix (modulo translation). The matrix A_t is estimated using a least squares procedure as described in section 3.3.1.

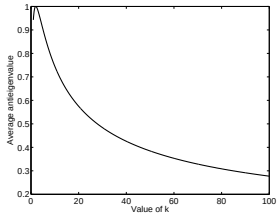


Figure 5.1: Average antieigenvalue for A in (5.5) as a function of k.

5.4.2 Numerical range of the operator

The numerical range of an operator A is defined as the set $W(A) = \{\langle A\mathbf{x}, \mathbf{x}\rangle, \mathbf{x} \in H, \|\mathbf{x}\| = 1\}$, where H is the Hilbert space. For example, consider an operator defined by the matrix $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$. Let $\mathbf{x} = (p, q)$. For simplicity, assume $\|\mathbf{x}\| = |p|^2 + |q|^2 = 1$. Then $A\mathbf{x} = (q, 0)$ and $\langle A\mathbf{x}, \mathbf{x}\rangle = qp$. A simple calculation shows that $W(A) = \{\mathbf{x} = (p, q) : |p|^2 + |q|^2 \leq \frac{1}{2}\}$ or the half disk. Closely related to the numerical range, we can define the angle of the operator $\cos A$ and the antieigenvalues of the operator A as discussed in section 5.3.

5.4.3 Choosing key frames

We compute antieigenvalues of A_t using (5.4) and use the mean antieigenvalue as a measure of relative significance of the frame in representing the activity. A small value of the mean antieigenvalue indicates that the minimum $\cos A_t$ is small or that the turning angle is large. This indicates a larger relative change in the state vector and hence significant for representing the activity. We illustrate the use of antieigenvalues in detecting key frames through a few examples in the 1-D case.

The state of the moving object is the pair $(x(t), \dot{x}(t))$. Suppose the transformation operator is given by

$$A = \begin{pmatrix} 2 & 0 \\ 0 & k \end{pmatrix}. \tag{5.5}$$

For differing values of k, this means that the change in the state of the object is due to a changing speed, while the position remains constant (modulo translation). Figure 5.1 shows the variation of the average antieigenvalue as the value of k is increased. We observe that, as expected, the average antieigenvalue varies inversely as the extent of change in the state of the moving object.

5.4.4 Matching sequences

We compute the similarity score between two video sequences by comparing the sequences of key frames. Clearly, an activity need not be repeated with the same timing scale from one instantiation to the next and the location of key frames may change slightly. To allow for non-linear time normalization while matching, we use dynamic time warping (DTW) [55]. The similarity score is computed by traversing the warping path, which gives the correspondence of the frames in the reference and probe sequences.

To place the proposed approach in context, we compare this to the eigenvalue based methods. Various approaches in the literature have used eigenvalue-based ideas to model activities in two main ways: for pre-processing or filtering the data and for extracting the dominant characteristics for representation. The basic hypothesis in all these approaches is that the dominant characteristics of the signal

are important. Also, the main characteristics are assumed to be highly structured and stationary. In such a setting, the eigenvectors capture the dominant characteristics and the eigenvalues represent the relative contribution of the eigenvectors for representation. For example, eigenfaces capture the dominant characteristics for face recognition [88]. By reconstructing the signal using the top few eigenvectors, it induces a smoothing operation on the original signal [29]. Zhong et al. [37] use this idea for activity classification where they cluster the sequence of frames into prototype classes.

5.4.5 Algorithm overview

- Pre-processing: Extract object trajectory from video and smooth it.
- For every time t, compute the state $\mathbf{x}(t) = (x(t), y(t), \dot{x}(t), \dot{y}(t))$. For computing $\dot{x}(t), \dot{y}(t))$, we use finite differencing over W frames of data.
- Compute the least squares estimate of the operator $A_t : \mathbf{x}(t_-) \rightarrow \mathbf{x}(t_+)$.
- Compute the antieigenvalues of the operator A_t using (5.4). Compute its mean.
- Recognition: compare the key frames detected from the average antieigenvalue sequence for the training using DTW.

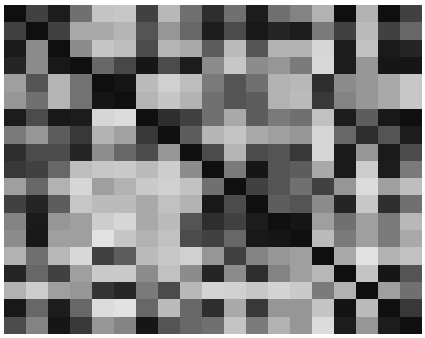


Figure 5.2: Confusion matrix for activities in the MOCAP dataset

5.5 Experiments

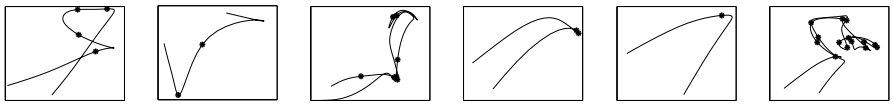
We demonstrate our approach to activity recognition using the MOCAP action dataset and the UCF human action dataset.

5.5.1 Motion Capture (MOCAP) dataset

A brief description of the MOCAP dataset is presented in section 1.3.4. There are 9 activities in the dataset and approximately 75 sets of observation overall. The tracks for an activity such as walking consists of multiple cycles of the activity. We divide the sequence into individual walking cycles and treat each half-cycle as an observation. Half-cycle refers to the part of the walking cycle starting from the standing pose, right (or left) leg forward, reaching the swing pose, and withdrawing the right (or left) leg to the standing pose. The number of observations is increased to 365 by treating similar trajectories of nearby locations as multiple samples, i.e., 2 locations near the abdomen are treated as multiple samples of the same location. To ensure that there is no bias due to the displacement, we use mean-subtracted

Table 5.1: MOCAP dataset: Closest-matching activities based on comparing event probability sequences. All activities were correctly recognized. Table shows the matches following the top match.

Test activity	Match #2	Match #3
Blind-walk	Normal walk	Normal walk
Prowl-walk	Jog	Exaggerated walk
Broom	Sit	Exaggerated walk
Crawl	Broom	Sit
Exaggerated walk	Sad walk	Normal walk
Jog	Jog2	Normal walk
Sit	Sit1	Neutral
Normal walk	Normal walk	Sad Walk
Sad walk	Exaggerated walk	Normal walk



(a) (b) (c) (d) (e) (f)

Figure 5.3: Sample trajectories from the UCF dataset along with the detected key frames

trajectories for all locations.

We compute the state vector for every time instant and estimate the transformation operator A_t as described in section 5.4. We compute the antieigenvalue and use its mean as a signature for the activity. The antieigenvalue sequences are matched using DTW. All the activities were correctly recognized. Table 5.1 summarizes the activities that were the closest matches following the top match. We observed that the different types of walking resembled each other while the similarity scores corresponding to *sitting*, *sweeping with a broom* were significantly larger. Figure 5.2 shows the confusion matrix across all activities. It may not be straightforward to associate a physical meaning to the detected key frames for activities such as walking, etc. other than saying a key frame was detected at the stance when the feet are maximally apart, and so on. In the UCF action dataset described below, the key frames are more readily apparent.

5.5.2 UCF human actions dataset

A brief description of the UCF dataset is provided in section 1.1.

The average antieigenvalue sequence was computed as outlined in section 5.4.5.

The key frames were identified by finding the minima in the average antieigenvalues. Figure 5.3 shows the key frames identified for some of the activity trajectories. The dots marked along the trajectory denote the key frames detected along the trajectory. Figure 5.3(a) shows the key frames for opening a door. In figure 5.3(b), the trajectory for picking up an object from the desk and putting it on the floor shows two key frames detected, one of which is the result of a sharp change in direction and the other a gradual change. The second sharp change is not detected due to boundary effects. In the case of erasing a white board, we observe a key frame when the eraser is picked up, and several key frames at the left side of the erasing back-and-forth action of the hand (figure 5.3(c). This means that each back and forth action of the hand may be considered as the past and future states separated by the key frames. Figures 5.3(d) and (e) show trajectories of picking up objects. They each have one key frame detected at approximately the instant the object is picked up. Figure 5.3(f) shows the trajectory of a random action. The lack of structure in the data is reflected by a large number of changes leading to the detection of several key frames.

