# Dynamic Texture Synthesis in Image and Video Processing

**XU, Leilei**

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

Abstract of thesis entitled:

Dynamic Texture Synthesis in Image and Video Processing

Submitted by XU Leilei

for the degree of Master of Philosophy

at The Chinese University of Hong Kong in September 2007

Dynamic textures are representations of such textured surfaces with repetitive, time-varying visual patterns which form an image sequence that exhibits certain stationarity properties in time. These include sea-waves, smoke, foliage, whirlwind, dense crowds, and traffic scenes. The aim of dynamic texture synthesis in our work is to learn some model which best represents the given sample textures, and then unseing the learnt model to predict and extrapolating the finite dynamic textures to an infinite one sharing similar dynamic behaviors with the original input.

We first propose the linear dynamic system (LDS) model from the gray level video sequence to colored dynamic textures by compressing the color channels in the YUV color space using laplacian pyramid and radial basis functions (RBF). The dynamic color texture synthesis model we propose is memory efficient and has the ability to better capture complex dynamic behaviors. Further, we develop the multi-resolution auto-regressive moving average model (MARMA) based on the first order linear dynamic system by transforming the dynamic textures into different frequency scales using multi-resolution analysis descriptors. We perform the comparatively study using three representative multi-resolution models: Laplacian pyramid, haar wavelet and the steerable pyramid. The multi-resolution analysis method forms a multi-level LDS model which not only improves the synthesis algorithm, but also shed a light on dynamic texture comparison. We also extend the current work to motion transfer using LDS model. We define the objective functions in appearance and dynamic behavior domains using steerable pyramid and KL divergence separately for solving the problem. The work needs further efforts on how to solve the optimization problem.

In summary, the dynamic color texture synthesis model we propose can deal with large size video sequence ; the synthesized results are as good as the LDS model applies in the RGB color space. Especially for the RBF based model, it is not only memory efficient, but also capturing the structural information better than the previous RGB color approach for the complex dynamic color textures. The multi-resolution auto-regressive moving average model can capture the local and global dynamic behaviors in different scales thus causes a better synthesized result than the LDS model. Our experiments show that steerable pyramid performs better than other multi-resolution analysis methods in our MARMA model. The dynamic textures models we developed can be widely applied in digital entertainment, interactive video gaming, and augmented virtual environments.

# 圖像視頻處理中的動態紋理模擬

徐磊磊

## 摘要

動態紋理是指那些由一些具有重複的、並隨時間變化的圖案形成的紋理而組成的圖像序列。這種紋理的圖像序列在時間域上具有某種不變特性。動態紋理包括海浪、飄動的煙、舞動的樹葉、吹動的風、擁擠的人群以及交通狀況等。動態紋理模擬的目的就是根據一小段輸入的示例紋理視頻，學習該段運動紋理的參數模型從而預測並將原始的有限長度的動態紋理視頻擴展成無限長的視頻，同時保證該擴展的紋理序列同原始輸入序列保持相似的運動規律。
論文中首先提出了將線性動力系統擴展到彩色文理演算法。該演算法使用 Laplacian pyramid 和 Radial basis function（RBF）來壓縮 YUV 色彩域中的彩色資訊，從而避免了大規模矩陣的複雜運算，並同時能夠更好的模擬動態紋理的運動特性。其次，我們提出了基於 multi-resolution 分析的 multi-resolution auto-regressive moving average 模型（MARMA）。該模型以線性動力系統爲基礎，使用 multi-resolution 分析方法，將原始的動態紋理序列轉化到不同的頻率範圍裏以改進動態紋理模擬效果。同時，我們也設計了一個比較研究，使用三種不同的 multi-resolution 分析方法：laplacian pyramid，haar 小波以及 steerable pyramid，進行動態紋理模擬。實驗結果顯示，我們的 MARMA 模型可以改善模擬的結果。我們還將目前的工作擴展到使用線性動力系統進行運動轉移。我們在表像域與運動域分別使用 steerable pyramid 和 KL divergence，定義了該問題的目標函數。進一步的工作將放在如何求解全局優化的問題上。
總的來說，我們提出的動態彩色問題模擬模型可以處理大規模的動態問題視頻，同時保證模擬效果與直接應用線性動力系統於 RGB 彩色空間的結果不相上下。特別是對於基於 RBF 的模型，它不僅僅節省存儲空間，同時也能更好的捕捉到動態紋理的結構資訊。MARMA 模型將拒不和全局的動態特性分成不同的層次從而改善了模擬效果。實驗結果顯示，steerable pyramid 比其他 multi-resolution 分析方法能更好的對紋理進行模擬。

# Acknowledgement

I would like to thank my supervisors Prof. Sun and Prof. Jia for suggesting this interesting research topic, providing the initial direction to the solution and helping and encouraging me throughout this work.

I want to thank my friends in the work group, Mr. Xiong Wei, Mr. Zhang Fan, Mr. Tao Chenjun and Mr. Zhao Chong, for giving me all those valuable suggestions. Special thanks for Mr. Shen Jianbin for helping me to finalize the idea. It was really a great experience and pleasure to work with you all.

Finally, I want to thank my family for supporting me. Special thanks to my husband.

This work is dedicated to my dear husband for encouraging me throughout the process.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The topics in *Texture* analysis and its applications have a long history in both computer graphics and computer vision societies. With the fast developing of computer hardware, especially on graphics cards, realistic rendering of dynamic textures are more commonly used in computer games, special effects in TV/film producing, and rich-media digital entertainment to create virtual environments which photo-realistically simulate the natural phenomenon. Figure 1.1 shows some examples of natural scenes containing textures. In this section we begin with the definition of texture and expand the 2D textures to the 3D textures, which we call the *Dynamic textures*. And then review the previous work on dynamic textures.

## 1.1 Texture and Dynamic Textures

Although texture has been studies extensively, there is still no general and precise definition for texture [9]. We list several representatives adopted in computer vision society here:

- We may regard texture as what constitutes a macroscopic region. Its structure is simply attributed to the repetitive patterns in which elements or primitives are arranged according to a placement rule. [47]

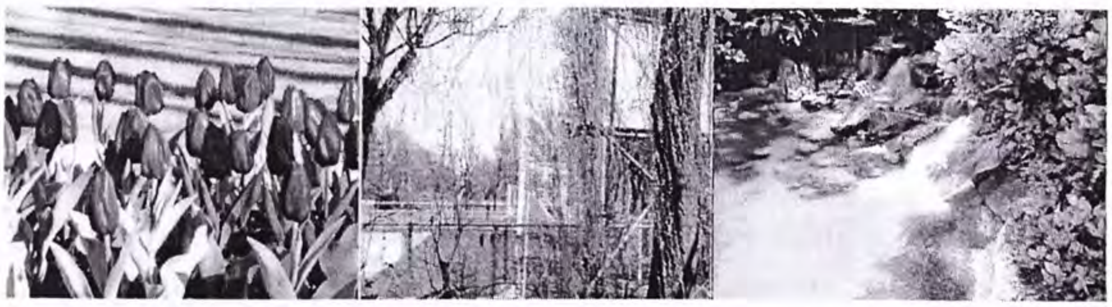

Figure 1.1: Flowers blowing, tree leaves, and the spring water are all representatives of dynamic textures in the natural scenes.

- A region in an image has a constant texture if a set of local statistics or other local properties of the picture function are constant, slowly varying, or approximately periodic. [42]
- The image texture we consider is nonfigurative and cellular...An image texture is described by the number and types of its (tonal) primitives and the spatial organization or layout of its (tonal) primitives... A fundamental characteristic of texture: it cannot be analyzed without a frame of reference of tonal primitive being stated or implied. For any smooth gray-tone surface, there exists a scale such that when the surface is examined, it has no texture. Then as resolution increases, it takes on a fine texture and then a coarse texture. [20]
- Texture is defined for our purposes as an attribute of a field having no components that appear enumerable. The phase relations between the components are thus not apparent. Nor should the field contain an obvious gradient. The intent of this definition is to direct attention of the observer to the global properties of the display i.e., its overall coarseness, bumpiness, or fineness. Physically, nonenumerable (aperiodic) patterns are generated by stochastic as opposed to deterministic processes. Perceptually, however,

the set of all patterns without obvious enumerable components will include many deterministic (and even periodic) textures. [36]

- Texture is an apparently paradoxical notion. On the one hand, it is commonly used in the early processing of visual information, especially for practical classification purposes. On the other hand, no one has succeeded in producing a commonly accepted definition of texture. The resolution of this paradox, we feel, will depend on a richer, more developed model for early visual information processing, a central aspect of which will be representational systems at many different levels of abstraction. These levels will most probably include actual intensities at the bottom and will progress through edge and orientation descriptors to surface, and perhaps volumetric descriptors. Given these multi-level structures, it seems clear that they should be included in the definition of, and in the computation of, texture descriptors.[60]
- The notion of texture appears to depend upon three ingredients: (i) some local order is repeated over a region which is large in comparison to the orders size, (ii) the order consists in the nonrandom arrangement of elementary parts, and (iii) the parts are roughly uniform entities having approximately the same dimensions everywhere within the textured region. [21]

These definitions are defined by different authors on different applications, so their focus on the definition of texture are deferred. The definition we are using in this paper is borrowed from Dorreto. et. al [11]: A texture is a realization from a stationary (informally, a signal is stationary if the statistics it exhibits in a region is invariant to the region's location) stochastic process with spatially invariant statistics.

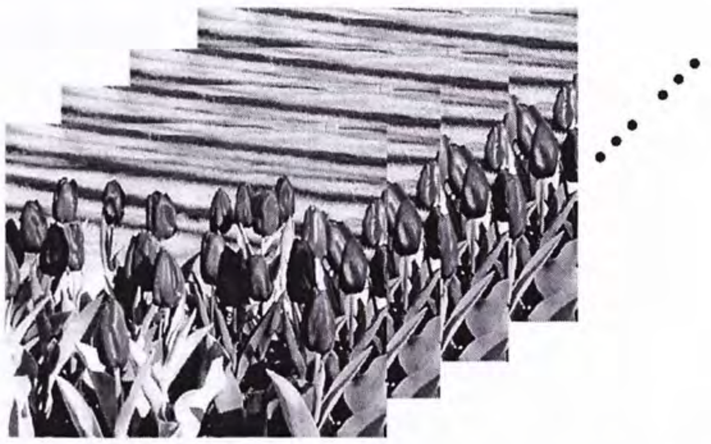
Figure 1.2: The tulips video sequence captured.

The expansion from 2D textures to 3D is natural ( Figure 1.2). Natural phenomenons, such as flowing water, blowing tree leaves, burning fires and so on are called dynamic textures. We can regard dynamic textures as the expansion of 2D textures into time domain. Formally, *Dynamic Textures* [43], also called temporal textures [46] or video textures [5] can be defined as repetitive patterns exhibiting both spatial and temporal coherence intrinsic in the process. It is the multidisciplinary research topic interested in graphics and vision communities.

## 1.2 Related work

In recently years, the study of dynamic textures has stimulated a growing interest in both computer graphics and vision communities. Plenty of methodologies, applications are proposed which can be found in almost every core journals and conference papers.

#### Graphics Methods

Computer Graphics methods, or particularly physics-based methods focus on rendering photorealistic video sequences resembling the natural world or nonphotorealistic cartoon dynamic textures which still preserves similar dynamic behaviors to the real world objects. These methods establish physical models using differential equations to render the objective textures. A good example of using physical models is illustrated in [8]. The study in physical model in computer graphics has a long history, and the synthesized results are promisingly in good quality. Also, they have a great potential for dynamic texture editing. But these methods are usually computationally intensive, and models are developed only for specific phenomenons.

#### Vision Methods

Computer vision methods start from a training sequence of dynamic textures video, and new frames are learnt from this. Most recent techniques in vision community can be categorized into *nonparametric* and *parametric* methods [7].

The nonparametric methods directly sample the original information from the input image or sequence of images. Wei and Levoy [52] sample original pixels from the training images using Markov Random Field texture models and generating textures through a deterministic searching process. Video Textures [38] and its extension [39] replay a video by reordering the image frames to achieve smooth transition between adjacent frames to preserve temporal continuity. Wavelet-structures are sampled from the input video using steerable pyramid in [58] to synthesize 2D and 3D textures. GraphCut Textures [50] copies 2D patches or 3D voxels to synthesize by finding the optimal seam for blending using the state-of-art Graph Cut technique. These nonparametric methods usually provide high quality visual effect but yield a limited amount of information for the intrinsic property of textures.

Parametric models, on the other hand provides better generalization and understanding of perceptual process, thus these models are nature choices for analysis and controlling of textures. Wang and Zhu [51], [57] view the images and videos as a superposition of image or video basis which are called *texton* and *movton* separately by applying different kernel functions at various locations, orientations and scales from the kernel bank in order to detect the two kinds of "features". This is useful for matching, tracking and other applications in texture or dynamic texture. Szummer and Picard [45] regard the motion as *temporal texture* and model the interaction of pixels within a local neighborhood over both space and time by adopting the spatio-temporal autoregressive(STAR) model. By relying on the spatio-temporally localized image features, these representations are incapable of abstracting the video into a pair of holistic appearance and motion components. The problem is addressed in [11] by providing an autoregressive random process (specifically, a *linear dynamic system*, LDS, also known as Kalman filter [54] model) using models and tools from control theory and provide a fast closed form solution for learning and synthesis dynamic textures. Fitzgibbon [15] added the rigid camera motion in combination with the stochastic motion patterns, so that the motion is registered properly. While Linear system is easy to model and calculate, other works, such as [29], [16], [30], model the dynamic system, which is truly a nonlinear one, as a combination of multiple linear dynamic systems by using an transition matrix indicating the likelihood of switching from one LDS to another. Yuan et.al [32] improve the visual quality of the synthesized result of Dorreto's by adding feedback controls and extending the LDS model to a closed-loop LDS model. Filip et. al [14] address the special problem of synthesizing dynamic color textures by analyzing the eigen-system of dynamic texture images and subsequent preprocessing and modelling of temporal interpolation eigen-coefficients using a causal auto-regressive model in RGB color space. Generally, these kind of parametric methods are less likely to generate dynamic textures with the same quality as the nonparametric models, but parametric models indeed provide useful information to tasks like dynamic texture segmentation [12], recognition [55], registration [15] and manipulation [13]. Most of parametric models are in space domain while still some of the models work in frequency domain based on the observation that working in frequency domain models take full advantage of spatial correlation than working in space domain [1].

## 1.3 Thesis Outline

The rest of the thesis is organized as follows. In Section 3 we review the linear dynamic system model, its formulation, learning and synthesis process, which is form the basis of all our methods. In Section 4 we present a novel approach for dynamic color synthesis in the YUV color-space. In Section 5, we conduct a comparative study on different multi-resolution analysis techniques based on the multi-resolution auto-regressive moving average (MARMA) model we present. Section 6 discusses the motion transfer framework while conclusion is drawn in Section 7.

