GLOBALLY-COORDINATED LOCALLY-LINEAR MODELING OF MULTI-DIMENSIONAL DATA

BY

CHE-BIN LIU

B.S., National Taiwan University, 1996

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Department of Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2006

Urbana, Illinois

To my parents,
my wife Jung
and my son Bennett.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Prof. Narendra Ahuja, for his guidance and support in the course of my graduate study. His passion for research and commitment to perfection give me lifelong inspiration.

Many thanks to my friend, Ruei-Sung Lin, for his collaboration on the problem of learning globally-coordinated nonlinear manifolds. I also thank every group member in the lab for their great friendship.

I sincerely thank my parents for their unconditional love and encouragement. Without their support, I would not have come to Illinois for the graduate study.

Most of all, I am extremely grateful to my wife, Jung, for her endless love and support. While being a hard-working engineer in the daytime, she is always a lovely wife and mother who takes excellent care of me and our son, Bennett, everyday after coming back from work. Without her support and understanding I would never have completed this thesis.

TABLE OF CONTENTS

PAGE

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

CHAPTER 1

INTRODUCTION

1.1 Motivation

This thesis is motivated by the objective of modeling a variety of spatiotemporal variations in video sequences. These include variations in raw color values as well as certain mappings of these values. Further, since an arbitrary scene consists of distinct objects occupying different parts at different times, video sequences possess different properties in different spatiotemporal segments. Therefore, our goal is local rather than global spatiotemporal modeling. The models we plan to develop will indeed apply to not just video but a variety of multivariate multi-dimensional data encountered in everyday life. For example, a one-dimensional (1D) waveform may be a sleep (electroencephalograph, or EEG) signal, a blood pressure record, or an electrocardiogram (ECG) profile over time (Figure 1.1). Ordinary photographs and sonograms are examples of two-dimensional (2D) data. A video sequence is three-dimensional (3D) data. Four-dimensional (4D) data are encountered, for example, as temporal records of dynamic 3D structures such as magnetic resonance volume images of heart.

Let us define a multivariate function $y = f(x)$, where $y \in \mathbb{R}^n$ and $x \in \mathbb{R}^m$, that represents a set of n-dimensional data y defined in a m-dimensional domain x. In this thesis, we are interested in the case where multi-dimensional value y is color, or properties computed from

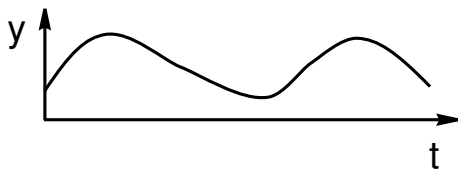


Figure 1.1 A one-dimensional waveform can be a sleep signal, a heart beating or a blood pressure record over time.

Figure 1.2 A 2D slice of a brain (CT brain image).

these values such as shape and texture, and the multivariate domain x can be space, time, or a combination of them, and we must partition the space such that the variation within each part is linear although not necessarily independent. For example, consider a video sequence of moving people. The pose of a given face changes linearly. However, this linear change has different parameters in regions occupied by different faces. Thus the ensemble of face regions in the video sequence can be modeled by locally linear models of pose variation. Similar locally linear models can be obtained for a video sequence of a dynamic texture where y represents texture properties.

As another example, consider an objet embedded within a 3D space domain, and adjacent 2D slices of the object, e.g., a brain, collected at a series of points along a particular axis. Suppose a specific 3D structure in the slice is represented by its 2D contour (see Figure 1.2).

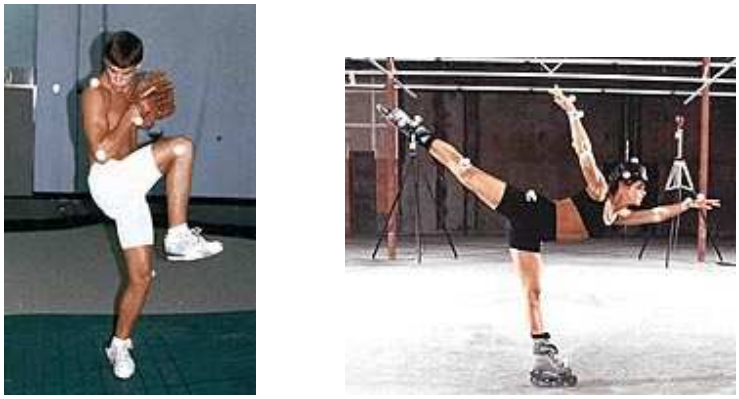


Figure 1.3 Motion capture technique is used to reproduce realistic motion sequences for video games, animation production, etc. Images are acquired from Motion Analysis Corporation.

If the 2D contour is represented by a set of N parameters (e.g. Fourier coefficients), then y is a N-dimensional vector. Since the brain structure is a continuous 3D surface, the 2D contours of two adjacent slices are very likely to be similar, and the contour representation y would therefore be continuous and locally linear along the axis x. A sequence of visual images of a fire can also be represented in a similar way, with y as an N-dimensional 2D contour and x the time each image is taken. However, fire contours exhibit a stochastic behavior, like dynamic texture video sequences, so not the contour but certain of its properties (e.g., contour center) are continuous.

Another example of multiple-object, locally linear variation is human motion data. One widely used method capturing such data is by recording a sequence of 3D locations of specific body parts by placing a number of optical/magnetic markers on a human actor and the 3D positions of these markers are measured over time (see Figure 1.3). If N markers are attached to the human actor, then the motion capture data can be expressed as a function with $3N$-dimensional values y in the time domain x. Note that the vector data y represent multiple objects (body parts), each of which undergoes a continuous and locally linear motion (e.g.

points located on hand, knee, foot). Some of the objects move under strong mutual constraints (e.g., shoulder and elbow, and knee and ankle are at fixed distances) whereas others move relatively independently (e.g. head and feet). The goal of modeling here is to partition the data and identify the locally linear model of each part. The global model exploits the relationships among parts, and thus is more efficient (lower-dimensions) than a union of independent linear models.

Another aspect of locally linear modeling concerns whether the models are obtained for the data in the original domain x or after reducing its dimensionality by rasterizing and obtaining models for lower dimensional x. Clearly, reducing the dimensionality obscures the true variation in the data, however, it has been often used in applications where the obscured characteristics are not important, or only for its simplicity. Consider an image volume with T frames of M-by-N gray-scale images. In its original space, the image volume can be expressed by a function defined in the domain $x \in \mathbb{R}^3$ within a $M \times N \times T$ volume, and the function has a one-dimensional gray-scale value y at any given x. Using this representation, properties of each pixel can be investigated in both spatial and temporal domain. If only temporal properties are relevant, it suffices to reexpress the data in terms of a $(M \times N)$-dimensional vector y of gray values, obtained, for example, by rasterizing each image, and the variation to be modeled as that of the rasterized vector with respect to time. Examples of such linear modeling of video data include [1, 2] in the temporal domain and [3] in the spatio-temporal domain.

We focus on developing models that capture variations in continuous, locally-linear, multi-dimensional data. We evaluate our models on the tasks of (i) prediction of dynamic object appearance, (ii) reproduction of dynamic object shape, appearance and motion, and (iii) recog-

nition of dynamic object based on its temporal variation in shape. The proposed approaches can be easily used in spatial only or spatiotemporal domain, and be used in different applications.

1.2 Problem Definition

In this thesis, we investigate continuous, locally-linear models of visual data. For a given visual data sequence $\{y_1, y_2, \ldots, y_n\}$, we learn a model $\mathcal{M}$ that characterizes temporal variations in $\{y_t\}$ which are continuous and locally linear. We then use the learned model $\mathcal{M}$ to perform the following visual tasks:

1. *Prediction*: predict the most probable y_{n+1} based on the learned model $\mathcal{M}$.

2. *Reproduction*: generate a new sequence $\{y'_1, y'_2, \ldots, y'_T\}$ given $\{y_1, y_2, \ldots, y_n\}$ based on the learned model $\mathcal{M}$ of a given $\{y_1, y_2, \ldots, y_n\}$.

3. *Recognition*: recognize a previously unseen member of a class of dynamic objects in terms of certain temporal features of the object motion selected based on the model $\mathcal{M}$ of the class.

In the following, we define each specific task for which we use locally linear modeling in this thesis.

1.2.1 Tracking of Objects

Given a video sequence of an object undergoing unknown pose changes, learn the appearance model of the object. Given another video sequence, locate the object and estimate its pose in each image frame.

In this task, we predict an object's 2D location and appearance over time. The object in each image is circumscribed in a bounding box which is described by its center and its width and height. The appearance of the object is captured by the image content in the bounding box, scaled to a standard size k-by-k. Each y_i is the image inside the box along with 2D location (center) and scale (width and height) of the object. Therefore, each y_i is a (k^2+4)-dimensional vector. Linear models are widely used to capture data sets with small variations. For large variations, a linear model may not suffice. However, if the object undergoes a continuous movement, its appearance change over a short period will be small, and therefore could be modeled as locally linear.

We apply our model to tracking human faces. Most other trackers track only the 2D location and the scale of a face. At time $t+1$, they sample/search possible bounding boxes in the four-dimensional space (i.e. location and scale) and select the one with image content having the highest likelihood of being the target face. We track the face pose as well which means we search for not only a high-likelihood face but also a high-likelihood pose. When multiple faces are present in the scene, tracking face pose prevents the tracker from drifting to a nearby face and thus facilitates more reliable tracking. Using a piecewise-linear model to characterize a face helps to better distinguish between poses.

1.2.2 Synthesis of Dynamic Textures

Videos of dynamic textures exhibit statistical variations with time. Given such a video sequence, model its temporal variations, and then generate a new video sequence based on the learned model.

Dynamic textures exhibit statistically repeated patterns in the temporal domain. Examples include wavy ocean, flapping flags, smoke, etc. Since we are concerned with the temporal relationships between successive images, we represent the input video as a sequence of image vectors over time. Here each of y_i's is a $(M \times N)$-dimensional vector if the video contains a sequence of M-by-N gray-scale images.

Previous works in the literature focus on linear modeling of such data, assuming the textural variation is small and the change in rasterized vector y over time is linear. In the real world, however, changes in the environment may cause changes in dynamic textures in different ways. Take a flag in the wind for example. When there is a light breeze, the flag might hang limp or move slowly. As the wind becomes stronger, the flag starts to fly, and flaps vigorously when the winds blow very strongly. Adding the directional changes of the wind, the flag shows distinguishable shapes, appearances and dynamics over time. When the direction and the speed of the wind remain constant, the flag motion may be characterized using linear models. Often times, the wind changes unpredictably, therefore linearity of dynamic textures occurs only in short time periods.

We employ a locally linear model to describe motion sequences of dynamic textures that involve large variations. Each linear component of the model captures a small variation within a subset of images acquired during a sufficiently short time period. Synthesis is achieved by creating images according to the learned model.

1.2.3 Synthesis of Articulated Motion

Given a sequence of 3D positional data of multiple objects that move under kinematic constraints, model their temporal variations and then generate a new motion sequence of these objects based on the learned model.

We use sequences of data representing 3D motion of human joints measured from a human actor wearing markers on joints. As mentioned in Section 1.1, we treat multiple objects moving under constraints as a single data vector at each time. Therefore, the input data y is a $3N$-dimensional vector if N markers are used. Example human motions include dancing, hand gestures, etc.

Previous works in the field of computer graphics have used locally linear models for human motion synthesis. The general solution is to identify some transition points for locally linear models, which pose extra constraints on the model. These transition points are analog to key frames and the synthesized motion is forced to go through transition points. Without transition points, however, their models do not ensure continuous motion because two successive data vectors may be captured by different linear components.

We parameterize our locally-linear model with a global coordinate system. Although two successive data vectors may be captured by different linear components, their representations in our model are still continuous. The continuous representation naturally corresponds to continuous variation of input data.

1.2.4 Recognition of Objects from Video

Given a video sequence of an object undergoing complex shape changes, learn the model that characterizes variations in object shape over time. Given other video sequences of the object, recognize the object based on the parameters of the learned model.

The visual data in this task are 2D shape representations of an object derived from images over time. As mentioned earlier, if a set of N parameters are used to describe a 2D shape, the input data y will be N-dimensional shape representation over time. We use a locally-linear model to describe the dynamic shape changes over sufficiently short time periods.

The model parameters can then be used as discriminating features for recognition of the dynamic object. Of course, using a locally-linear model for recognition is more complex than a linear model.

1.3 Thesis Overview

The general goal of our work is to develop models to capture variations in continuous, locally-linear, multi-dimensional data. We demonstrate the use of the models mainly in the temporal domain, though it can also be used in spatial and spatio-temporal domains. The applications shown in this thesis include tracking, synthesis, recognitions and retrieval of dynamic objects based on shape, appearance and motion of dynamic objects.

In Chapter 2, we first relate our problem to nonlinear dimensionality reduction and intrinsic manifold learning. We use a global coordination model that enables us to model high-dimensional nonlinear data in a mixture of globally-coordinated linear subspaces. A two-stage learning algorithm is adopted for the global coordination model, where we first learn a mixture of linear subspaces for the input data and then align these linear subspaces in a probabilistic

formulation. We extend the model to a dynamic global coordination model and compare it with other linear or nonlinear dynamical models. We also validate the capability of the model to characterize continuous, locally-linear multi-dimensional data.

Chapter 3 instantiates the use of the dynamic global coordination model for data prediction. We develop an approximate inference algorithm for this dynamical model with an application to nonlinear appearance based object tracking. Specifically, we approximate the nonlinear manifold of object appearance using a mixture of locally linear models. These local models are then aligned so that we obtain a coherent representation to parameterize the nonlinear manifold. Unlike conventional tracking methods that track only 2D position and the scale of the object in videos, our model allows us to simultaneously track object pose in the low-dimensional subspace. As a result, our proposed method is robust in the presence of similar objects or temporary occlusion as explained earlier.

Chapter 4 proposes non-parametric dynamic models that operate in globally-coordinated locally-linear subspaces for different types of motion parameters. We demonstrate applications to dynamic texture synthesis and human motion synthesis. We are able to synthesize non-stationary dynamic textures which are beyond the capabilities of existing approaches based on linear dynamical models. We also relax the transition constraints that have been posed by other nonlinear models used for human motion synthesis. We show that our model captures continuous, complex motion as a continuous, low-dimensional trajectory in the globally-coordinated subspace.

Chapter 5 investigates a special case where a linear model is adopted to capture temporal variations in dynamic 2D object shape under the assumption that the dynamics are linear over a short period of time [4]. We represent 2D region shape in terms of the spatial frequency con-

tent of the region contour using Fourier coefficients. The temporal changes in these coefficients are used as the temporal signatures of the shape changes. Specifically, we use an autoregressive model of the Fourier coefficient series. The efficacy of the model is evaluated by several applications. First, we use the model parameters as discriminating features for object recognition and classification. Second, we show the use of the model for synthesis of dynamic shape using the model learned from a given image sequence. A nonlinear model for synthesis is also employed to enrich synthesis results. Third, we show that, with its capability of predicting shape, the model can be used to predict contours of moving regions which can be used as initial estimates for the contour based tracking methods.

In chapter 6, we show that various forms of temporal variations in image content, when used together, have the potential to provide important information for content based video retrieval. We explore the use of different motion representations and evaluate them in retrieving various motion patterns. Our approach assumes that each dynamic object has been tracked and circumscribed in a minimal bounding box in each video frame. We represent the motion attributes of each object in terms of changes in the image context of its circumscribing box, which we call a 2D motion model. We demonstrate the use of the proposed 2D motion model in retrieving objects undergoing complex motion [5].

In chapter 7, we conclude the thesis and present some future directions in which the current work could be extended.

The electronic version of this thesis, experimental results, videos and related publications are available at `http://vision.ai.uiuc.edu/~cbliu/`.

1.4 Contributions

This thesis makes major contributions to the modeling and analysis of continuous, locally-linear, multi-dimensional spatio-temporal data.

- Our approach has a better descriptive power than linear models because our approach is capable of capturing nonlinear but locally-linear variations.

- Toward locally-linear modeling, our approach has advantages over those using mixtures of linear subspaces in that our approach aligns the linear subspaces within the same global coordinate system. As a result, we are able to represent continuous multi-dimensional data using continuous low-dimensional coordinates.

- Our approach is also different from those that use nonlinear embedding which also projects continuous multi-dimensional data onto a low-dimensional subspace while preserving the original continuity properties. Whereas nonlinear embedding has no mapping function to back-project data from the low-dimensional subspace, our model provides a two-way mapping between the original data space and its corresponding set of low-dimensional subspaces. Therefore, while we have the advantage of analyzing multi-dimensional data in low-dimensional subspaces, we can map the results of the analysis back to the original data space, which is essential for many computer vision tasks such as tracking or synthesis that are defined in the original space.

- Our approach extends the previous work on the global coordination model to temporal analysis of continuous, multi-dimensional data. By contrast, the previous work has addressed monolithic modeling of multi-dimensional variation in the data, and further, has been demonstrated on only synthetic data sets. We have developed algorithms for time-

varying data analysis, applied them to full-scale, real-world data sets, and used them in new applications.

We show that the new modeling features of our approach improve the performance of existing approaches in many different applications.

- In object tracking, our approach is the first one to track nonlinear appearance variations by using low-dimensional representation of the appearance change in globally-coordinated linear subspaces.

- In dynamic texture synthesis, we are able to model non-stationary dynamic textures, which cannot be handled by any of the existing approaches.

- In human motion synthesis, we show that synthesis can be performed without using specific transition points, or key frames.

Finally, we use a linear model for recognition of dynamic objects by using locally-linear models of temporal shape variations over short time periods. We demonstrate this simple model in an involving application in fire detection.

CHAPTER 2

DYNAMIC GLOBAL COORDINATION MODEL

In this chapter, we present a global parameterization approach and extend it to a dynamical model to capture variations of multi-dimensional data on a nonlinear manifold. A nonlinear manifold can usually be approximated by a mixture of locally linear models. In such mixture models, each linear component has its own mapping between the high-dimensional input space and the low-dimensional embedding, which results in different coordinate systems and causes difficulties for many computer vision task such as object tracking and motion synthesis. There are also methods in nonlinear dimensionality reduction that map a manifold into a single low-dimensional coordinate system. But these methods preserve only spatial relationships among manifold points, and therefore do not provide a mapping that infers values of manifold points in the input space from the low-dimensional representation.

To overcome the aforementioned problems, we employ a global coordination model [6]. we first use a mixture of linear subspaces to model the input data. We then align these linear subspaces in a probabilistic formulation so as to parameterize the nonlinear manifold within a global coordinate system. We extend the global coordination model to a dynamic Bayesian network. This extended model serves as the foundation of our thesis that enables us to model the temporal variations of continuous, locally-linear, high-dimensional visual data.

2.1 Introduction

Often times, high-dimensional data lie on or near a low-dimensional manifold embedded in the input space. It is a common assumption and has been empirically validated, particularly in visual images taken from a single object or multiple interactive objects. Therefore, given a set of high-dimensional data, we want to learn the intrinsic structure of its manifold.

To find the intrinsic structure of a manifold is a dimensionality reduction problem. When the input data lie on or near a linear subspace, classical techniques such as principal component analysis (PCA) are effective to find such a low-dimensional embedding. However, many real-world data yield large variations and their manifolds are more likely to be nonlinear and cannot be fully captured by a simple PCA model.

The existing methods for nonlinear dimensionality reduction are generally in two categories: (1) global nonlinear projection, or (2) locally linear projection. Methods using global nonlinear projection aim at preserving spatial relationships among the input data on the manifold. They map a nonlinear manifold into a low-dimensional global coordinate system, in the hope of discovering the intrinsic degrees of freedom of the input data. Examples of these methods include Isomap [7] and Locally Linear Embedding (LLE) [8]. However, the high-dimensional coordinates in the input space are not preserved when mapping from the input space to the low-dimensional embedding. For many applications in computer vision, a mapping from the embedding to the input space is needed to reconstruct data in the original space.

Methods using locally linear projection characterize a nonlinear manifold by fitting multiple locally linear models (e.g. mixtures of Probabilistic PCA (MPPCA) [9]). Their mappings preserve information in the original space, so data can be reconstructed given low-dimensional representations. However, each local model has its own mapping between data and its corre-

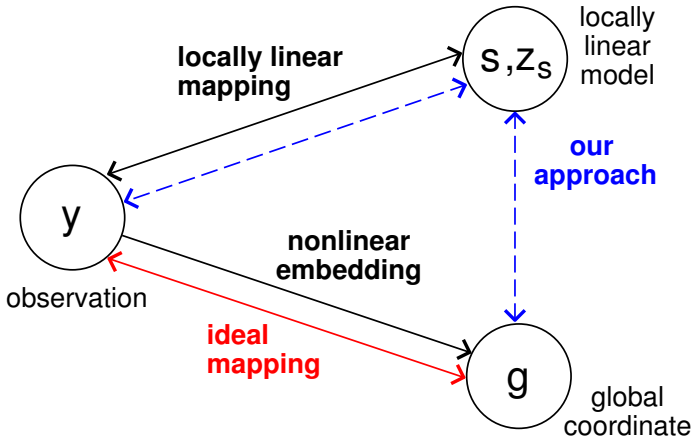


Figure 2.1 A mixture of locally linear model offers a two-way mapping, but lacks a coherent coordinate system. Nonlinear embedding offers a global coordinate system, but lacks the mapping from the global coordinate to the input space. We can map a mixture of linear subspaces into a new coordinate system to achieve an ideal two-way manifold mapping.

sponding linear subspace, resulting in different coordinate systems. For the lack of a single and coherent coordinate system, these methods are not appropriate to describe continuous motions.

Figure 2.1 depicts different approaches for nonlinear dimensionality reduction. An ideal approach should have a single coherent coordinate system and provide a two-way mapping between the input space and the low-dimensional manifold coordinate. To achieve such a mapping, we use a two-stage mapping approach through a mixture of linear subspaces. The essential idea is to align internal coordinate systems of different locally linear models into a single global coordinate system.

2.2 Global Coordination Model

Roweis et al. [6] present the first global coordination model shown in Figure 2.2. They use a mixture of factor analyzers (MFA) for the locally linear models. The joint distribution over

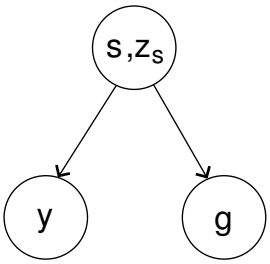


Figure 2.2 The global coordination model proposed by Roweis et al. If given s and z_s, the mappings to high-dimensional input data y and to low-dimensional global coordinate g are both linear. Since s and z_s are latent variables, the mapping between y and g is nonlinear.

observed and hidden variables in the MFA is expressed as

$$P(y, s, z_s) = P(y|s, z_s)P(z_s|s)P(s), \tag{2.1}$$

where y is the high-dimensional observed variable, s is the discrete hidden variable indexing different linear models, and the continuous hidden variable z_s is the internal coordinate of the s-th linear model. The MFA model assumes data are sampled from different linear models with prior probabilities $P(s)$, and within each component, the local coordinates of data are Gaussian: $P(z_s|s) \sim \mathcal{N}(0, I)$. The linear mapping from data to local coordinate is expressed as

$$y = \Lambda_s z_s + \mu_s + u_s, \tag{2.2}$$

where Λ_s is the transformation matrix, μ_s is the mean, and $u_s \sim \mathcal{N}(0, \Psi_s)$ is the noise term. Conventional learning algorithms for the MFA model estimate parameters $\{\Lambda_s, \mu_s, \Psi_s, P(s)\}$ without encouraging models that align their local coordinates. Therefore, when traversing a continuous path on the manifold in the high-dimensional space, the internal representations in the MFA model can change unpredictably.