Comparison with the UCF method[1]: Rao et al. treat activities as a sequence of *dynamic instants* that are defined as the points of maximum curvature along the trajectory [1]. The key frames in the proposed approach are detected based on changes in the data including changes in direction and changes in speed. The comparison of recognition rates are given in figure 5.4.

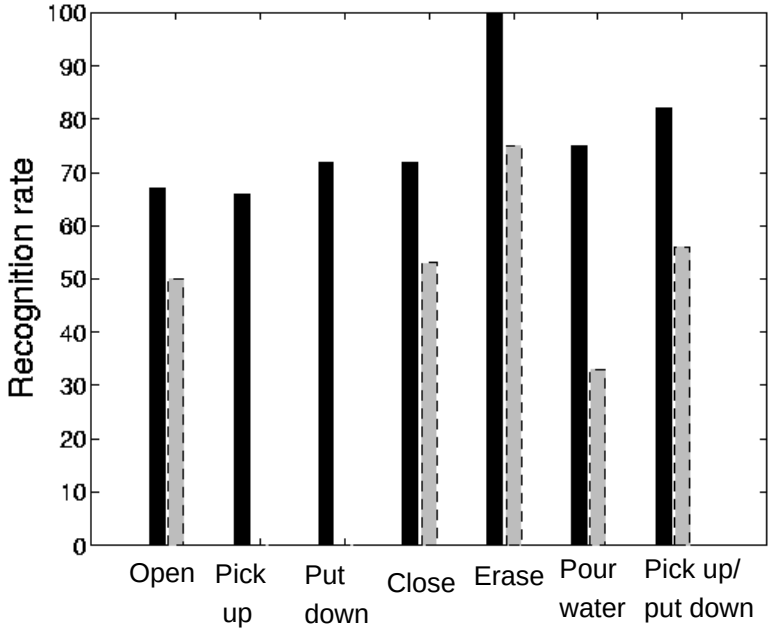


Figure 5.4: UCF dataset: Comparing recognition rates. Solid black bar represents proposed method, dashed gray bar are the rates reported in [1]

5.6 Summary

We have presented a key frame based activity representation using the largely unexplored theory of antieigenvalues. Key frames are characterized using changes in the data, rather than dominant, persistent properties. This allows a natural way to detect both subtle and sudden changes, which are often more interesting than the portions of the data that are normally observed.

Chapter 6

Spatio-temporal Volumes

6.1 Introduction

The motivation for representing human activities using spatiotemporal volumes stems from a twofold objective of selecting useful features for recognition and providing an intuitive way of visualization. The aim is to characterize the topology of these activity volumes in an algebraic framework so that features that capture the topology can be extracted. As an example of typical activities of interest, consider a surveillance scenario where two people approach and meet each other. This creates two cylinder-like structures in the spatiotemporal volume that merge into one. The persistence of the two persons at the place of meeting creates a maximum. In general, the surface of spatiotemporal volumes consist of maxima, minima and saddle points (collectively, known as critical points), and homotopic sections between them. Since spatiotemporal volumes are created by moving objects, it is sensible to express the topology of such volumes using motion properties.

In the mathematical abstract, topology is defined as "the study of open sets". Intuitively, this refers to the shape or structure that is invariant to deformations that are one-to-one and continuous. These include all deformations that preserve local structures such as holes and attached segments. Polygonal meshes and implicit surfaces are some of the ways of representing topological shapes. Topological invari-

ants can be computed using combinatorial structure of faces, edges and vertices. For example, Euler characteristic χ was defined in this way.

$$\chi = \#\text{ vertices - }\#\text{ edges + }\#\text{ faces} = 2 - 2g,$$

where g is the genus (or the number of 'handles').

Another way of representing surfaces is due to Morse theory. It uses critical points of a smooth, real-valued function defined over the manifold. It reveals the connection between differential geometry of a surface, its algebraic topology and stability of dynamical systems evolving on the manifold. The topology changes only at the critical points, i.e., connected sections can merge or split or local extrema occurs. Morse theory has been used in many areas including graphics, image compression and geographic information systems ([89], [90], [91]). In many of these applications, the focus of research is on accurate rendering of images or shapes. For activity modeling, however, capturing the global structure or feature selection and extraction is more important.

If all critical points of a smooth, real-valued function defined on the manifold are non-degenerate, then it is said to be a Morse function. We use the dynamics of moving objects to construct Morse functions for spatiotemporal volumes. Assuming that the dynamics is linear within intervals, a system matrix that captures the dynamics is estimated. A suitable distance metric on the space of such matrices is defined and the distance is used to construct a Morse function. Critical points of the Morse function are detected. The Hessian at the critical points captures the topology of the spatiotemporal volumes, and are used to compare activities formed

by moving objects. We demonstrate an application to activity classification in a surveillance scenario using the TSA airport surveillance dataset.

Critical points and critical values may be viewed as activity descriptors that are based on motion when a suitable real-valued function is used in their extraction. Local descriptors such as wavelet coefficients and SIFT features have been studied extensively in the context of image matching and object recognition ([92], [93]). One of the objectives of these methods is to find stable keypoints in images, which possess desirable properties such as scale and affine invariance. To see the connection between Morse functions and local descriptors of images, consider an image $I(x, y)$, where (x, y) specifies the pixel location, and I is the intensity. Instead of representing an image as $I(x, y)$, it can be expressed as the tuple $(x, y, I(x, y))$, i.e., in a 3-D cube, x and y axes specify a co-ordinate system for pixels, and the intensity $I(x, y)$ is plotted along the z-axis. We show that the SIFT algorithm is equivalent to characterizing the topology of $(x, y, I(x, y))$ with a particular choice of Morse function $f(x, y, I(x, y)) = I(x, y)$. The keypoints of I in the SIFT algorithm after the post-processing steps form critical points of the Morse function and the keypoint descriptors approximate the Hessian at its critical values.

The rest of the chapter is organized as follows. Section 6.2 presents a brief overview of some results in Morse theory. Section 6.3 describes the connection between critical points of Morse functions and SIFT features. Section 6.4 describes the construction of Morse functions based on dynamics for activity modeling. Section 6.5 discusses some practical issues and describes an application of the proposed approach to activity classification. Section 6.6 shows experimental results using the

TSA airport surveillance dataset.

6.1.1 Related work

Niyogi and Adelson [94] used xyt slices for modeling gait patterns in one of the earlier works based on spatiotemporal volumes. Cootes et al. [95] developed active shape models, where training samples are used to fit deformable shapes. More recently, level set methods have been used instead of active surfaces, where a suitable energy functional is minimized. A framework for tracking and spatiotemporal segmentation that can handle occlusions is described in [96].Spatiotemporal volumes have been used for detecting and tracking humans in surveillance scenarios ([97]).

Image-level local descriptors have been used for matching ([98], [99]). Filter-based approaches have been developed for detecting feature points and edges ([100], [101]). More recently, SIFT features have been successfully used for applications such as image matching and object recognition ([93]). Motion descriptors were extracted from a motion stabilized spatiotemporal volume to recognize actions in far field video [102].