---

□ **End of chapter.**

# Chapter 2

# Image/Video Processing

Video analysis, also called image sequence analysis, is based on processing in groups of two frames, which is the overwhelming majority of studies to date. Differences between two related frames are used to infer the dynamics occurring in the video, or help on extracting the main object out of an image sequence, or video compression and so on.

In this chapter, we will review some of the commonly used techniques in image/video processing.

## 2.1 Bayesian Analysis

Because of inevitable uncertainties in vision processes, principles from statistics, probability and information theory are often used as the formal basis. Generally, when we have the knowledge about the data distribution but no appreciable prior information about the quantity being estimated, we may use the *maximum likelihood (ML)* criterion [35]; when the situation is the opposites, i.e., when we have only prior information, then we may use the *maximum entropy* criterion; when we know both prior and likelihood distributions which is the usual situation in vision processing, we can compute the *maximum of a posterior* (MAP for short) probability. The Bayesian criterion permits to express this conditional probability as multiplying the *prior* term and *likelihood* term.

According to Bayes rule, the posterior probability of the solution $f$ (such as in image segmentation, $f$ is the optimal partition) w.r.t the observed data $d$ ($d$ can be any observed data, such as the image, the video, or the features got from the original data) can be computed by using the following equation

$$P(f|d) = \frac{p(d|f)P(f)}{p(d)} \tag{2.1}$$

where $P(f)$ is the prior probability of solution $f$, $p(d|f)$ is the conditional probability density of the observations $d$, also called the *likelihood* function of $f$ for $d$ fixed, and $p(d)$ is the density of $d$ which is a constant when d is given, so equation 2.1 can be written as below

$$P(f|d) \propto P(f,d) = p(d|f)P(f) \tag{2.2}$$

Where the probability density is represented by the joint probability. Then the MAP estimate is equivalently found by

$$f^* = arg\ max_{f \in F}\{p(d|f)P(f)\} \tag{2.3}$$

where $F$ is the set of all possible configurations of $f$. Thereby separating image-based cues (fist term) form geometric properties (second term). The Bayesian framework has become increasingly popular to tackle many ill-posed problems in computer vision. Firstly, the conditional probability $p(d|f)$ of an observation given a model state is often easier to model than the posterior distribution, it typically follows from a generative model of the image formation process. Secondly, the term $p(f)$ in ( 2.3) allows to introduce prior knowledge stating which interpretations of the data are *a priori* more or less likely. Whenever available, such *a priori* knowledge may help to cope with missing low-level information.

## 2.2 Markov Property

In image/video processing, the content of each pixel, area, or frame in the image or image sequence is not independently or insolated occurred. There is strong relationship between the neighborhood pixels and the selected pixel, between the surrounding area and the selected area, and between the previous frames and the current frames. *Markov Chains* and its generalization *Markov Random Fields* are used to formally describe this property.

Markov random field (MRF) provides a convenient and consistent way of modelling the *a prior* probability of some context dependent entities such as image pixels and other spatially correlated features. By relating MRFs with GIBBS distributions, several vision problems could be modelled in a mathematically sound and tractable way in the Bayesian framework.

The definition of Markov random field is like this: First, there is a set $S$, a family of random variables $F = \{F_1, \cdots, F_m\}$, and a a neighborhood system $N$ for the set $S$. Each random variable $F_i$ of the family $F$, which is also called a random field, takes a value $f_i$ in $L$. The configuration $f$ is defined as $f = \{f_1, \cdots, f_m\}$. The neighborhood system $N$ is defined as $N = \forall i \in S$ where $N_i$ is the set of sites neighboring $i$. There are two properties of the neighboring relationship,

1. a site $i$ is not neighboring to itself: $i \notin N_i$
2. the neighboring relationship is mutual: $i \in N_i' \Leftrightarrow i' \in N_i$

Then $F$ is said to be a Markov random field on $S$ with respect to a neighborhood system $N$ if and only if the following two conditions are satisfied:

$$P(f) > 0, \forall f \in \mathbf{F} \text{ (poritivity)} \tag{2.4}$$

$$P(f_i | f_{\mathcal{S}-\{i\}}) = P(f_i | f_{\mathcal{N}_i}) \text{ (Markovianity)} \tag{2.5}$$

where $s-\{i\}$ is the set difference, $f_{s-\{i\}}$ denotes the set of labels at the sites in $s-\{i\}$ and $f_{N_i} = \{f_{i'} \mid i' \in N_i\}$ stands for the set of labels at the sites neighboring $i$ [26].

Markov random field is a useful tool on image segmentation, but it is hard to determine the joint probability from the definition directly. The Hammersley-Clifford theorem [19] establishes the equibalence of these two types of properties. The theorem sates that *$\mathcal{F}$ is an MRF on $\mathcal{S}$ with respect to $\mathcal{N}$ if and only if $\mathcal{F}$ is a GRF on $\mathcal{S}$ with respect to $\mathcal{N}$.*

A set of random variables $\mathcal{F}$ is said to be a *Gibbs random field (GRF)* on $\mathcal{S}$ with respect to $\mathcal{N}$ if and only if its configurations or labeling obey a *Gibbs distribution*:

$$P(f) = Z^{-1} \times e^{-\frac{1}{T}\mathcal{U}(f)} \tag{2.6}$$

where

$$Z = \sum_{f \in \mathcal{F}} e^{-\frac{1}{T}\mathcal{U}(f)} \tag{2.7}$$

is a normalizing constant called the *partition function.* $T$ is a constant called the *temperature* which shall be assumed to be 1 unless otherwise stated, and $\mathcal{U}(f)$ is the energy function. The energy

$$\mathcal{U}(f) = \sum_{x \in \mathcal{C}} V_c(f) \text{or} \tag{2.8}$$

$$\mathcal{U}(f) = \sum_{\{i\} \in \mathcal{C}_1} V_1(f_i) + \sum_{\{i,i'\} \in \mathcal{C}_2} V_2(f_i, f_{i'}) + \sum_{\{i,i',i''\} \in \mathcal{C}_3} V_3(f_i, f_{i'}, f_{i''}) + \cdots \tag{2.9}$$

is a sum of *clique potentials* $V_c(f)$ over all possible cliques $\mathcal{C}$. The value of $V_c(f)$ depends on the local configuration on the clique $c$.

An MRF is characterized by its local property whereas a GRF is characterized by its global property. And the joint probability in MRF can be calculated from GRF by specifying the clique potential function $V_c(f)$ and chooses appropriate potential functions for desired system behavior.

## 2.3 Graph Cut

Robust Graph Cut algorithm is first presented by Yuri Boykov [4]. Graph Cut algorithm is used to solve these vision problems that can be formulated in terms of energy minimization, i.e. one seeks the solution $f$ that minimizes the energy 2.8. And this is just consistent with equation 2.2, in which $p\{d|f\}$ measures the disagreement between the hidden variable $f$ and the observed data $d$ while $p(f)$ measures the extent to which $f$ is not piecewise smooth.

Typically, the form of $E_{data}$ is:

$$E_{data}(f) = \Sigma_{i \in \mathcal{S} D_i(f_i)} \tag{2.10}$$

where $D_i$ measures how appropriate a label $f_i$ is for the site $i$ given the observed data.

And $E_{smooth}$ is typically considered as the form below

$$E_{smooth} = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) \cdot \delta(f_p, f_q) \tag{2.11}$$

The choice of $E_{smooth}$ is a critical issue as *discontinuity-preserving* has to be considered seriously together with smoothness constraints, otherwise $E_{smooth}$ will makes $f$ smooth everywhere even when at the boundary conditions. Function $\delta$ here just acts as the discontinuity preserving parameter. Most commonly, $\delta$ is defined as

$$\delta(f_p, f_q) = \begin{cases} 1 & \text{if } f_p \neq f_q \\ \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

A growing number of publications in vision use graph-based energy minimization techniques for applications like image segmentation [28], restoration [18], stereo [3], augmented reality [48],and others. The graphs corresponding to these applications are usually huge 2D or 3D grids, and min-cut/max-flow algorithm efficiency is an issue that cannot be ignored. [2] gives an efficient algorithm to compute the min-cut/max-flow.

## 2.4 Belief Propagation

In the literature of probabilistic graph models, a Markov network is an undirected graph as shown in Figure 2.1. Nodes $f_p$ are hidden variables and nodes $d_p$ are observed variables. By denoting $F = \{f_p\}$ and $D = \{d_p\}$, the posterior $P(F|D)$ can be factorized as according to MAP-MRF framework and GRF (2.2):

$$P(F|D) \propto \prod_p \psi_p(f_p, d_p) \prod_p \prod_{q \in N(p)} \psi_{pq}(f_p, f_q) \qquad (2.13)$$

where $\psi_{pq}(f_p, f_q))$ is called the compatibility matrix between nodes $f_p$ and $f_q$, and $\psi_p(f_p, d_p)$ is called the local evidence for node $f_p$. In fact, $\psi_p(f_p, d_p)$ is the observation probability or the likelihood $p(d|f)$ and $\psi_{pq}(f_p, f_q)$ is the a priori in the Bayesian model ( 2.2). If the number of discrete states of $f_p$ is $L$, $\psi_{pq}(f_p, f_q)$ is an $L \times L$ matrix and $\psi_p(f_p, d_p)$ is a vector with $L$ elements. We take the negative log properties of 2.13 and then obtain the same form as in 2.8. We use this formulation because it is lest sensitive to numerical artifacts, and uses the energy function definition more directly.

BP approach can be used to find an approximate minimum cost labeling of energy functions in the form of equation 2.8 [53].The max-product BP algorithm works by passing messages around the graph defined by the four-connected image grid. Each message is a vector of dimension given by the number of possible labels. Let $m_{pq}^t$ be the message that node $p$ sends to a neighboring node $q$ at time $t$. When using negative log probabilities just as the MAP-GRF equivalence, all entries in $m_{pq}^0$ are initialized

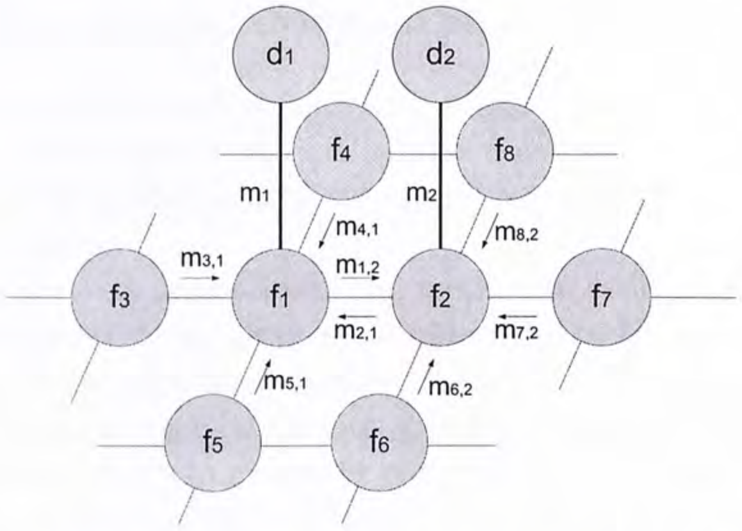

Figure 2.1: Local message passing in a Markov Network. Orange nodes are hidden variables. Green nodes are observable variables. In the "max-product" algorithm, the new message sent from node $f_1$ to $f_2$ is: $m_{1,2}^{new} \leftarrow k\max_{f_1} \varphi_{12}(f_1, f_2)m_1 m_{3,1} m_{4,1} m_{5,1}$. The belief at node $x_1$ is computed as: $b_1 \leftarrow k m_1 m_{2,1} m_{3,1} m_{4,1} m_{5,1}$.

to zero, and at each iteration new messages are computed in the following way,

$$m_{pq}^{t}(f_q) = \min_{f_p}(V(f_p, f_q) + D_p(f_p) + \sum_{s\in\mathcal{N}(p)\backslash q} m_{sp}^{t-1}(f_p)) \quad (2.14)$$

where $\mathcal{N}(p) \setminus q$ denotes the neighbors of $p$ other than $q$. After $T$ iterations, a belief vector is computed for each node,

$$b_q(f_q) = D_q(f_q) + \sum_{p\in\mathcal{N}(q)} m_{pq}^{T}(f_q) \quad (2.15)$$

Finally, the label $f_q^*$ that minimizes $b_q(f_q)$ individually at each node is selected. As BP could approximate the MRF-modelled problem in polynomial time, it has been applied successfully to some vision problems: J.Sun et. al applied BP to solve stereo matching problem [44] by first formulate the problem as an Markov network model and then apply BP to approximate the result.

## 2.5 Expectation-Maximization

An *expectation-maximization* (EM) algorithm is used in statistics for finding maximum likelihood estimates of parameters in probabilistic models, where the model depends on unobserved latent variables.

EM is an iterative optimization method to estimate some unknown parameters $\Theta$, given measurement data $U$. However, we are not given some hidden nuisance variables $J$, which need to be integrated out. In particular, we want to maximize the posterior probability of the parameters $\Theta$ given the data $U$, marginalizing over $J$:

$$\Theta * = \arg\max_{\Theta} \sum_{J \in \mathcal{J}^n} P(\Theta, J|U) \tag{2.16}$$

The intuition behind EM is an old one: alternate between estimating the unknowns $\Theta$ and the hidden variables $J$. This idea has been around for a long time. However, instead of finding the best $J \in \mathcal{J}$ given an estimate $\Theta$ at each iteration, EM computes a distribution over the space $J$.

EM alternates between performing an expectation (E) step, which computes an expectation of the likelihood by including the latent variables as if they were observed, and a maximization (M) step, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found on the E step. The parameters found on the M step are then used to begin another E step, and the process is repeated.

## 2.6 Principle Component Analysis

Principle Component Analysis (PCA) is a useful statistical technique that has found application in fields such as face recognition and image compression and is a common technique for finding patterns in data of high dimension. Data are expressed in a way that their similarities and differences are highlighted. Particularly, when the dimension of the data is high, patterns in data can be hard to find, while PCA is powerful and straight-forward tool for analyzing data.

The most common definition of PCA, due to Hotelling [22], is that, for a set of observed $d$-dimensional data vectors $t_n, n \in \{1 \cdots N\}$, the $q$ principle axes $w_j, j \in \{1 \cdots q\}$, are those orhonormal axes onto which the retained *variance* under projection is maximal. It can be shown that the vectors $w_j$ are given by the $q$ dominant eigenvectors (i.e. those with the largest associated eigenvalues) of the sample covariance matrix $S = \Sigma_n (t_n - \bar{t})(t_n - \bar{t})^T / N$ such that $Sw_j = \lambda_j w_j$ and where $\bar{t}$ is the sample mean. The vector $x_n = W^T(t_n - \bar{t})$, where the orthogonal columns of $W = (w_1, w_2, \cdots, w_q)$ span the space of the leading $q$ eigenvectors of $S$, is thus a $q$-dimensional reduced representation of the observed vector $t_n$.