Assume that there is a global coordinate g that parameterizes the manifold everywhere. Based on the graphical model in Figure 2.2, we have

$$P(g, s, z_s) = P(g|s, z_s)P(z_s|s)P(s). \tag{2.3}$$

We further assume that the global coordinate is a linear mapping from local coordinates with the transformation matrix A_s, the mean κ_s, and the noise[1] $v_s \sim \mathcal{N}(0, \Phi_s)$ as

$$g = A_s z_s + \kappa_s + v_s. \tag{2.4}$$

From (2.1) to (2.4), we have

$$P(y|s, z_s) \sim \mathcal{N}(\Lambda_s z_s + \mu_s, \Psi_s),$$

$$P(g|s, z_s) \sim \mathcal{N}(A_s z_s + \kappa_s, \Phi_s).$$

With z_s being integrated out, we also have

$$P(y|s) \sim \mathcal{N}(\mu_s, \Psi_s + \Lambda_s \Lambda_s^T),$$

$$P(g|s) \sim \mathcal{N}(\kappa_s, \Phi_s + A_s A_s^T).$$

The inference about the global coordinate g conditioned on an input data y can be written as

$$P(g|y) = \sum_s P(g|y, s)P(s|y), \tag{2.5}$$

[1]Noise is ignored in [6], which results in a deterministic mapping between local and global coordinates.

where

$$P(g|y,s) = \int P(g|s,z_s)P(z_s|s,y)dz_s. \tag{2.6}$$

Note that both $P(g|s,z_s)$ and $P(z_s|s,y)$ are Gaussians, so is $P(g|y,s)$. And since $P(s|y) \propto P(y|s)P(s)$ can be computed and be viewed as a weight, $P(g|y)$ is a mixture of Gaussians. That is, in spite of the linear mapping between the mixture model $\{s,z_s\}$ and the global coordinate g, the mapping between g and input data y is nonlinear.

Roweis et al. [6] propose an EM algorithm to learn the global coordination model. The algorithm penalizes disagreement on global coordinates mapped from different local models, if the likelihoods of a data point being generated from these local models are all high. However, their algorithm suffers from inefficient training and serious local minima problems.

2.3 Two-Stage Learning of Global Coordination Model

Instead of using the EM algorithm proposed by Roweis et al. [6] to learn the global coordination model, we adopt post-coordination methods proposed by Teh [10] and Brand [11], where global coordination is invoked after conventional algorithms are used to learn the mixture of locally linear models. We call such an approach a two-stage learning algorithm because it sequentially performs two tasks: mixture model learning and local model alignment.

2.3.1 Learning Mixtures of Probabilistic PCA

Unlike conventional PCA that does not correspond to a probability density, we use probabilistic PCA (PPCA) [9] which is formulated within a maximum likelihood framework. For

complex input data, we use a mixture of PPCA model (MPPCA) whose parameters can be estimated by using an EM algorithm.

For a PPCA model, the observed data y is mapped to a latent variable z as

$$y = Wz + \mu + \varepsilon, \tag{2.7}$$

where $y \in \mathbb{R}^q$, $z \in \mathbb{R}^d$, $q > d$, W is a linear projection matrix, μ is the mean of observation y, and $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ is an isotropic noise. Therefore, we have

$$y \sim \mathcal{N}(\mu, \sigma^2 I + WW^T),$$

and

$$P(y) = (2\pi)^{-q/2}|C|^{-1/2}\exp\{-\frac{1}{2}(y-\mu)^T C^{-1}(y-\mu)\}, \tag{2.8}$$

where $C = \sigma^2 I + WW^T$. For a mixture model with S PPCA models, we have

$$P(y) = \sum_{s=1}^{S} \pi_s P(y|s), \tag{2.9}$$

where $P(y|s)$ is the s-th PPCA model as (2.8) and π_s is the corresponding mixing parameter with $\pi_s \geq 0$ and $\sum_s \pi_s = 1$.

To obtain model parameter $\{\pi_s, \mu_s, W_s, \sigma_s^2\}$, we can maximize the log-likelihood of observed data y:

$$\mathcal{L}(y) = \sum_{n=1}^{N} \ln(\sum_{s=1}^{S} \pi_s P(y_n|s)), \tag{2.10}$$

where N is the number of observed data. Details of EM learning of MPPCA model parameters can be found in [9]. Practically, the algorithm converges to a good estimate of model parameters within 15 to 30 iterations, depending on data complexity. A rough initialization usually leads to a good parameter estimation and reduces the number of iterations.

2.3.2 Local Model Alignment of MPPCA

After learning a MPPCA model, we use a post-coordination method proposed in [10] to align its local PPCA models, which results in a global coordination model. Given a learned mixture of S PPCA models, for each data point y_n, the s-th PPCA has a d-dimensional internal coordinate z_{ns} for y_n and an associated responsibility r_{ns}, where $r_{ns} = P(y_n|s)$ and $\sum_s r_{ns} = 1$. We assume there is a linear mapping between local representations and global coordinates, with linear projection L_s and mean l_s^0. The global coordinates g_n are defined as the weighted sum of the projections by each local model:

$$\begin{aligned} g_n &= \sum_s r_{ns} g_{ns} = \sum_s r_{ns}(L_s z_{ns} + l_s^0) \\ &= \sum_s \sum_{i=0}^{d} r_{ns} z_{ns}^i l_s^i = \sum_j u_{nj} l_j, \end{aligned} \tag{2.11}$$

where l_s^i is the i-th column of L_s, z_{ns}^i is the i-th entry of z_{ns}, and $z_{ns}^0 = 1$. After vectorizing index pair (i, s) into a single index j and defining matrix U as $u_{nj} = r_{ns} z_{ns}^i$ and j-th row of L as $l_j = l_s^i$, we can write a linear equation system

$$G = UL, \tag{2.12}$$

where $j = j(i, s)$, $u_{nj} = r_{ns} z_{ns}^i$ and $l_j = l_s^i$, with fixed U and unknown L.

To determine L, we need to minimize a cost function that incorporates the topological constraints that govern g_n. Hence, the cost function is selected based on LLE's idea [8]: preserving the same neighborhood structure between the high-dimensional input space and the low-dimensional embedding. For each data point y_n, we denote its nearest neighbors as y_m ($m \in \mathcal{N}_n$) and minimize

$$\mathcal{E}(Y, W) = \sum_n \parallel y_n - \sum_{m \in \mathcal{N}_n} w_{nm} y_m \parallel^2 \tag{2.13}$$

with respect to W subject to $\sum_{m \in \mathcal{N}_n} w_{nm} = 1$. The weights w_{nm} are unique and can be estimated by constrained least squares. These weights represent the locally linear relationships between y_n and its neighbors. Accordingly, we define a similar cost function

$$\begin{aligned} \mathcal{E}(G, W) &= \sum_n \parallel g_n - \sum_{m \in \mathcal{N}_n} w_{nm} g_m \parallel^2 \\ &= \text{trace}(G^T (I - W^T)(I - W) G) \\ &= \text{trace}(L^T A L) \end{aligned} \tag{2.14}$$

with respect to G, where $A = U^T (I - W^T)(I - W) U$. Since $\mathcal{E}$ is invariant to translations and rotations of G, and $\mathcal{E}$ scales as G is scaled, we define the following two constraints

$$\frac{1}{N} \sum_n g_n = \frac{1}{N} \overrightarrow{1}^T G = \frac{1}{N} \overrightarrow{1}^T U L = 0 \tag{2.15}$$

and

$$\frac{1}{N} \sum_n g_n g_n^T = \frac{1}{N} G^T G = L^T B L = I_d, \tag{2.16}$$

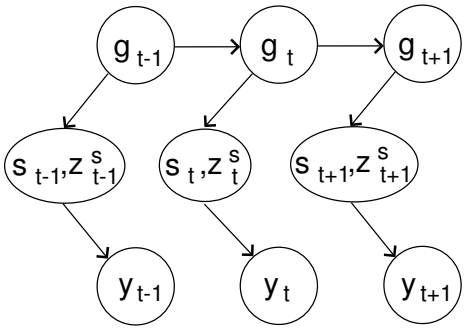


Figure 2.3 The graphical model of our dynamic global coordination model (DGCM).

where $B = \frac{1}{N}U^T U$. Now that the cost function (2.14) and the constraint (2.16) are both quadratic, we can determine the optimal L, without local minima problems, by solving generalized eigenvalue system $Av = \lambda Bv$ subject to $\frac{1}{N}\overrightarrow{1}^T UL = 0$. The solution for L is the matrix with its columns formed by the second to $(d+1)$-th smallest generalized eigenvectors.

Note that the entire learning algorithm does not involve temporal information associated with the input data. Since we limit the input data of concern to be temporally continuous, the learning algorithm ensures that the temporal neighbors of an input data y_t are among the spatial neighbors of its global coordinate g_t. Therefore, a sequence of temporally continuous input data corresponds to a continuous trajectory of low-dimensional coordinates on the manifold.

2.4 Dynamic Global Coordination Model

If we model the temporal dynamics of our intrinsic parameters using the Markovian assumption, for intrinsic parameters g_t, we will have $P(g_t|g_{t-1}, \ldots, g_1) = P(g_t|g_{t-1})$. With this assumption, we can concatenate the global coordination model as a dynamic Bayesian network as shown in Figure 2.3. We call this dynamical model a dynamic global coordination model (DGCM).

2.4.1 Comparisons to Other Dynamical Models

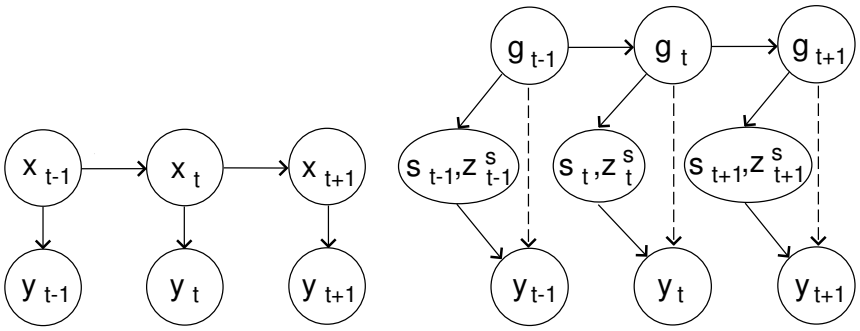
Our dynamic global coordination model has a similar structure to a linear dynamical system (LDS) as illustrated in Figure 2.4(a). Compared to a LDS, we replace the hidden state variable x with global coordinate g, and the inference from g to the input data y is nonlinear through the mixture model $\{s, z\}$. Since latent variables (s_t, z_t^s) are temporally independent, they can be marginalized out at each time t as shown in Figure 2.4(b).

A general switching LDS (SLDS) [12] illustrated in Figure 2.4(c) consists of multiple linear dynamical systems and a transition probability $P(s_t|s_{t-1})$ to select the current local model. Our dynamical model is very similar to a SLDS since both are based on a mixture of locally linear models. The difference is that our state variable s_t is temporally independent, and we switch local models depending on state g in the globally-coordinated space. Therefore, our dynamical model ensures data continuity. To ensure data continuity in SLDS, transition constraints between locally linear models have to be posed [13].

2.5 Model Validation

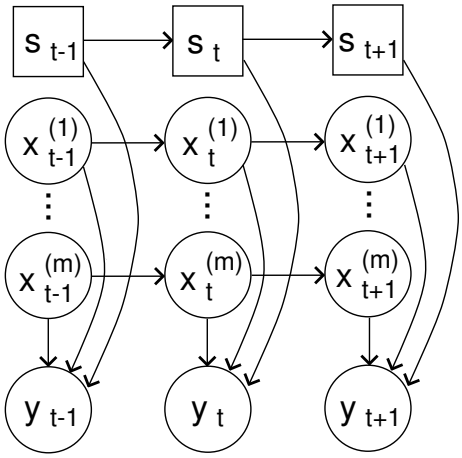
Before evaluating the proposed model in different application scenarios in following chapters, we will validate the capability of the model to characterize continuous, locally-linear multi-dimensional data.

We validate the model by analyzing images acquired from a video where the pose of a human face changes over time. Figure 2.5 shows some selected frames of the video. We use a baseline appearance based tracker to crop out face images and scale these images to 19 by 19 pixels. The pose variations are large enough so that the baseline tracker loses track quickly. Therefore, we restart the tracker every 50 frames and obtain $2,200$ frames of face images. That is, we use



(a) LDS

(b) DGCM



(c) SLDS

Figure 2.4 The graphical models of three dynamical models.

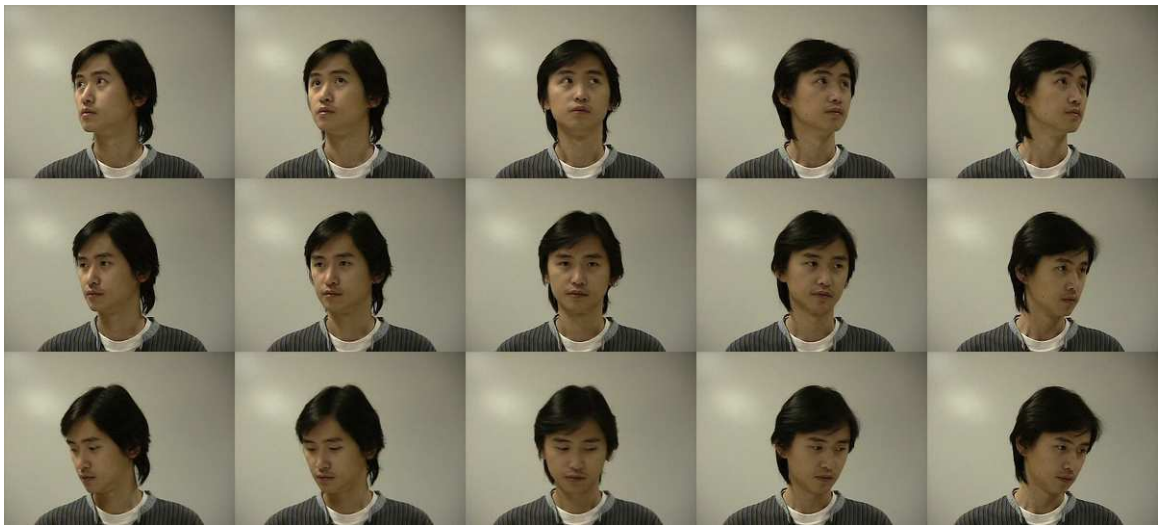


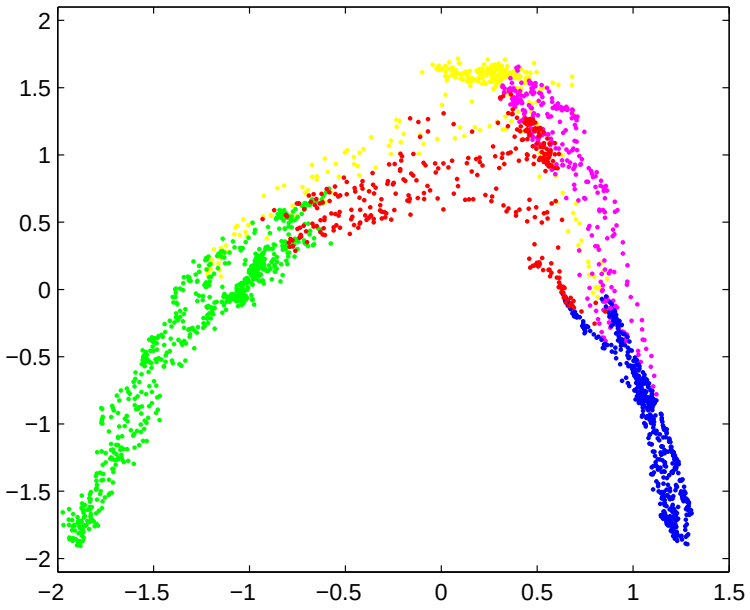
Figure 2.5 Selected frames of the face video used to construct a globally-coordinated mixture of linear subspaces for the face.

2,200 time-varying, 361-dimensional image vectors in the experiment. Note that the cropped face images do not well align, compared to the hand-cropped training images used by most of the face-related research, which makes this sequence of face images even more challenging.

We learn a two-dimensional global coordination model using these cropped face images with a mixture of five 13-dimensional probabilistic PCA (PPCA) models. To obtain a better mixture model, we initialize the learning by hand-picking five images corresponding to the face looking forward, upward, downward, left, and right, and performing single PPCA learning for 300 closest images for each hand-picked image. After learning the mixture model, the mean images of the five local models are shown in Figure 2.6(a). Using the face images y and their corresponding local coordinates $\{s, z_s\}$ in the mixture of PPCA models, we learn the mapping between local latent variables $\{s, z_s\}$ and the two-dimensional globally-coordinated intrinsic variable g based on the algorithm described in Section 2.3.2.

The learned two-dimensional global coordinates of the face images are depicted in Figure 2.6(b). The color of each point denotes the assigned cluster label of the mixture of PPCA

(a) The mean faces of the five local PPCA models.

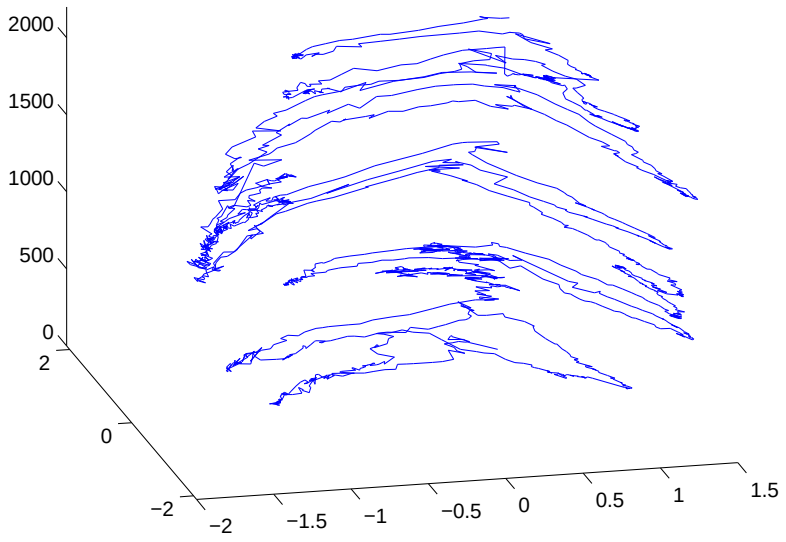


(b) Two-dimensional globally-coordinated mixture of five local PPCA models.

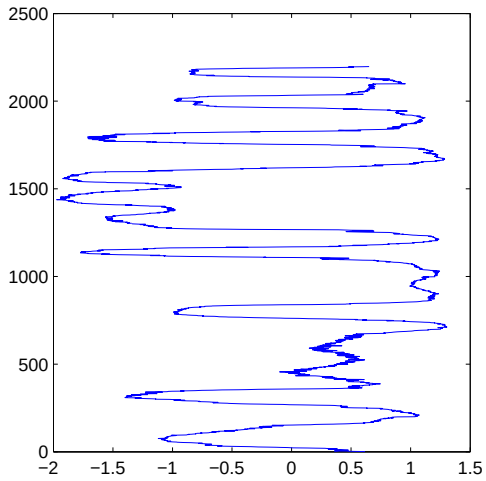
Figure 2.6 (a) The means of the five learned local PPCA models corresponding to different image clusters. The five models (from left to right) correspond to faces that look upward, right, forward, left, and downward. (b) The two-dimensional global coordinates of the cropped face images. Points of the same color are from the same local PPCA model. Models from left to right in (a) associate with colors yellow, green, magenta, blue and red, respectively.

models, which is determined by comparing $P(s|y)$. The colors assigned to the PPCA models, from left to right in Figure 2.6(a), are yellow, green, magenta, blue and red, respectively. It shows that the property of local linearity in input images is well captured in the mixture model. We also notice that the three PPCA models, corresponding to looking upward, forward and downward, are collapsed together in the globally-coordinated space. This is because we have to scale face images to a same size for learning, and the images corresponding to the these three models, in particular, become more similar. It can be further verified by comparing their means in Figure 2.6(a) (the first, third and fifth images).

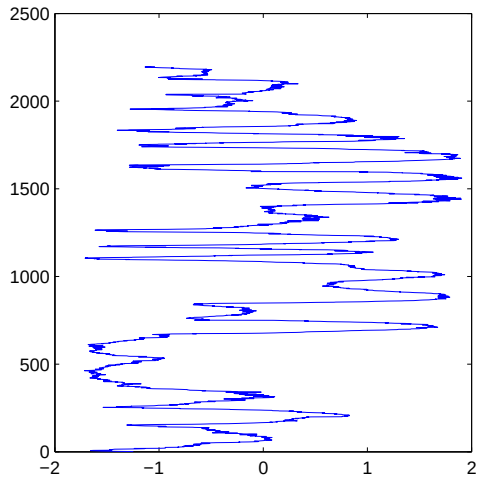
Next, we validate that the model does capture the continuity properties of the input data. To help visualize the dynamics in the globally-coordinated space, we link global coordinates of successive face images with line segments and add a time axis to the two-dimensional globally-coordinated space, resulting in Figure 2.7. It shows that the continuous face motion does correspond to a continuous trajectory in the two-dimensional globally-coordinated space, thus validates our earlier assumption.



(a) X-Y-T view



(b) X-T view



(c) Y-T view

Figure 2.7 Continuous face motion as a continuous, low-dimensional trajectory in the globally-coordinated space.

CHAPTER 3

DYNAMIC GLOBAL COORDINATION MODELS OF DYNAMIC OBJECT APPEARANCE WITH APPLICATION TO OBJECT TRACKING

In this chapter, we present a dynamic inference algorithm for object tracking using the dynamic global coordination model that we present in Chapter 2. We assume that the object undergoes a complex motion which involves large variations in object appearance, so a mixture of linear appearance models is better to capture such large variations than a single linear model. Unlike other nonlinear models such as a SLDS that has been employed for visual tracking, our dynamical model captures complex motion as a continuous trajectory in a low-dimensional globally-coordinated space. Since our model has a similar structure to a LDS, we are also able to derive an efficient approximate inference algorithm for our dynamic global coordination model. We track the object using a Rao-Blackwellized particle filter to integrate out part of the latent variables in our dynamical model and reduce the dimensionality needed for particle sampling. Experimental results demonstrate the improved performance of the proposed model on face tracking in scenes with significant clutters and temporary occlusions which pose difficulties for other tracking methods.