Solutions to the Poisson equation have been used to find closed contours of shapes in [103]. This was extended to the detection of closed boundaries of space-time volumes in [104], and the detected shapes were used for action classification. They use multigrid methods for efficient computation. This is similar to the approach adopted in [90], which uses the multigrid technique to solve for Laplacians subject to Dirichlet boundary conditions. It requires initialization (or specifying

constraints at known points). It is perhaps worth mentioning that we are not concerned with reconstructing spatiotemporal volumes. The proposed approach, on the other hand, approximates the objects in the scene using motion blobs that are detected using a separate pre-processing step. The dynamics of the blobs is used to find features from spatiotemporal volumes for activity classification.

Morse theory has its foundations in smooth functions on manifolds [105]. It has been extended to piecewise linear functions on meshes [106]. A graphical structure is used to represent the topology of surfaces, which encodes parent-child relationships in contours. Perturbations are used to simplify the structure in terrain data, and in general 2-manifolds, in [89].

6.2 Morse theory

This section briefly reviews Morse theory. Milnor [105] gives an accessible and insightful description. Let $f : M \rightarrow \mathbb{R}$ be a smooth real-valued function defined on a manifold M. A point $p \in M$ is said to be a critical point if $\partial f(p)/\partial x_i = 0$, $i = 1, 2, \ldots, n$. The real number $f(p)$ is said to be a critical value. A critical point p is said to be non-degenerate if and only if its Hessian

$$Hess_p(f) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}(p)\right]$$

is non-singular; otherwise it is degenerate.

A function f is said to be Morse if and only if all its critical points are non-degenarate. Degenerate critical points are unstable and can be perturbed to render

the function Morse, since the set of Morse functions on any M is dense. Alternatively, any bounded smooth function $f : M \to \mathbb{R}$ can be uniformly approximated by a smooth function g, which has no degenerate critical points.

We are interested in a 2-manifold M closed and embedded in $\mathbb{R}^3$. According to the Morse lemma, a Morse function can be expressed as $f(x) = f(p) \pm x_1^2 \pm x_2^2$ in the neighborhood of every non-degenerate critical point p. The number of minus signs is the index of p. It is the number of negative eigenvalues of $Hess_p(f)$, counting multiplicities, and characterizes the type of critical point, i.e., minimum has index 0, saddle has index 1 and maximum has index 2.

Let M^a denote a section of the manifold such that $M^a = f^{-1}(-\infty, a] = \{p \in M : f(p) \leq a\}$. For real numbers $a < b$, M^a is diffeomorphic to M^b if the set $f^{-1}[a, b]$ is compact and contains no critical points of f. Further, M^a is a deformation retract of M^b and the set $f{-}1[a, b]$ is said to be of the same homotopy type. This implies that there are no topological changes on a homotopic section of a manifold that contains no critical points. It gives a convenient way to partition the overall structure into homotopic segments.

We recount the topology of the torus to illustrate the use of critical points (figure 6.1). Consider a doughnut being held vertically before dunking it in coffee. The height h of a point on the surface from the base forms a Morse function that has four critical points: a minimum at the base, a maximum at the top and two saddles along a vertical line on the inner circle. The topology of the dunked portion of the doughnut changes at the four instants when the critical points are submerged in the coffee. At the saddles, there is one principal direction along which the Morse

function (height) increases, and one direction where it decreases. At the minima, clearly there are no directions where the function decreases whereas at the maximum, there are two principal directions along which the function decreases. The behavior about a critical point is captured by the Hessian of the Morse function. To complete the characterization of the topology, handlebodies are used to connect neighboring critical points.

In practical applications, spatiotemporal volumes do not lend themselves to Morse functions that are smooth. The function, rather than the surface of the spatiotemporal volume, is smoothed. It may be worth noting that there is no unique Morse function for a given manifold. However, if a function is Morse, then the critical points and the Hessian at the critical points completely specify the topology. In the case of activities, the underlying dynamics of the moving blobs provides a natural way to define a Morse function whereas in 3-D object modeling, the height function can be used.

6.3 Morse function for SIFT features

Lowe [93] described a method for extracting keypoints called SIFT features. It relies on identifying distinctive, stable features or keypoint descriptors that are invariant to scale, image noise and rotation. Since critical points, together with associated critical values and handlebodies are sufficient to characterize topology of objects such as spatiotemporal volumes, it is reasonable to look for connections between the two approaches. The SIFT features are extracted by convolving the

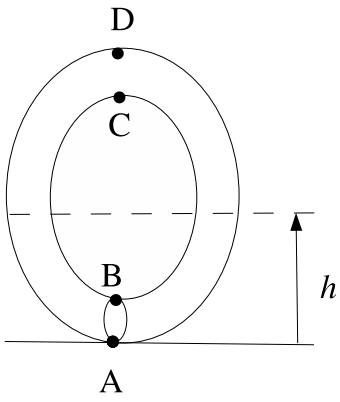


Figure 6.1: Topology of torus is characterized by the four critical points A, B, C, D of the height function h; A is a minimum, B, C are saddle points and D is a maximum. Since all critical points are non-degenerate, h is a valid Morse function.

image with a difference of Gaussian (DOG) filter and identifying points of local extrema. The computation is done across scales so that *stable* extremum points are obtained. Since the DOG operator picks up edges as well as isolated maxima and minima, post-processing steps such as edge-elimination and low contrast removal are needed. Every keypoint is assigned an orientation based on a gradient histogram in the neighborhood.

Consider the image I as a 3D object $(x, y, I(x, y))$, where $I(x, y)$ is the intensity at pixel location (x, y). Analogous to the height function $h = z$ in the case of the torus (figure 6.1), consider a candidate Morse function $f(x, y, (I(x, y))) = I(x, y)$. If the image has edges, clearly the critical points along edges will be degenerate (with the rank of Hessian being 1). Post-processing methods are used for edge-removal and elimination of low-contrast keypoints based on thresholding. This is equivalent to removing degenerate critical points in the candidate Morse function so

that the refined function becomes Morse. The orientation assignment to keypoints finds directions of maximum (or minimum) gradient, i.e., the eigenvector directions of the Hessian operator within the local neighborhood of the point. Thus, choosing $f(x, y, (I(x, y))) = I(x, y)$ as a Morse function, gives a Morse theoretic interpretation to the SIFT features.

6.4 Morse Function for Activities

This section describes the computation of Morse functions using dynamics of moving objects that are represented by moving blobs. It may be useful to illustrate the intuition behind the use of dynamics for characterizing spatiotemporal volume through an example. In an airport surveillance scenario, a common activity is that of a luggage cart approaching a plane, stopping for a few minutes when the luggage is loaded before turning and exiting the scene. This creates a spatiotemporal volume that resembles a bent pipe, where the location of the 'bend' corresponds to the stopping of the luggage cart. The fine details of the surface of the volume depends on the shape of the luggage cart. A detailed shape information may be required to distinguishing between two types of cart, or more generally different types of 3D objects. But the low resolution and noise in video streams may not allow for exact reconstruction. More importantly, we are not interested in these details. It is sufficient to treat the luggage cart as a motion blob for the purpose of activity classification. Similarly, a plane taxiing to the gate would be represented by the volume carved out by its motion blob. We require an algebraic way of representing

spatiotemporal volumes in order to compare them.

In the luggage cart example, there are three parts in the activity as described: enter, stop at plane, turn and exit. When describing the activity, we associate a meaning to these parts due to important changes in dynamics of the object at these time instants. So it is desirable that these parts are reflected in the representation of spatiotemporal volumes. In other words, the Morse function should attain criticality to mark these events.

6.4.1 Motion blobs

Postion and velocity of the centroid of the moving object is said to be its state $z(t)$. We assume that the parameters of dynamics are constant for W frames. The value of W depends on the type of data. For instance, it may be reasonable to assume $W = 100$ or 4 second in far field surveillance data. Within a sliding window of size W, state z_t is estimated. Let the state evolution be modeled using a linear transformation, i.e.,

$$z_{t+1} = Az_t + v_t, \tag{6.1}$$

where v is the error in approximation. The parameters of the matrix A are estimated using the least squares procedure (section 3.3.1).