A complementary property of PCA is that the projection onto the *principle subspace* minimizes the squared reconstruction error $\Sigma \parallel t_n - \hat{t}_n \parallel^2$. The optimal linear reconstruction of $t_n$ is given by $\hat{t}_n = Wx_n + \bar{t}$.

The original definition of PCA is absent of an associated probabilistic model for the observe data, which prohibit the comparison with other probabilistic techniques, while not facilitating statistical testing and also not permitting application of Bayesian methods. The problem is addressed by Tipping [49]. Another disadvantage is that PCA only defines a *linear* projection of the data, which cause the scope of its application somewhat limited. Several *nonlinear* principal component analysis are developed based on the linear one.

---

□ **End of chapter.**

# Chapter 3

# Linear Dynamic System

For a single textured image, it is a realization from a stationary stochastic process with spatially invariant statistics [56]. But for a sequence of time varying textures, it is clear that there is a temporal coherence between neighboring textures in time domain. The modeling of the dynamic system is often very complex and a nonlinear model may elegantly capture the dynamic behaviors more precisely. But for computational feasibility and speed, nonlinear dynamic system models are often mapped to linear models which is easy to solve and fast to process. For most of the dynamic texture models, there is one underlying model which is called *Linear dynamic system.* The aim of dynamic texture synthesis using the linear dynamic system (LDS for short) model is to learn or identifying models parameters that are optimal in the sense of maximum likelihood or minimum prediction error variance, and then using the learnt model parameters to predict and extrapolating the finite sequence of images to an infinite length of dynamic video sequence sharing the similar dynamic behaviors with the original input. The model is elegant and popular because of its simplicity and also its ability in capturing most of the dynamic textures. In this section, we will introduce the model and discuss its properties in details.

## 3.1 System Model

[11] defines that in dynamic textures, individual images are realizations of the output of a dynamical system driven by an independent and identically distributed (IID) process and one of the system that models the process is a second-order stationary process represented by a discrete time LDS with white, zero-mean gaussian noise [31] [40]. The observation or input of the system is a sequence of $\tau$ images represented by matrix $Y = [y(1) \dots y(\tau)] \in \mathcal{R}^{m \times \tau}$ with each image represented by column vector $y(t) \in \mathcal{R}^m$; $X = [x(1) \dots x(\tau)] \in \mathcal{R}^{n \times \tau}$ with $x(t) \in \mathcal{R}^n$ standing for the hidden state vector at time $t$ encoding the evolution of the image sequence and the initial condition $x(0)$ known. Typically, $m \gg n$ and with values of $n$ in the range of 5 to 35. Both the observation and state variables are corrupted by additive gaussian noise, which is also hidden. The basic generative model can be written as:

$$\begin{cases} x(t+1) = Ax(t) + Bv(t) & v(t) \sim \mathcal{N}(0,1) \\ y(t) = Cx(t) + w(t) & w(t) \sim \mathcal{N}(0,R) \end{cases} \tag{3.1}$$

where $A \in \mathcal{R}^{n \times n}$ is the state transition matrix and $C \in \mathcal{R}^{m \times n}$ is the observation matrix which encodes the appearance information. $v(t)$ and $w(t)$ are random variables representing the state evolution and observation noises, respectively, which are independent of each other and the values of $x$ and $y$. The two noise terms are both temporally white and spatially Gaussian distributed with zero mean and covariance matrices, which we denote $Q = BB^T$ and $R$. A dynamic texture model is completely specified using the parameters $\Theta = \{A, C, Q, x(0)\}$.

There are several general properties of the linear dynamic system model which can be strong clues for dynamic texture segmentation, optimization, recognition and manipulation.

## 3.2 Degeneracy and Canonical Model Realization

Notice that there is degeneracy in the model 3.1. The consequence of the degeneracy is that the choice of $\Theta$ is not unique. Suppose we have a solution $\{A, C, Q, x(0)\}$ that could generate the observation $y(t)$. Then it is easily seen that $\{TAT^{-1}, CT^{-1}, TQT^{T}, Tx(0)\}$ can also generate the same sample path $y(t)$, with $T \in \mathcal{R}^{n\times n}$ any invertible matrix.

Reasonable assumptions should be made in order to obtain unique solution to the problem, which is called the *canonical model realization* [24]. One of the canonical models often used is to work with models in which $Q$ is the identity matrix, as all of the structures in matrix $Q$ can be moved into the matrices $A$ and $C$. And also because the swapping of columns of $C$ and $A$ will not affect the final result of $y(t)$, typically, $C$ is chosen an ordering based on the norms of the columns of $C$. Another canonical solution is used in [11] for calculating the closed form solution for LDS model, which we will discuss in detail in Section 3.3.

## 3.3 Learning of Dynamic Textures

The learning of dynamic textures is also called the system identification problem. The learning process of the LDS model in equation 3.1 is to learn the model parameter $\Theta$ given the observation vectors $Y = \{y(1)\cdots y(\tau)\}$. And the problem can be posed as an MAP problem which maximizes the posterior probability of $\max(P(A, C, Q|\{y(t)\}_{t=1\cdots\tau}))$. But as the distribution for $P(A, C, Q)$ are independent and uniformly distributed, the object function could be written as an Maximum Likelihood formulation in which we wish to estimate the model parameters $\Theta$ for which the observed data ($Y = \{y(1)\cdots y(\tau)\}$) are the most likely.

$$\text{given } y(1)\cdots y(\tau), \text{ find}$$

$$\hat{A}, \hat{C}, \hat{Q}, x(0) = \arg\max_{C,A,Q,x(0)} p(y(1), \cdots, y(\tau)) \tag{3.2}$$

$$\text{subject to ( 3.1)}$$

The main idea of the model 3.1 is that the hidden state sequence $x(t)$ should be an informative lower dimensional projection or explanation of the complicated observation sequence $y(t)$. A PCA dimension reduction process is adopted in [11] to reduce the dimension of $y(t)$ to a lower dimension vector $x(t)$ which represent the most informative information of the observed data and forms this closed-form but less optimal solution for the linear system.

First, several assumptions are made in order to get a unique solution 3.2:

$$m >> n; rank(C) = n \tag{3.3}$$

and

$$C^T C = I_n \tag{3.4}$$

Furthermore, the mean value is subtracted from the input sequence in order to ensure the input of the system to have zero mean $\Sigma_t\, y(t) = 0$, by subtracting the mean from each frame. We denote $Y_1^{tau} \doteq [y(1)\cdots y(\tau)] \in \mathcal{R}^{m\times\tau}$ with $\tau > n$, and similarly for $X_1^\tau \doteq [x(1)\cdots x(\tau)] \in \mathcal{R}^{n\times\tau}$. Let $Y_1^\tau = U\Sigma V^T; U \in \mathcal{R}^{m\times n}, U^T U = I_n; V \in \mathcal{R}^{\tau\times n}, V^T V = I_n$ be the *singular value decomposition (SVD)* with $\Sigma = diag\{\sigma_1 \cdots \sigma_n\}$ and $\sigma_i$ be the $i_{th}$ singular values, then( [17])

$$\hat{C}(\tau) = U \tag{3.5}$$

$$\hat{X}(\tau) = \Sigma V^T \tag{3.6}$$

The state matrix $A$ is found as the minimizer

$$A = \arg\min_{A} \parallel X_1^{\tau} - AX_0^{\tau-1} \parallel_F \tag{3.7}$$

which is easily computed. Finally the driving covariance $Q$ is approximated as the sample covariance of the residuals $E_1^{\tau} = X_1^{\tau} - AX_0^{\tau-1}$, given by

$$Q = \frac{1}{\tau - 1} E_1^{\tau} E_1^{\tau T} \tag{3.8}$$

The matlab code for the learning process is given in Table 3.1

The algorithm is fast as the closed-form solution can be found without doing iteration.

## 3.4 Synthesizing Dynamic Textures

After getting the system parameters $\Theta$, we can move backward start from initial state $x(0)$ and the LDS model parameters all the way down to generate new and ideally infinite number of state vectors across the time domain. Then using the appearance model to synthesize and predict new sequence of textured images obtaining the same dynamic behaviors as the original training data.

The Matlab code for the synthesis process of the closed-form solution ( 3.3) is given in Table 3.2. The synthesis process can be done similarly for other methods using EM algorithm as the basic idea.

We summarize the learning and synthesis process of linear dynamic system in Figure 3.1

## 3.5 Summary

In this chapter, we give a brief introduction to the basic model we are working, the linear dynamic system which is a novel

Table 3.1: Function for learning LDS model parameters

```
function [x0, Ymean, Ahat, Bhat, Chat] = dytex(Y, n, nv)


tau = size(Y,2);

Ymean = mean(Y,2);

[U,S,V] = svd(Y − Ymean ∗ ones(1,tau),0);

Chat = U(:,1 : n);

Xhat = S(1 : n,1 : n) ∗ V(:,1 : n)′;

x0 = Xhat(:,1);

Ahat = Xhat(:,2 : tau) * pinv(Xhat(Xhat(:,1 : (tau − 1))));

Vhat = Xhat(:,2 : tau) - Ahat ∗ Xhat(:,1 : (tau − 1));

[Uv,Sv,Vv] = svd(Vhat,0);

Bhat = Uv(:,1 : nv) * Sv(1 : nv,1 : nv) / sqrt(tau − 1);
```

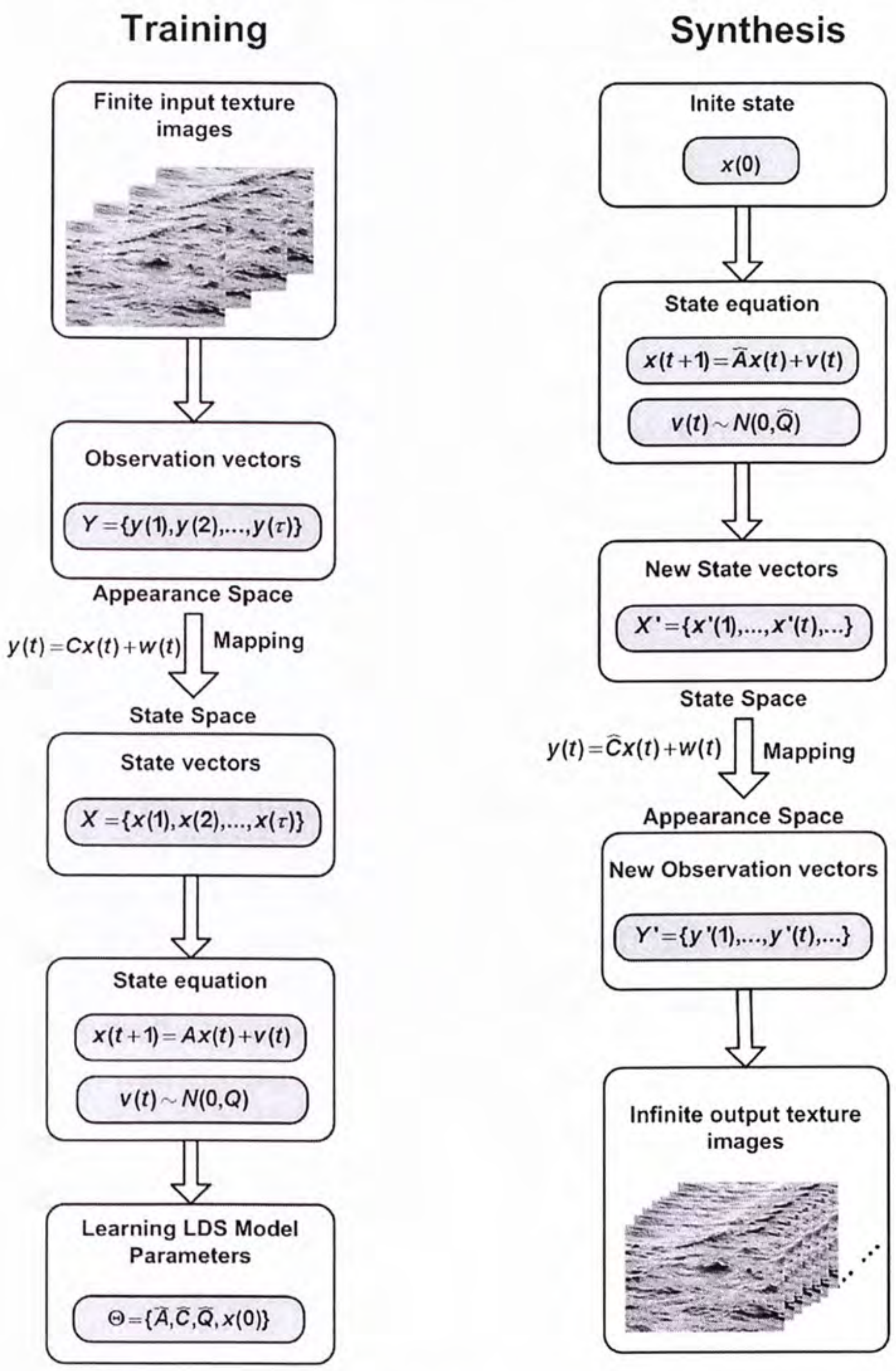
Training
Finite input texture images
Observation vectors
$Y=\{y(1),y(2),...,y(\tau)\}$
Appearance Space
$y(t)=Cx(t)+w(t)$
Mapping
State Space
State vectors
$X=\{x(1),x(2),...,x(\tau)\}$
State equation
$x(t+1)=Ax(t)+v(t)$
$v(t)\sim N(0,Q)$
Learning LDS Model Parameters
$\Theta=\{\widehat{A},\widehat{C},\widehat{Q},x(0)\}$
Synthesis
Inite state
$x(0)$
State equation
$x(t+1)=\widehat{A}x(t)+v(t)$
$v(t)\sim N(0,\widehat{Q})$
New State vectors
$X'=\{x'(1),...,x'(t),...\}$
State Space
$y(t)=\widehat{C}x(t)+w(t)$
Mapping
Appearance Space
New Observation vectors
$Y'=\{y'(1),...,y'(t),...\}$
Infinite output texture images

Figure 3.1: Framework of dynamic texture analysis and synthesis using LDS. (a) In the training process, a finite sequence of input images is used to train an LDS model with the system matrix $A$, the observation matrix $C$ and the system noise $v(t)$ (with its variance $Q$) using either methods (the EM based methods or the closed form solution presented). (b) In the synthesis process, a new (possibly infinite) sequence is generated using the learnt LDS and by sampling system noise.

Table 3.2: Function for Synthesize dynamic texture

```
function $[I]$ = synth($x0$, $Ymean$, $Ahat$, $Bhat$, $Chat$, $tau$)

$[n, nv] = \text{size}(Bhat);$
$X(:,1) = x0;$
for $t = 1 : tau$
    $X(:, t+1) = Ahat * X(:, t) + Bhat * randn(k, 1);$
    $I(:, t) = Chat * Xhat(:, t) + Ymean;$
end;
```

representation for dynamic textures. Also we adopt the closed form solution for learning and synthesis of sequences from training data given by Dorreto et al. We also reviews the properties and applications on the LDS model. The framework is verified to have a strong ability to capture complex visual phenomena, which is useful in video compression, classification, recognition, synthesis and editing of image sequences.