3.1 Introduction

The objective of this work is to develop a model for object tracking on nonlinear appearance manifolds. An appearance manifold refers to a collection of raw images of an object taken under different viewing parameters (e.g. varying poses and illuminations) forming a continuous set of points in high-dimensional space [14]. When the viewing parameters contain large variations, the appearance manifold is likely to be nonlinear. Although appearance images of an object are in a high-dimensional space, these images usually lie on or near a low-dimensional manifold embedded in the input space. Therefore, given a set of appearance images, we can learn the intrinsic structure of their appearance manifold. Since we are interested in temporally continuous motions, tracking an object can be done by tracking a continuous trajectory on the appearance manifold.

As mentioned in Chapter 2, manifold learning is a dimensionality reduction problem, and there are two main types of algorithms for nonlinear dimensionality reduction. Mixtures of linear subspace methods are generative models with good probabilistic interpretation. A problem with subspace methods is that when mixtures of linear subspaces are used, parameters in each subspace are expressed using respective local coordinates. By contrast, nonlinear embedding methods map the input images into a low-dimensional space where there is a global coordinate system describing the spatial relationships among any two images. However, their mappings do not preserve appearance information of each image.

For applications in visual tracking, both methods possess attractive properties. Mixture of linear subspace methods are good candidates for measurement functions, while nonlinear embedding methods are good for modeling appearance dynamics in a low-dimensional coordinate system.

In this work, we use the global coordination model to represent appearance images, which retains the advantage of both mixture of linear subspaces and nonlinear embedding approaches. This generative model allows us to map to and from appearance images and their intrinsic structures in a low-dimensional, globally-coordinated space. We extend the model to a dynamic Bayesian networks (DBN) for object tracking. Due to the complexity of our model, exact inference on our DBN is intractable, so an efficient approximate inference algorithm is proposed. We also show that the inference can be carried out by using a Rao-Blackwellized particle filter which facilitates efficient tracking.

3.2 Related Work

Tracking moving objects based on their appearance has been an ongoing research area in computer vision [15, 16]. Early work on appearance based tracking use appearance models mainly as a measurement function [17, 18]. The appearance dynamics are not taken into account in their tracking algorithms. Recently, Khan et. al. [19] present an approach with appearance parameters included in their dynamical model. They use the probabilistic PCA (PPCA) [20] model to reduce the dimensionality of appearance images, and apply Kalman filter to model the dynamics of the latent variables of PPCA. Their model fits nicely into a Rao-Blackwellized particle filter for efficient tracking. Comparing with Khan's method, our approach uses a mixture of PPCA models to better model the appearance variations. A mixture model breaks down the Gaussian property, making it difficult to directly apply a Rao-Blackwellized particle filter to tracking problems.

There has been some work that tries to obtain globally-coordinated latent variables of mixture models and use those variables for object tracking [21]. But most of them are based on

switching linear dynamical systems (SLDS). Some SLDS-based tracking approaches use only a single linear subspace, but with switching linear dynamics [22]. Our globally-coordinated DBN is different from these approaches because we track the latent variables directly without resorting to any state switching models. As a result, our model is capable of more accurately describing the actual underlying appearance dynamics as a continuous process.

3.3 Tracking on a Globally-coordinated Appearance Manifold

In Chapter 2, we present our generative model and explain the graphical inference between appearance images and their corresponding globally-coordinated intrinsic parameters in a low-dimensional space. With this inference method, given a sequence of appearance images, we can infer intrinsic parameters of these images and track appearance variations using these parameters. Because the spatial relationships among appearance images are preserved in this intrinsic globally-coordinated space, a continuous appearance variation is analogous to a continuous trajectory in this space. This property allows us to use simple dynamics models, such as Brownian motion, to describe appearance dynamics.

3.3.1 Complete Dynamic Global Coordination Model for Object Tracking

In this work, we predict an object's 2D location and appearance over time. The object in each image is circumscribed in a bounding box which is described by its center and its width and height. The appearance of the object is captured by the image content in the bounding box. As described in Section 2.4, we extend the global coordination model to a dynamic

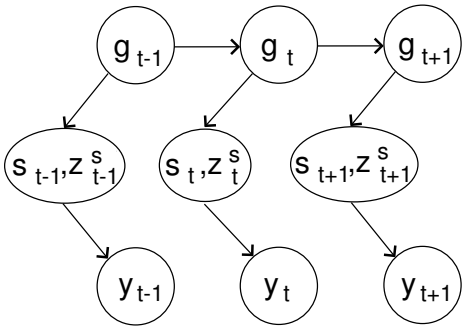


Figure 3.1 The graphical model of our dynamic global coordination model.

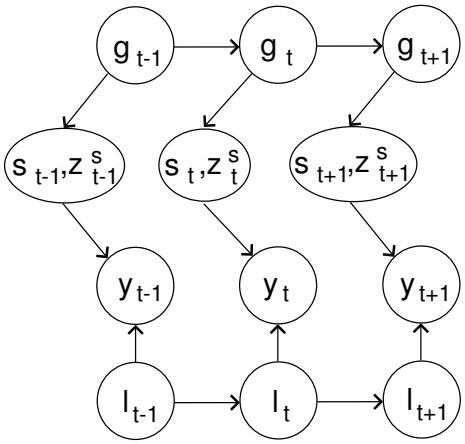


Figure 3.2 Our complete dynamic global coordination model for tracking the appearance, 2D location and scale of an object in a video.

global coordination model as in Figure 3.1 to track the continuous appearance change of an object. However, this graphical model does not take into account the 2D location (center of the bounding box) and the scale (width and height of the bounding box) of the tracked object in the video. Let l_t denote the location variable describing the 2D location and the scale of the tracked object at video frame t. By including the location variables, our graphical model becomes the one shown in Figure 3.2, where inference on this dynamical model is nontrivial.

3.4 Dynamic Inference for Dynamic Global Coordination Model

For the dynamic inference for our dynamic global coordination model, we need to estimate the posterior distribution $P(g_t|y_{1:t})$. We will show that the probability distribution $P(g|y)$ can be approximated as a Gaussian in spite of its formulation in a mixture model. As a result, the posterior distribution can be recursively estimated by using a Bayes filter as

$$P(g_t|y_{1:t}) = \kappa P(y_t|g_t) \int P(g_t|g_{t-1}) P(g_{t-1}|y_{1:t-1}) dg_{t-1}. \tag{3.1}$$

In Equation (3.1), if we have $P(g_t|y_t)$, $P(y_t|g_t)$ can be computed using the Bayes rule:

$$P(y_t|g_t) = \frac{P(g_t|y_t)P(y_t)}{P(g_t)}. \tag{3.2}$$

And because a continuous object motion corresponds to a continuous trajectory in the globally-coordinated space, we can use simple models, such as a Brownian motion model, to capture the dynamics of g_t.

Exact inference on our graphical model is impossible since the probability distribution $P(g|y)$ is a mixture of Gaussians as shown in Equation (2.5). In our dynamical model, as time progresses, the number of model parameters in the distribution $P(g_t|y_{1:t})$ increases exponentially, which makes exact inference intractable. Including the location variables further complicates the inference problem, since $P(g_t, l_t|y_t)$ is unlikely to be any analytical parametric distributions.

Under such circumstances, using a particle filter is a good option for approximate inference for our graphical model. However, directly applying particle filter to our model is infeasible. Since our latent variables contain (g_t, l_t), a large number of particles are needed to approximate probability distributions in this joint latent variable space. Nevertheless, if we can integrate

out part of the latent variables to reduce the number of dimensionality needed for sampling, particle filter approach can be incorporate into our dynamical model, which is the central idea of a Rao-Blackwellized particle filter.

3.4.1 Approximate Inference

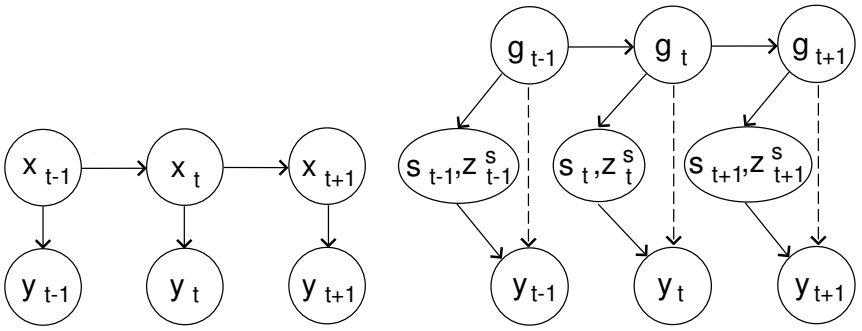
Ideally, the mapping between y and g should be a one-to-one mapping so that when Gaussian noise is added, the distribution $P(g|y)$ would be a Gaussian. When learning the global coordination model, we do impose constraints to preserve locally spatial relationship between appearance images (see Equation (2.14)). Therefore, even though distribution $P(g|y)$ is a mixtures of Gaussians, the overall distribution should still be a Gaussian. This observation motivates us to approximate $P(g|y)$ using a single dynamic Gaussian.

Denote (μ_t, Σ_t) as the mean and the covariance matrix of the Gaussian that we use to approximate $P(g|y)$, and denote (μ_t^s, Σ_t^s) as the mean and the covariance matrix for Gaussian distribution $P(g|y, s)$. From Equation (2.5), we re-write $P(g|y)$ as

$$P(g|y) = \sum_s \mathcal{N}(g; \mu_t^s, \Sigma_t^s) P(s|y).$$

To ensure $\mathcal{N}(\mu_t, \Sigma_t)$ well approximate $P(g|y)$, we minimize the weighted KL-distance between $\mathcal{N}(\mu_t, \Sigma_t)$ and $P(g|y, s)$ of different local models:

$$(\mu_t, \Sigma_t) = \arg\min_{\mu,\Sigma} \sum_s P(s_t|y_t) KL(\mathcal{N}(\mu_t^s, \Sigma_t^s) || \mathcal{N}(\mu, \Sigma)). \tag{3.3}$$



(a) Kalman filter model (b) DGCM

Figure 3.3 When $P(g|y)$ is approximated as a Gaussian, our dynamic global coordination model is like a Kalman filter. But our measurement model $P(y_t|g_t)$ changes over time, in contrast to a fixed measurement model $P(y_t|x_t)$ in a Kalman filter model.

The solution of (μ_t, Σ_t) can be written as

$$\begin{aligned} \mu_t &= \sum_s P(s_t|y_t)\mu_t^s, \\ \Sigma_t &= \sum_s P(s_t|y_t)(\Sigma_t^s + (\mu_t - \mu_t^s)(\mu_t - \mu_t^s)^T). \end{aligned} \tag{3.4}$$

With $P(g|y)$ being approximated as a Gaussian, our dynamic global coordination model is like a Kalman filter. However, unlike a Kalman filter that has a fixed measurement model $P(y_t|x_t)$ representing the probability of observing image y_t given hidden state x_t, our measurement model $P(y_t|g_t)$ changes dynamically.

3.4.2 Rao-Blackwellized Particle Filter

In a Rao-Blackwellized Particle Filter (RBPF), part of the latent variables are integrated out and are represented by an analytical distribution. The likelihood $P(g_t, l_t|Y_t)$ can be decomposed

as

$$P(g_t, l_t|Y_t) = P(g_t|l_t, Y_t)P(l_t|Y_t) \tag{3.5}$$

where $Y_t = \{y_1, y_2, \ldots, y_t\}$. We have shown that $P(g_t|l_t, Y_t)$ is an analytical distribution in our dynamical model by dynamically approximating $P(g_t|y_t)$ as a Gaussian. We can also apply a particle filter to approximate the distribution of $P(l_t|Y_t)$. At the same time, we expand the particle filter to become three tuples so as to include the analytical distribution $P(g_t|l_t, Y_t)$. That is, a Rao-Blackwellized particle filter is a set of N particles $\{(s^{(i)}, w^{(i)}, \alpha^{(i)}(g))\}_{i=1}^{N}$, where $s^{(i)}$ is a sample of l, $w^{(i)}$ is its associated weight and distribution $\alpha^{(i)}(g)$ corresponds to $P(g_t|l_t, Y_t)$. In a Rao-Blackwellized particle filter, the likelihood $P(g_t|l_t, Y_t)$ is approximated as

$$P(g_t, l_t|Y_t) \approx \sum_i w^{(i)}\delta(s^{(i)})\alpha^{(i)}(g). \tag{3.6}$$

According to the Bayes filter, given a sequence of observations Y_t, we estimate the current object location and scale as

$$P(l_t|Y_t) = \kappa \int_{g_t} P(Y_t|g_t, l_t) \int_{l_{t-1}} \int_{g_{t-1}} P(g_t, l_t|g_{t-1}, l_{t-1})P(g_{t-1}, l_{t-1}|Y_{t-1}). \tag{3.7}$$

In our model, we assume the dynamics of g and l are independent. That is,

$$P(g_t, l_t|g_{t-1}, l_{t-1}) = P(g_t|g_{t-1})P(l_t|l_{t-1}). \tag{3.8}$$

We also use Brownian motion models for both dynamics:

$$P(g_t|g_{t-1}) \sim \mathcal{N}(g_{t-1}, Q), \tag{3.9}$$

$$P(l_t|l_{t-1}) \sim \mathcal{N}(l_{t-1}, R), \tag{3.10}$$

where Q and R are predefined covariance matrices.

With all the model parameters being defined, our Rao-Blackwellized particle filter is described as Algorithm 3.1.

Algorithm 3.1 A Rao-Blackwellized particle filter for tracking using DGCM

Starting with particles $\{(s_{t-1}^{(i)}, w_{t-1}^{(i)}, \alpha_{t-1}^{(i)}(g_{t-1}))\}_{i=1}^{N}$:

1. Re-sample the particles from $\{s_{t-1}^{(i)}\}_{i=1}^{N}$ according to $\{w_{t-1}^{(i)}\}_{i=1}^{N}$, and denote the new selected particles as $\{s_t^{(i)}\}_{i=1}^{N}$.
2. Drift $\{s_t^{(i)}\}_{i=1}^{N}$ according to $P(l_t|l_{t-1})$.
3. Update distributions $\alpha_{t-1}^{(i)}(g_{t-1})$ to $\{\alpha_t^{(i)}(g)\}$ according to $P(g_t|g_{t-1})$ and $P(y_t|s_t^{(i)}, g_t)$:

$$\alpha_t^{(i)}(g) = \kappa^{(i)} P(y_t|s_t^{(i)}, g_t) \int P(g_t|g_{t-1}) \alpha_{t-1}^{(i)}(g_{t-1}) dg_{t-1} \tag{3.11}$$

4. Set $w_t^{(i)} = \kappa^{(i)}$.

3.5 Experiments

We test our globally-coordinated dynamical model by tracking human faces. To collect training images, we take a video of a face undergoing large pose variations in a clear, white background as shown in Figure 2.5. As described in Section 2.5, we obtain $2,200$ face images by interactively using a baseline tracker. These face images are then scaled to the same size (19 by 19 pixels) and used to learn a two-dimensional globally-coordinated space.

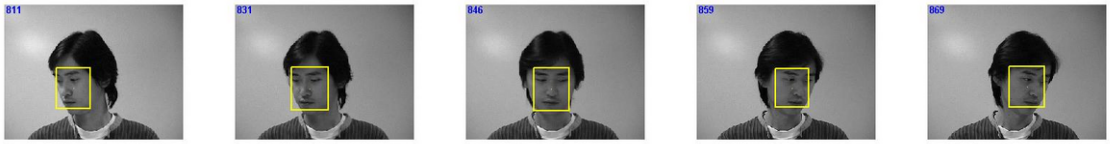


Figure 3.4 Tracking results of frame 811, 831, 846, 859, 869 in the training video.

3.5.1 Tracking Performance Evaluation

Before demonstrating our tracker in more challenging scenarios, we first evaluate its tracking performance with a basic setup. We choose to track the training video because we can take the known data and parameters (2D locations and scales of the bounding window and 2D global coordinates as face poses) as the ground truth and assess the tracking performance by comparing our tracking results to the the ground truth.

For this experiment, instead of using the mixture model that we learned in Chapter 2, we learn a new mixture of five 15-dimensional PPCA models largely correspond to different poses from left to right. We initialize a tracking window around the face in the first frame of the training video using the ground-truth data. We use 500 particles to track the face in the remaining video frames. Figure 3.4 shows tracking results from some selected frames.

There are two variables in our graphical model (Figure 3.2) that infer the observed face image y in the entire image: location and scale of the tracking window (l) and appearance parameters in the globally-coordinated space (g). Therefore, we evaluate the accuracy of our tracker in terms of the similarity in tracked regions and appearance parameters between our tracker and the ground truth.

The location parameter in our model represents the configuration of the tracking window as $l = (x, y, w, h)$, where (x, y) denotes the center of the window and (w, h) denotes the width and height of the window. Note that it is trivial in our model to include rotation and skew

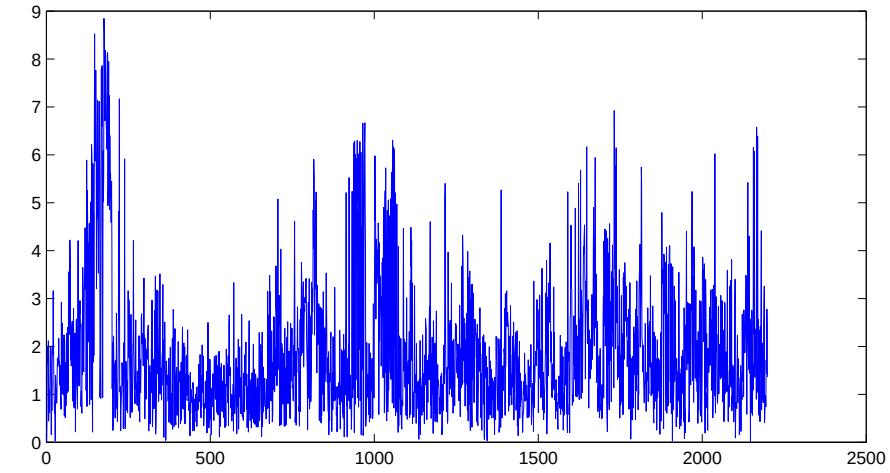
parameters to configure the tracking window in that it only requires a larger number of particles to be sampled in a larger parameter space.

Let $l^T = (l_x^T, l_y^T, l_w^T, l_h^T)$ and $l^G = (l_x^G, l_y^G, l_w^G, l_h^G)$ denote the tracking window and ground-truth region, respectively. We use L^2 distance $S_d = \sqrt{(l_x^T - l_x^G)^2 + (l_y^T - l_y^G)^2}$ and area ratio $S_a = (l_w^T l_h^T)/(l_w^G l_h^G)$ to measure the similarity between the tracking window and ground-truth region. These measurements are plotted in Figure 3.5 over time. The 2D location deviations are usually very small and the area ratios between the tracking region and ground-truth region are usually around one, which shows that our tracker follows the target very well.

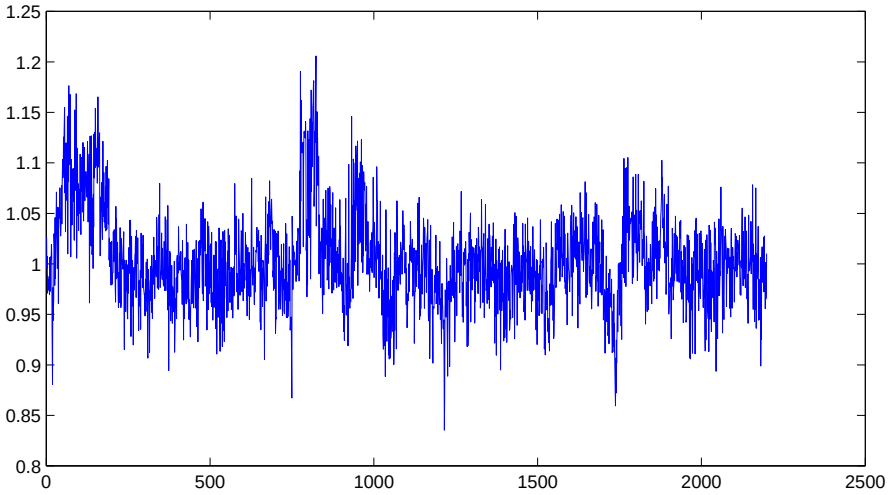
In Figure 3.6 we plot both tracked trajectory and ground-truth trajectory for two short periods of about two seconds for each. Figure 3.7 shows both trajectories in each dimension vs time and Figure 3.8 displays the L^2 distance between the tracked pose and the ground truth in the globally-coordinated space at each image frame. For the most of time, our tracker is very accurate in pose estimate. We also observe that the ground-truth trajectories are smoother and our track sometimes over-shoots. It is mainly because we define a larger covariance Q in Equation (3.9), which is necessary to track new video sequences since they might contain faster face movement.

3.5.2 Face Tracking in the Presence of Other Faces

Our second experiment tests our tracker by tracking a target face in a cluttered background where multiple faces are present in the scene. All faces in the images undergo obvious pose variations. We initialize a tracking window of the target face in the first frame of the test video, and use 500 particles to track target face in the remaining image frames. We use the

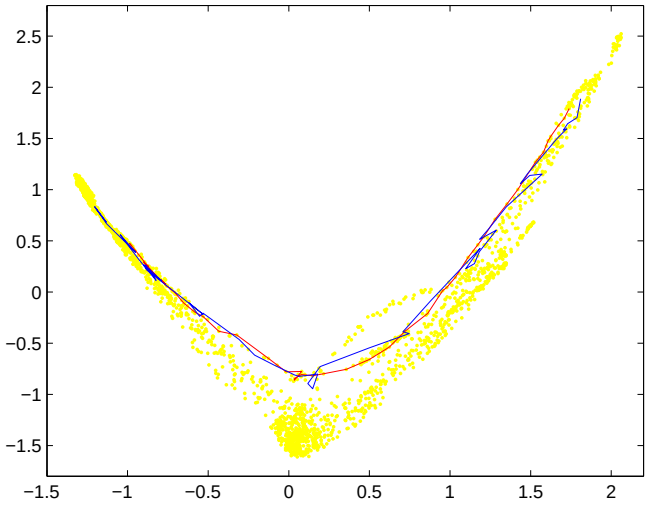


(a) 2D location deviation

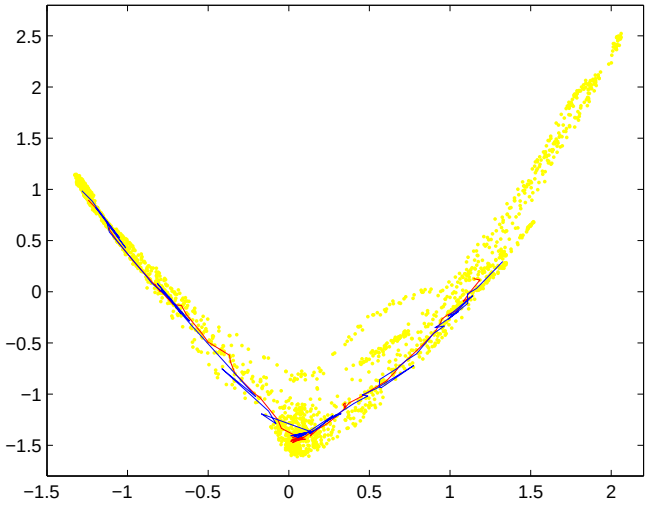


(b) Area ratio

Figure 3.5 (a) 2D location deviation of the tracking window from the ground-truth region measured in pixels over time. (b) Area ratio of the tracking window over ground-truth region over time.