6.4.2 Candidate Morse function

Let $f(x, y, t)$ denote the value of the real-valued Morse function at pixel (x, y). Let $I(x, y, t)$ represent the image frame $I(x, y)$ at time t. Our goal is to estimate f

using dynamics of short time trajectories of motion blobs such that critical points and the Hessian at those points describe the spatiotemporal volumes.Given a video stream, moving objects are detected and the region $M(x, y, t)$ containing motion is said to be the region where the candidate Morse function is valid:

$$M(x, y, t) \quad = \quad \mathbf{I}_{I(x,y,t) \neq \{I(x,y,t-\Delta)\}}, \tag{6.2}$$

where $\{I(x, y, t-\Delta)\} = \{I(x, y, t-\Delta), I(x, y, t-\Delta+1, \ldots, I(x, y, t-1))\}$ for some constant Δ consecutive frames and $\mathbf{I}$ is the indicator function.

Every point (x, y, t) is associated with a moving blob. A candidate Morse function $f^0(x, y, t)$ is initialized based on the least squares estimates of dynamics (section 6.4.1) as follows. A distance function between dynamics of the moving blobs is computed within the region of interest $M(x, y, t)$. All the pixels in a blob are assumed to follow the same dynamics, i.e., $z_{t+1} = Az_t + v_t$, where $z_t \in V \subset \mathbb{R}^3$ and V is a spatiotemporal volume cube $\{V(x, y, t)\}$, $x \in [x_1, x_2]$, $y \in [y_1, y_2]$, $t \in [t_1, t_1+W]$ and (x_1, x_2, y_1, y_2) gives a rectangular blob.

Using a segment of frames $\{z_{t-W}, z_{t-W+1}, \ldots, z_{t-1}\}$, parameters of A_1, such that $z_t = A_1 z_{t-1} + v_t$ are estimated. Similarly for the segment $\{z_t, z_{t+1}, \ldots, z_{t+W-1}\}$, A_2 is estimated such that $z_t = A_2 z_{t-1} + v_t$ for the new segment. The distance between A_1 to A_2 forms the value of candidate Morse function at the t^{th} frame that separates the two segments.

6.4.3 Motion-based critical point detection

The candidate Morse function is computed by comparing the distance between a reference video sequence using the metric described in chapter 3. Using the candidate Morse function, maxima are computed using appropriate filters defined over a local cubical region in space and time. A maximum in $f(x, y, t)$ indicates that the motion at that point underwent a perceptible change. On the other hand, a minimum denotes that the point is strongly correlated with its past as expected. This makes it difficult to isolate minimum points for the candidate Morse function. At the first step, we only look for maxima. After the candidate Morse function has been refined, minima can be detected.

Only those critical points that lie within the region $M(x, y, t)$ defined in (6.2) are considered relevant for modeling the activity. The candidate Morse function is refined to obtain a smooth Morse function that retains the critical points detected within M.

6.4.4 Refining Morse function

The candidate Morse function f is smoothed using a Gaussian kernel and the critical points are recomputed based on the smoothed f. If the critical points in the region of interest M are non-degenerate, then the smoothed function is a Morse function. On regions of the spatiotemporal surface that are flat, the critical points are degenerate, i.e., the Hessian at the point is rank-deficient. Degenerate critical points, however, cannot be used to characterize the surface topology. To overcome

this problem, we collapse flat regions into a point.

The critical points are recomputed and checked for degeneracy. Small, smooth perturbations are added to the Morse function. The disadvantage is that this creates additional critical points. However, since Morse functions exist on any smooth manifold N and, moreover, form an open dense subset of smooth functions on N, it is reasonable to assume that the number of artificially created critical points is small.

6.5 Approach

In this section, some practical issues in implementation and application to activity classification are described.

6.5.1 Pre-processing

A stationary camera is used to capture far-field surveillance scene. The motion trajectories are extracted as described in section 3.2. The 4-tuple representing the 2-D position and velocity is said to be the state of the object. Assuming that the motion is piecewise linear, parameters of the linear dynamical model for the 2-D position and velocities are estimated. For discrete time instants, state evolution can be written as given in (6.1).

6.5.2 Critical points

Distance between matrices A's as the object moves is used to construct a candidate Morse function and it is refined as described in sections 6.4.2 and 6.4.4. The calculus for finding critical points on a manifold resembles the familiar first order and second order conditions involving several variables in differential calculus except for the restriction that global coordinates cannot be used. Instead, the standard way of doing differential calculus on such a topological surface is by covering it with overlapping patches. The patches are homeomorphic to open 2-D rectangles such that the transitions between them define differentiable function in $\mathbb{R}^2$. This enables us to restrict the function to be smooth as a function on rectangles. Critical points of the function are those where the gradient takes the value zero. Critical values and the Hessian at the critical points are computed as activity descriptors.

6.5.3 Activity classification

The topology of spatiotemporal volume provides a representation of the inherent characteristics of the different activity components, e.g., it is invariant to translation and rotation on the ground plane. The topology is specified by the critical points of the Morse function and the Hessian at the critical points capture the topology (along with homotopic sections that connect consecutive critical points). So critical points and the eigenvalues of the Hessian at these points can be regarded as descriptors that capture points of significance in activities. The overall similarity score between two video sequences is calculated using the average of distances at each

(a) (b)

Figure 6.2: TSA dataset: Motion trajectories and critical points for a sequence with ground crew movement.

critical point. Suppose a video sequence V_1 has n critical points $p_1, p_2, \ldots, p_n \in M$ and a second sequence V_2 has m critical points $q_1, q_2, \ldots, q_m \in M$. The distance $D(V_1, V_2)$ is:

$$D(V_1, V_2) = \frac{1}{n}\sum_{i=1}^{n} d(p_i, q_K) + \frac{1}{m}\sum_{j=1}^{m} d(q_j, p_L),$$

where $d(p_i, q_K)$ represents the Euclidean distance between the diagonal values of the Hessian at points p_i, q_K and

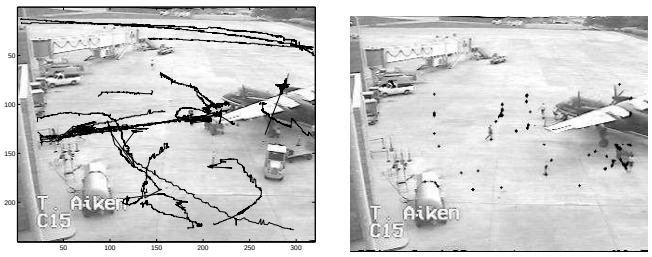
$$q_K = \arg\min_{k=1}^{m} d(p_i, q_k).$$

Similarly,

$$p_L = \arg\min_{l=1}^{n} d(q_j, p_l).$$

6.6 Experiments

The TSA dataset (section 1.3.2) is divided into segments of 1000 frames each. Activities are labeled as follows: plane arrival, vehicle (luggage cart, jeep) movement, passenger embarking and disembarking, ground crew movement. Motion trajecto-

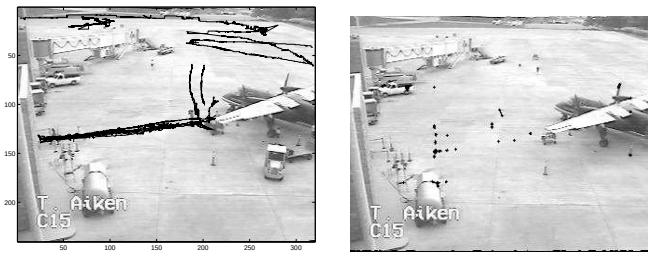


(a) (b)

Figure 6.3: TSA dataset: Motion trajectories and critical points for a sequence showing passengers disembarking and walking to the gate.

ries are extracted and system matrices that approximate the dynamics within 100 frames of a sliding window are computed. As the object moves, the distance between the matrices for the different time windows is computed (section 6.4.2). This forms the Morse function value for the region within the blob whose centroid lies on the trajectory. Critical points are computed, and the Hessian at those points is evaluated. This is used to compare two video sequences.