□ **End of chapter.**

# Chapter 4

# Dynamic Color Texture Synthesis

In previous chapters, we explored the basics of linear dynamic system: the model formulation, its properties, and the learning and synthesis process. Generally, the learning and synthesis methods are done for the grey-level input videos. Although the method of dynamic texture synthesis can be expanded into color videos, due to the limit of the memory size, the method can be slow or even unavailable when the input color video size is large. And they do not consider the intrinsic relationship among color channels. We present a new dynamic color texture synthesis algorithm by using the radial basis function (RBF for short) to compress the input of the colored dynamic texture video. The compressed information is then passed to the standard linear dynamic system to finish the learning and synthesis process. Matrix computation is small in size which makes the color video synthesis robust for large video input.

## 4.1 Related Work

Dynamic color texture synthesis is an advance of gray-level video synthesis problem in parametric modeling of dynamic textures in image and video processing. Doretto's linear dynamic system [11] is the first which addresses the colored dynamic textures problem in this area. It is done by setting the input matrix formed by the three unfolded RGB channels ordered one after the other. The problem of the method is that, we have to deal with large matrix computation which will be unavailable when the input video size is large. Filip's [14] method is specially designed to deal with colored dynamic textures, but its results are blurring.

Our consideration here is to choose a more suitable space to represent color [37] in color images which may lead to a more efficient use of the LDS model in terms of its ability in reducing the size of the input matrix $Y$ from the training sequence.

## 4.2 System Model

We work in the $YUV$ color space, commonly used in video processing. $Y \in \mathcal{R}^{m \times \tau}$ (Here we use the same $Y$ as in LDS model. In most cases, it stands for intensity) is the monochromatic luminance channel, which we refer to simply as intensity, while $U \in \mathcal{R}^{m \times \tau}$ and $V \in \mathcal{R}^{m \times \tau}$ are the chrominance channels, encoding the color [23]. One of the commonly adopted theory in image compression is that the structure information of the image is more important than the color information in the YUV color space, thus, a compression or smoothing can be done in the U, V channels without large loss of quality. Figure 4.1 shows the framework of our dynamic color textures learning and synthesis process.

There are a lot of ways to compress the U, V channels. But generally, the chosen compression method should not employ too much extra memory compared to the memory reduced for computing. Thus a PCA based compression is not applicable in our case, as it indeed need a lot of extra memory for calculating which is also very time consuming. In our testing, we

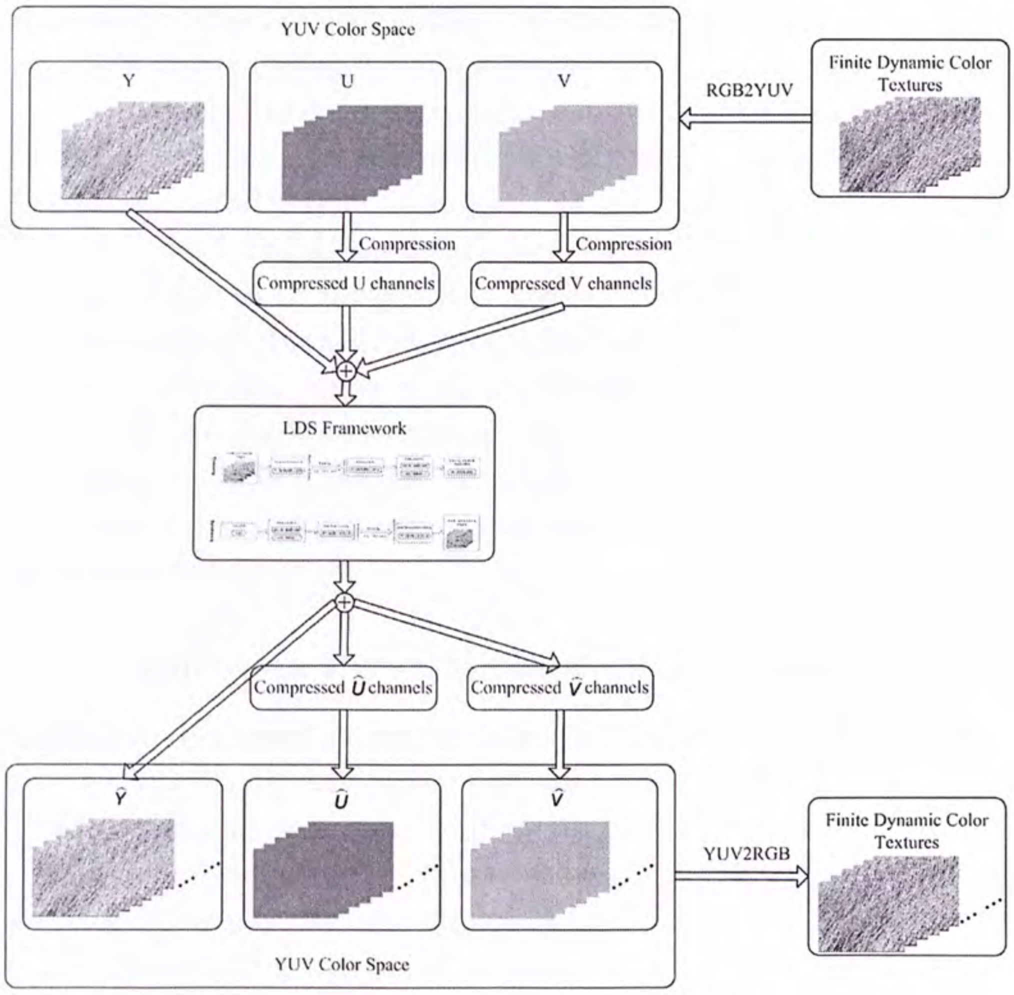

Figure 4.1: Dynamic Color Textures Synthesis framework (DCTS model). Sample frames are transferred to YUV color space first. U, V channels are compressed and are passed into the LDS model together with Y channel to synthesize new data. The newly generated compressed data are used to reconstruct the $U$, $V$ channels separately. Then the YUV data are transformed back into RGB channels to get the final synthesized results

try to use two different kinds of techniques to compress $U$, $V$ channels. First, we apply the well-known laplacian pyramid to the color channels, and only use part of the pyramid levels to reconstruct the color channels. Second, we use the radial basis function to compress the color channels. Radial basis functions are commonly used in data interpolation in multi-dimensions, especially, widely adopted in surface fitting from very small number of scattered data. Leung et. al use RBF to compress the BRDF value for image compression. In our test, we will show its strong ability in reconstructing the color channel values for the very complicated textured images by applying local support (In dynamic textures, there is no definite global correlation, thus a PCA based compression again will not be taken into our consideration, as the dimension reduction in PCA is a global one). Section 4.2.1 and 4.2.2 will discuss the two kinds of compression methods in details.

### 4.2.1 Laplacian Pyramid-based DCTS Model

Laplacian Pyramid is an overcomplete multi-resolution analysis method which decomposes image into different levels representing different scales or frequencies of the original image in a pyramid like structure. We may view the laplacian pyramid as a set of band-pass filtered copies of the image. Given the laplacian pyramid, the exact reconstruction of the original image is done by expanding, then summing all the levels of the Laplacian pyramid. For compression, a gaussian smoothed version of the original image could be got from the sum of all laplacian images at the same level and above.

### 4.2.2 RBF-based DCTS Model

RBF network can be regarded as a three layer network shown as in Figure 4.2. The problem of color dynamic textures compres-

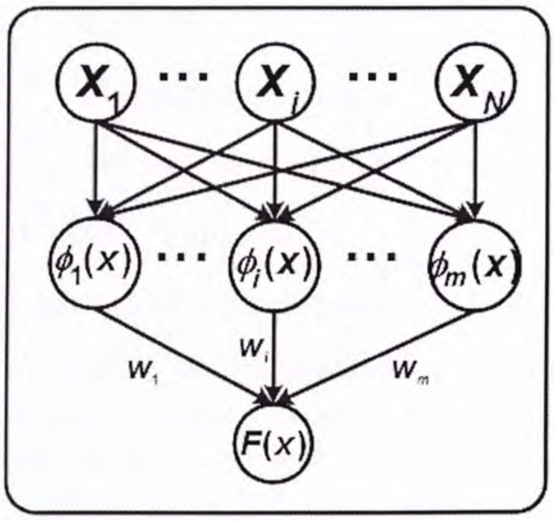

Figure 4.2: The radial basis function network. Each of the $N$ components of the input vector $x$ feeds forward to $M$ basis functions whose outputs are linearly combined with weights $\{w_j\}_{j=1}^{M}$ into the network output $F(x)$

sion can be stated as given a set of fixed points $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N \in R^n$ (where $\mathbf{x} = (x, y, z)$) on the volume of video in $R^3$ and a set of function values $f_1, f_2, \cdots, f_N \in R$, find an interpolant $F : R^3 \to R$ such that

$$F(x_i) = f_i \quad, i = 1, 2, \cdots, N \tag{4.1}$$

With the use of radial basis functions the interpolation function is defined as

$$F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi(\| \mathbf{x} - \mathbf{x}_i \|) + p(\mathbf{x}) \tag{4.2}$$

where $p(\mathbf{x})$ is a polynomial, $\alpha_i$ are coefficients corresponding to each center $\mathbf{x}_i$ and $\| \cdot \|$ is the Euclidean norm on $R^3$. The basis function $\phi$ is a real valued function which is called the radial basis function. Usually, there are some popular basis functions:

1. $\phi(r) = r$, linear radial basis function

2. $\phi(r) = r^3$, triharmonic
3. $\phi(r) = \sqrt{r^2 + c^2}$, multi-quadric
4. $\phi(r) = \frac{1}{\sqrt{r^2+c^2}}$, inverse multi-quadric
5. $\phi(r) = exp(-cr^2)$, Gaussian
6. $\phi(r) = r^2 \log(r)$, Thin-plate spline

where $r = \parallel \mathbf{x} - \mathbf{x}_i \parallel$. The polynomials $p(\mathbf{x})$ can be chosen as constant, linear, and quadratic, whose degree is generally lower than four in order to easily control the fitting results.

### Center Points

The position and the number of center points are crucial to the final interpolation results in the RBF framework. As textures are defined to be repetitive patens exhibiting spatial coherence, thus uniformly choosing the center points in each frame or each texture is reasonable. The center points chosen for the whole video are: $C = \{C_1, \cdots, C_\tau\}$ where $C_i = \{c_1, \cdots, c_m\}$ is the center points uniformly chosen in frame $i$ of the dynamic texture video sequence which has $\tau$ frames.

So far until now, we only consider the spatial coherence of texture for each frame separately. We disregard the unique property of dynamic textures, that is the temporal coherence between frames. For color values of points in current frame are much related to the ones in its previous $k$ frames and next $k$ frames. We call this "Markov" property borrowed from Markov process and the centers in previous and next frames are called neighborhood of the center points in the current frame:

$$F(x) = \sum_{i=1}^{N} \alpha_i \phi(\parallel \mathbf{x} - \mathbf{x}_i \parallel) + p(\mathbf{x}) \tag{4.3}$$

where center points $\mathbf{x}_i$ of the pixel $\mathbf{x}$ in frame $t$ contains the centers not only in current frame $t$, but also the frames ahead of and next to it. That is: $\mathbf{x}_i \in \{C_{t-k}, \cdots, C_{t-1}, C_t, C_{t+1}, \cdots, C_{t+k}\}$. In our paper, generally, we only consider the two nearest neighbor frames. That is, we set $k = 1$.The centers are chosen to be distributed in the same position in each frame (in the time domain), while guarantee uniformness in spatial domain.

**Polynomials**

We have stated that the polynomial part of the RBF framework should not be larger than the order, or it will be hard to control the interpolation. In fact, if we regard the polynomial part as the partly finished product of the final fitting of the color values, then blending the two parts by summing the two functions is equivalent to deforming the partly finished values into the desired values using a set of radial basis functions.

In our texture value fitting problem, as textures are composed of repetitive patterns, thus color values occurred in the texture are limited, and uniformly distributed. With this view in mind, we choose linear polynomial as for a local area or single pattern (or texton) the color values do not vary much from pixels to pixels. Additional so-called natural constraints are added to the coefficients $\alpha$ that must satisfy the following constraints: $\sum_{i=1}^{N} \alpha_i = 0$ and $\sum_{i=1}^{N} \alpha_i x_i = \sum_{i=1}^{N} \alpha_i y_i = \sum_{i=1}^{N} \alpha_i z_i = 0$, and $N = (2 * k + 1) * m$.

Equation 4.2 and the constrains give rise to the following linear system:

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1N} & 1 & x_1 & y_1 & z_1 \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2N} & 1 & x_2 & y_2 & z_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{NN} & 1 & x_N & y_N & z_N \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 \\ x_1 & x_2 & \cdots & x_N & 0 & 0 & 0 & 0 \\ y_1 & y_2 & \cdots & y_N & 0 & 0 & 0 & 0 \\ z_1 & z_2 & \cdots & z_N & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \\ \alpha_{N+1} \\ \alpha_{N+2} \\ \alpha_{N+3} \\ \alpha_{N+4} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.4}$$

The solution of the system composed of the weighting coefficients and the polynomial coefficients for the interpolation function $F(\mathbf{x})$. Thus, the final compressed $U$, $V$ channels are composed of $m$ center point values, and $N + 4$ weights. Figure 4.3 shows the framework of learning and synthesis process for dynamic color textures using RBF.

## 4.3 Experimental Results

All of our testing dynamic color textures are from Dynamic Texture Database[1] which provides hundreds of dynamic textures samples for the research of this area. The synthesis procedures are written in Matlab, running on Intel 2GHz CPU personal computer with 1GB RAM.