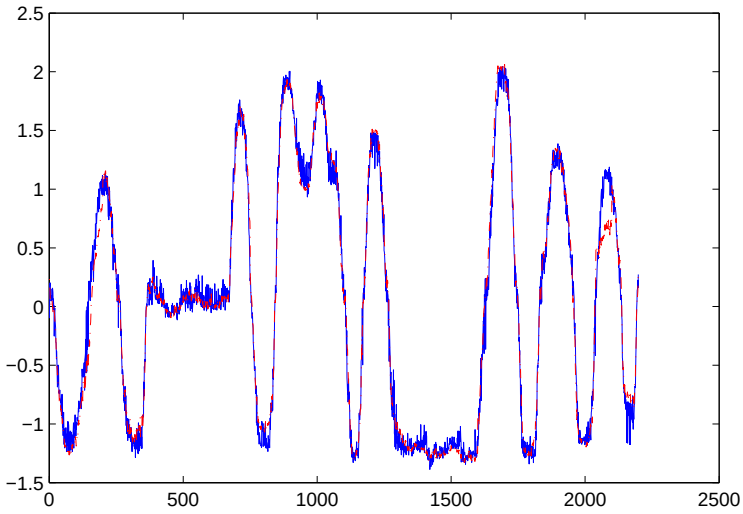


(a) Frame 801 to 870

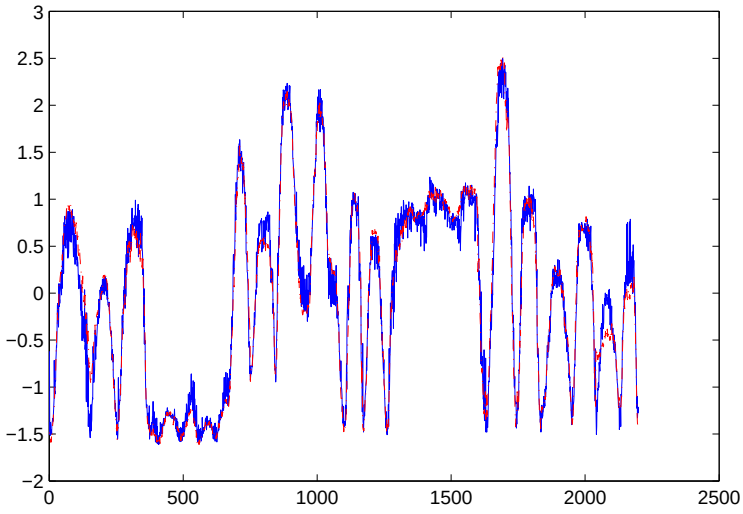


(b) Frame 1071 to 1130

Figure 3.6 The trajectory plots in the globally-coordinated space during two short periods. The red trajectories are the ground-truth trajectories, while the blue ones are tracked by our tracker.



(a) X-T view



(b) Y-T view

Figure 3.7 The trajectory vs time plots in X-T and Y-T views. The red trajectories are the ground-truth trajectories, while the blue ones are tracked by our tracker.

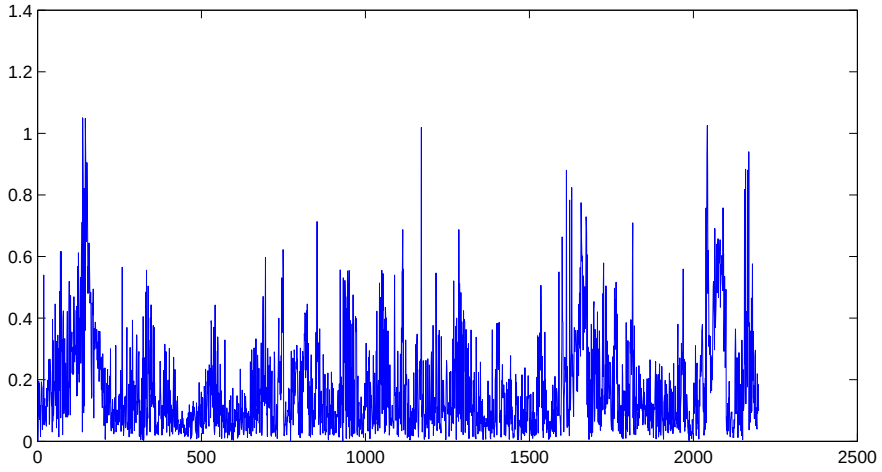


Figure 3.8 Deviation of tracked pose against ground truth measured by L^2 distance in the globally-coordinated space over time.

globally-coordinated space that we learn in Chapter 2. Note that the environments of training scene and test scene are different.

Although multiple similar objects moving nearby makes tracking a challenging problem, our tracker tracks the target face very well. Figure 3.9 shows several snapshots of our tracking process. Below each image frame we show the learned globally-coordinated space with the projection of the tracked face at each step. In the bottom, the yellow dots are the projected training face images. The green dots represent the tracking trajectory till ten frames back, and the blue dots show the path of the most recent ten frames.

Our graphical model for visual tracking provides extra robustness for trackers because we track object position as well as its pose (appearance coefficients). The current pose estimation becomes a strong prior while tracking the next frame and prevents our tracker from being distracted by other objects with similar appearances. Some other trackers may perform well, but they do not simultaneously provide positional and pose information at each frame.

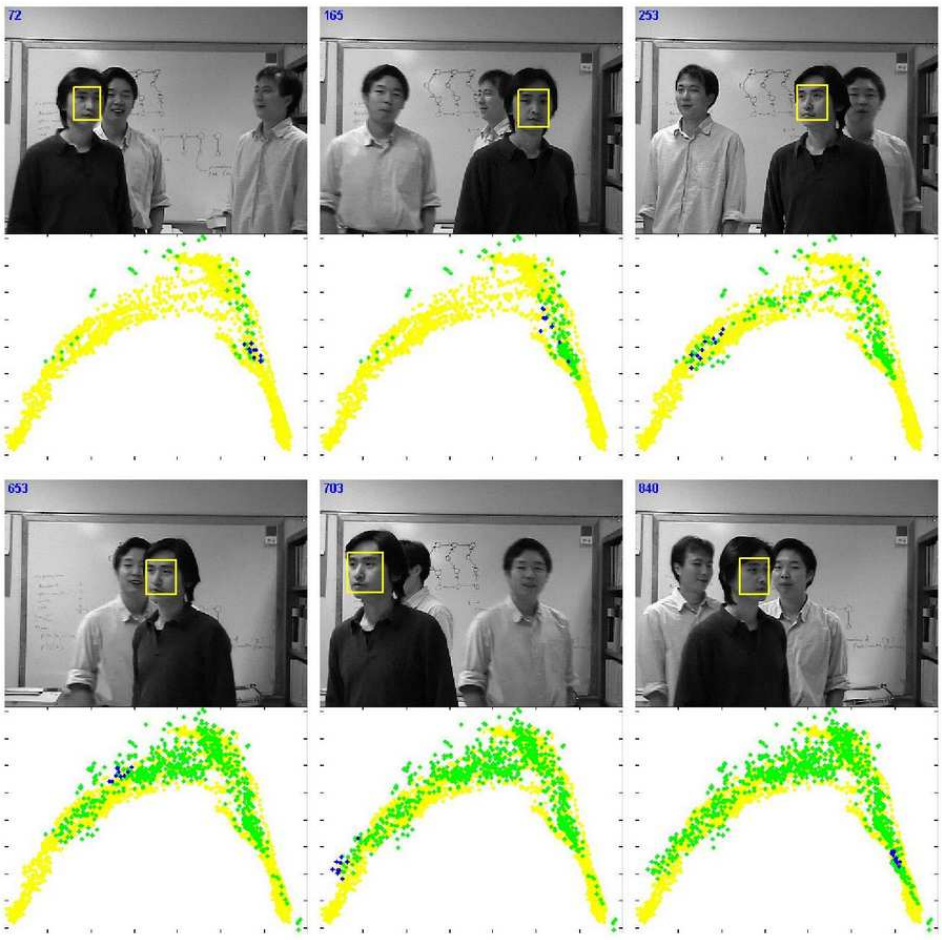


Figure 3.9 We model the appearance manifold of a face using a mixture of five globally-coordinated linear subspaces. We demonstrate the use of the model by tracking a face in the presence of other faces. In each sub-figure, the top half shows our tracking window. The bottom half shows the intrinsic coordinates of input face images (yellow) and the tracked poses (green, and blue for the last 10 image frames) in the globally-coordinated space.

3.5.3 Face Tracking with Temporary Occlusion

Our third experiment examines the robustness of our tracker with temporary occlusion. In the current implementation, we use a threshold for the likelihoods of the particles, and reject tracking results if all particles have small likelihoods. This method works very well to detect occlusion. However, the main problem is how to find the target face when it appears in the scene again.

One way to handle this problem is to expand the ranges of searching parameters over time due to the increasing uncertainty under occlusion. Under occlusion for τ frame, we rewrite (3.5) as

$$P(g_{t+\tau}, l_{t+\tau}|Y_{t+\tau}) = P(g_{t+\tau}|l_{t+\tau}, Y_{t+\tau})P(l_{t+\tau}|l_t) \tag{3.12}$$

where t denotes the time of last observation without occlusion. Therefore, we need to drift particles in a large spatial space and update their distributions with larger covariances with increasing τ from those of time t. This means more particles are needed to find the target after occlusion.

In the current experiment, we use the same number of particles and do not change the covariances of appearance distributions because our test videos involve only short periods of occlusion and we assume the appearance of the target does not change significantly. Figure 3.10 shows our tracking results with temporary occlusion. In the test, the target face moves around the same location, and another face moves in front of the target face. When the likelihoods of all particles are small, we declare occlusion. We then sample particles in a larger space as the occlusion lasts. Notice that the second person has a very different face pose (facing to the

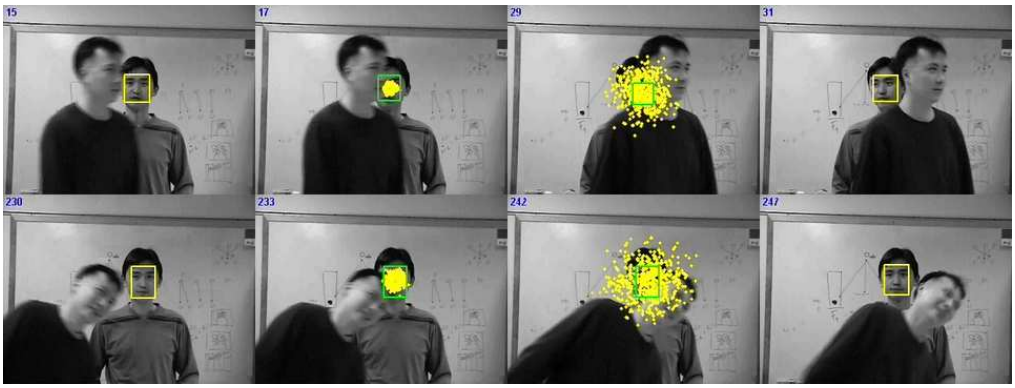


Figure 3.10 Green-colored window indicates occlusion is detected. The yellow dots show sampled particles with their associated 2D positions. The longer the occlusion lasts, the larger space the particles are sampled.

right) from the target face (facing forward). Since our approach tracks appearance coefficients, it does prevent the tracker from being distracted by the second face.

3.6 Conclusions

In this chapter, we present a dynamic inference algorithm for nonlinear appearance based object tracking using the dynamic global coordination model that we propose in Chapter 2. We also apply a Rao-Blackwellized particle filter to facilitate efficient object tracking. Our dynamical model captures continuous motion as a continuous low-dimensional trajectory. By tracking appearance on a nonlinear low-dimensional manifold in addition to the object's location and scale, the tracking performance is more robust in the presence of other similar objects.

Our main goal is to develop a generative model capable of describing the process of appearance variation over time. That is, we aim at tracking and understanding simultaneously. The tracker understands the underlying process that causes appearance changes and uses this information for robust tracking. Our experiments validate our proposed model and that the inference procedure is correct.

CHAPTER 4

DYNAMIC GLOBAL COORDINATION MODELS OF DYNAMIC MOTION DATA WITH APPLICATION TO COMPLEX MOTION SYNTHESIS

In this chapter, we formulate the problem of motion synthesis as a nonlinear manifold learning and traversing problem. The motions of interest are characterized by changes in spatial or spectral parameters. For continuous changes of such parameters, it is commonly assumed that all these parameters lie on or close to a low-dimensional manifold embedded in the original input space. For complex motions, the manifolds are usually nonlinear. We use the dynamic global coordination model as the generative model for the input data. With the nonlinear manifold being globally parameterized, we overcome motion discontinuity encountered in switching linear dynamical models. We use a nonparametric method to describe the complex dynamics of motions on the manifold. We test our approach in both spatial (motion capture data) and spectral (dynamic textures) domains. The experimental results suggest that our approach is able to synthesize smooth, complex, articulated human motion and texture motions, and has potential applications to other motion synthesis problems.

4.1 Introduction

In this chapter, we use the term *motion* to refer to the temporal changes in spatial or spectral parameters of objects or image sequences. This definition includes independent point motion, single rigid body motion, single articulated body motion, motions of multiple rigid bodies, etc. There is a broad interest in modeling changes in the parameters of different types of motions. For example, research in human motion analysis concerns temporal changes of the configurations of human body parts. These configurations include the 3D position, orientation, or rotation parameters of a hierarchical kinematic model. Also, research in dynamic texture analysis concerns temporal changes in pixel intensities in an image sequence. By complex motions, we refer to model parameters that have multi-modal distributions, which can be often seen in the real world. For instances, a person may hop, skip, jump, tiptoe, leap, etc., in a dancing sequence, and a flapping flag may exhibit various distinguishable shapes with changes in wind. In this work, we are interested in developing a general framework to model and synthesize complex motions. In particular, we present and evaluate our approach to learning and synthesis of human motions and dynamic textures, which represent the classes of articulated motions and independent point motions, respectively.

4.1.1 Previous Work on Dynamic Textures

In the literature, depending on the techniques or applications, dynamic textures are also called temporal textures, video textures, graphcut textures, and dynamic scenes. Regardless the names, their ultimate goals are the same: to provide a continuous, infinitely varying sequence of images given a finite set of images [23]. Unlike conventional 2D image textures, dynamic textures stress on temporal stationarity instead of spatial stationarity.

Beyond stationary dynamic textures, people are also interested in non-stationary dynamic textures. There are many examples of non-stationary dynamic textures. For example, a flapping flag displays different types of motions in response to changing speeds or directions of wind blows. If the configuration of the wind blow is fixed in a short period of time, the flag motion would be temporally stationary during the period. However, an observer is more likely to see a flag exhibiting a piecewise temporal-stationary motion. This observation applies to most natural phenomena.

The existing methods for dynamic texture synthesis can be categorized into two classes: parametric and non-parametric. The non-parametric methods are mostly extended from techniques used for 2D image texture synthesis. For example, Wei and Levoy [24] synthesize new video pixel-by-pixel in the raster scan order. They ignore the underlying texture dynamics. It can generate novel images, but is limited to only both spatially and temporally stationary textures. Kwatra et al. [25] propose a graph-cut based seam optimization scheme to synthesize video sequences from example videos. This method is arguably the best method to date in terms of generating high-quality images with minimal spatial and temporal discontinuity. However, it takes a great deal of time in synthesis, so this method is impractical for many real-world applications. Also, to encourage continuity in synthesized video, it might distort or break object structures in videos, which sometimes results in worse perception than spatial or temporal discontinuity. Schödl et al. [23] generate long video sequences by rearranging original frames from given videos. The basic idea is to identify smooth transitions from frames to frames so that they can synthesize an endless video by looping the input video. However, the synthesized video contains only images from the input video [1]. In general, non-parametric methods

[1]Graph-cut method generates new video with only images from the input video when it performs only temporal translation for textures without spatial stationarity.

reuse segments of original videos with optional image processing during frame transitions (e.g. spatial graph cut, morphing), so the synthesized video quality is perceptually almost the same as original videos. They can also handle non-stationary video textures because they simply reuse original video segments. But they need to store entire original video clips in memory, and may either require intensive computation or cannot produce novel images.

The parametric methods usually involve dimensionality reduction of example videos [1, 2, 26, 27], and are commonly referred to as dynamic texture analysis. It is assumed that the images of a video example lie on or near a low-dimensional manifold embedded in the image space, usually called an appearance manifold [14]. They use principal component analysis (PCA) to construct a linear subspace, approximating the appearance manifold, that denotes the appearance of images in a given video. The evolution of images, as a sequence of PCA coefficients, is modeled by an autoregressive process (AR). The generative model of dynamic textures can be viewed as a linear dynamical system (LDS) as Figure 4.2(a), and can be written as

$$\begin{cases} x_t = Ax_{t-1} + v_t, & v_t \sim \mathcal{N}(0, Q) \\ y_t = Cx_t + w_t, & w_t \sim \mathcal{N}(0, R) \end{cases} \tag{4.1}$$

where y is the observed image, x is the hidden state variable, C is the output matrix mapping observations to state variables, A is the transition matrix of AR process, v and w are zero-mean Guassian noise sources, and the subscript t denotes time (or image index). Szummer and Picard [3] do not model video textures at image level, but use a spatio-temporal autoregressive (STAR) model at the pixel level to represent the relationships between a pixel and its neighborhoods. Such pixel-level dynamical models experience difficulties in selecting the appropriate size and topology of the neighborhood. A good model of this approach also requires a large

number of model parameters. Most importantly, such a model is not capable of synthesizing rotation-like motion patterns. Although LDS based models require much fewer model parameters and has a greater capability of capturing different motion types, the visual quality of synthesized video is usually unsatisfactory when the scenes contain large temporal appearance variation and/or shape variation. Regardless the types of given video, these methods tend to suffer a decreasing image quality as synthesis proceeds.

As Yuan [27] points out, the LDS based method produces good-quality dynamic textures only if it is an oscillatory system. That is, for all eigenvalues σ_i of A, $|\sigma_i| \leq 1$ and there exists j such that $|\sigma_j| = 1$. Otherwise, the synthesized dynamic textures will gradually decay or diverge. To overcome this problem, they incorporate a feedback control that results in a non-causal system. Therefore, they first need to generate reference states $\{x_1, x_2, \ldots, x_N\}$ by patching video segments, and then iteratively smooth out the discontinuity in the sense of system fitting. Although using this method one will obtain better results, it does not predict new states on the fly. Furthermore, this improved dynamics model still cannot well model videos with large temporal appearance variation and/or shape variation. Non-stationary dynamic textures are left out as well. As Fitzgibbon [2] suggests, nonlinear component analysis might enhance the performance of video texture modeling. An important fact is that the appearance manifold of a given video example is rarely linear.

4.1.2 Previous Work on Human Motion Synthesis

In human motion synthesis, generating realistic, continuous and new motion sequences using motion capture data has been of great interest. Generally, versatile human poses are divided into different clusters resulting in multi-modality in data distribution. A major issue of human

motion synthesis is how to generate smooth transitions between different clusters. For example, motion graph [28, 29] has been created to characterize possible transitions between clusters of similar poses. The transition points are predetermined, and novel motions are generated by looping and reordering given motion capture data [30, 31]. By specifying initial and final poses, a minimal-cost path of poses can be determined by dynamic programming [32]. However, these methods do not involve new poses.

Dimensionality reduction for multi-modal motion data has been adopted for human motion synthesis [33, 34] using K-means clustering and PCA. But these methods do not find intrinsic, low-dimensional manifold of the motion data. Li et al. [13] modify switching LDS by setting end constraints for each LDS to ensure continuous transitions between local models. These end constraints represent transition points that connect different linear subspaces. Synthesized motions have to go through these pre-selected transition points in order to correctly convert coordinate systems between subspaces, which limits its descriptive capability for motion transitions.

4.1.3 System Overview

The investigations in the above two research fields give hints to successful synthesis of complex motions. First, dimensionality reduction is essential for capturing high-dimensional data, and providing means for generating new configurations in the original data space. Second, multiple models are needed for complex multi-modal data. Third, smooth transitions between different models have to be ensured for realistic motion synthesis. Accordingly, we formulate the problem of complex motion synthesis, for human motion and dynamic textures, or other motion synthesis tasks, as an intrinsic nonlinear manifold learning and traversing problem.

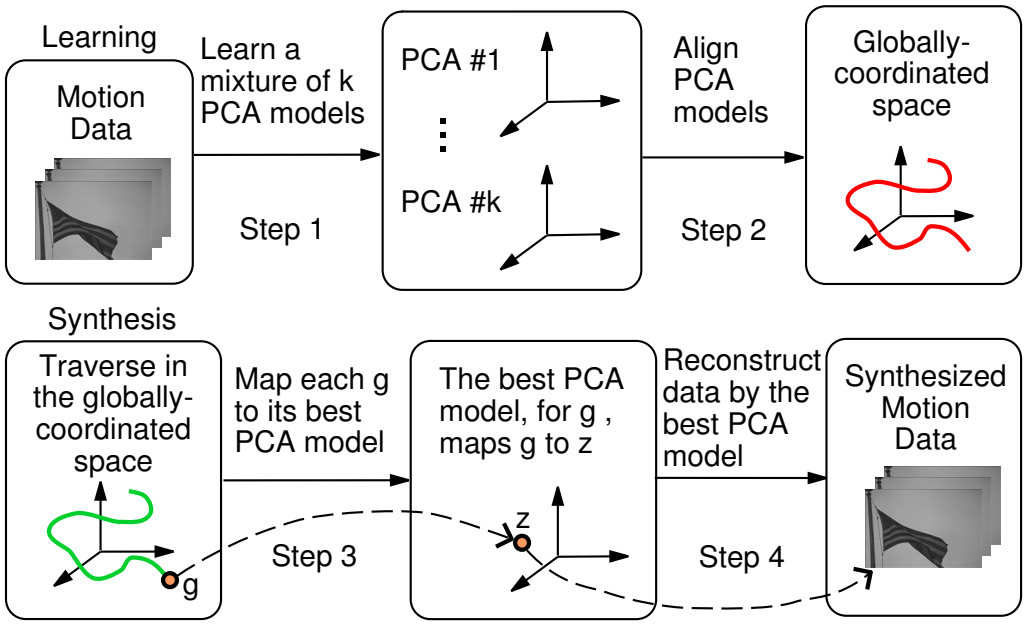


Figure 4.1 Overview of our approach to learning and synthesis of a given motion sequence.

Our approach is conceptually illustrated in Figure 4.1. In the learning stage, we learn a mixture of PCA models that best represents the entire motion sequence. The different PCA subspaces are then aligned into a globally-coordinated space within a maximum likelihood framework. The motion data are projected onto this globally-coordinated space to form a continuous trajectory, and their projected coefficients are stored, not raw data. In the synthesis stage, we pick an initial point, usually a projection of given motion data, in the global subspace. We synthesize motion sequences by traversing in the globally-coordinated space, according to the local dynamics of projected motion data. For each point g in the globally-coordinated space, we find the most probable PCA model associated with the point, and compute its mapping z in the selected PCA subspace. This PCA model then reconstructs a frame of motion data using the mapping z.