Figures 6.2-6.5 show the activities in the scene with the motion trajectories and the critical points detected. Each image represents a segment in the video sequence and is labeled by the dominant activity in the segment. A segment can have multiple labels if there is more than one activity occurring simultaneously. In figure 6.2, a ground crew person walks to the truck and drives away. Also, a vehicle crosses the tarmac at the top of the scene. In figure 6.3, the dominant activity consists of people disemarking, luggage carts moving and ground crew moving. The critical points detected are shown in figure 6.3(b). In figure 6.4 passengers disembark and walk to the gate. Also, a luggage cart goes to the plane while a truck crosses



(a) (b)

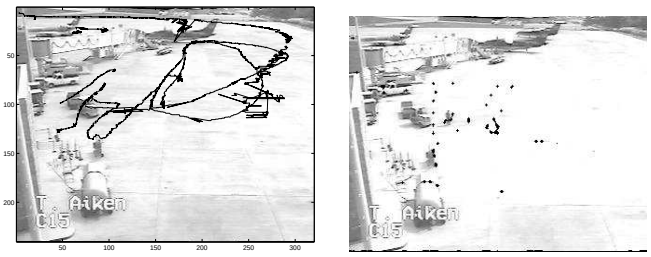
Figure 6.4: TSA dataset: Motion trajectories and critical points for a sequence showing passengers embarking and ground vehicles moving.

the scene. Figure 6.5(a) shows a plane entering, after which a luggage cart goes to the plane.

The classification rate for the activities is as follows: plane arrival (100%), vehicle movement (82%), passengers embarking (69%), passengers disembarking (79%), ground crew movement (82%). Ground crew movement was sometimes wrongly classified as passengers embarking or disembarking, when they followed similar trajectories prior to the plane's arrival and departure. The motion of vehicles in some instances was wrongly identified as people. This was due to a failure in the preprocessing steps when a vehicle was detected as multiple moving blobs.

6.7 Summary

Morse theory gives an effective way of combining algebraic and geometric properties for representing spatiotemporal volumes. We have presented a method for constructing a Morse function based on the underlying dynamics of the activity. The candidate Morse function is refined to remove degenerate critical points. Preliminary



(a) (b)

Figure 6.5: TSA dataset: Motion trajectories and critical points for a sequence with vehicle movement.

experiments with the TSA airport surveillance dataset shows encouraging results for activity classification. A Morse theoretic interpretation to the computation of SIFT features was described.

Chapter 7

Summary and Future Work

7.1 Event probability sequences

A sequence of instantaneous events is said to represent an activity. They provide a compact model of relevant parts for recognizing activities. We formulate the event detection problem within the HMM framework and compute the probability of an event occuring at every time instant. Training is a two-step procedure in which the HMM parameters are estimated before computing the event probability sequences for every trajectory. Multiple trajectories are used to train an HMM whereas every trajectory is associated with an event probability sequence. Depending on changes in the viewing direction, multiple trajectories of an activity may be used to train separate HMMs. As part of future work, we plan to develop efficient methods to check for such conditions.

The event probability is based on a simple step edge with a certain support region, i.e., for $p = 3$ we have an event if $Q = \{2, 2, 2, 1, 1, 1\}$. We categorize events based on the two states involved in the transition, the region of support and the probability of the transition. This can be modified to include more complex patterns so that the events of various types can be detected; for example, an event of the type $Q = \{2, 1, 2, 1, 2, 1\}$. Discovering such event patterns using multiple observations of an activity is an interesting problem.

Applications to anomaly detection using event probability sequences were described. As part of future work, we intend to quantify the types of anomalies that can be detected.

Another direction we wish to pursue is to develop perceptual ways of organizing events that occur due to object interaction. It would allow the system to focus on those parts of the scene that affect events and activities of interest while ignoring distracting motion in the scene.

7.2 Coarse to fine event hierarchy

We presented five criteria and quantitative measures for evaluating the effectiveness of coarse-to-fine hierarchical representations of human activities. Experiments using both indoor (UCF human action dataset) and outdoor (TSA airport tarmac surveillance dataset) sequences demonstrate the usefulness of hierarchical structures for activity modeling. It was shown that a reduced frame rate of computing event probability sequences did not have a significant impact on consistency in events detected. The effect of reduced spatial resolution on activity recognition was demonstrated using the TSA dataset. At low resolution, it was not possible to reliably extract motion trajectories of individual objects. Instead, aggregate information was used to identify activities in video segments. This reduces computational time required, but compromises the level of detail in modeling. Fine details of activities were extracted using motion trajectories extracted at the original video resolution. As part of future work, we will address minimalism and stability of

hierarchical activity representations.

7.3 Epitomic representation of activities

We have presented an epitomic representation for modeling activities using piecewise linear segments. An epitome is said to be a tuple consisting of the estimated system matrix, initial value of the state and statistics (mean and covariance) of the input signal. The Iwasawa matrix decomposition was used to factorize the system matrices into three components that represent rotation, scaling and projective action of the state vector. The decomposition not only provides a physical insight into the effect of epitomes, but also allows us to compute geodesic distances using Riemannian metrics on each component. The usefulness of each component was demonstrated using activity recognition, key frame detection and clustering. Experimental results with the UCF action database and the TSA surveillance dataset are encouraging.

Instead of treating the input signal as residual error in modeling, we intend to explore its role in linking the statistical estimates of the model with semantics of activities.

Star diagrams were introduced as synthesized trajectories of epitomes. The shape of star diagrams formed by connecting end points of successive trajectories can be used as a signature of activities. We wish to pursue this direction to model activity trajectories.

7.4 Spectral methods

Antieigenvalues were introduced as a way to characeterize key frames in activities, which are based on changes. As part of future work, we wish to develop more efficient ways of calculating antieigenvalues. Presently, it is necessary to compute eigenvalues since the problem is formulated using cosine of the operator. Alternatively, sine of operator can be used to find approximations, which does not involve an eigenvalue computation.

7.5 Spatio-temporal volumes

Morse functions based on dynamics were used to model the spatio-temporal volumes carved out by objects as they move in the scene. Also, Morse functions were shown to be related to SIFT features through a particular choice of the Morse function based on image intensity. We wish to pursue this work further and attempt to relate semantics of activities with homotopic sections of activity volumes. We plan to investigate the utility volume representation of activities that are captured using moving cameras.

Chapter A

Baum-Welch algorithm

This section contains a brief overview of the Baum-Welch algorithm that was introduced in [82]. There are several sources that offer a detailed explanation (e.g., [26], [82]). Let $\lambda = (A, B, \Pi)$ represent an HMM, where $A = [a_{ij}]$ is the transition probability matrix, B contains emission probabilities conditioned on the current state and Π is the initial distribution of states. Let $O = \{o_1, o_2, \ldots, o_T\}$ be the observation sequence. The Baum-Welch algorithm is an expectation-maximization (EM) algorithm that can compute parameters of A, B and Π such that the likelihood function $P(O|\lambda)$ is maximized. It involves the following steps:

- Choose the initial parameters of λ. (usually through a k-means procedure).
- Re-estimate the parameters using the forward and backward variables. This is equivalent to computing estimates such that:

$$\bar{a}_{ij} = \frac{\text{expected \# transitions from state } i \text{to } j}{\text{expected \# transitions from state } i}$$

$$\bar{b}_j(o) = \frac{\text{expected \# times in state} j \text{ and observing } o}{\text{expected \# times in state } j}$$

- Iterate until convergence.

Chapter B

Proof of Proposition 1 (Chapter 2)

Proposition 1 (Sufficient condition for event sequence to be view invariant): For the event probability sequences to be invariant under changing viewing conditions, the associated HMMs must be conforming.