[1]http://www.cwi.nl/ projects/dyntex/

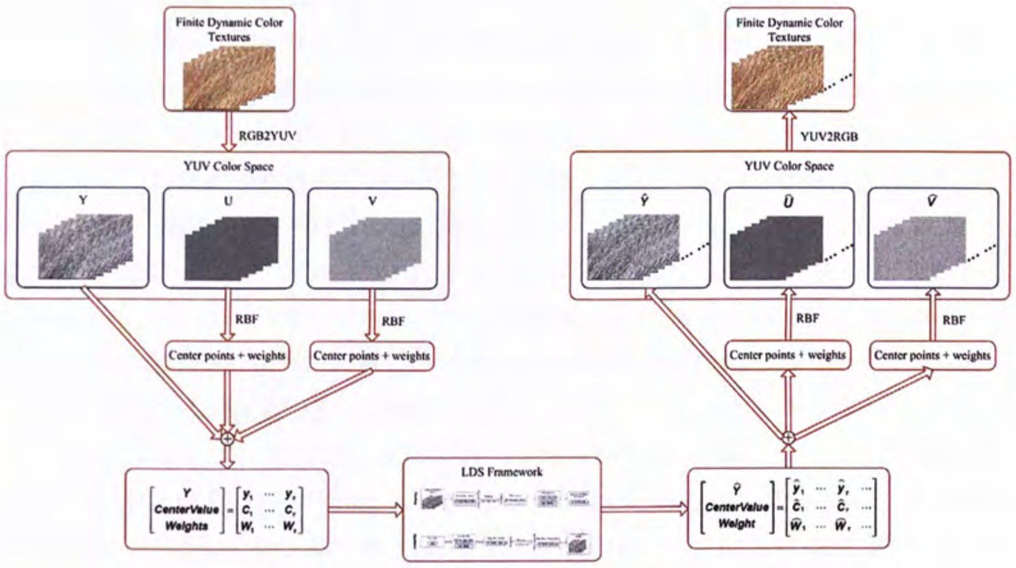

Figure 4.3: Dynamic Color Textures Synthesis framework. Sample frames are transferred to YUV color space first. U, V channels are compressed using RBF networks and the center values, weights together with Y channel are passed into the LDS model to synthesize new data. The newly generated center values, and weights are used to reconstruct the U, V channels separately. Then the YUV data are transformed back into RGB channels to get the final synthesized results

In our experiments, we also employ the dynamic color texture synthesis method mentioned by Doretto (directly pass all the RGB data to be processed by the LDS framework) for comparison with the two methods presented in our approach. Because the three methods, laplacian pyramid compression, RBF compression, or Doretto's direct method, have different size of the input matrix to the LDS model, and the synthesized results are also determined by the input matrix. Another factor that will influence the quality of the synthesized results is the number of principle components $n$ we choose during the dimension reduction process in the LDS model. We have to determine a suitable $n$ for comparing the results. The input matrix size of the three methods are bounded by $(m\times\tau, 3m\times\tau]$, where $m$ is the pixel number for one channel in one frame, and $\tau$ is the frame number. It is clear that, the input matrix in laplacian pyramid compression and RBF compression should be larger than the matrix purely got from one single channel, and should be smaller than the matrix containing three-channel values which is got by directly applying LDS model in RGB color space. From Doretto's tests, we know that, the reconstruction error does not drop much when $n$ varies from 45 to 65. That is to say the quality of the final synthesized results will not improve much when $n$ increases from 45 to 65. We did the same experiments to calculate the reconstruction error when the input is all three color channels which is three times larger than Doretto's testing data, similar conclusion can be drawn except that in this situation, $n$ varies from 50 to 70. From the reason above, we choose $n = 60$ as our testing base. Figure 4.4 to Figure 4.9 show the testing results. The first line shows the images from the original video sequence. The second 2 rows show the synthesized results using LDS in RGB color space. The third 2 rows show the results of using laplacian pyramid. The last 2 rows are the synthesized results using RBF method.

Table 4.1: Comparison of memory and time usage using the three methods for straw video

| | Memory | Time |
|---|---|---|
| Direct LDS model | $122 \times 148 \times 3 \times 100$ | 39 s |
| LP based DCTS | $(122 \times 148 + 404 \times 2) \times 100$ | 55 s |
| RBF based DCTS | $(122 \times 148 + (25 + 25 \times 3) \times 2) \times 100$ | 674 s |

Table 4.2: Compression ratio for the straw video

| | Compression ratio for the whole volume | compression ratio for U/V channel |
|---|---|---|
| LP based DCTS | 34.8% | 2.2% |
| RBF based DCTS | 33.5% | 0.55% |

Table 4.1 shows the comparison of memory and time usage using the three methods. $122 \times 148$ are the row number times column number, and 100 is the number of frames. For LDS model in RGB color space, the memory required is all the three channels information of the input data. For Laplacian pyramid based DCTS model, the memory usage is the gray channel plus the compressed U and V channel information (in this example, it is 404 extra units for U, V channel each). For RBF based model, the extra memory used are the center points values, and the corresponding weights (the centers are chosen every 900 pixels in each frame). Although, RBF-based method is memory efficient, we have to sacrifice time to get the memory efficiency. Most of the time cost in RBF based method is spent on calculating the weighting and reconstructing the lost data.

Table 4.2 shows the compression ratio. The compression ratio for the whole volume is defined as the total memory used using the LP or RBF methods divided by the total memory used using the direct LDS model. The compression ratio for U/V channel shows the most efficient memory reduction saved for the color channel. From the tests we see that RBF-based DCTS model is the most memory efficient one, but it gets the best visual effects among all the testing

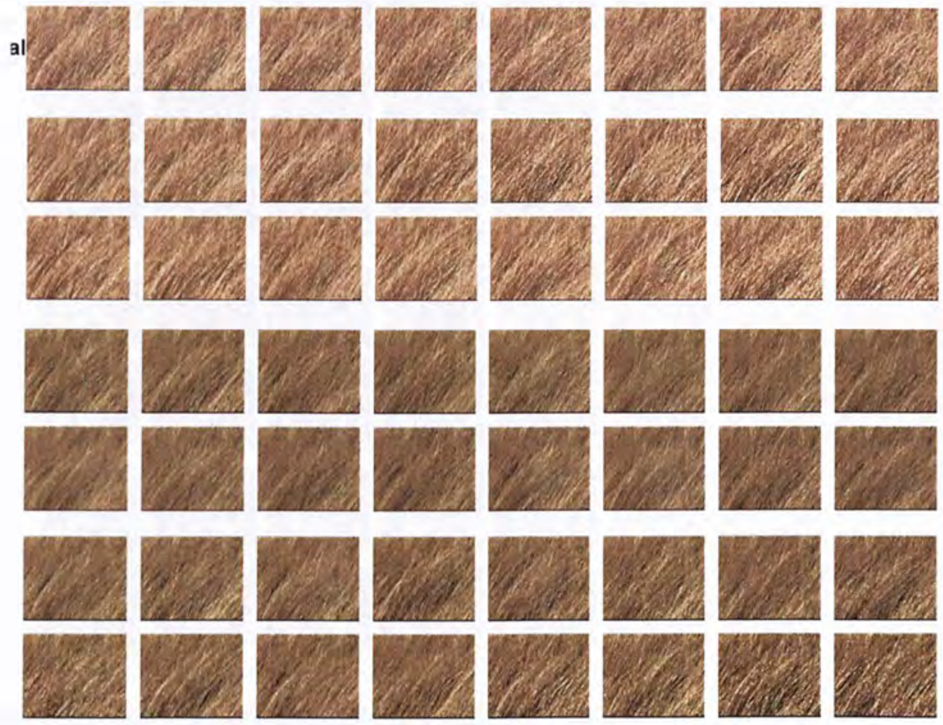

Figure 4.4: Straw sequence with size $120 \times 148$ and 100 frames. From top to bottom: First row shows samples from the original straw texture. Second 2 rows show samples from the direct LDS model used by Doretto. Third 2 rows contain samples generated using Laplacian Pyramid based DCTS model and the last 2 rows are the synthesized results using RBF based DCTS model

Figure 4.6 shows the comparison between the synthesized result of using RBF-based method and simply down-sampling of the $U, V$ channels. The size of the down-sampled $U$ channel image is $8 \times 10$ which is close to the extra size used for $U$ channel in

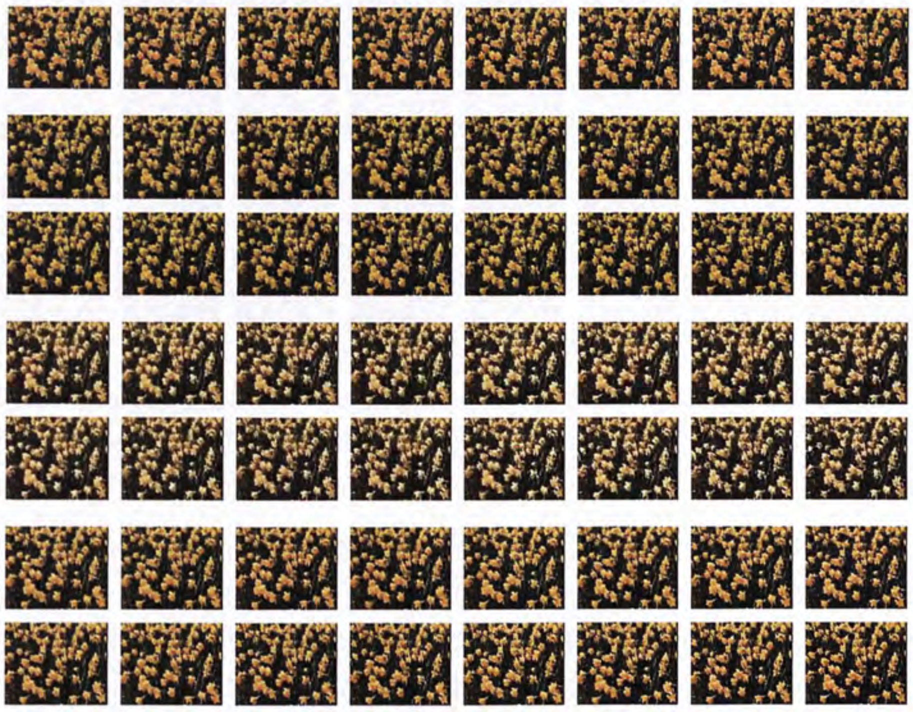

Figure 4.5: Flower sequence with size $148 * 120$ and 100 frames. From top to bottom: First row shows samples from the original straw texture. Second 2 rows show samples from the direct LDS model used by Doretto. Third 2 rows contain samples generated using Laplacian Pyramid based DCTS model and the last 2 rows are the synthesized results using RBF based DCTS model

RBF-based one $(25 + 25 \times 3)$ for each frame. It is obvious that, if we downsample the $U$ channel image to the compression ratio of RBF-based one, the synthesize result is much worse than the RBF-based DCTS method. We can see that the yellow color for the flowers are almost lost in the simple downsampling tests.

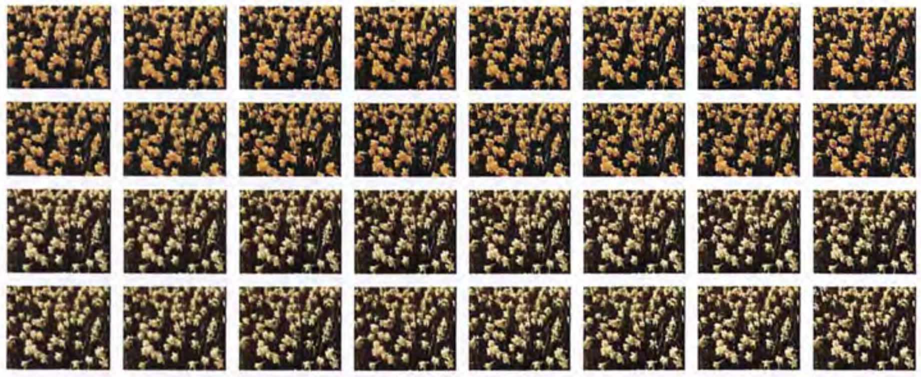
Figure 4.6: Comparison of synthesized results using RBF-based method (first 2 rows) and simply downsampling (second 2 rows). The downsampled size of the $U$ channel is $8 \times 10$

The fire sequence(Figure 4.9) gives us a good example where LDS model fails to capture the too complex dynamic behavior. Among the synthesized results for the three methods, we can observe more preserved structural information from the RBF based results. The reason why RBF based DCTS can improve the results than the direct LDS model lies on two sides: first is that the input matrix to be processed by the RBF-based DCTS model is much smaller than the matrix in the direct LDS model, and most part of the matrix is the $Y$ channel, or the structural information. After the dimension reduction, principle components tend to reflect or preserve more about the structure. Second is our consideration in choosing the center points. The RBF functions in our system not only reflect the spatial correlation in a single frame, but also stress the temporal relation between neighborhood frames.

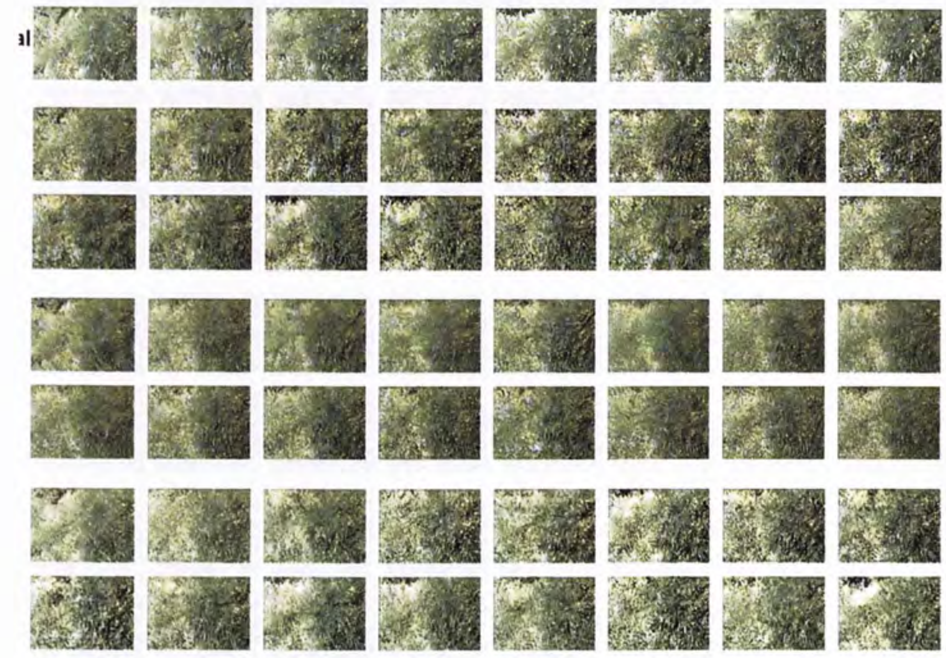

Figure 4.7: Tree sequence with size 148 * 108 and 100 frames. From top to bottom: First row shows samples from the original straw texture. Second 2 rows show samples from the direct LDS model used by Doretto. Third 2 rows contain samples generated using Laplacian Pyramid based DCTS model and the last 2 rows are the synthesized results using RBF based DCTS model