Our model is similar to a general SLDS in the sense that we use a mixture of linear subspace representations. But we do not have to deal with explicit model switchings and coordinate conversions. This is because we align the linear models and obtain a global coordinate representation. Our underlying nonlinear manifold still uses a switching linear model, but it is done implicitly via the nonlinear mapping between the globally-coordinated space and original data space. A continuous, multi-modal motion has a continuous coordinate representation in our globally-coordinated space. As a result, our approach is suitable for continuous, multi-modal motion synthesis, and it requires no constraints, or specific transition points, for model transitions.

4.2 Dynamics Model in Global Subspace

We use the dynamic global coordination model (Figure 4.2(c)) that we present in Chapter 2 as our generative model for continuous, multi-modal motion data synthesis. Compared to the LDS (Figure 4.2(a)), we replace the hidden state variable x with global coordinate g, and the inference from g to y is nonlinear through the mixture model $\{s, z\}$. Compared to a general SLDS (Figure 4.2(b)), where it defines a transition probability $p(s_t|s_{t-1})$ to select the current local model, we implicitly switches model depending on state g so the synthesized motion is continuous. Compared to the constrained SLDS [13], where it switches model only at transition points (pre-determined states) to avoid random switches like SLDS, our state variable g has no constraints.

Note that the SLDS has been employed for tracking without problems of random model switches. This is because in the tracking problem we have current observations that can be used to update the predicted current model s_t and state $x_t^{(s_t)}$. In a motion synthesis problem,

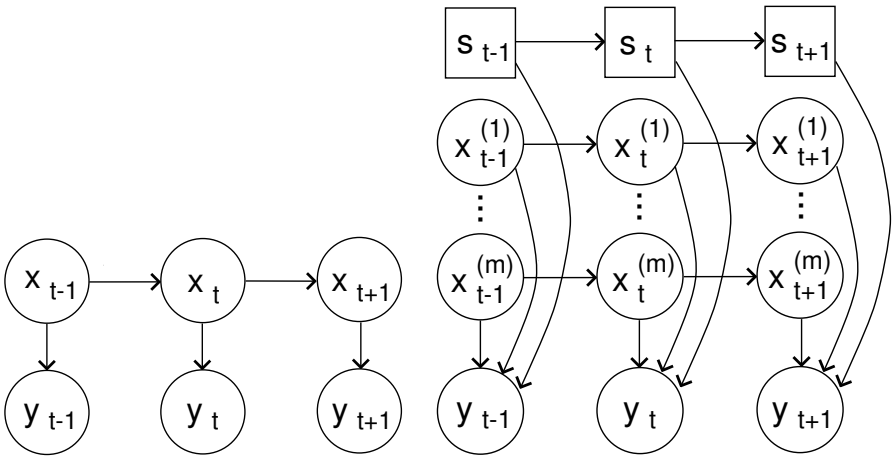
we must ensure the prediction is good enough in the sense of generating continuous and realistic motion data.

4.2.1 Nonparametric Dynamics

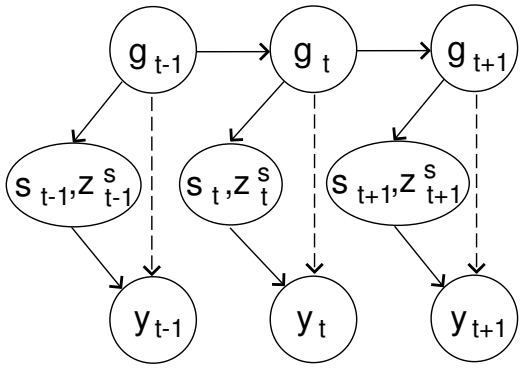
With our generative model, next, we need to determine the dynamics model $p(g_t|g_{t-1})$ in the global subspace. The purpose of this work is to synthesize complex motions, where in general it is reasonable to assume that the dynamics are piecewise linear. The problem with using linear dynamics is that the motion data sequences may be chopped into many small subsequences. In addition, we determine the mixture model without fitting dynamics at the same time so that we can keep a smaller number of local models. Therefore, we adopt a nonparametric dynamics model that learns or samples local motions. This also makes our approach suitable for more general motions.

The essential idea of our nonparametric dynamics is to traverse along the learned trajectory in the global subspace without drifting far away from it. Therefore, we sample and learn motions captured in the given motion data with *spatial locality* and *temporal similarity* constraints. Figure 4.3 helps to visualize our motion prediction process. We denote the projections of given motion data as g with subscripts indicating temporal indices. We also denote the current position in the global subspace as x_t, conventionally representing the current state in dynamic models.

To advance x_t to x_{t+1}, we first find the nearest neighbors of x_t among $\{g\}$. We prefer using motions of these neighbors (spatial locality) while encouraging temporal smoothness (temporal similarity). In Figure 4.3, although g_i is closer to x_t, we prefer using the motion from g_j to g_{j+1} because the motion used to advance g_{j-1} is more similar to that of x_{t-1}. After advancing



(a) LDS (b) SLDS



(c) DGCM

Figure 4.2 Three dynamical models for motion data synthesis. Previous work on dynamic texture synthesis usually uses the LDS model. The SLDS model is used for human motion synthesis. We use the proposed DGCM model for general motion data synthesis.

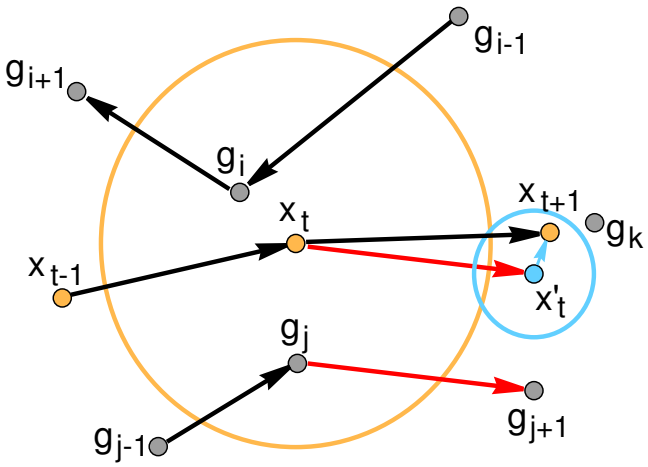


Figure 4.3 An illustration of our nonparametric dynamics in the globally-coordinated subspace.

to x'_t, we perturb it with noise for motion variations. However, to avoid drifting away from the given projected trajectory, we favor sample noise values that pull x'_t toward given data (spatial locality). In Figure 4.3, we eventually obtain the next position x_{t+1} as the predicted state in that it is close to g_k. The entire motion synthesis algorithm is described in Algorithm 4.1. Using this algorithm, if we set the initial condition as $x_1 = g_1$ and $x_2 = g_2$, and let $\sigma_h = \sigma_p = 0$, we are able to reconstruct the input motion data.

Alternatively, after step 2, we can select g_i based on weights $W_i^{(1)} W_i^{(2)}$. Then we learn a local p-order AR process, $g_t = \sum_{k=1}^{p} A_k g_{t-k} + w + v_t$, within a small time window around g_i. We then obtain x_{t+1} by computing $x_{t+1} = \sum_{k=1}^{p} A_k x_{t-k+1} + w + v_t$, where the noise term v_t is now used as a control parameter that pulls prediction toward input motion data. If we compare these two methods, the first one is suitable for fast motion, and the second one discourages novel motions and guides the synthesized sequence along the original trajectory. Therefore, the decision to use which sampling method depends on the user.

Hundreds or a couple of thousands of motion data are a sparse data set in a global subspace, normally no larger than 20 dimensions in our experiments. So we can speed up the sampling

Algorithm 4.1 The motion synthesis algorithm for globally-coordinated subspace model

1. *Sampling neighbors*: At x_t, find its K nearest neighbors $\{g_i\}$, $i \in \mathcal{N}_t$, with weights $W_i^{(1)} \sim \mathcal{N}(x_t, \sigma_h^2)$.

2. *Temporal smoothness*: Compute the motion similarity between x_t and each g_i:

$$\cos(\theta_i) = \frac{\langle dx_t, dg_i \rangle}{\sqrt{\langle dx_t, dx_t \rangle \langle dg_i, dg_i \rangle}}, \tag{4.2}$$

   where $dx_t = x_t - x_{t-1}$, $dg_i = g_i - g_{i-1}$, and $\langle \cdot, \cdot \rangle$ denotes an inner product. Scale the motion similarity to $W_i^{(2)} = \exp(\alpha(\cos(\theta_i) - 1))$ where α is a constant.

3. *Noise perturbation*: Sample noise $\{v_j\} \sim \mathcal{N}(0, \sigma_p^2)$. For each (i, j) pair, we form position candidates at time $t + 1$:

$$x_{t+1}^{(i,j)} = x_t + dg_{i+1} + v_j. \tag{4.3}$$

4. *Drift prevention*: Weigh each position candidate using the Parzen-window approach, so we have

$$p(x_{t+1}^{(i,j)}) = W_i^{(1)} W_i^{(2)} \sum_k \varphi(\frac{x_{t+1}^{(i,j)} - g_k}{h}), \tag{4.4}$$

   where h is the window width and φ is the window function.

5. *Normalization*: Normalize weights so that $\sum_{i,j} p(x_{t+1}^{(i,j)}) = 1$.

6. *Prediction*: Sample x_{t+1} with the weight $p(x_{t+1}^{(i,j)})$.

algorithm by ignoring outlying data. For instance, if the current position x_t is closest to the example g_i, then we can take the nearest neighbors of g_i as the nearest neighbors of x_t, which can be computed beforehand. In addition, in (4.4), we drop the summation operator and compute the window function using only the nearest neighbor g_k because when h is small, the probability of the occurrence of additional examples is very small. Hence, the synthesis can be performed in real time.

4.3 Experiments on Human Motion Synthesis

The input to the human motion synthesis algorithm are motion capture data which are given in the form of 4x4 matrices of 3D transformations using homogeneous coordinate representations. The data contains rotation and translation parameters of 20 body parts of a hierarchical kinematic model. We assume the translation parameters are fixed in the hierarchical model except the one associated with the root which represents the global position of the human body. We use the displacement instead of the global position to accumulate translations in synthesis. We also convert 3x3 rotation matrices into exponential map representations, each of which contains only three parameters encoding the rotation axis and angles. When expressing rotations in exponential maps, we ensure that the representation is continuous over time, instead of enforcing the range $[0, \pi)$ for rotation angles.

We represent human motion data with 60-dimensional rotation parameters. Translations are not treated as a part of the aforementioned algorithms. We sample translation around the nearest neighbors of the predicted rotation. We use four test sequences each of which has between 200 and 500 frames. We use a mixture of two 12-dimensional PPCA models for the danger sequence, and use mixtures of three 12-dimensional PPCA models for other

Figure 4.4 Three selected frames of our synthesized *bow* sequence.

Figure 4.5 Three selected frames of our synthesized *ballet* sequence.

three sequences. Other than the bow sequence that has no repeated motion, we synthesize 1,000 frames for all other input data. A few frames of each rendered motion are shown in the Figures 4.4, 4.5, 4.6 and 4.7. All these sequences contain easily distinguishable poses. The results show that our method is able to produce smooth and realistic human motions.

It is also possible to mix different input sequences to train a globally-coordinated space. If two sequences contain similar motions, these two sequences will be aligned automatically by our approach. Path planning using dynamic programming also can be done within our proposed framework. However, these are out of the scope of the thesis.

Figure 4.6 Three selected frames of our synthesized *disco* sequence.

Figure 4.7 Three selected frames of our synthesized *danger* sequence.

4.4 Experiments on Dynamic Texture Synthesis

For dynamic texture synthesis, the motion data are raw image vectors. The image sequences used in our experiment are taken from MIT temporal texture database [35]. Most image sequences in the database have resolutions of 170 by 115 and contain 120 to 150 frames. They include both stationary and non-stationary dynamic textures. Here, we will compare our method to the LDS based method which is implemented by PCA+AR approach. The order of AR model is automatically determined by Schwarz's Bayesian Criterion [36]. Because the example sequences are short, we use up to three PPCAs for each mixture model.

For stationary dynamic textures, we take a river sequence of 120 frames as an example. We use a 20-dimensional PCA for the LDS method, and align two 20-dimensional PPCA models into a 20-dimensional globally-coordinated subspace for our method. The synthesis results in Figure 4.8(a) show that our method is able to produce high-quality images during extrapolation, while the LDS method produces images with decreasing visual quality over time. Although a closed-loop LDS (CLDS) [27] can fix this problem for stationary dynamic textures, we show in Figure 4.9, 4.10 and 4.11 that, for non-stationary dynamic textures, using a single PCA model results in obvious visual artefact even in image reconstruction, while a mixture model alleviates the problem. The artefact is more serious when shape variations are significant (see Figure 4.12). Synthesis results of this flag sequence are compared in Figure 4.13(a). Due to its fast motion, there is still some small artefact around the flag produced by our method, although when being displayed as a movie, this artefact looks like motion blur. It all thanks to our locally-linear models that confine the artefact to be local and minimal, in contrast to the videos synthesized by the LDS method. All synthesized videos are available at `http://vision.ai.uiuc.edu/~cbliu/`.

Figure 4.8(b) and 4.13(b) show the projected trajectories in the globally-coordinated subspaces. Note that an input sequence has to form loops so that our synthesized sequence can be longer than the original video sequence, which holds true for most approaches in motion synthesis. Interpolated samples have been inserted to small gaps on projected trajectories, as shown in both figures, to create additional loops.

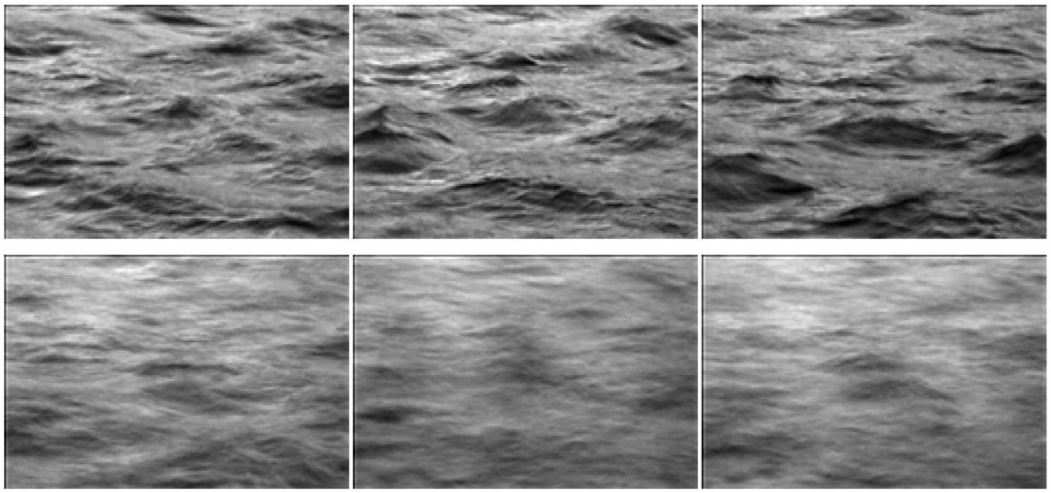
4.5 Discussion

In Table 4.1, we compare our approach with other major approaches for dynamic texture synthesis.

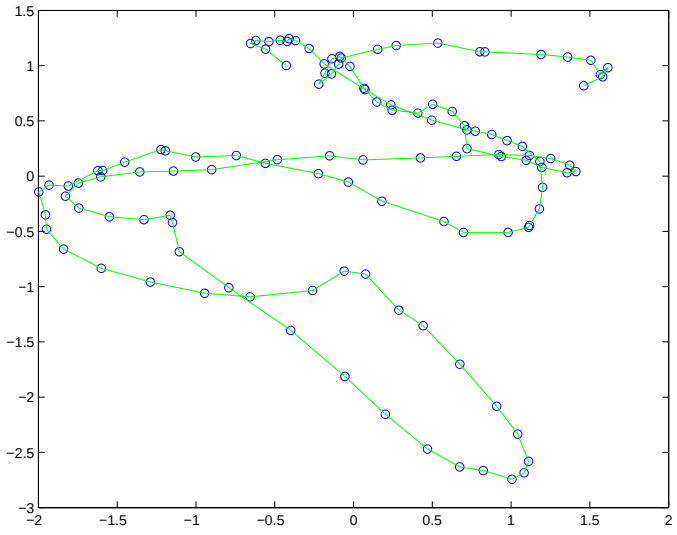
To illustrate the complexity in space, let m and n denote the number of pixels in an image and the number of images in a video, respectively. We also assume that d-dimensional PCA models are used whenever needed. We use k PCA models in our dynamical model and align them to a d-dimensional space, where $k, d < n$ and $k, d \ll m$. And usually, $k < d$.

For non-parametric method, obviously, the space complexity is $O(mn)$. For LDS based methods, the space complexity is $O(m)$ because it needs a linear transformations from image to PCA subspaces, the mean of images, AR parameters, and noise covariances ($md + m + d^2 = O(m)$). For the CLDS based method, other than parameters of the PCA and AR models, it needs to store all PCA coefficients of images. So its space complexity is $md + m + d^2 + nd = O(m + n)$.

For our model, we need to store low-dimensional coordinates (nd), k linear transformations for the PCA mixture (kmd), and k linear transformations from PCA subspace to the globally-coordinated space (kd^2). Isotropic noise in PPCA models and some Gaussians in the algorithm



(a) Comparisons of synthesis result.



(b) Low-dimensional trajectory in the globally-coordinated space.

Figure 4.8 (a) Frame 120, 160, 200 of the synthesized river sequences by our method with a mixture of two PPCA models (top) and the LDS method (bottom). (b) The 20D trajectory of the river sequence projected onto the first 2D of the globally-coordinated space.

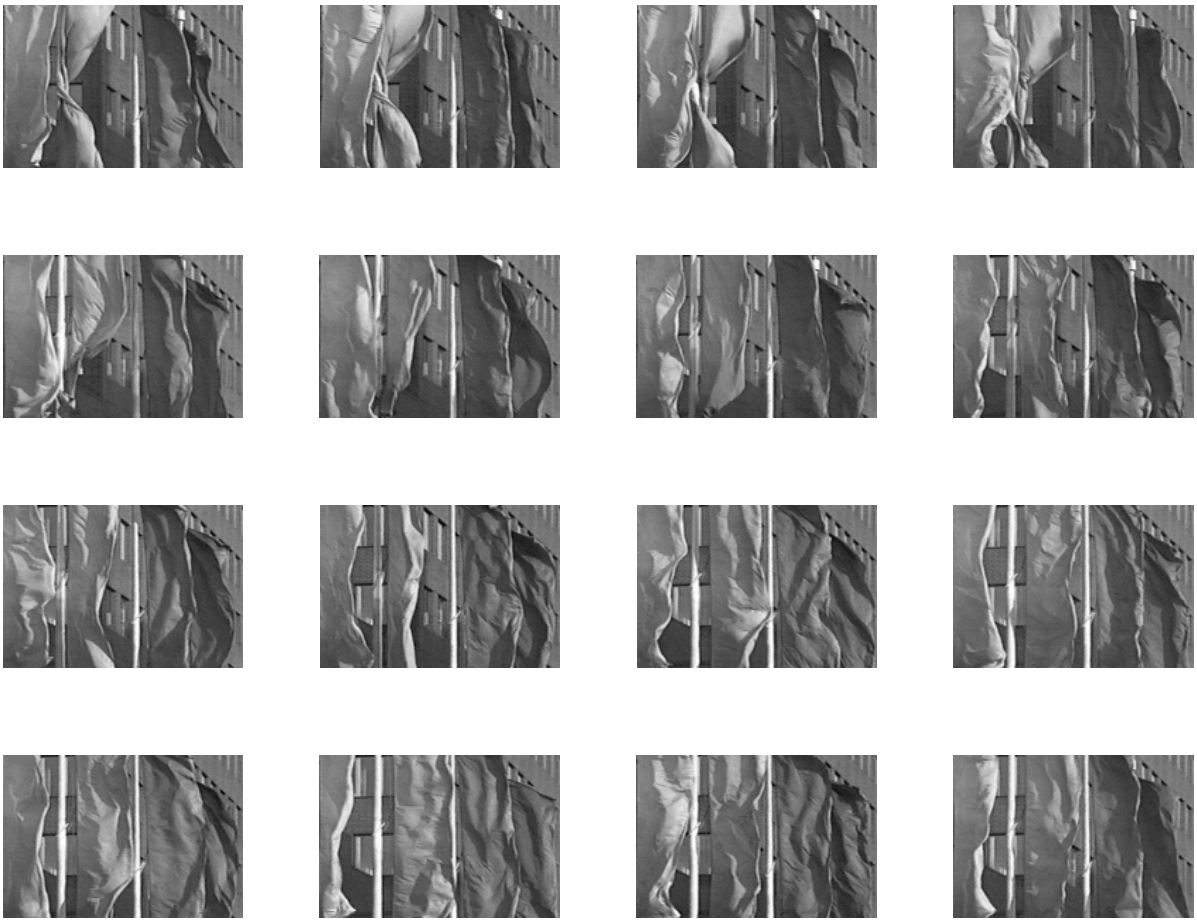


Figure 4.9 Selected frames of an original flag sequence from the temporal texture database.

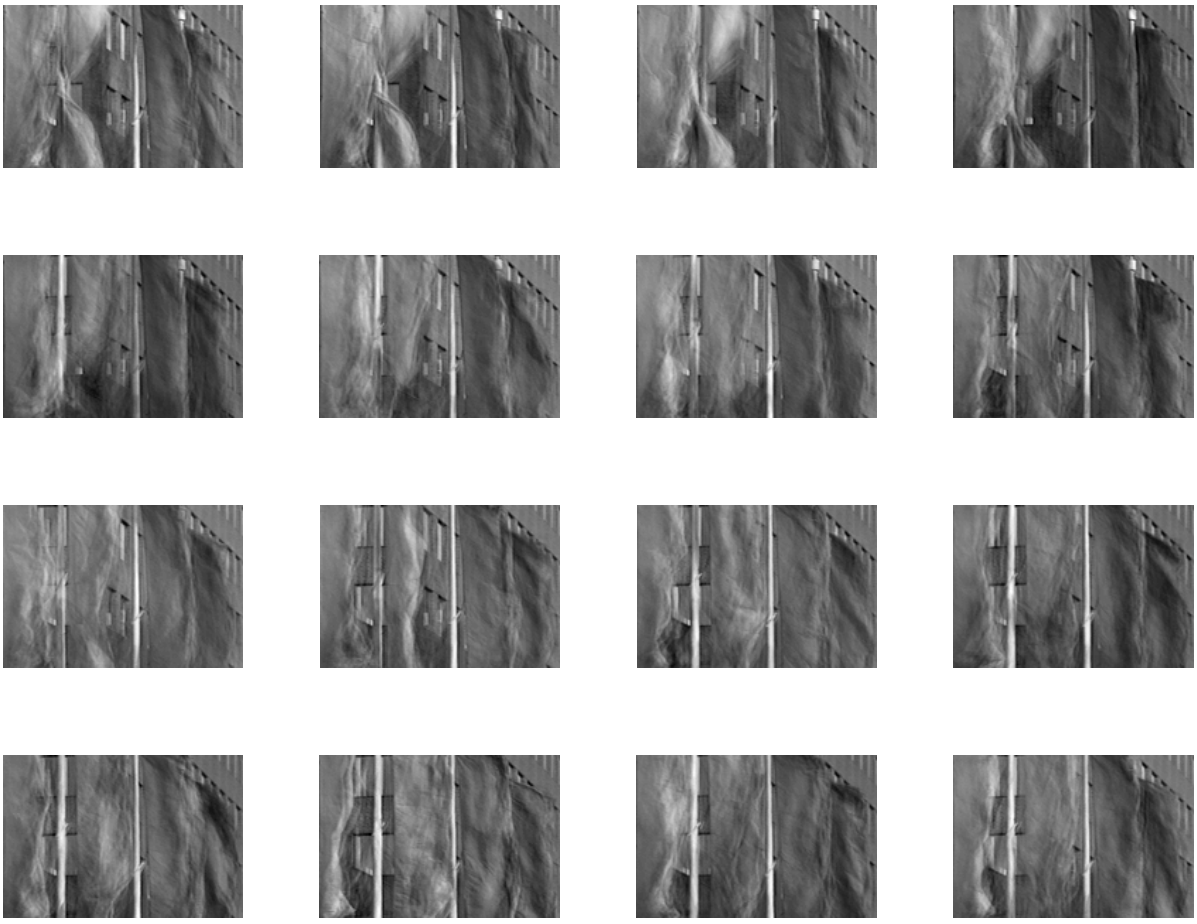


Figure 4.10 Reconstructed frames corresponding to Figure 4.9 using a single PCA model.