Proof: The basic idea of the proof is that for every event of type (k_i, l_i) detected for a viewing direction, there must be a corresponding event of type $(\tilde{k}_i, \tilde{l}_i)$ when viewed from a different direction.

Let $\lambda = (A, B, \Pi)$ be an HMM that generates output symbols O, and $X(\lambda)$ the associated topological space. The structure of the probability transition matrix A and the state distributions B reflect the topology of the HMM. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be an affine transformation of O. We can construct another HMM $\tilde{\lambda}$, with the same number of states, that generates $\tilde{O} = f(O)$ such that $F : X(\lambda) \rightarrow X(\tilde{\lambda})$ is a homeomorphism.

(i) By construction, the two HMMs have the same topology (number of states, model structure etc). Since the two sets of observation O and $\tilde{O}$ are related by a non-trivial affine map, we may write $P(O \in S/\lambda) = P(f(O) \in \tilde{S}/\tilde{\lambda})$ for some open S. So, the map F is surjective. If $O_1, O_2 \in X(\lambda)$ are both mapped to $\tilde{O} \in X(\tilde{\lambda})$ under F, choose the observation sequence that is generated by the optimal state sequence. So, the mapping F is made injective. (ii) Every open set U in $X(\tilde{\lambda})$ has

an open set $F^{-1}(U) \in X(\lambda)$ as long as the dimension of O and $\tilde{O}$ are the same. Every open ball in $X(\lambda)$ is transformed to an open ball in $X(\tilde{\lambda})$ as long as the dimension of O and $\tilde{O} = f(O)$ are the same. It follows that F is continuous. (iii) By interchanging the position of (O, λ) and $(\tilde{O}, \tilde{\lambda})$, we see that F^{-1} is continuous. From (i)-(iii), F is a homeomorphism. We see that these assumptions may break down at orthogonal viewing directions.

Chapter C

Definitions

Definition 2 *The general linear (matrix) group GL, special linear group SL, real symplectic group Sp, orthogonal group O, special orthogonal SO group are defined as follows:*

$$\begin{aligned} GL(n) = \quad GL(n,\mathbb{R}) &= \{A \in \mathbb{R}^{n\times n} : \det A \neq 0\} \\ SL(n) = \quad SL(n,\mathbb{R}) &= \{A \in GL(n) : \det A = 1\} \\ Sp(n) = \quad Sp(2n,\mathbb{R}) &= \{A \in SL(2n) : A^T J_n A = J_n\} \\ O(n) = \quad O(n,\mathbb{R}) &= \{A \in GL(n) : AA^T = I\} \\ SO(n) = \quad SO(n,\mathbb{R}) &= \{A \in O(n) : \det A = +1\}, \end{aligned}$$

where $J_n = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}$ and I_n is the $n \times n$ identity matrix.

Definition 3 *A function $f : U \to \mathbb{R}^n$ is differentiable of class C^r if the partial derivatives of the component functions $f^1, \ldots, f^n$ through order r are continuous on the open set U. If f is of class C^r for all finite r, it is said to be of class C^∞ or a smooth function.*

Definition 4 *Let $\gamma : [-1, 1] \to M$ be a parameterized curve on a manifold M. Let $\mathcal{G}$ be the set of all differentiable parameterized curves through $x \in M$, such that $\gamma(0) = x$. Let $\mathcal{F}$ be the set of all functions that are defined and differentiable near*

$x \in M$. The derivative is defined as

$$\frac{d}{dt} f \cdot \gamma \Big|_{t=0}$$

The equivalence class formed by all such curves $\gamma \in \mathcal{G}$ are called tangent vectors at $x \in M$. The set of all tangent vectors at a point forms a vector space. It is called the tangent space at $x \in M$ and is denoted as TM_x. The disjoint union of tangent spaces at each point is called the tangent bundle TM, i.e., $TM = \bigcup_{x \in M} TM_x$.

Definition 5 *Let M be a C^r manifold. Suppose we have an inner product $\langle v, w \rangle$ defined on each tangent space of M, which is differentiable of class C^q, $0 \leq q < r$. Such an inner product is called a Riemannian metric on M.*

Chapter D

Iwasawa decomposition

Theorem 2 *Let $F \in Sp(2n, \mathbb{R}) \subset SL(2n, \mathbb{R})$, $n > 1$ such that*

$$F = \begin{pmatrix} F_1 & F_2 \\ F_3 & F_4 \end{pmatrix}$$

for $F_i \in \mathbb{R}^{n \times n}$, $i = 1, \ldots, 4$. Let

$$M = F^T F = \begin{pmatrix} M_1 & M_2 \\ M_2 & M_4 \end{pmatrix} \in Sp(2n, \mathbb{R}),$$

where M_1 is positive definite. Then Iwasawa decomposition of $F = KAN$ is given by

$$A = \begin{pmatrix} D^{1/2} & 0 \\ 0 & D^{-1/2} \end{pmatrix}, N = \begin{pmatrix} R & RM_1^{-1}M_2 \\ 0 & (R^{-1})^T \end{pmatrix}$$

and $K = PN^{-1}A^{-1}$.

Proof 1 *From the definition of M, $M_1 = F_1^T F_1 + F_3^T F_3$, $M_4 = F_2^T F_2 + F_4^T F_4$ and $M_2 = F_1^T F_2 + F_3^T F_4$. Since $M \in Sp(2n, \mathbb{R})$, $M_1^T M_2 = M_2^T M_1$ and $M_1^T M_4 - M_2^T M_2 = I$. Following a simple calculation, M can be written as:*

$$M = \begin{pmatrix} I_n & X \\ 0 & I_n \end{pmatrix} \begin{pmatrix} M_1 & 0 \\ 0 & (M_1^{-1})^T \end{pmatrix} \begin{pmatrix} I_n & X \\ 0 & I_n \end{pmatrix}, \tag{D.1}$$

where $X = M_1^{-1}M_2$. Let $M_1 = R^T D R$ be the Cholesky factorization of M_1, where R is an upper triangular matrix and D is a diagonal matrix with positive

entries. Since $K \in SO(n, \mathbb{R})$ *we have*

$$\begin{aligned} M &= P^T P = (KAN)^T KAN \\ &= N^T A^T K^T KAN = N^T A^T AN, \end{aligned} \tag{D.2}$$

The result follows from (D.1) and (D.2).

Bibliography

[1] C. Rao, A. Yilmaz, and M. Shah, "View-invariant representation and recognition of actions," *International J. Comput. Vision*, vol. 63, pp. 257–285, 1989.

[2] Y. A. Ivanov and A. F. Bobick, "Recognition of visual activities and interactions by stochastic parsing," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, pp. 852–872, 2000.

[3] D. D. Vecchio, R. M. Murray, and P. Perona, "Decomposition of human motion into dynamics based primitives with application to drawing tasks," *Automatica*, vol. 39(12), pp. 2085–2098, 2003.

[4] V. T. Vu, F. Bremond, and M. Thonnat, "Temporal constraints for video interpretation," in *Proc. European Conference on Computer Vision*, 2002.

[5] R. Chellappa, N. P. Cuntoor, S. W. Joo, V. Subrahmanian, and P. Turaga, *Understanding Events: How Humans See, Represent, and Act on Events.* Oxford University Press, To appear, ch. Computational Vision Approaches for Event Modeling.

[6] S. Tsuji, A. Morizono, and S. Kuroda, "Understanding a simple cartoon film by a computer vision system," in *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 1977, pp. 609–610.

[7] B. Neumann and H. J. Novak, "Event models for recognition and natural language descriptions of events in real-world image sequences," in *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 1981, pp. 724–726.

[8] A. Kojima, T. Tamura, and K. Fukunaga, "Natural language description of human activities from video images based on concept of hierarchy of actions," *International J. Comput. Vision*, vol. 50(2), pp. 171–184, 2002.