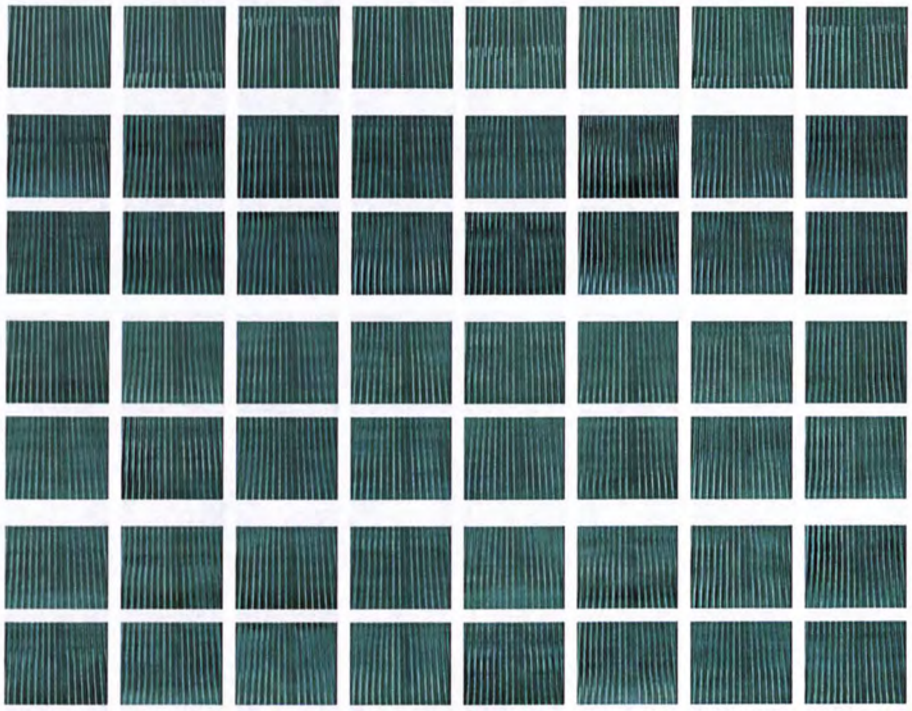

Figure 4.8: Elevator sequence with size $148 * 120$ and 100 frames. From top to bottom: First row shows samples from the original straw texture. Second 2 rows show samples from the direct LDS model used by Doretto. Third 2 rows contain samples generated using Laplacian Pyramid based DCTS model and the last 2 rows are the synthesized results using RBF based DCTS model

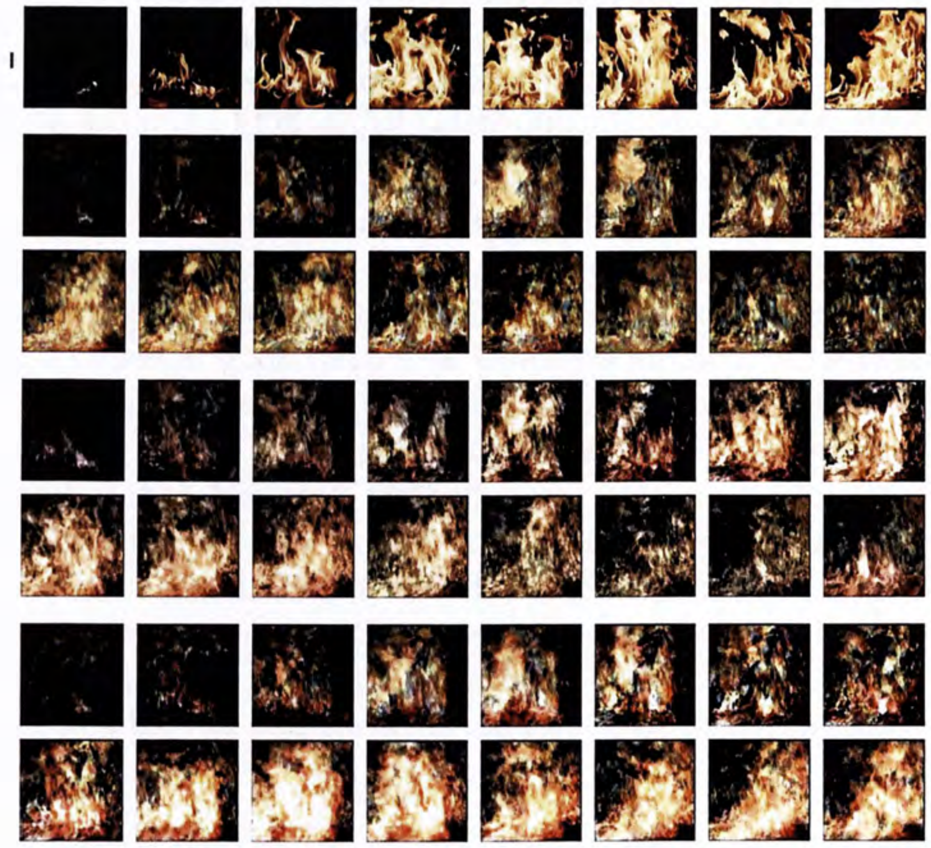

Figure 4.9: Fire sequence with size 148 * 148 and 100 frames. From top to bottom: First row shows samples from the original straw texture. Second 2 rows show samples from the direct LDS model used by Doretto. Third 2 rows contain samples generated using Laplacian Pyramid based DCTS model and the last 2 rows are the synthesized results using RBF based DCTS model

## 4.4 Summary

In this chapter, we present a framework for dynamic color texture synthesis by doing compression of the $U$, $V$ channels in YUV color space. Our DCTS model is a memory efficient for LDS model to process. We have tested two kinds of compression methods: wavelts based DCTS model and RBF based DCTS model. Both two models could reduce the data size to be passed as the input of the LDS model, while still generate good quality textures.

The RBF based DCTS model especially perform well. The method is memory efficient. Only $\frac{3}{400} * m$ extra space is needed for each frame in our experiments. It also can generate new dynamic textures as good quality as Doretto's method. And even for some dynamic textures exhibiting complex dynamic behavior, our RBF based DCTS model can preserve more structural information than Doretto's method.

The future work will be put on to add control parameters on the RBF based DCTS model. By only changing the center points and their corresponding weights, we can control the size of dynamic texture pattern, the speed of evolution, and other properties which is addressed in the paper Editable Dynamic Textures.

□ **End of chapter.**

# Chapter 5

# Dynamic Textures using Multi-resolution Analysis

Traditional LDS model is applied using the raw image vectors. There are also frequency domain representations for the LDS model by applying Fourier transform to the original video. But these two representations are not adequate for representation and analysis of textural signals. These signals typically contain details at different scales, orientations and locations. Many recent research efforts suggest that a better representation for such signals are multi-resolution structures, which is usually done by applying multi-resolution analysis to get the representations for different scales, orientations and locations. More complex and detailed texture appearance characteristics could be captured and represented in different scales and orientations using multi-resolution analysis (MRA).

Our work is just based on the above consideration. We decompose the texture properties, which are hidden or fused into the dynamic texture video sequence, into difference resolution levels where the coarse levels represent the texture pattern while the detail levels or the high frequency signals represent the uniqueness of the texture. For different levels of signals, they should have different dynamic behaviors. Thus, we present the Multi-resolution Auto-regressive Moving Average model (MARMA) to better represent and capture more complex dynamic textures than purely applying the LDS model in the space domain.

The following Sections are arranged as follows: Section 5.1 presents the MARMA model and addresses the learning and synthesis process; We discuss the criteria for choosing multi-resolution descriptors in Section 5.2 and briefly introduce the three MRA methods we are using for the comparative study; Section 5.3 shows the experimental results for the MARMA model and conclusion comes in Section 5.4.

## 5.1 System Model

We believe that the original LDS model is not enough to capture the complex motion for dynamic textures, one reason is that it does not fully take the advantage of the spatial relation between neighborhood pixels (the LDS model only considered the evolution in time domain). And another consideration is that, for patterns in deferent scales or frequencies, the behaviors or the evolving properties should vary from each other. As it is well-known that, multi-resolution analysis allows a signal (in our case, the dynamic textures) to be viewed in multiple resolutions. Each resolution reflects a different frequency. Thus, we consider to transform the original dynamic texture sample into different levels to capture motions ranging from local to global, while considering spatial correlation between pixels in each frame using multi-resolution analysis.

Our multi-resolution auto-regressive moving average model, or MARMA for short, is graphically shown in Figure 5.1. Our model takes the original video sequence as input and then transform the video into its multi-resolution representation by applying some chosen multi-resolution descriptor frame by frame. Then, different levels of resolution are then processed by the

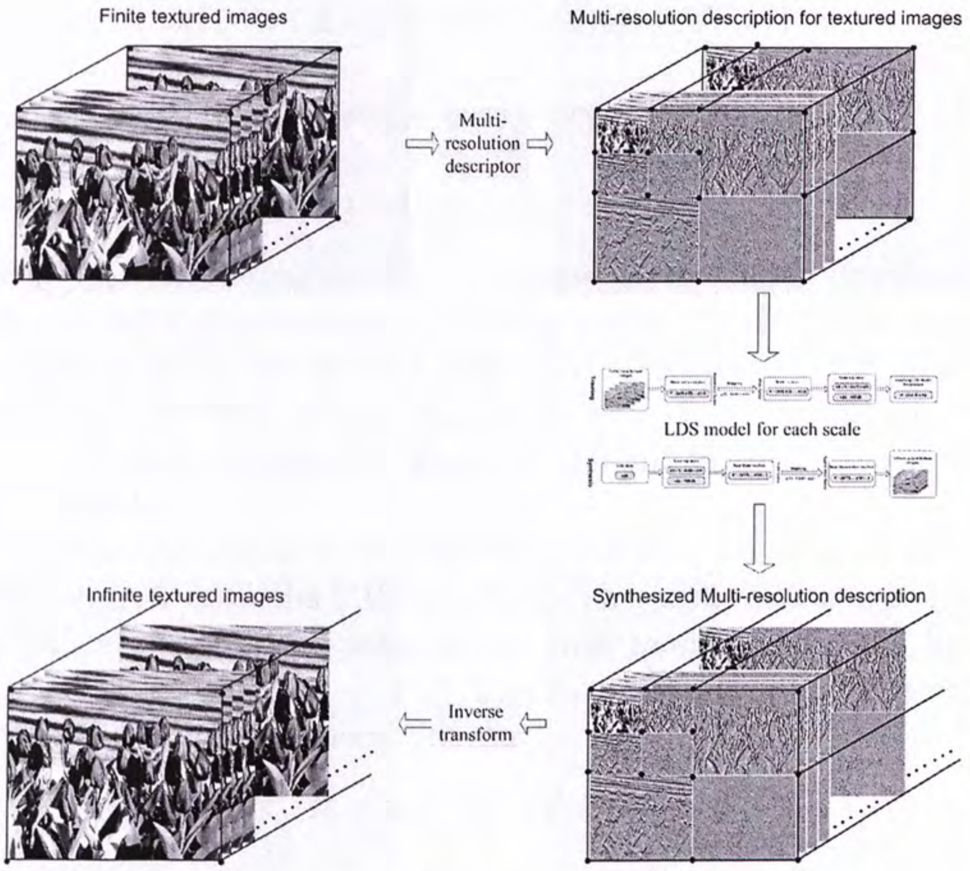
Finite textured images
Multi-resolution description for textured images
Multi-resolution descriptor
LDS model for each scale
Synthesized Multi-resolution description
Infinite textured images
Inverse transform

Figure 5.1: 5-level laplacian pyramid

state-of-art LDS model independently to get the synthesized resolution level features. After that, we use inverse transform of the multi-resolution analysis to transform these features back to form a new synthesized dynamic texture video. The system model could be formulated as:

$$\begin{cases} s(t) = As(t) + Bv(t) & ,v(t) \sim \mathcal{N}(0,1) \\ f(t) = Cs(t) + w(t) & ,w(t) \sim \mathcal{N}(0,R) \\ f(t) = MRA(y(t)) & \end{cases} \tag{5.1}$$

where $y(t)$ is the $t_{th} = 1, \cdots, \tau$ frame of the input dynamic texture video sequence, and $f(t)$ is the result of multi-resolution transform result for frame $t$, and $s(t)$ is the state variable in frequency domain. $f(t) = d_{mn}(t)$ for $m = 1, \cdots, m$ and $n = 1, \cdots, N$ (M,N are given). $d_{mn}(t)$ is the $m_{th}$ feature in level $n$ for frame $t$.

Given the sample video, our aim is to find the system variables $A, B, C$ after the MR transform. This is the classic system identification problem, same as the LDS model, which can be defined as: given $f(1), \cdots, f(\tau)$, find an optimal solution subject to the maximum likelihood criteria:

$$\widehat{A}, \widehat{B}, \widehat{C}, \widehat{R} = \arg \max_{A,B,C,R} f(1), \cdots, f(\tau) \tag{5.2}$$

The closed form solution for Equation 5.2 can be found in Dorreto's method, which is reviewed in Chapter 3.

## 5.2 Multi-resolution Descriptors

There are many multi-resolution descriptions that are widely used in image and video processing , such as compression and classification. Generally, in our problem, two considerations should be taken into account here for choosing the suitable wavelet functions:

1. The multi-resolution analysis transform should has inverse transform
2. There should be no reconstruction error for accuracy of the synthesized results

Those multi-resolution features satisfying the two requirements could be applied in our MARMA framework. In the following, we will investigate the MARMA framework with different multi-resolution features: Laplacian pyramid, Haar wavelet transform, and steerable pyramid, which are commonly used MRA functions in image and video processing, to find out whether different multi-resolution descriptors could bring us different degree of accuracies in the final synthesized results.

### 5.2.1 Laplacian Pyramids

In image processing, images are often decomposed into information at different scales. Fine scale information is obtained by subtracting the blurring image (often got by convolving the original image with lowpass filters) from the original image. The invertible linear transform that maps input image to the two images blurred and fine. The inverse transform maps the fine image and the blurred image to the result. Often, we will want further subdivisions of scales by decomposing the coarse scale image into medium corse and very coarse images by applying the same splitting technique. Repeating the process by splitting off finer and finer details from the blurred image, we can get the data structure known as a *Laplacian Pyramid*. Figure 5.2 shows a 5 level laplacian pyramid in which each level represents different scales of the image.

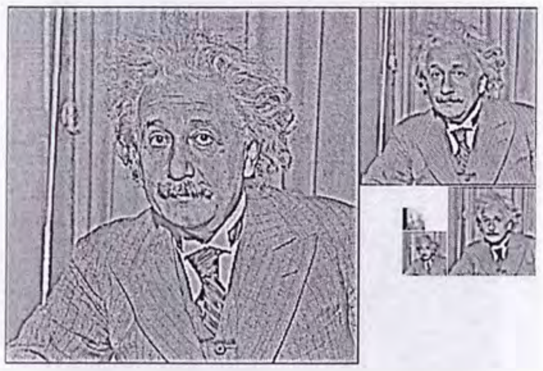

Figure 5.2: 5-level laplacian pyramid

Two things about Laplacian pyramids are a bit unsatisfactory. First, there are more pixels (coefficients) in the representation than in the original image. Specifically, the 1-dimensional transform is overcomplete by a factor of $\frac{4}{3}$, and the 2-dimensional transform is overcomplete by a factor of 2. Secondly, the "band-pass" images do not segregate information according to orientation.

## 5.2.2 Haar Wavelets

Although Haar wavelets date back to the beginning of the twentieth century, but it still has a lot of advantages that a lot of applications have the preference to use it than other wavelet functions[1].