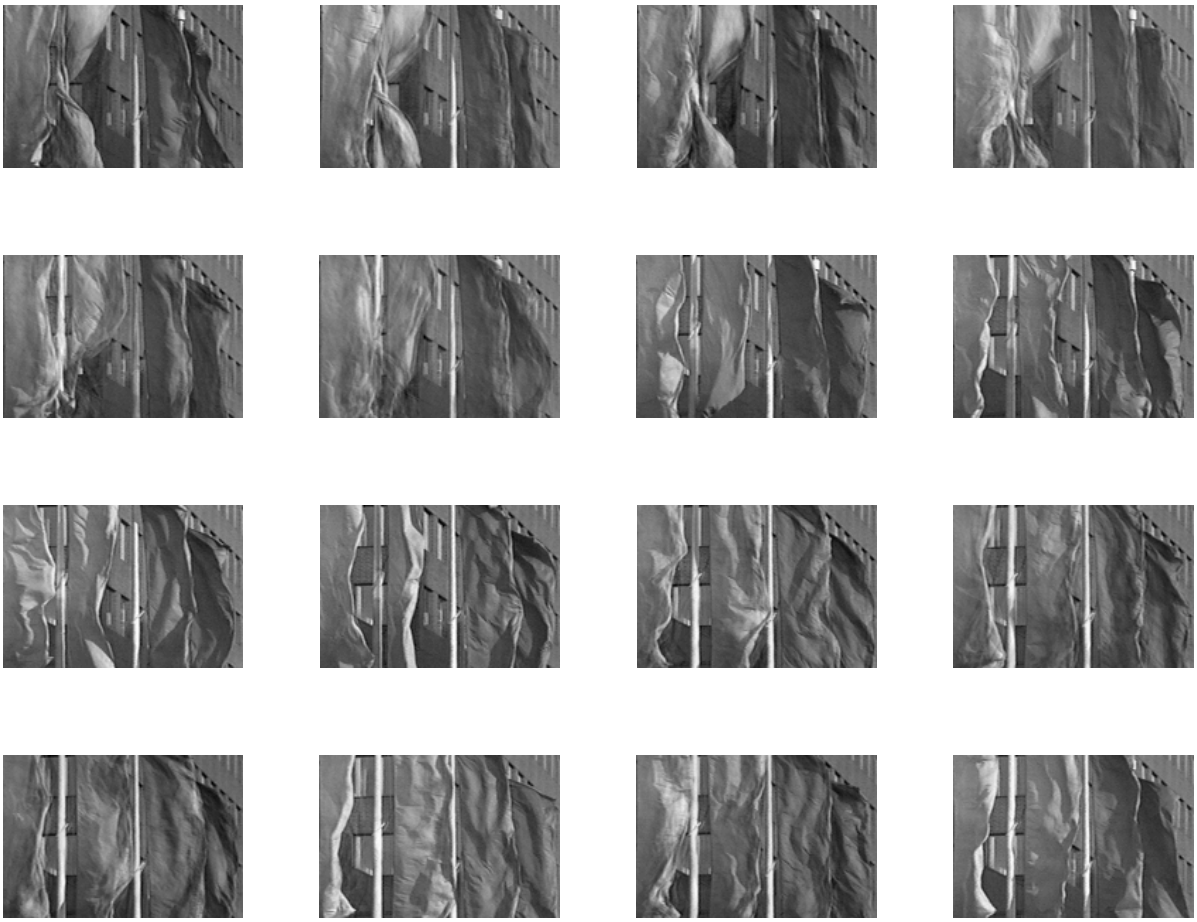


Figure 4.11 Reconstructed frames corresponding to Figure 4.9 using our method with a mixture of three PPCA models. These reconstructed images are significantly crisper than the ones shown in Figure 4.10.

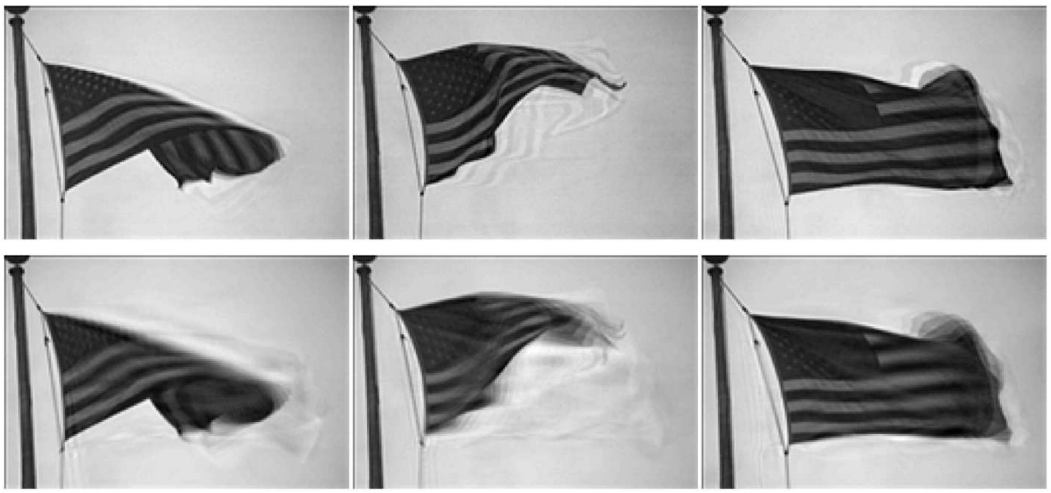
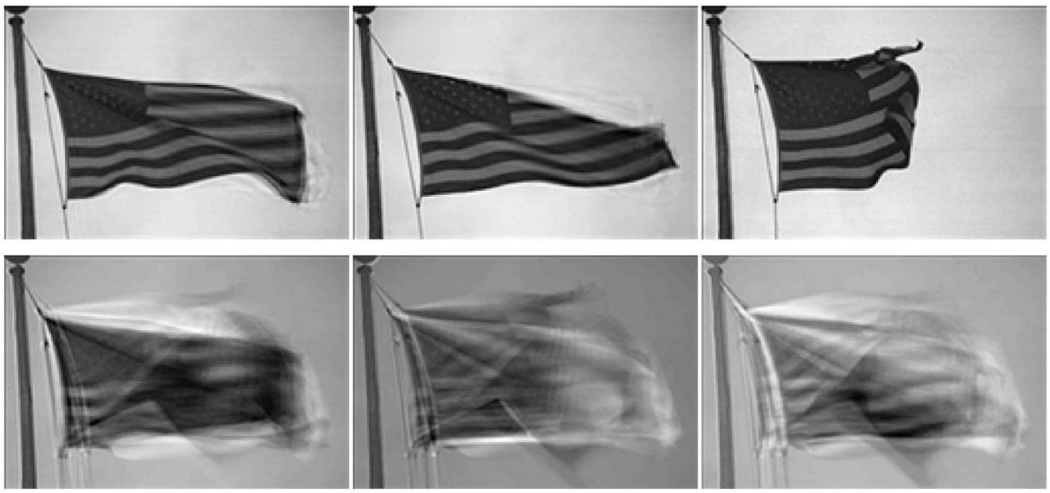


Figure 4.12 Reconstructed frame 20, 40, 60 of the flag sequence by our method with a mixture of three PPCA models (top) and the LDS method (bottom).

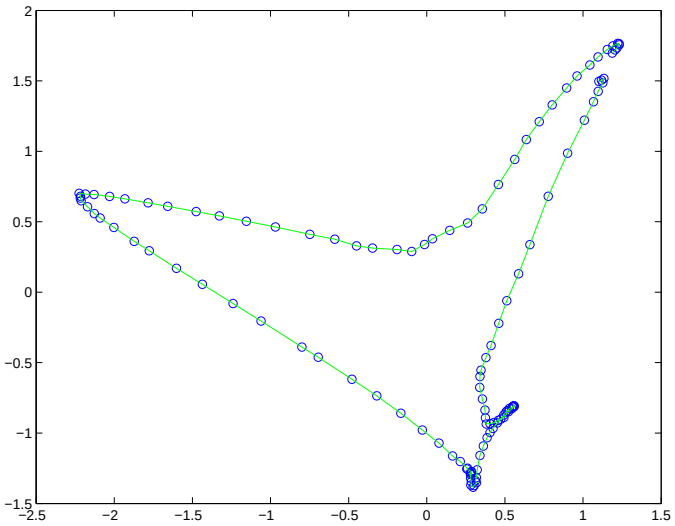
all take constant space. So our space complexity is $O(m+n)$. Even when the image sequences are short, we save about 50% in space for videos of 120-150 frames like those in MIT database.

4.6 Conclusions

In this chapter, we propose a new approach to learning and synthesizing continuous complex motions. We have shown that realistic synthesis of complex motions can be carried out by mixtures of linear subspace models with global coordination. Our underlying idea is similar to SLDS, but our dynamic model ensures continuous motion and does not require constraints for local model transitions. We have demonstrated our approach in two classes of motions: articulated motions and independent point motions. In particular, compared to the literature in dynamic texture analysis, our method enhances the visual quality in synthesis of stationary dynamic textures, and is able to model non-stationary dynamic textures which cannot be handled by any of the existing approaches in the literatures.



(a) Comparisons of synthesis result.



(b) Low-dimensional trajectory in the globally-coordinated space.

Figure 4.13 (a) Frame 100, 210, 290 of the synthesized flag sequences by our method with a mixture of three PPCA models (top) and the LDS method (bottom). The bottom-row images are lighter due to Matlab display program which scales intensity values. (b) The 20D trajectory of the flag sequence projected onto the first 2D of the globally-coordinated space.

Table 4.1 Comparisons of different methods for dynamic texture synthesis.

	Video Textures [23]	Graphcut Textures [25]	Dynamic Textures [1, 2]	Dynamic Textures with CLDS [27]	Our Method
Dimensionality Reduction	No	No	Yes	Yes	Yes
Video Data Storage	$O(mn)$	$O(mn)$	$O(m)$	$O(m+n)$	$O(m+n)$
High-quality Synthesized Images	Yes	Yes	No	Yes, if input is temporally stationary	Yes
Fast/Large Input Motion	Yes	Yes	No	No	Yes
Online Synthesis	Yes	No	Yes	No	Yes
Novel Image Synthesis	No	Yes, around seams	Yes	Yes	Yes
Non-stationary Textures	Yes	Yes	No	No	Yes

CHAPTER 5

LINEAR MODELS OF DYNAMIC 2D SHAPE AND APPEARANCE

In this chapter, we present an approach to model gradual changes in the 2D shape of an object. We represent 2D region shape in terms of the spatial frequency content of the region contour using Fourier coefficients. The temporal changes in these coefficients are used as the temporal signatures of the shape changes. Specifically, we use an autoregressive model of the Fourier coefficient series. We demonstrate the efficacy of the model on several applications. First, we use the model parameters as discriminating features for object recognition and classification. Second, we show the use of the model for synthesis of dynamic 2D shape using the model learned from a given image sequence. We also explore the use of a nonlinear dynamic model to enhance synthesis results. Third, we show that, with its capability of predicting shape, the model can be used to predict contours of moving regions which can be used as initial estimates for the contour based tracking methods.

5.1 Introduction

Changes in the shape of a dynamic object offer important cues for object recognition. In this chapter, we are concerned with models of gradual changes in the shape of a 2D region. We

present a simple model of shape variation which was seen limited use in the past work. This model models the changes in the 2D shape of a region in terms of the changes in its contour representation. Specifically, an autoregressive time series model of the changes in the Fourier coefficients of the region contour is used. We use it to model, recognize, and synthesize 2D dynamic shape. We present applications to (i) modeling fire motion and detecting fire in video sequences, (ii) classification of objects based on changes in 2D shapes, (iii) synthesis of novel image sequences of evolving 2D shapes, and (iv) object boundary prediction for use by contour tracking methods.

The 2D shape representation and its use has received much attention in computer vision. A survey of shape analysis methods can be found in [37]. Pavlidis [38] proposed the following three classifications for shape based methods using different criteria. (i) *Boundary* (or *External*) or *Global* (or *Internal*): Algorithms that use region contour are classified as external and boundary, such as Fourier transforms based approaches; Those that use interior region for the analysis are classified as internal and global, such as moment based methods. (ii) *Numeric* or *Non-numeric*: This classification is based on the result of the analysis. For instance, medial axis transform generates a new image with a symmetric axis, and is categorized as non-numeric. In contrast, Fourier and moment based methods produce scalar numbers, and thus are in numeric category. (iii) *Information Preserving* or *Non-preserving*: Approaches that allow users to reconstruct shapes from their shape descriptors are classified as information preserving. Otherwise, they are information non-preserving.

We propose a dynamic shape model that describes shape at any given time using Fourier transform coefficients and an autoregressive (AR) model to capture the temporal changes in these coefficients. The Fourier description is boundary, numeric, and information preserving.

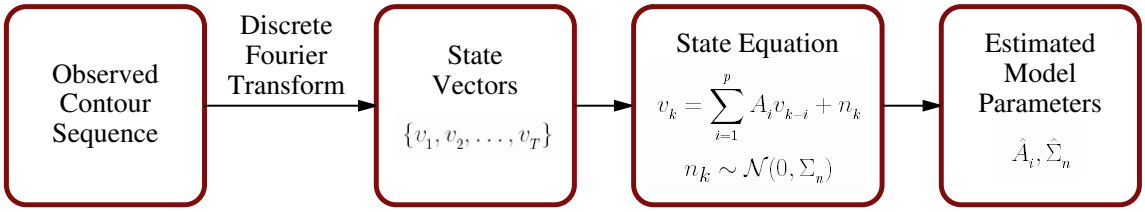


Figure 5.1 System overview of our approach. For an observed contour sequence of length T, we represent the contour at each time instant with its Fourier coefficients v_t. The Fourier coefficient series is then fitted in an autoregressive model. The estimated model parameters $\{\hat{A}_i, \hat{\Sigma}_n\}$ are used to capture the stochastic characteristics of temporal shape variation.

The autoregressive model is a simple probabilistic model that has shown remarkable effectiveness in the mapping and prediction of signals. Our analysis of dynamic shape is illustrated in Figure 5.1. As Srivastava [39] points out, the temporal change of Fourier representation may not be linear. However, a linear model is more manageable to approximate such a process, and requires a small number of observations to estimate parameters. We also elaborate a nonlinear dynamic model for synthesis of dynamic 2D shape to show a potential extension of our proposed approach.

The remainder of this chapter is organized as follows. We review related work in Section 5.2. In Section 5.3, we present our dynamic shape model and its parameter estimation. In Section 5.4, we apply the proposed approach to modeling and detection of fire in video sequences. In Section 5.5, we classify several objects and visual phenomena based on their evolving region contours. In Section 5.6, we apply the learned model to synthesis of evolving shape sequences. Section 5.7 uses our model to predict object shape in a video sequence for object contour tracking. Section 5.8 discusses the limitations of the proposed model. Section 5.9 summarizes the contribution of this work.

5.2 Related Work

Our proposed approach is related to models of active contour tracking, e.g. [40], that predict contour motion and deformation to account for dynamic object shape. For example, Terzopoulos and Szeliski [41] incorporate Kalman filtering with the original snake model [42]. Blake et al. [43] propose a contour tracking method that works particularly well for affine deformation of object shape. Snake based methods process the contour directly in the spatial domain and consider local deformations [42, 44]. By contrast, in our representation, shape information is distributed in each coefficient of FD. Thus, our approach works on global deformations. Only a few methods, such as [43, 45], consider both local and global deformations. Local deformations of all contour points comprise too large a data set to be convenient for shape recognition and classification. In addition, models of active contour tracking predict inter-frame motion and deformation. By contrast, we model global temporal characteristics of the entire contour sequence. Most importantly, most work on deformable shape modeling is aimed at region contour identification by using a deformable, evolving snake to converge on the desired contour. Instead, in our work, the evolving shape description is aimed at describing a temporal changing shape.

There is some work using level sets to represent dynamic shape such as [46]. The advantage of the level set based method is its ability to handle topology changes. However, as will be shown later, our model requires significantly less computation.

We also relate our dynamic shape model to linear dynamical systems (LDS) as illustrated in Figure 2.4(a). The LDS based approaches have been applied to modeling and synthesis of dynamic textures [1, 2, 26]. These methods use principal component analysis (PCA) for the mapping between observed images y and hidden states x. The dynamics of textures is carried

out by an autoregressive model. Similar to LDS, we map between observed contours and hidden states (Fourier coefficients) using Fourier transform.

5.3 Dynamic Shape Model

Our dynamic shape model includes two parts: a spatial representation of 2D shape and a temporal representation of shape variation. The detailed model and its parameter estimation are described in the following sections. The model is also illustrated in Figure 5.1.

5.3.1 Spatial Representation of Shape

Fourier Descriptors (FD), the Fourier Transform coefficients of the shape boundary, represents a 2D shape using an 1D function. There are several variations of Fourier based 1D boundary representation in literature [47]. In this work, we use Persoon and Fu's method [48] for its simplicity.

Given an extracted region in an image, we first retrieve its boundary using eight-connected chain code. Assume that we have N points from the chain code representation of the boundary. We express these points in complex form: $\{z_i | z_i = x_i + jy_i\}_{i=1}^{N}$ where (x_i, y_i) are the image coordinates of boundary points as the boundary is traversed clockwise. The coefficients of the Discrete Fourier Transform (DFT) of $\{z_i\}_{i=1}^{N}$ are

$$a_k = \frac{1}{N} \sum_{i=1}^{N} z_i \exp(-j \frac{2\pi}{N} ik), \tag{5.1}$$

where $k = -\lfloor \frac{N-1}{2} \rfloor, \ldots, \lfloor \frac{N}{2} \rfloor$. If M harmonics are used ($M \leq \lfloor \frac{N-1}{2} \rfloor$), the coefficients $\{a_m\}_{m=-M}^{M}$ are the Fourier Descriptors used to characterize the shape.

To reconstruct L boundary points $\{\widetilde{z}_l\}_{l=1}^{L}$ using M harmonics, we perform inverse DFT as:

$$\widetilde{z}_l = \sum_{m=-M}^{M} a_m \exp(j\frac{2\pi}{L}ml), \tag{5.2}$$

where $l = 1, \ldots, L$.

Note that $a_0 = \frac{1}{N}\sum_{i=1}^{N} z_i$ represents the center of gravity of the 1D boundary, which does not carry shape information. We neglect this term to achieve translation invariance for recognition and classification. We keep this term for synthesis and shape prediction because it accounts for scale changes.

Most related works in Fourier based shape description discuss about similarity measures that make FD invariant to relevant transformations, e.g., rotation, translation and scaling. The requirement for each invariance depends on the applications. In this work, we have to avoid rotation invariance because we need to reconstruct the 2D shape. Since rotation invariance is not relevant, we can always choose the starting point as the topmost boundary pixel along the vertical axis through the center of gravity of the entire shape. Our representation approximates scale invariance (if we drop a_0 term) since we have dense sampling of points along region boundary using chain code. Chain code expression discretizes the arc and Equation (5.1) normalizes the arc length [1].

5.3.2 Temporal Representation of Shape Variation

The stochastic characteristics of boundary motion are estimated by an autoregressive model of changes in Fourier coefficients of the region boundary. The autoregressive (AR) model is

[1]Note: scale invariance is achieved if the distances between a pixel and its eight neighbors are considered as equal.

used based on the assumption that each term in the time series depends linearly on several previous terms along with a noise term [49]. In this work, the AR model is used to capture different levels of temporal variation in FDs.

Suppose v_k are the m-dimensional random vectors observed at equal time intervals. The m-variate AR model of order p (denoted as AR(p) model) is defined as

$$v_k = w + \sum_{i=1}^{p} A_i v_{k-i} + n_k. \tag{5.3}$$

The matrices $A_i \in R^{m \times m}$ are the coefficient matrices of the AR(p) model, and the m-dimensional vectors n_k are uncorrelated random vectors with zero mean. The m-dimensional parameter vector w is a vector of intercept terms that is included to allow for a nonzero mean of the time series.

Our dynamic shape model uses FDs to represent shape, so the random vector v_k is in a form of FD at time k. To select the optimum order of the AR model, we adopt Schwarz's Bayesian Criterion [36] which chooses the order of the model so as to minimize the forecast mean-squared error. We estimate the parameters of our AR model using Neumaier and Schneider's algorithm [50] which ensures the uniqueness of estimated AR parameters using a set of normalization conditions.

5.4 Application I: Recognition

In this section, we will show that using the temporal information of shape variation improves recognition results that use shape only. We choose the problem of fire recognition in video sequences as an example because of its potential usefulness in the real world.

5.4.1 Vision Based Fire Detection

Fire has diverse, multi-spectral signatures, several of which have been utilized to devise different methods for its detection. Most of the methods can be categorized into smoke, heat, or radiation detection. A detailed survey can be found in [51]. Each fire detection method is better suited to a distinct environment. Vision based fire detection has the following advantages over the other methods. First, it has fast response to fires. As the radiation based method, it detects fires as soon as they appear in sight. Second, it directly senses the location of fire (in 2-D), not just radiation which comes from its general vicinity. Third, it is capable of analyzing existing images or image sequences so that it can be used for multimedia database retrieval. Although it is a line of sight method, there are scenarios where fire is visible and is indeed a strong cue, complementing any smoke and smouldering nearby.

However, there are only a few papers about fire detection in computer vision literature. Healey et al. [52] use a purely color based model. Phillips et al. [53] use pixel colors and their temporal variations. These methods have the following two drawbacks. First, a region composed of fire-colored pixels is too simple a model of fire since fire also has spatial structure, namely the core is brighter than the periphery. Second, temporal variation in image pixel color does not capture the temporal property of fire which is more complex and requires a region level representation. For example, pixels of the core of the fire exhibit less temporal variation than the other pixels.