[9] L. Zelnik-Manor and M. Irani, "Event based analysis of video," in *Proc. IEEE International Conf. Comput. Vision and Pattern Recognition*, 2001.

[10] M. T. Chan, A. Hoogs, J. Schmiederer, and P. M, "Detecting rare events in video using semantic primitives with hmm," in *Proc. IEEE International Conf. on Pattern Recognition*, vol. 4, Cambridge, UK, 2004, pp. 150–154.

[11] G. Medioni, I. Cohen, F. Bremond, S. Hongeng, and R. Nevatia, "Event detection and analysis from video streams," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 8, pp. 844–851, 2001.

[12] A. R. J. Francois, R. Nevatia, J. Hobbs, and R. C. Bolles, "Verl: An ontology framework for representing and annotating video events," *IEEE Multimedia*, vol. 12(4), pp. 76–86, 2005.

[13] N. Johnson and D. Hogg, "Learning the distribution of object trajectories for event recognition," *Image and Vision Computing*, vol. 14(8), pp. 609–615, 1996.

[14] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24(9), pp. 1450–1464, 2006.

[15] C. Stauffer, "Learning a factorized segmental representation of far-field tracking data," in *Proc. IEEE Workshop on Event Mining*, vol. 7, 2004, pp. 115–121.

[16] V. Parameswaran and R. Chellappa, "View invariants for human action recognition," in *Proc. IEEE Comput. Vision and Pattern Recognition*, 2003.

[17] T. Syeda-Mahmood, "Segmenting actions in velocity curve space," in *Proc. ICPR*, 2002.

[18] T. Syeda-Mahmood, A. Vasilescu, and S. Sethi, "Recognizing action events from multiple viewpoints," in *Proc. IEEE Workshop on Detection and Recognition of Events*, 2001.

[19] C. Stauffer and E. Grimson, "Learning patterns of activity using real-time tracing," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22(8), pp. 747–757, 2000.

[20] N. Vaswani, A. R. Chowdhury, and R. Chellappa, "Activity recognition using the dynamics of the configuration of interacting objects," *IEEE Trans. Image Processing*, vol. 14(10), pp. 1603–1616, 2005.

[21] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *Int. Journal of Computer Vision*, vol. 9(2), pp. 137–154, 1992.

[22] L. Toressani and C. Brelger, "Space-time tracking," in *Proc. ECCV*, 2002.

[23] A. K. R. Chowdhury and R. Chellappa, "A factorization approach for activity recognition," in *Proc. IEEE Workshop on Event Mining*, vol. 4, Madison, WI, USA, 2003, pp. 41–46.

[24] Y. Caspi, D. Simakov, and M. Irani, "Feature-based sequence-to-sequence matching," *International Journal of Computer Vision*, vol. 68(1), pp. 53–64, 2006.

[25] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *Proc. IEEE International Conf. Comput. Vision and Pattern Recognition*, 1992, pp. 994–999.

[26] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech processing," *Proc. IEEE*, vol. 63, pp. 257–285, 1989.

[27] J. Aggarwal and Q. Cai, "Human motion analysis:a review," *Computer Vision and Image Understanding*, vol. 73, no. 3, pp. 428–440, 1999.

[28] T. Starner and A. Pentland, "Real-time american sign language recognition from video using hidden markov models," in *Proc. SCV*, 1995, pp. 265–270.

[29] A. Kale, A. N. Rajagopalan, A. Sundaresan, N. P. Cuntoor, A. K. Roy-Chowdhury, V. Kruger, and R. Chellappa, "Identification of humans using gait," *IEEE Trans. Image Processing*, pp. 1163–1173, 2004.

[30] R. Hamid, Y. Huang, and I. Essa, "Argmode - activity recognition using graphical models," in *Proc. IEEE Comput. Vision and Pattern Recognition*, vol. 4, Madison, WI, USA, 2003, pp. 38–43.

[31] Y. Shi and A. F. Bobick, "P-net: A representation for partially-sequenced, multi-stream activity," in *Proc. IEEE Workshop on Event Mining*, vol. 4, Madison, WI, USA, 2003, pp. 40–47.

[32] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition," in *Proc. IEEE Comput. Vision and Pattern Recognition*, San Juan, Puerto Rico, 1997, pp. 994–999.

[33] Z. Ghahramani and M. I. Jordan, "Factorial hidden markov models," *Machine Learning*, vol. 29(2-3), pp. 245–273, 1997.

[34] D. Koller and U. Lerner, *Sequential Monte Carlo Methods in Practice*. Springer, 2001, ch. Sampling in Factored Dynamic Systems, pp. 445–464.

[35] M. Brand and V. Kettnaker, "Discovery and segmentation of activities in video," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 844–851, 2000.

[36] N. Oliver, E. Horvitz, and A. Garg, "Layered representations for human activity recognition," in *Proc. IEEE International Conf. on Mulitmodal Interfaces*, Pittsburgh, PA, USA, 2002, pp. 3–7.

[37] H. Zhong, J. Shi, and M. Visontai, "Detecting unusual activity in video," in *Proc. IEEE Comput. Vision and Pattern Recognition*, Washington, D.C, 2004, pp. 819–826.

[38] I. Koprinska and S. Carrato, "Temporal video segmentation: a survey," *Signal Processing: Image Commun.*, vol. 16, no. 5, pp. 477–500, 2001.

[39] J. R. Kender and B. L. Yeo, "Video scene segmentation via continuous video coherence," in *Proc. IEEE Comput. Vision and Pattern Recognition*, Santa Barbara, CA, USA, 1998, pp. 367–373.

[40] M. Isard and A. Blake, "A mixed-state condensation tracker with automatic model switching," in *Proc. IEEE Comput. Vision and Pattern Recognition*, 1998.

[41] Z. Ghahramani and G. E. Hinton, "Variational learning for switched state-space models," *Neural Computation*, vol. 12, no. 4, pp. 969–996, 1998.

[42] A. B. Kurzhanski and P. Varaiya, "Control design of an automated highway system," *Journal of Optimization Theory and Applications*, vol. 108(2), pp. 227–251, 2001.

[43] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: a study in multiagent hybrid systems," *IEEE Trans. Automatic Control*, vol. 43(4), pp. 509–521, 1998.

[44] C. Jordan, "Essai sur la geometrie a n dimensions," *Bulletin de la Societe Mathematique*, vol. 3, pp. 103–174, 1875.

[45] H. Hotelling, "Relation between two sets of variates," *Biometrika*, vol. 28, pp. 321–372, 1936.

[46] K. D. Cock and B. D. Moor, "Subspace angles and distances between arma models," in *Proceedings of MTNS*, 2000.

[47] R. Martin, "A metric for arma processes," *IEEE Trans. on Signal Processing*, vol. 48(4), pp. 1164–1170, 2000.

[48] A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto, "Detecting unusual activity in video," in *Proc. CVPR*, 2001, pp. 819–826.

[49] Y. Sheikh and M. Shah, "Exploring the space of an action for human action recognition," in *Proc. ICCV*, 2005.

[50] R. Kjeldesn and J. Kender, "Finding skin color images," in *Proc. Face and Gesture Recognition*, 1996.

[51] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 7, pp. 629–639, 1990.

[52] B. Georis, M. Maziere, F. Bremond, and M. Thonnat, "A video interpretation platform applied to bank agency monitoring," in *Proc. Workshop on Intelligent Distributed Surveillance Systems (IDSS)*, 2004.

[53] T. Izo and W. E. L. Grimson, "Simultaneous pose estimation and camera calibration from multiple views," in *Proc. IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, vol. 1, Washington, D.C, 2004, pp. 14–21.