Given an array of $n$ samples, the 1D haar transform is performed by first finding the **average** of each pair of samples

[1]Advantages of haar wavelet:

- It is computationally simple
- It is fast
- It is memory efficient, since it can be calculated in place without a temporary array
- It is exactly reversible without the edge effects that are problem with other wavelet transforms

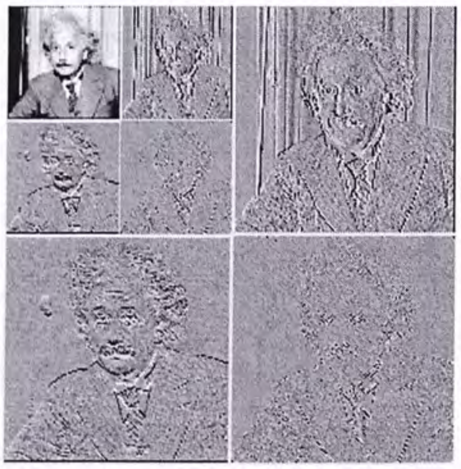

Figure 5.3: 2-level haar wavelet

(smoothing) and filling the first half of the array with averages. Then finding the **difference** between each average and the samples it was calculated from, and filling the second half of the array with differences. Repeating the process on the fist half of the array. A 2D haar transform could be performed by first applying a 1D haar transform on each row, and then on each column. Figure 5.3 shows two level haar wavelet.

### 5.2.3 Steerable Pyramid

The Steerable pyramids [41], together with Gabor wavelet, is commonly recognized as the best of all texture feature extractor in classification of texture images [27]. It is a linear multi-scale, multi-orientation image decomposition method, unlike the most discrete wavelet transforms such as Haar wavelet and Dyadic wavelet, which is non-orthogonal and over-complete. The representation of the wavelet form is designed to be translational and rotational invariant.

The system diagram for steerable pyramid is shown in Fig. 5.4, illustrating the analysis (the left hand side) /synthesis (the right hand side) process of steerable pyramid for a single stage. The

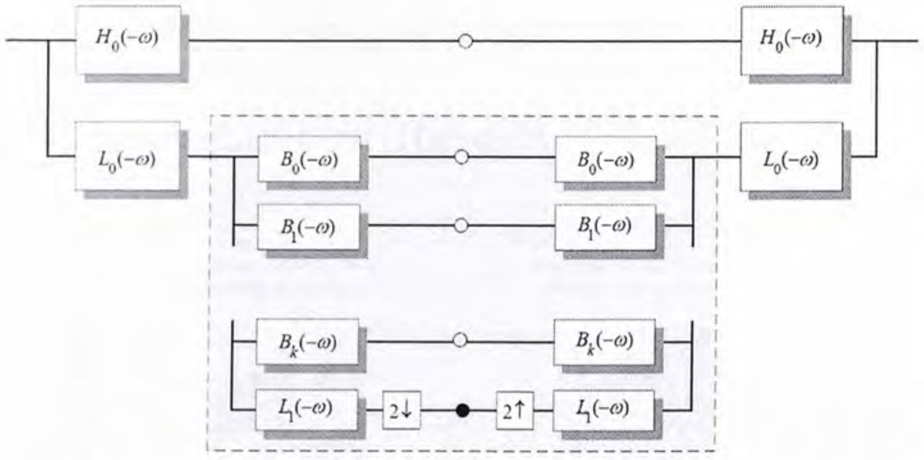

Figure 5.4: System diagram for the Steerable pyramid, illustrating the filtering and sampling operations, and the recursive construction.

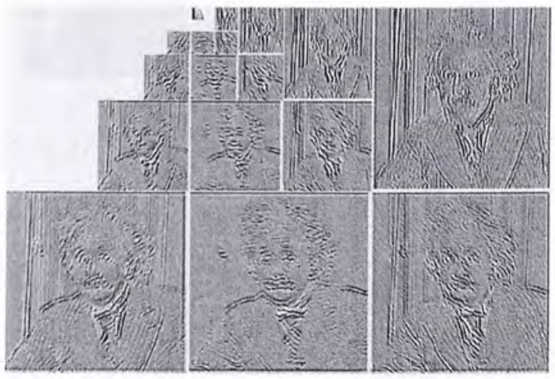

Figure 5.5: 4-level steerable pyramid

circles in between represent the decomposed subband images. Boxes containing 2 $\downarrow$ and 2 $\uparrow$ mean downsampling and upsampling by a factor of 2. The image is first decomposed into lowpass and highpass subband, using the filters $L_0$ and $H_0$. the lowpass band continues to be divided into a set of oriented bandpass subbands $B_0, \cdots, B_k$ and a lowpass subband $L_1$. The lower subband is subsampled by a factor of 2 along the $x$ and $y$ directions. The recursive construction of a pyramid is achieved by inserting a copy of the diagram contents enclosed by the dashed rectangle at the location of the solid circle (i.e., the lowpass branch).

Figure 5.5 shows a 4-level steerable pyramid.

## 5.3 Experimental Results

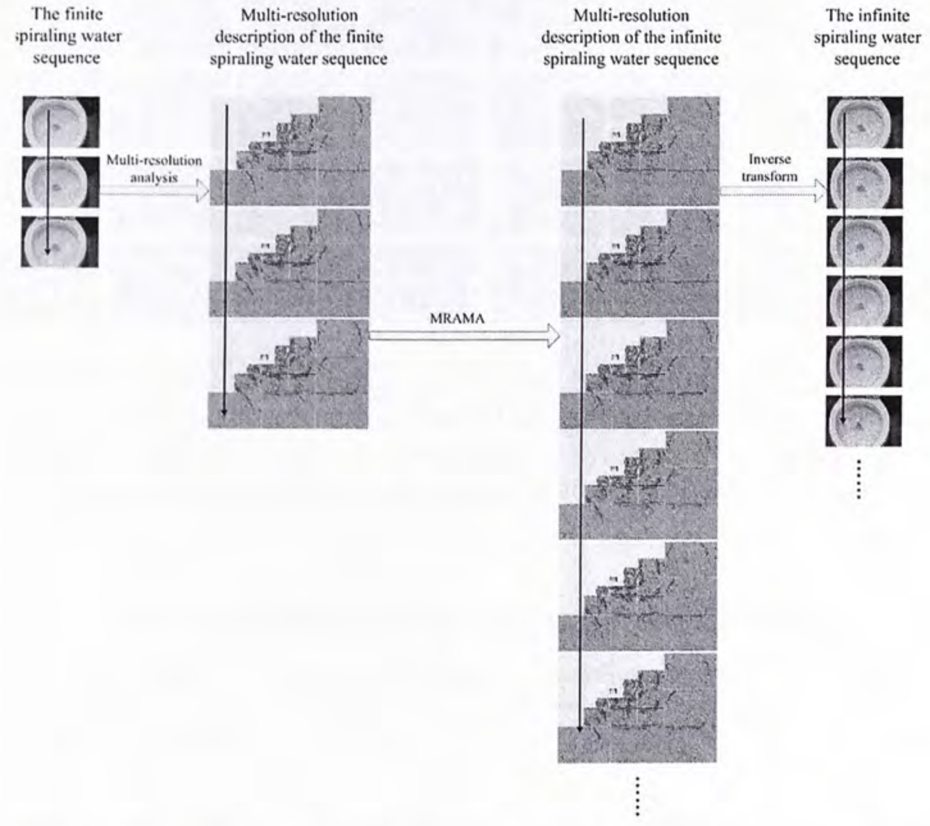
The finite spiraling water sequence
Multi-resolution description of the finite spiraling water sequence
Multi-resolution description of the infinite spiraling water sequence
The infinite spiraling water sequence
Multi-resolution analysis
Inverse transform
MRAMA

Figure 5.6: Spiraling-water. The figure shows how an "infinite length" texture sequence is synthesized from a typically "short" input texture sequence by transforming to frequency domain to synthesize dynamic behaviors at different scales.

Figure 5.6 illustrates the fact that an "infinite length" texture sequence can be synthesized from a typically "short" input sequence by just drawing IID samples $v(t)$ from a Gaussian distribution on the multi-resolution level of the original video sequence using our MARMA model. The frames are from the spiraling-water sequence originally having 120 frames of dimension $112 \times 168$. A new sequence of frames are generated using $n = 40$ principle components having 200 frames.

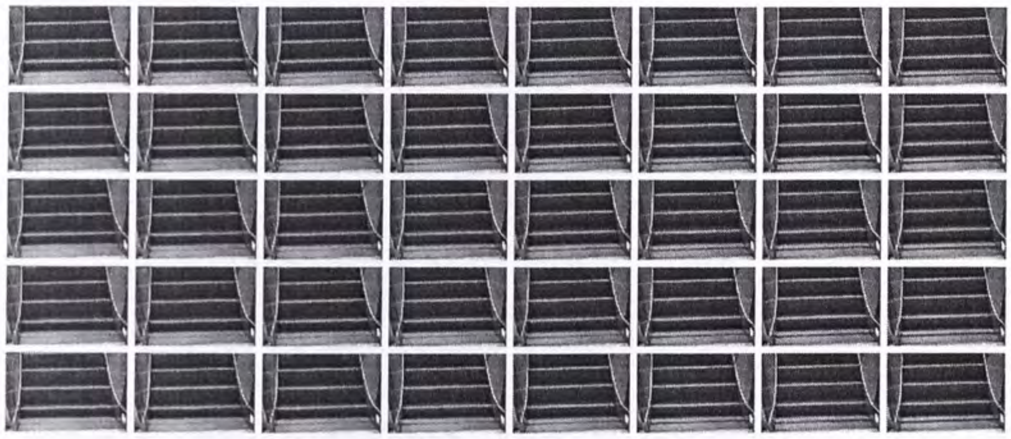

Figure 5.7: Elevator. From top to bottom: Samples of the original video, samples of extrapolated sequence using LDS model, samples of extrapolated sequence of MARMA using 3-level laplacian pyramid, 3-level haar wavelet, and 3-level steerable pyramid (all using $n = 40$ components, $\tau = 120$)

Table 5.1: Reconstruction error for the elevator video.

| $\|\|Y - Y'\|\|$ | LDS | Laplacian Pyramid | Wavelet Analysis | Steerable Pyramid |
|---|---|---|---|---|
| $n = 20$ | 9.0101 | 8.1769 | 7.1832 | 7.1832 |
| $n = 40$ | 7.2715 | 6.6609 | 6.4641 | 6.1451 |
| $n = 80$ | 7.0586 | 6.4426 | 6.4639 | 5.9789 |

Figure 5.7 to 5.13 show the behavior of the algorithm on a representative set of experiments (the training sequence are from the MIT Temporal Texture database[2]). In each case, the first row shows sample frames from the original video. On the second row, we show the synthesized result using the LDS model [11]. The third to fifth row show the results of our MARMA model

[2]ftp://whitechapel.media.mit.edu/pub/szummer/temporal-texture/

using Laplacian pyramid, haar wavelet and steerable pyramid sequentially. We choose $n = 40$ principle components in these testing results.

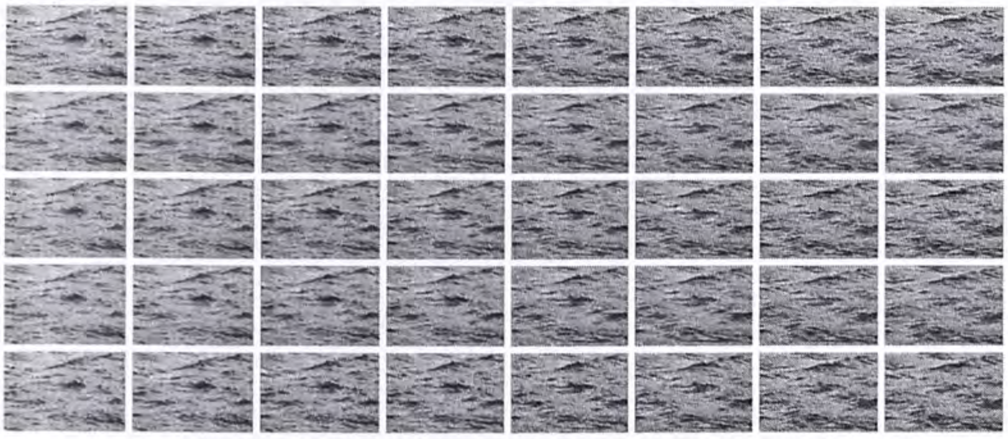

Figure 5.8: Ocean wave. From top to bottom: Samples of the original video, samples of extrapolated sequence using LDS model, samples of extrapolated sequence of MARMA using 3-level laplacian pyramid, 3-level haar wavelet, and 3-level steerable pyramid (all using $n = 40$ components, $\tau = 120$)

Table 5.2: Reconstruction error for the ocean wave video.

| $\|\|Y - Y'\|\|$ | LDS | Laplacian Pyramid | Wavelet Analysis | Steerable Pyramid |
|---|---|---|---|---|
| $n = 20$ | 28.7696 | 25.8103 | 28.0077 | 24.4732 |
| $n = 40$ | 22.0595 | 20.2793 | 20.4455 | 17.3135 |
| $n = 80$ | 17.015 | 15.0768 | 16.5541 | 14.9716 |

From Table 5.1 to 5.7, the result shows the reconstruction errors using LDS model, and our model using laplacian pyramid, haar wavelet and steerable pyramid. A dynamic model for the sequence is first obtained by fixing the number of states, $n$, of the system. Then using the states estimated as part of the learning process, the original sequence is reconstructed. Then, the error in reconstruction is calculated as the norm of the difference between the original images and the reconstructed images. The reconstruction error can be regarded as a function of the number of components selected, that is $n$. It is obvious that the reconstruction error is monotonically increasing as $n$ increases. Tables from 5.1 to 5.7 also shows the trend when $n = 20, n = 40$ and $n = 80$.