5.4.2 Fire Models

Fire has unique visual signatures. Color, geometry, and motion of fire region are all essential for recognition. A region that corresponds to fire can be captured in terms of (1) spectral

characteristics of the pixels in the region, and (2) the spatial structure defined by their spectral variation within the region. The shape of a fire region usually keeps changing and exhibits a stochastic motion, which depends on surrounding environmental factors such as burning materials and air flow.

The pixels in a fire region have characteristic color spectra and the pixels with different spectra have characteristic relative locations. In color images, we might see bright white color in the core, and yellow, orange and red away from the core. In grayscale images, we see that core is brighter than the periphery. Note that a fire region may include multiple bright cores which correspond to multiple hot spots. This can be viewed as a large fire composed of multiple sources of fires as illustrated in Figure 5.2. Thus, the fire region in a single image can be modeled as follows: (i) It stands in high contrast to its surroundings; (ii) It exhibits a structure of nested rings of colors, changing from white at the core to yellow, orange and red in the periphery.

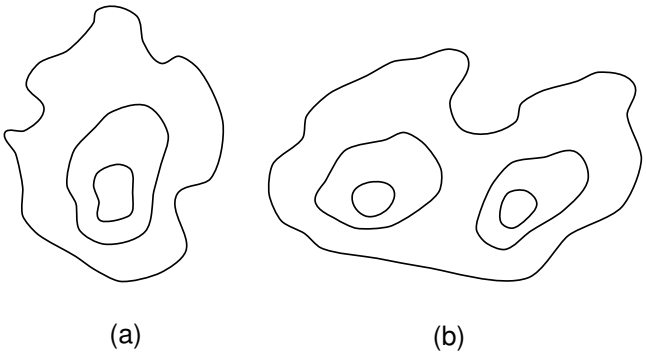


Figure 5.2 Examples of the nested ring structure of fire regions. (a) A fire region with a single core. (b) A fire region with two cores.

A fire in motion has a relatively static general shape (determined by the shape of burning materials) and rapidly changing local shapes in the unobstructed part of the border. The lower frequency components of fire region boundary are relatively steady over time, and the higher

frequency components change in a stochastic fashion. Accordingly, we can use a stochastic model to capture the characteristic random motion of fire boundary over time.

5.4.3 Fire Detection Algorithms

Our fire detection algorithms include two main steps: (i) Extract potential fire regions in each image; and (ii) Represent each extracted region using FD and AR parameters, and then use a classifier to recognize fire regions.

We detect potential fire regions based only on the fire spectral and spatial models. We first extract high intensity regions (in grayscale) possibly corresponding to fire cores, which we called seed regions. We grow each seed region by following spectral gradients of the image and adding neighbor pixels if they have colors given by the fire spectral model with sufficiently high likelihood. The fire spectral model is represented by a mixture of Gaussian distributions of interior fire color in HSV space [54].

For each potential fire region, we represent it independently by taking the magnitude of its FDs. We then find spatially matching regions in previous images of the sequence, and estimate parameters of the AR model for the corresponding fire regions. The FD and estimated AR parameters are both used as features of current region. We use a two-class Support Vector Machine (SVM) classifier [55] with radial basis function (RBF) as its kernel function for fire region recognition. Accordingly, $f(x)$ represents the decision whether feature vector x is fire:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \exp(-\gamma |x - x_i|^2) - b \tag{5.4}$$

where $\{x_i\}_{i=1}^{N}$ are training data of two classes (fire or non-fire), $y_i \in \{+1, -1\}$ is the class label of x_i, α_i is a Lagrange multiplier, and γ is the bandwidth of the kernel function. Finding an optimal hyperplane to separate training data is equivalent to finding all the nonzero α_i. A training sample x_i corresponding to nonzero α_i is called a supported vector (SV) of the optimal hyperplane. The classification result of x is determined by the sign of $f(x)$.

5.4.4 Experimental Results

The video clips used in our experiments are taken from a random selection of commercial/training video tapes. They include different types of fires such as residential fire, warehouse fire, and wildland fire. We use images captured at day time, dusk or night time to evaluate system performance under different lighting conditions. We also use other image sequences containing objects with fire-like appearances such as sun and light bulbs as negative examples. The video clips that we tested our algorithm on contain a total of 3956 image frames in 36 sequences. Figure 5.3 shows some selected fire images used in our experiments. The (red) contours depicted in the images are the detected fire region contours.

In our test data, the potential region extraction algorithm extracted a total of 1319 fire-like region contours, 1089 of which were true fire region contours. For shape representation in terms of Fourier Descriptors, we find that using 40 coefficients (i.e. $M = 20$) is sufficient to approximate the relevant properties of the fire region contours. In this experiment, we assume that different FDs at any given time k are independent of each other, so we have diagonal coefficient matrices in our AR model, where $A_i(m, n) = 0$ if $m \neq n$. Thus it can be viewed as modeling $2M$ independent time series. We also find that the AR(1) model yields the minimal

Figure 5.3 Selected fire images used in experiments.

forecast mean-squared error. Therefore, we use 40 AR coefficients to represent the stochastic characteristics of the temporal changes in FDs.

Table 5.1 Recognition rate of fire and non-fire contour recognition.

Experiments	Fire Contours	Non-Fire Contours
Use shape only (FD)	0.996	0.904
Use shape + evolution (FD + AR)	0.999	1.0

We tested our algorithms in two ways: The first set of experiments with only spatial information of region contours (FD only as the feature vector), and the second set of experiments with spatial and temporal information of region contour evolution (FD and AR parameters as the feature vector). In the second set of experiments, we required that a fire contour be seen in at least previous four frames. Note that three frames are the minimum requirement to estimate parameters of our AR(1) model. For each set of experiments, we repeated the test ten times using 10% of fire and non-fire region contours to train the SVM classifier, and the other 90% of region contours for test. In this way, we used many more fire examples than counter examples on training. This was intended to tilt the detector in favor of false positives vs false negatives as corroborated by the experimental results. The average recognition rate is shown in Table 5.1. It is clear that temporal information of shape evolution indeed improved the detection performance and significantly reduced the false alarm rate.

5.5 Application II: Classification

In this section, we demonstrate that the temporal information of shape variation alone is a good discriminant for classifying several objects and visual phenomena. Under our proposed

framework, we show that object shape variation is indeed an important visual cue for object classification.

Follow the model presented in Section 5.3. Assume that M harmonics in the FDs are used to represent the region boundary of an object in each image of the sequence, and AR(1) model is used to describe boundary dynamics. We then have $2M$ AR coefficients to represent the temporal characteristics of the evolving object shape in an image sequence. Let $\{a_n\}$ and $\{b_n\}$ be AR coefficients modeling a dynamic shape α and a dynamic shape β, respectively. We define the distance between the two dynamic shape sequences as

$$d(\alpha, \beta) = (\sum_{n=1}^{2M} |a_n - b_n|^2)^{1/2}. \tag{5.5}$$

A simple nearest-neighbor classifier using metric (5.5) is used for classification.

5.5.1 Experimental Results

The image sequences used in the experiments include two running human sequences, three waving flag sequences, and two fire sequences. The fire contours are extracted as described in [56]. The region boundaries of flags and running human are semi-automatically extracted using active contour method [42] for each image frame. We use forty FDs to approximate each object boundary. The AR parameters are estimated using each whole sequence. Therefore, the estimated AR parameters represent the global dynamics of the object boundary in a sequence. The experiments are done using the cross-validation method. Only one out of seven image sequences is misclassified, where a running human sequence is classified as a waving flag sequence (see Table 5.2).

Table 5.2 Nearest-neighbor classification results.

Input test sequences (for each, the nearest-neighbor is identified)	# of input sequences labeled as		
	Human	Flag	Fire
Running Human	1	1	0
Flapping Flag	0	3	0
Fire	0	0	2

5.6 Application III: Synthesis

In this section, we apply our model to synthesis of dynamic shape. In particular, we synthesize fire boundary sequences, where the dynamic shape model is obtained from a fire image sequence in as described Section 5.4. We choose fire as an example because fire region can be modeled as nested subregions, where each subregion shows temporal variation.

Synthesis of dynamic shape is a novel topic in computer vision. The most relevant work are those of image based dynamic/temporal texture synthesis. Some of them use only local image structures and ignore the underlying dynamics [24]. Some other works that learn the underlying dynamics in pixel level [3] or in image subspace [1] do not use region level image structures. Instead, they learn the global dynamics of the whole image. In our method, we learn the dynamics of regions using region boundaries.

Many physics based methods have been proposed to produce visual phenomena such as fire [57, 58, 59]. However, since these methods do not learn dynamics from images, they are not capable of generating subsequent images based on a given image. Image based method, such as [1], generates an image sequence if given an initial image and the learned image dynamics. But the resulting images will show significant artifacts if the region of motion is not fixed. Our approach is image based, and it directly deals with temporal variation of regions.

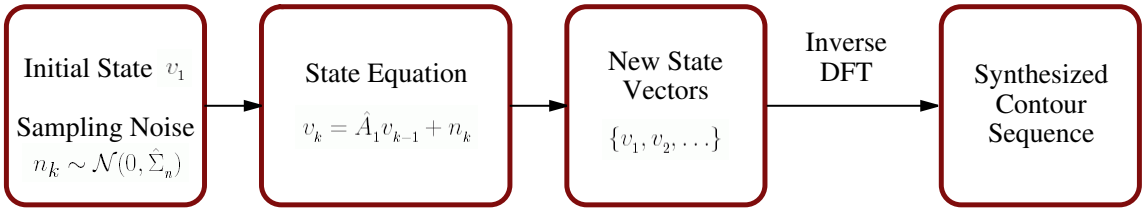


Figure 5.4 Our procedure for synthesis of dynamic 2D shape.

5.6.1 Synthesis Algorithm

We synthesize a new sequence by simulating the AR model learned from the given image sequence. The synthesis process is illustrated in Figure 5.4. For a given initial image, we retrieve the object boundary in the image and represent it using Fourier descriptors. We perform desired number of iterations of the AR model to simulate FDs for the entire synthesis sequence. The new 2D shape sequence is reconstructed using the simulated FDs by Equation (5.2).

5.6.2 Synthesis Using Nonlinear Dynamic Model

To enhance the synthesis capability of our proposed model, we can incorporate a nonlinear dynamic model to replace our AR model. It is reasonable to consider that, in general, the dynamics is piecewise linear. Therefore, we consult to a switching linear dynamical system (SLDS). A SLDS includes m linear dynamical systems (LDS) and a discrete hidden Markov model $P(s_t|s_{t-1})$ that selects one of the m LDSs at each time instant, as shown in Figure 2.4(c).

Within our proposed framework, the linear mapping between observed contours and hidden states is provided by Fourier transform. As a result, we have a switching autoregressive (SAR) model to describe changes in a Fourier coefficient series. Using the notations in Equation 5.3,

we can write a SAR model as

$$v_k^{(j)} = w^{(j)} + \sum_{i=1}^{p} A_i^{(j)} v_{k-i}^{(j)} + n_k^{(j)}, \tag{5.6}$$

where $j = 1 \ldots m$.

In the case of fire, the shape of a fire region usually keeps changing, e.g., because of air flow. The air flow, being unpredictable, lends a distinct character to the way in which the region shape changes with time. Accordingly, a SAR model, being able to describe a set of different dynamics, is more descriptive in such a situation.

5.6.3 Synthesis Results

In this experiment, we use a fire sequence as a training example. A fire region is modeled as a nested ring structure where each ring is associated with a color spectrum. Although the changes in color is continuous, we threshold the fire region (by grayscale intensity) into three subregions. Each region boundary in the given image sequence are independently modeled by our approach. The color spectra of each region are modeled as a mixture of Gaussian. Once the parameters of three AR models have been estimated, we use the mean boundaries in the given sequence as initial boundaries, and simulate the AR models to generate subsequent boundaries. An inner region boundary is confined to its outer region boundary so that we maintain the nested ring structure. To avoid spin-up effects, the first thousand time steps of the AR models are discarded. The pixel colors of each region are drawn from respective color models. Figure 5.5 shows the nested ring model, an example fire image of the input video and some selected synthesized fire image frames.

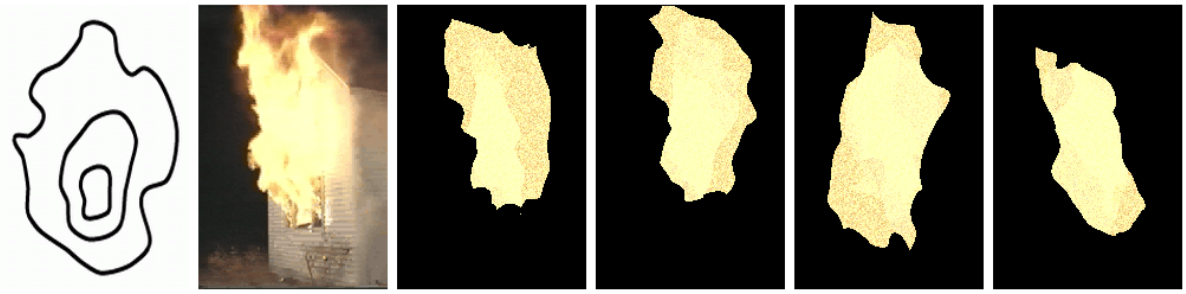


Figure 5.5 Leftmost image: A nested ring structure models the fire region. Second image: An example fire image from the given video sequence. Others: Selected frames of the synthesized fire image sequence.

Our method is capable of solving the following two problems: Given a fire image sequence, (i) generate a new sequence of fire shapes, where both shapes and dynamics are similar to the given image sequence; (ii) also given an initial fire shape, generate a new sequence of fire shapes, where the dynamics is similar to the given image sequence. To achieve photo-realistic fire rendering, since we can solve problem (i), we need only a more sophisticated model that enforces spectral gradient to fill colors in the synthesized fire region. For non-photo-realistic fire rendering, such as cartoon drawing, we ask artists to draw fire regions as nested rings and assign a color for each subregion. Our approach will automatically generate subsequent images based on the learned dynamical model. The synthesized sequence can then be overlaid into other image sequences.

5.7 Application IV: Shape Prediction

The capability of predicting shape comes naturally in our dynamic shape model. In this section, we apply our method to tracking deformable objects. The contour based tracking methods consist two parts: obtaining an initial contour and conforming the initial contour to

object boundary. A good initial contour estimate provides a predicted contour closer to true object boundary in both geometry and position.

Most works on contour tracking are based on the active contour model (or snake model) proposed by Kass et al. [42]. Some works assume that the motion of the object is slow and its deformation is small [60]. So the optimal contour estimate in the previous image frame is used as the initial contour in the current frame. When the changes in shape are large, these methods are very likely to fail. Other works that estimate motion and deformation are compared to our method in Section 5.2.

Using our proposed framework, the contours are again represented by FDs. To account for large changes in shape, we estimate our AR model locally using a small number of previous image frames. A first-order AR model is estimated. Then the initial contour is predicted by Equation 5.3 with $n_k = 0$. Note that the zeroth term of FDs has positional information. So our dynamical model simultaneously predicts the 2D position and shape for the current image frame. Any contour based tracking methods can then be used to conform the contour to object boundary.

5.7.1 Experimental Results

We test our algorithms using a Bream sequence, where a fish initially swims to the right, makes a sharp turn, and then swims to the left. We choose this image sequence because there are large changes in shape when the fish makes a sharp turn, which makes the tracking challenging. We compare our method to the method that predicts only shape translation but not shape deformation.

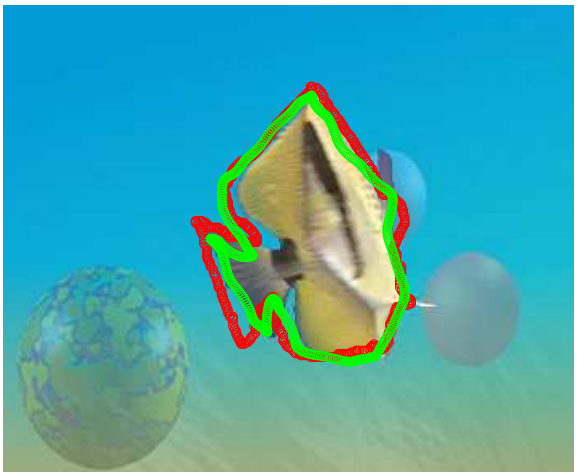


Figure 5.6 The green contour is predicted by our dynamic shape model, and the red contour is the optimal contour of the previous image frame with predicted translation.

Figure 5.6 shows the estimated initial contours of both methods. It is clear that our method accounts for scale change in horizontal dimension, but the other method does not. The fin on the upper right side of the fish is partially occluded in the previous image frame. Both methods do not predict this discontinuous change in shape. But our method does move the fin upward according to its appearance in previous image frames. The quality of the converged contour by any snake model will benefit from a better initial shape prediction.

5.8 Limitations

In Section 5.3.1, we approximate scale invariance for FD by densely sampling along the boundary to obtain the chain-code. However, for small regions, the spatial quantization is likely to introduce considerable noise to the FD. To avoid this problem, we eliminate regions smaller than a certain size. Consequently, our model does not detect small or far away fires. Small regions are expected to increase misclassification rate and synthesis results are better for larger regions.

The AR model is a linear dynamical system. There may be cases where linear model is not sufficient. In such cases, nonlinear dynamical model can be adopted under the proposed framework. For example, we have shown that a switching AR model can be used to enhance synthesis results. Similarly, any other shape description method with boundary, numeric, and information preserving properties may be used in place of FD.

5.9 Conclusion

In this chapter, we have proposed a novel model for dynamic shape. Although both FD and AR model have been well established, using them together to analyze temporal shape variation is not discussed in literature. Traditional shape analysis focuses on spatial similarity, but not temporal similarity. The autoregressive model has been applied mainly to model 1D signals [49] and 2D pixel interdependences [1, 3]. We are not aware of any work on AR modeling of region shape changes.

CHAPTER 6

MOTION MODELS OF 2D OBJECTS

In this chapter, we will show that temporal variations in image content provide useful information for content based video retrieval. In particular, we explore the use of three motion representations and apply and evaluate them in retrieving a variety of motion patterns. Our approach assumes that each dynamic object has been tracked and circumscribed in a minimal bounding box in each video frame. We represent the motion attributes of each object in terms of changes in the image context of its circumscribing box, which we call a 2D motion model. The changes are described via motion templates [61], self-similarity plots [62], and image dynamics [1]. Initially, defined criteria of the retrieval process are interactively refined using relevance feedback from the user. Experimental results demonstrate the use of the proposed 2D motion models in retrieving objects undergoing complex motion.

6.1 Introduction

Recently, some motion representations have been proposed to recognize different motion patterns such as human gaits, activities, periodic motions and texture motions. However, the existing content-based video retrieval (CBVR) approaches focus on low-level motion features such as pixel-level optical flow or affine parameters for motion content indexing, in addition

to other visual features such as color, shape or texture. The main disadvantage of using low-level motion features lies in the recognition of complex motion patterns such as gaits. Such complex motion patterns can be effectively tackled using higher-level motion representations, which might be region based or image based, for example. But such an extension is not straightforward for video retrieval and often depends on many assumptions.

6.1.1 Motivation and Approach

According to the motion classification tree of objects proposed by Kambhamettu et al. [63], most real-world motions can be classified as rigid, articulated, elastic (deformable motion with topological invariance), or fluid. For example, vehicle movement is a rigid motion; animal/human movements are articulated motions in general; deformable objects affected by external force such as a dropping sheet of paper exhibit elastic motion; motions exhibiting topological variations and turbulent deformations are viewed as fluid motion.

These different types of movements become apparent via different characteristics of motion extracted from images and can then be used for retrieval. For instance, articulated motion can be characterized through periodicity of motion observed in videos and can be found in many biological movements, such as human or animal gaits. Apart from gaits, articulated motion also includes interesting kinds of movements, such as moving body parts like a man swimming in sports video, that people are interested in querying. These movements are generally localized in nature and can be characterized through region based motion features like motion presence and motion recency, which we will discuss in the later sections of this chapter. Elastic and fluid motion, on the other hand, varies continuously across objects. The difference between these two classes of motions lies in the continuity of the object itself. When observing these types

of motions, people usually have prior knowledge about the object and pay attention to the deformations or topological changes of the observed object. Either deformation or topological change is usually an important signature of object identity. Rigid motion characterizes poses and translations of rigid objects and it corresponds to affine parameter estimation in image analysis that has no information about object identification. To recognize a rigid object, people consult to shape, color or other visual cues other than motion to determine the object class.

Based on the aforementioned visual properties, we investigate four motion properties: periodicity, presence, recency, and image dynamics. To represent these movements, we also adopt appearance based methods, rather than model based approaches, because (1) the types of dynamic objects of interest are unknown, and (2) the dynamic objects present in the video database may have huge variety so that no single model fits well all dynamic objects. In this work, we represent motion periodicity using modified similarity plots [62] where we use normalized cross-correlation to measure image similarity. Then we proceed to represent motion presence and motion recency using temporal template approach [61] that reveals the tendency of movement and have been successfully used to recognize human activities. Following which we use image-level dynamics, motivated by [1] that characterizes the temporal changes between image frames, to capture variations of the object motion without object models. These motion properties can be harnessed to cover a wide range of interesting motion patterns and can be used to retrieve videos by queries that analyze the high-level motion content in videos.

Our system makes two assumptions. First, we assume that the dynamic objects in an image sequence have been tracked so that a minimal bounding box circumscribed for each dynamic object is available in any given video frame. Second, we assume that there is only one foreground object in each circumscribing box, and that the backgrounds do not change

significantly over a short time period. There have been some tracking techniques which find a minimal bounding box for a moving object. Although in some cases tracking methods might fail to locate moving objects, we maintain the first assumption by interactively working with tracking methods. The second assumption can be removed if the dynamic object can be automatically segmented. However, this is a very difficult problem, especially when the types of moving objects are unknown. Therefore, our second assumption ensures that the dynamic objects in corresponding bounding boxes can be reasonably matched (or aligned). With these two assumptions, our objective becomes: given a sequence of bounding boxes whose changes in the image context representing the motion properties of a dynamic object, we find similar dynamic object sequences based on the similarity of changes in their image contexts.

6.2 Method

Assume that a dynamic object has been tracked and a minimal bounding box around the object in any give video frame is available. Note that the bounding boxes of a moving object may have different sizes in different video frames. Therefore, we first align the bounding boxes so that the appearances of the object are best matched. We then capture the motion content of the bounding box sequence using three representations described later. To retrieve similar sequences, we first compute the similarity measures between the representations of the given sequence and those in database. The similarity between sequences is interactively refined by integrating similarity measures according to user's feedbacks.