[54] N. P. Cuntoor, B. Yegnanarayana, and R. Chellappa, "Interpretation of state sequences in hmm for activity representation," in *Proc. IEEE Conf. on Acoustic Speech and Sig. Processing*, vol. 2, Philadelphia, PA, USA, 2005, pp. 709–712.

[55] B. H. Juang, "On the hidden markov model and dynamic time warping for speech recognition - a unified view," *Technical Journal*, vol. 63, pp. 1213–1243, 1984.

[56] J. Bilmes, "A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *Technical Report ICSI-TR-97-021*, 1997.

[57] I. Csiszar and P. Shields, "The consistency of the bic markov order estimator," *Annals of Statistics*, vol. 28, pp. 1601–1619, 2000.

[58] M. Fels and P. J. Olver, "On relative invariants," *Math. Annals*, pp. 701–732, 1997.

[59] A. Kuijper and L. M. J. Florack, "The hierarchical structure of images," *IEEE Trans. Im. Proc.*, vol. 12(9), pp. 1067–1080, 2003.

[60] D. Gavrila, "Multifeature hierarchical template matching using distance transform," *Proc. IEEE ICPR*, 1998.

[61] F. Fleuret and D. Geman, "Coarse to fine face detection," *IJCV*, vol. 41, pp. 85–107, 2001.

[62] O. Faugeras, *Three dimensional computer vision: A geometric viewpoint.* MIT Press, 1993.

[63] D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three dimensional shapes," *Proc. Royal Society, London*, vol. B:200, pp. 269–274, 1978.

[64] G. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis: forecasting and control.* Prentice Hall, 1994.

[65] M. J. Allen and W. M. Yen, *Introduction to measurement theory.* Waveland Press, Long Grove, IL, 2002.

[66] A. A. C. Wren, T. Darrell, and A. Pentland, "Pfinder: Real time tracking of the human body," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, pp. 780–785, 1997.

[67] I. Haritaoglu, R. Cutler, D. Harwood, and L. S. Davis, "Backpack: Detection of people carrying objects using silhouettes," in *Proc. IEEE Comput. Vision and Pattern Recognition*, 1999.

[68] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *IJCAI*, pp. 674–679, 1981.

[69] K. Iwasawa, "On some types of topological groups," *Annals of Mathematics*, vol. 50, pp. 507–558, 1949.

[70] S. Helgason, *Differential geometry, lie groups and symmetric spaces.* New York: Academic Press, 1978.

[71] A. Terras, *Harmonic analysis on symmteric spaces and applications II.* Berlin: Springer-Verlag, 1988.

[72] D. Serre, *Matrices: Theory and applications.* Springer-Verlag, 2002.

[73] D. Bao, S. S. Chern, and Z. Shen, *An Introduction to Riemann-Finsler Goemetry.* New York: Springer, 2000.

[74] B. Hanzon, *Identifiability, recursive identification and spaces of linear dynamical systems.* Amsterdam: CWI, 1990.

[75] A. Weinstein, "Almost invariant submanifolds for compact group actions," *J. Eur. Math. Soc.*, vol. 2(1), pp. 53–86, 2000.

[76] E. Klassen, A. Srivastava, W. Mio, and S. Joshi, "Analysis of planar shapes using geodesic paths on shape spaces," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26(3), pp. 372–383, March 2004.

[77] M. Gromov, *Metric Structures for Riemannian and Non-Riemannian Spaces*. New York: Birkhauser, 1999.

[78] A. Mielke, *Geometry, dynamics and mechanics*. Springer-Verlag, 2002, ch. Finite elastoplasticity, Lie groups and geodesics on SL(d), pp. 445–464.

[79] B. Sin and J. H. Kim, "Nonstationary hidden markov model," *Signal Processing*, vol. 46, no. 1, pp. 31–46, 1995.

[80] Y. Ephraim, A. Dembo, and L. R. Rabiner, "A minimum discrimination information approach for hidden markov modeling," *IEEE Trans. on Information Theory*, pp. 629–639, 1998.

[81] M. Vidyasagar, *Nonlinear systems analysis*. Prentice Hall, 1993.

[82] L. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains," *Ann. Math. Stat.*, vol. 41, no. 1, pp. 164–171, 1970.

[83] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 63, no. 1, pp. 268–278, March 1973.

[84] A. Kavcic and J. M. F. Moura, "The viterbi algorithm and markov noise memory," *IEEE Trans. on Information Theory*, vol. 46, no. 1, pp. 291–30, January 2000.

[85] F. DeNatale, O. Mayora-Ibarra, and L. Prisciandaro, "Interactive home assistant for supporting elderly citizens," in *Proc. EUSAI Workshop on Ambieng Intelligence Technologies for Wellbeing at Home*, 2004.

[86] K. Gustafson, "Antieigenvalues," *Linear Algebra and Appln.*, vol. 208, pp. 437–454, 1994.

[87] K. Gustafson and M. Seddinghin, "Antieigenvalue bounds," *J. Math. Anal. and Appln.*, vol. 143, pp. 327–340, 1989.

[88] M. A. Turk and A. Pentland, "Face recognition using eigenfaces," in *Proc.CVPR*, 1991.

[89] H. Edelsbrunner, J. Harer, and A. Zomorodian, "Hierarchical morse-smale complexes for piecewise linear 2-manifolds," *Discrete and Computational Geometry*, vol. 30, pp. 87–107, 2003.

[90] X. Ni, M. Garland, and J. C. Hart, "Fair morse functions for extracting the topological structure of a surface mesh," in *SIGGRAPH*, 2004, pp. 613–622.

[91] Y. Shinagawa, T. Kunii, and Y. Kergosien, "Surface coding based on morse theory," *IEEE Computer Graphics and Applications*, vol. 11(5), pp. 66–78, 1991.

[92] M. Lades, J. C. Vorbrueggen, J. Buhmann, and J. Lange, "Distortion invariant object recognition in the dynamic link architecture," *IEEE Trans. on Computers*, vol. 42(3), pp. 300–311, 1993.

[93] D. Lowe, "Distintintive image features from scale invariant keypoints," *International Journal of Computer Vision*, vol. 60(2), pp. 91–110, 2004.

[94] S. Niyogi and E. Adelson, "Analyzing and recognizing walking figures in xyt," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 469–474.

[95] F. Cootes, C. Taylor, D. Cooper, and J. Graham, "Active shape models-their training and application," in *CVIU*, vol. 61(1), 1995, pp. 38–59.

[96] J. Konrad and M. Ristivojevic, "Joint space-time image sequence segmentation: Object tunnels and occlusion volumes," in *Proc. ICASSP*, 2004, pp. 9–12.

[97] Y. Ricquenbourg and P. Bouthemy, "Real-time tracking of moving persons by exploiting spatiotemporal image slices," *IEEE Trans. PAMI*, vol. 22(8), pp. 797–808, 2000.

[98] T. Lindeberg, "Detecting salient blob-like image structures and their scales with a scale space primal sketch," *International Journal of Computer Vision*, vol. 11(3), pp. 283–318, 1993.

[99] H. Moravec, "Rover visual obstacle avoidance," *IJCAI*, pp. 785–790, 1981.

[100] J. Canny, "A computational approach to edge detection," *IEEE Trans.on Pattern Analysis and Machine Intelligence*, vol. 8(6), 1986.

[101] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vision Conference*, 1988, pp. 147–151.

[102] A. Efros, A. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," in *Proc. ICCV*, 2003.

[103] L. Gorelick, M. Galun, E. Sharon, R. Basri, and A. Brandt, "Shape representation and classification using the poisson equation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[104] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions and space-time shapes," in *Proc. IEEE International Conference on Computer Vision*, 2005.

[105] J. Milnor, *Morse theory*. Princeton University Press, 1963.

[106] T. F. Banchoff, "Critical points and curvature for embedded polyhedral surfaces," *American Mathematical Monthly*, vol. 77, pp. 475–485, 1970.