6.2.1 Image Alignment of Dynamic Objects

Normalized cross-correlation (NCC) [64] is used to match the appearances of dynamic objects in the bounding boxes. This method shifts a template image t over a search image f, measuring normalized cross-correlation at each point. The NCC value at (u, v) over the window $W_{x,y}$ is defined as

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}][t(x-u, y-v) - \bar{t}]}{\left\{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}]^2 \sum_{x,y}[t(x-u, y-v) - \bar{t}]^2\right\}^{0.5}} \tag{6.1}$$

The point associated with the maximal NCC value is selected as the best match. Although the original method requires that the search image be larger than the template image in both dimension, we extend the search region to include some neighbor pixels of the target bounding box. To help finding best match by NCC, we place spatial constraints on the search window to prune the candidate matches. This alignment process is fully automatic. The computations for all motion representations in the following are done in overlap regions of bounding boxes.

6.2.2 Self-Similarity Plots

Cutler et al. [62] developed an approach to detection of periodic motion by using similarity plots, where they analyze periodic signals using an auto-correlation function. The idea is to use similarity plot to encode the projection of spatio-temporal dynamics of moving objects, and then analyze similarity plot for object classification.

Figure 6.1 shows the similarity plot of a twenty-frame human running sequence. The value at pixel (x, y) of the plot represents the similarity, defined in this work as the value of normalized cross-correlation, between overlap regions of the bounding boxes in image frame x and y. The

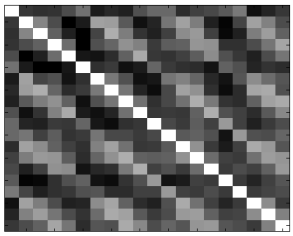


Figure 6.1 The similarity plot of a human running sequence.

bright gray lines parallel to the white diagonal in the plot indicate periodic motion in the given sequence.

Two features of a similarity plot are defined in this work. These are concerned with if the given sequence is periodic, and second, the length of a periodic cycle. For instance, as indicated in Figure 6.1, the motion of the object is periodic with a cycle length about six image frames, which corresponds to a half gait, or a stride.

6.2.3 Temporal Templates

Davis et al. [65, 61] introduced two temporal templates, motion-energy image (MEI) and motion-history image (MHI), to respectively represent the presence and the recency of object movement. Let $D(x, y, t)$ be a binary value indicating regions of motion at frame t. An MHI H_τ is defined as

$$H_\tau(x, y, t) = \begin{cases} \tau, & \text{if } D(x, y, t) = 1; \\ max(0, H_\tau(x, y, t-1) - 1), & \text{otherwise,} \end{cases} \tag{6.2}$$

where τ denotes the desired length of history. An MEI E_τ is defined as

$$E_\tau(x, y, t) = \begin{cases} 0, & \text{if } H_\tau(x, y, t) = 0; \\ 1, & \text{if } H_\tau(x, y, t) > 0. \end{cases} \tag{6.3}$$

In our system, we obtain D using image differencing. To distinguish between motion patterns, seven of Hu's moments are computed over MHIs and MEIs respectively, which are translation- and scale-invariant. Then the Mahalanobis distance of Hu's moments of two MHIs or MEIs is used to measure the similarity between motions of τ image frames. Note that different motion patterns may need different lengths of history to be best described by these temporal templates. Therefore, we compute MHI and MEI with four different lengths ($\tau = 5, 10, 15, 20$).

Image sequence registration is a problem when using this method to compare motion patterns in two sequences. We overcome this problem by shifting one sequence and computing motion similarity for every possible sequence alignment. The measurement of the best similarity in temporal templates for the eventual similarity integration is chosen.

6.2.4 Image Dynamics

We model image-level dynamics in image subspace, which is similar to Soatto's [1] and Brand's [66] approaches. The image subspace is spanned by a set of basis images. The input image sequence is projected onto the subspace frame by frame and the projections form a trajectory in the subspace. We model the evolution of this trajectory using a first-order auto-regressive (AR) model. Therefore, the temporal behavior of an image sequence is captured by the evolution of the moving trajectory in image subspace.

Assume that we have n frames in an image sequence, and each image frame of the sequence is represented as a column vector $I_i \in \mathbf{R}^m$ in the raster scan order. Let μ be the mean of the images and $I'_i = I_i - \mu$. We use a matrix $X = [I'_1 I'_2 \dots I'_n]$ to denote the whole input image sequence around the mean image. Using the algorithm in [67], we find the eigenvectors $\{e_j\}_{j=1\dots k}$, which correspond to the largest k eigenvalues, of the covariance matrix XX^T. Therefore, we

represent each image frame as $I_i = VP_i + \mu$, where $V = [e_1 e_2 \dots e_k]$ and $P_i = V^T(I_i - \mu)$. P_i is the projection of I_i in the subspace spanned by V. Furthermore, we treat the projections P_i as the k-dimensional random vectors observed at equal time intervals. The first-order k-variate AR model is defined as $P_i = AP_{i-1} + n_i$. The matrices $A \in \mathbf{R}^{k \times k}$ are the coefficient matrices of the AR model, and the k-dimensional vectors n_i are uncorrelated random noise with zero mean. Note that the AR model for each sequence are defined in different subspaces. Therefore, to measure the similarity between image sequences, we compute Martin's distance between AR models defined by $\{A, V\}$ pairs [68].

6.2.5 Integration of Similarity Measures

The respective similarity measures for three motion representations are all integrated to measure motion-content similarity in the circumscribing boxes of dynamic objects. Since the quality of retrieval results is subjective to user's visual perception, the ways to integrate different similarity measures may vary depending on the dynamic object of query. There have been some systems that require users to specify weights for their queries, which often leads to unsatisfactory results. In our system, we linearly combine similarity measures and dynamically adjust their weights according to user's interactive feedback [69]. Such relevance feedback based retrieval approach has been empirically proved to be very effective.

6.3 Experimental Results

The video clips used in our experiments were randomly collected from TV programs or recordings of street scenes. In most cases, the videos involve camera motions. Currently, fifty image sequences have been used in our experiments. All images are converted into gray-level

before we apply the algorithms. The retrieval results of five queries of different motion patterns are shown in figure 6.2.

The first test is a human walking video. Such motion pattern is periodic and has important signatures in motion presence and motion recency, where the MEI shows the motion is around the lower part of the object and the MHI indicates the major movement is toward right. The third best sequence of this query involves walking with an angle to the camera plane, which decreases the similarity in motion presence to the test sequence. The forth best sequence involves walking toward the opposite direction of the test sequence, and the features for motion recency show the differences.

In the second test, we use a tennis video where a player back to the camera performs a right-handed swing including his follow-through. This is a full body motion, but the motion in image context has emphasis on arm swing (including the racket) and leg movement. The similarity measures in motion presence and motion recency are most relevant in retrieving similar videos. Note that the videos of left-handed swings or swings of a player facing the camera are not retrieved.

The third test video is about the movement of a bird's wings. Although the motion of flapping wings is periodic, in most cases the movement is too fast for the system to detect its periodicity. As a result, motion presence is much more relevant to such fast movement in the retrieval process than the other properties. The forth test is a flowing river sequence. Our system relies on image-level dynamics to retrieve similar videos of such a no-where static scene.

The last test is a video of a moving car exhibiting a rigid motion. Our system is able to separate rigid motion from non-rigid motion, but has no discrimination among rigid objects. The system retrieves vehicle sequences because they are the only rigid objects in our database.

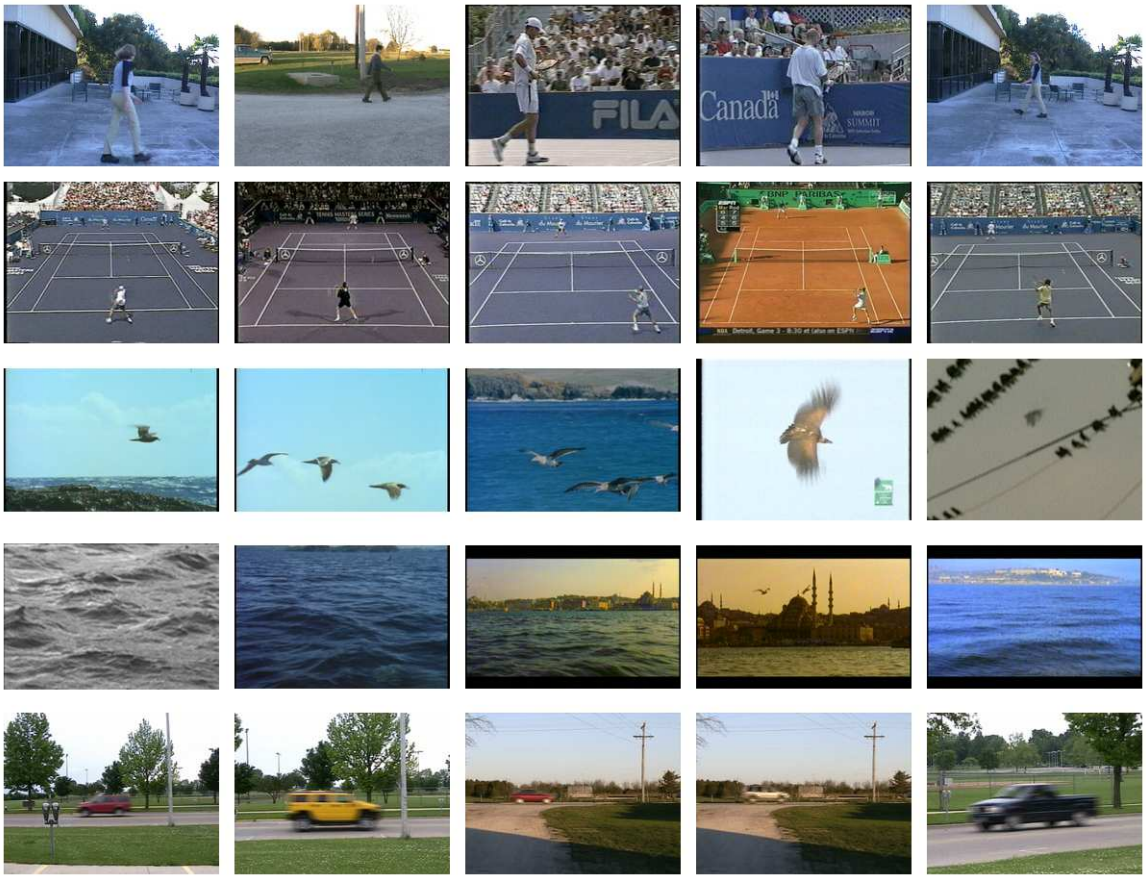


Figure 6.2 In each row, the leftmost image is an example image frame of the query sequence of a dynamic object. The corresponding four most similar sequences are shown in the right.

6.4 Discussion

In this chapter, we propose to use 2D motion models to retrieve dynamic objects in videos. The motion content in terms of changes in the image context of the circumscribing box of dynamic objects is considered. Although a few assumptions have been made to implement the system, the results suggest that higher-level motion representations certainly help to retrieve a wide range of similar motion patterns. Other video contents such as color and shape are not considered in this work, though using them will surely improve overall performance. For videos of multiple moving objects, the relationships between the corresponding circumscribing boxes

can be further explored so that a query with higher-level concept such as "object A chasing object B" can be answered. Future research should include other motion representations in order to cover more real-world motion patterns.

CHAPTER 7

CONCLUSIONS

In this thesis, we have considered the problem of modeling and analysis of continuous, locally-linear, multi-dimensional spatio-temporal data. Our work extends the previously reported theoretical work on the global coordination model to temporal analysis of continuous, multi-dimensional data. We have developed algorithms for time-varying data analysis and used them in full-scale, real-world applications. The applications demonstrated in this thesis include tracking, synthesis, recognitions and retrieval of dynamic objects based on their shape, appearance and motion. Experiments show that the new modeling features of our approach improve the performance of existing approaches in most applications.

A major direction for future research beyond this thesis will be to exploit the temporal correlation of the input data to learn their nonlinear manifolds and mappings. Most applications shown in this thesis involve time series analysis, but the temporal correlation of the data is currently not used for nonlinear manifold learning. Although promising results have been demonstrated using our current approach, using temporal correlation might improve manifold learning and further enhance the performance of our algorithms.

The proposed approach in this thesis has advantages over existing approaches to analyzing complex spatio-temporal data. We believe it would find more applications in the future.

REFERENCES

[1] S. Soatto, G. Doretto, and Y. Wu, "Dynamic textures," in *IEEE International Conference on Computer Vision*, 2001, vol. 3, pp. 439–446.

[2] Andrew Fitzgibbon, "Stochastic rigidity: Image registration for Nowhere-Static scenes," in *IEEE International Conference on Computer Vision*, 2001, pp. 662–669.

[3] Martin Szummer and Rosalind W. Picard, "Temporal texture modeling," in *IEEE International Conference on Image Processing*, 1996, vol. 3, pp. 823–826.

[4] Che-Bin Liu and Narendra Ahuja, "A model for dynamic shape and its applications," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2004, vol. 2, pp. 129–134.

[5] Che-Bin Liu and Narendra Ahuja, "Motion based retrieval of dynamic objects in videos," in *ACM Multimedia*, 2004, pp. 288–291.

[6] Sam Roweis, Lawrence Saul, and Geoffrey E. Hinton, "Global coordination of local linear models," in *Neural Information Processing Systems*, 2001, vol. 14, pp. 889–896.

[7] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec 2000.

[8] Sam Roweis and Lawrence Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec 2000.

[9] Michael E. Tipping and Christopher M. Bishop, "Mixtures of probabilistic principal component analysers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.

[10] Yee Whye Teh and Sam Roweis, "Automatic alignment of local representations," in *Neural Information Processing Systems*, 2002, vol. 15, pp. 841–848.

[11] Matthew Brand, "Charting a manifold," in *Neural Information Processing Systems*, 2002, vol. 15, pp. 961–968.

[12] Zoubin Ghahramani and Geoffrey E. Hinton, "Variational learning for switching state-space models," *Neural Computation*, vol. 12, pp. 831–864, 2000.

[13] Yan Li, Tianshu Wang, and Heung-Yeung Shum, "Motion texture: A two-level statistical model for character motion synthesis," in *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002, pp. 465–472.

[14] H. Murase and Shree K. Nayar, "Visual learning and recognition of 3D objects from appearance," *International Journal of Computer Vision*, vol. 14, no. 1, pp. 5–24, 1995.

[15] Shai Avidan, "Support vector tracking," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2001, vol. 1, pp. 184–191.

[16] Zia Khan, Tucker Balch, and Frank Dellaert, "An MCMC-based particle filter for tracking multiple interacting targets," in *Proceedings of the Eighth European Conference on Computer Vision*, Tomás Pajdla and Jiři Matas, Eds., 2004, LNCS 3024, pp. 279–290.

[17] Michael J. Black and Alan D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.

[18] Greg Hager and Peter Belhumeur, "Real-time tracking of image regions with changes in geometry and illumination," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1996, pp. 403–410.

[19] Zia Khan, Tucker Balch, and Frank Dellaert, "A Rao-Blackwellized particle filter for eigentracking," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2004, vol. 2, pp. 980–986.

[20] Michael E. Tipping and Christopher M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society, Series B*, vol. 61, no. 3, pp. 611–622, 1999.

[21] Qiang Wang, Guangyou Xu, and Haizhou Ai, "Learning object intrinsic structure for robust visual tracking," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2003, vol. 2, pp. 227–233.

[22] V. Pavlovic, J. M. Rehg, and J. MacCormick, "Learning switching linear models of human motion," in *Neural Information Processing Systems*, 2000, vol. 13.

[23] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa, "Video textures," in *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, July 2000, pp. 489–498.

[24] Li-Yi Wei and Marc Levoy, "Fast texture synthesis using tree-structured vector quantization," in *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 479–488.

[25] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Transactions on Graphics, SIGGRAPH 2003*, vol. 22, no. 3, pp. 277–286, July 2003.

[26] Neill W. Campbell, Colin Dalton, David Gibson, and Barry Thomas, "Practical generation of video textures using the auto-regressive process," in *British Machive Vision*, 2002, pp. 434–443.

[27] Lu Yuan, Fang Wen, Ce Liu, and Heung-Yeung Shum, "Synthesizing dynamic texture with closed-loop linear dynamic system," in *ECCV*, 2004, vol. 2, pp. 603–616.

[28] Okan Arikan and David A. Forsyth, "Interactive motion generation from examples," in *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002, pp. 483–490.

[29] Lucas Kovar, Michael Gleicher, and Frédéric Pighin, "Motion graphs," in *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002, pp. 473–482.

[30] Michael Gleicher, Hyun Joon Shin, Lucas Kovar, and Andrew Jepsen, "Snap together motion: Assembling run-time animation," in *Proceedings of the 2003 Symposium on Interactive 3D graphics*, 2003, pp. 181–188.

[31] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard, "Interactive control of avatars animated with human motion data," in *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002, pp. 491–500.

[32] Okan Arikan, David A. Forsyth, and James F. O'Brien, "Motion synthesis from annotations," *ACM Trans. Graph. (SIGGRAPH 2003)*, vol. 22, no. 3, pp. 402–408, 2003.

[33] Richard Bowden, "Learning statistical models of human motion," in *IEEE Workshop on Human Modelling, Analysis and Synthesis*, 2000.

[34] L. M. Tanco and A. Hilton, "Realistic synthesis of novel human movements from a database of motion capture examples," in *Proceedings of the Workshop on Human Motion*, 2000, pp. 137–142.

[35] Martin Szummer, "MIT temporal texture database," ftp://whitechapel.media.mit.edu/pub/szummer/temporal-texture/.

[36] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, pp. 461–464, 1978.

[37] Sven Loncaric, "A survey of shape analysis techniques," *Pattern Recognition*, vol. 31, no. 8, pp. 983–1001, 1998.

[38] T. Pavlidis, "A review of algorithms for shape analysis," *Computer Graphics and Image Procesing*, vol. 7, pp. 243–258, 1978.

[39] Anuj Srivastava, Washington Mio, Eric Klassen, and Xiuwen Liu, "Geometric analysis of constrained curves for image understanding," in *Proc. Second IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, 2003.

[40] Michael Isard and Andrew Blake, "Contour tracking by stochastic propagation of conditional density," in *European Conference on Computer Vision*, 1996, vol. 1, pp. 343–356.

[41] D. Terzopoulos and R. Szeliski, "Tracking with kalman snakes," in *Active Vision*, A. Blake and A. Yuille, Eds., pp. 3–20. MIT Press, Cambridge, MA, 1992.

[42] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, pp. 321–331, 1987.

[43] A. Blake, R. Curwen, and A. Zisserman, "Affine-invariant contour tracking with automatic control of spatiotemporal scale," in *ICCV*, 1993, pp. 66–75.

[44] A. Pentland and S. Sclaroff, "Closed-form solutions for physically based shape modeling and recognition," *IEEE Trans. on PAMI*, vol. 13, no. 7, pp. 715–729, 1991.

[45] D. Terzopoulos and D. Metaxas, "Dynamic 3d models with local and global deformations: deformable superquadrics," *IEEE Trans. on PAMI*, vol. 13, no. 7, pp. 703–714, 1991.

[46] Anthony J. Yezzi and Stefano Soatto, "Deformotion: Deforming motion, shape average and the joint registration and approximation of structures in images," *International Journal Comput. Vision*, vol. 53, no. 2, pp. 153–167, 2003.

[47] Shunji Mori, Hirobumi Nishida, and Hiromitsu Yamada, *Optical Character Recognition*, John Wiley & Sons, 1999.

[48] E. Persoon and K. Fu, "Shape discrimination using fourier descriptors," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, no. 3, pp. 170–179, March 1977.

[49] Helmut Lütkepohl, *Introduction to Multiple Time Series Analysis*, Springer-Verlag, 1991.

[50] Arnold Neumaier and Tapio Schneider, "Estimation of parameters and eigenmodes of multivariate autoregressive models," *ACM Transactions on Mathematical Software*, vol. 27, no. 1, pp. 27–57, 2001.

[51] William D. Davis and Kathy A. Notarianni, "NASA fire detection study," Tech. Rep., National Institute of Standards and Technology, March 1996.

[52] G. Healey, D. Slater, T. Lin, B. Drda, and D. Goedeke, "A system for real-time fire detection," in *Computer Vision and Pattern Recognition*, 1993, pp. 605–606.

[53] W. Phillips, III, M. Shah, and N. da Vitoria Lobo, "Flame recognition in video," in *Fifth IEEE Workshop on Applications of Computer Vision*, December 2000, pp. 224–229.

[54] Ming-Hsuan Yang and Narendra Ahuja, "Gaussian mixture model for human skin color and its application in image and video databases," in *Conference on Storage and Retrieval for Image and Video Databases (SPIE 99)*, 1999, vol. 3656, pp. 458–466.

[55] Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, second ed., 1999.

[56] Che-Bin Liu and Narendra Ahuja, "Vision based fire detection," in *17th International Conference on Pattern Recognition*, 2004, vol. 4, pp. 134–137.

[57] W. T. Reeves, "Particle systems – a technique for modeling a class of fuzzy objects," *ACM Transactions on Graphics*, vol. 2, pp. 91–108, April 1983.

[58] K. Sims, "Particle animation and rendering using data parallel computation," *ACM Computer Graphics (SIGGRAPH '90)*, vol. 24, no. 4, pp. 405–413, 1990.

[59] Jos Stam and Eugene Fiume, "Depicting fire and other gaseous phenomena using diffusion processes," *Proceedings of ACM SIGGRAPH*, pp. 129–136, 1995.

[60] F. Leymarie and M. Levine, "Tracking deformable objects in the plane using an active contour model," *IEEE Trans. on PAMI*, vol. 15, no. 6, pp. 617–634, 1993.

[61] J. Davis and A. Bobick, "The representation and recognition of action using temporal templates," in *Computer Vision and Pattern Recognition*, 1997, pp. 928–934.

[62] R. Cutler and L. Davis, "Robust periodic motion and motion symmetry detection," in *Computer Vision and Pattern Recognition*, 2000, pp. II:615–622.

[63] C. Kambhamettu, D. B. Goldgof, D. Terzopoulos, and T. S. Huang, "Nonrigid motion analysis," in *Handbook of PRIP: Computer Vision*, vol. 2, pp. 405–430. Academic Press, 1994.

[64] J. P. Lewis, "Fast normalized cross-correlation," *Vision Interface*, pp. 120–123, 1995.

[65] A. Bobick and J. Davis, "Real-time recognition of activity using temporal templates," in *IEEE Workshop on Applications of Computer Vision*, 1996.

[66] Matthew Brand, "Subspace mappings for image sequences," *Statistical Methods in Video Processing*, 2002.

[67] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[68] P. Saisan, G. Doretto, Ying Nian Wu, and S. Soatto, "Dynamic texture recognition," in *Computer Vision and Pattern Recognition*, 2001, vol. 2, pp. 58–63.

[69] Y Rui, Thomas S. Huang, M. Ortega, and S Mehrotra, "Relevance feedback: A power tool for interactive content-based image retrieval," *IEEE Tran. Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 644–655, 1998.

AUTHOR'S BIOGRAPHY

Che-Bin Liu received the BS degree in computer science and information engineering from the National Taiwan University in 1996. In 1998, he attended the Department of Computer Science at the University of Illinois at Urbana-Champaign. He was a research assistant under Prof. Narendra Ahuja in the Computer Vision & Robotics Lab at the Beckman Institute from 1999 to 2005. He is currently a senior technical staff at Epson Palo Alto Lab. His research interests include computer vision, computer graphics, image processing, and pattern recognition.