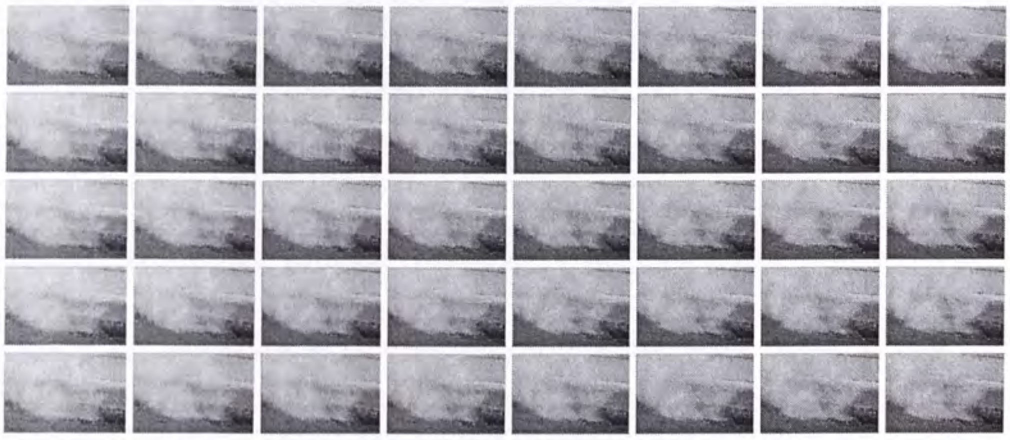

Figure 5.9: Smoke. From top to bottom: Samples of the original video, samples of extrapolated sequence using LDS model, samples of extrapolated sequence of MARMA using 3-level laplacian pyramid, 3-level haar wavelet, and 3-level steerable pyramid (all using $n = 40$ components, $\tau = 120$)

Table 5.3: Reconstruction error for the smoke video.

| $\|\|Y - Y'\|\|$ | LDS | Laplacian Pyramid | Wavelet Analysis | Steerable Pyramid |
|---|---|---|---|---|
| $n = 20$ | 5.3722 | 4.5393 | 4.5584 | 4.5182 |
| $n = 40$ | 4.8676 | 4.221 | 4.3346 | 3.7865 |
| $n = 80$ | 4.6453 | 4.128 | 3.7731 | 3.7276 |

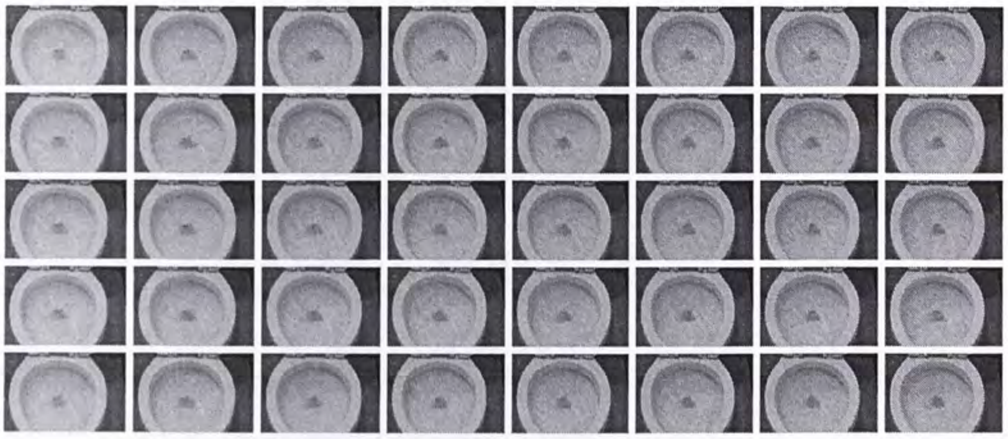

Figure 5.10: Toilet. From top to bottom: Samples of the original video, samples of extrapolated sequence using LDS model, samples of extrapolated sequence of MARMA using 3-level laplacian pyramid, 3-level haar wavelet, and 3-level steerable pyramid (all using $n = 40$ components, $\tau = 120$)

Table 5.4: Reconstruction error for the toilet video.

| $\|\|Y - Y'\|\|$ | LDS | Laplacian Pyramid | Wavelet Analysis | Steerable Pyramid |
|---|---|---|---|---|
| $n = 20$ | 5.7778 | 5.6136 | 5.6049 | 5.4474 |
| $n = 40$ | 5.4544 | 5.3002 | 5.2139 | 4.9938 |
| $n = 80$ | 4.9587 | 4.8463 | 4.8153 | 4.7758 |

## 5.4 Summary

In this chapter, we propose a scheme to learn and synthesize dynamic texture sequences, based on the algorithm provided by [11]. The proposed scheme uses multi-resolution descriptors of frames in the sequence, instead of the frames themselves to learn the dynamics of the texture sequence. In doing this, we have shown that similar results as the ones obtained using the original LDS model can be achieved. Since each of the multi-resolution analysis contains contributions from pixels in the neighborhood,

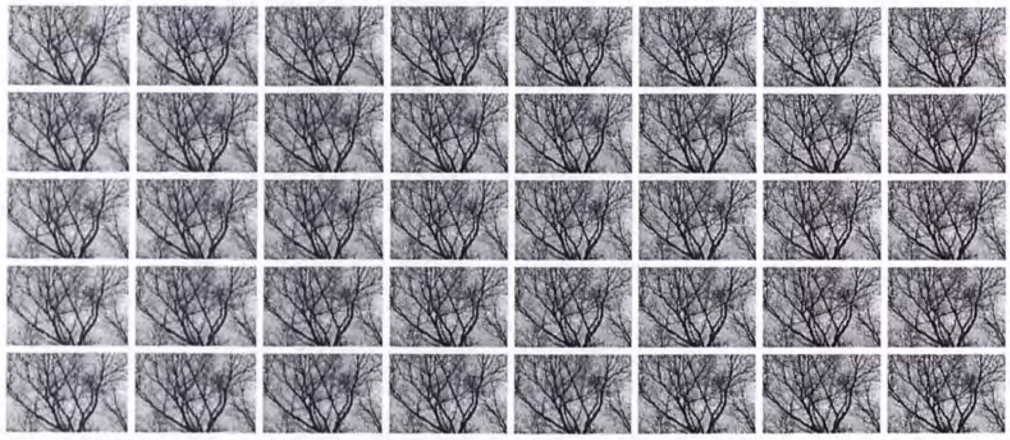

Figure 5.11: Tree. From top to bottom: Samples of the original video, samples of extrapolated sequence using LDS model, samples of extrapolated sequence of MARMA using 3-level laplacian pyramid, 3-level haar wavelet, and 3-level steerable pyramid (all using $n = 40$ components, $\tau = 120$)

Table 5.5: Reconstruction error for the tree video.

| $\|\|Y - Y'\|\|$ | LDS | Laplacian Pyramid | Wavelet Analysis | Steerable Pyramid |
|---|---|---|---|---|
| $n = 20$ | 24.9935 | 22.8309 | 24.2517 | 20.9935 |
| $n = 40$ | 16.3066 | 14.7844 | 14.5836 | 14.4518 |
| $n = 80$ | 15.5745 | 13.9985 | 14.1822 | 13.842 |

and more importantly, dynamic behaviors are separated into local and global basis, the proposed scheme enables us to harness better. Also, we present a comparative study on choosing different kind of multi-resolution analysis techniques. From the tested examples, we found that, the better the multi-resolution descriptors describing the textured images, the better the synthesized result, and because steerable pyramid are verified to be one of the best descriptors in representing textures, it performs better than other two multi-resolution methods.

Our future work will focus on extending the same scheme to

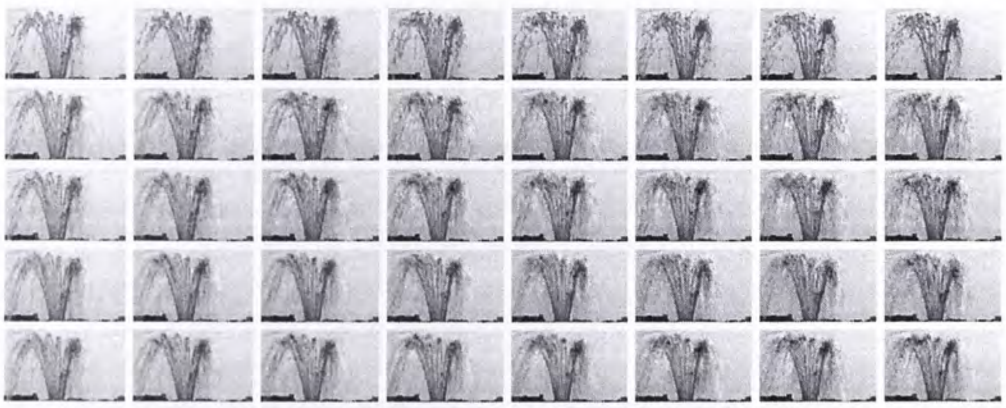

Figure 5.12: Fountain. From top to bottom: Samples of the original video, samples of extrapolated sequence using LDS model, samples of extrapolated sequence of MARMA using 3-level laplacian pyramid, 3-level haar wavelet, and 3-level steerable pyramid (all using $n = 40$ components, $\tau = 120$)

Table 5.6: Reconstruction error for the fountain video.

| $\|\|Y - Y'\|\|$ | LDS | Laplacian Pyramid | Wavelet Analysis | Steerable Pyramid |
|---|---|---|---|---|
| $n = 20$ | 35.319 | 31.5948 | 25.2886 | 22.3614 |
| $n = 40$ | 23.4547 | 22.7937 | 21.1182 | 19.9252 |
| $n = 80$ | 21.468 | 20.4588 | 20.2605 | 19.8335 |

color images, and try to investigate the usage of this scheme for dynamic texture recognition. We believe that, the multi-resolution description of the dynamic behavior should perform better in dynamic texture recognition by carefully choosing the weights for different levels of dynamic behaviors than the original LDS model.

□ **End of chapter.**

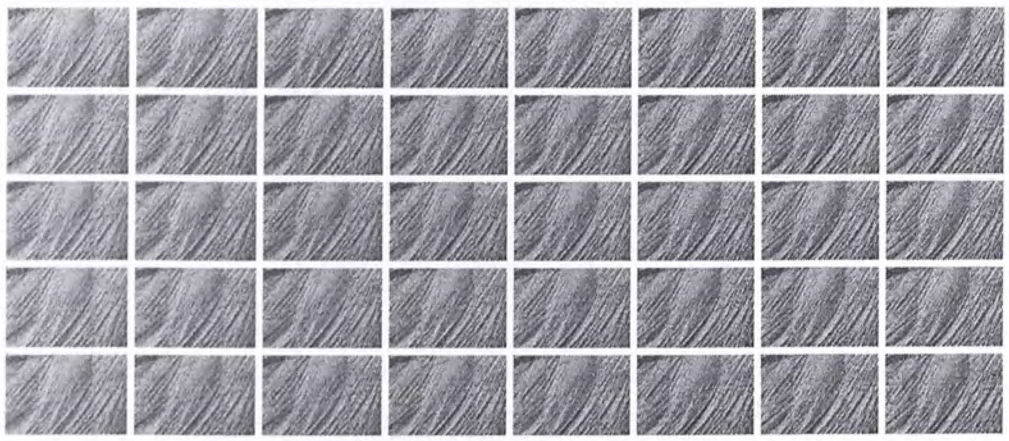

Figure 5.13: Plastic. From top to bottom: Samples of the original video, samples of extrapolated sequence using LDS model, samples of extrapolated sequence of MARMA using 3-level laplacian pyramid, 3-level haar wavelet, and 3-level steerable pyramid (all using $n = 40$ components, $\tau = 120$)

Table 5.7: Reconstruction error for the plastic video.

| $\|\|Y - Y'\|\|$ | LDS | Laplacian Pyramid | Wavelet Analysis | Steerable Pyramid |
|---|---|---|---|---|
| $n = 40$ | 55.1431 | 23.8466 | 20.237 | 13.4551 |
| $n = 80$ | 28.2665 | 22.1416 | 13.7776 | 11.5155 |
| $n = 20$ | 16.8635 | 14.9321 | 13.72 | 11.1025 |

# Chapter 6

# Motion Transfer

In the previous chapters, we are focusing on dynamic texture synthesis from a given input sequence of image. In fact, it could be regarded as a special case of the concept of what we call as *Motion Transfer*. The aim of motion transfer is to transform the dynamic behaviors of a given dynamic textures video to a static image to synthesize a new dynamic textures video which has similar dynamic behavior as the given video while the appearance of the synthesized video is similar to the static image[1].

In[51], the problem of motion transfer is implicitly addressed by transfer the trajectories of the learned textons and movetons to the destination image. But the method does not perform well when the number of the basic elements are too large. Our concern here is to give a former definition of motion transfer under the power of state-of-art linear dynamic system framework which is verified to have great performance in dynamic texture analysis, recognition, and editing.

[1]The traditional synthesis process can been seen in the motion transfer basis as: the input is the video sequence and the image which is the first frame (or any other frame in the sequence) of the video, and the aim is to transfer the dynamic behavior of the video to the image to get a new video sequence with arbitrary length.

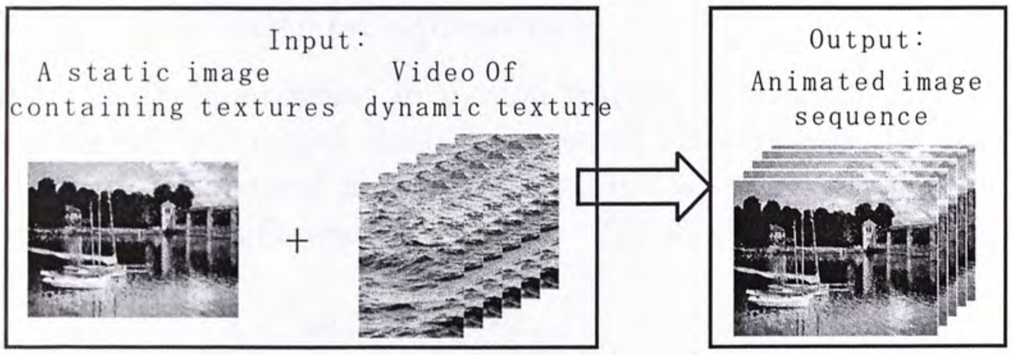

Figure 6.1: Motion transfer: a given static scene picture, and a dynamic texture video (the water sequence in this example),to synthesize a new video sequence to make the water in the image to flow like the given video

## 6.1 Problem formulation

Under the framework of the linear dynamic system, the problem can be formulated as the following:

Table 6.1: Problem Formulation for Motion Transfer

| | |
|---|---|
| Input: | a static image $I$ |
| | a video sequence $Y_1^\tau = \{z(1) \cdots z(\tau)\}$ with $\tau$ frames |
| Output: | a video sequence $Z_1^\tau = \{y(1) \cdots y(\tau)\}$ with $\tau$ frames |
| Constraints: | 1. $Z_1^\tau$ should share similar appearance with $I$ |
| | 2. $Z_1^\tau$ should similar dynamic behavior with $Y_1^\tau$ |

The similarity measurements of the appearance and dynamic behavior between the output video and the inputs should be qualitatively defined.

### 6.1.1 Similarity on Appearance

One of the constraints in motion transfer is that the all the frames in the output video sequence $Z_1^\tau$[2] should look like the given static image $I$ in appearance. That is to say, we want to minimize the difference between all $\tau$ frames in $Z$ and $I$:

$$S_{app} = \arg\min_{Z_1^\tau} \sum_{t=1}^{\tau} \parallel z(t) - I \parallel \tag{6.1}$$

where $\parallel x - y \parallel$ is the measurement of difference between $x$ and $y$.

Multi-resolution analysis is a good way to measure the similarity between static textured images [27] [10] [33]. Generally, the similarity process can be divided into two stages: First is *Feature Extraction*, where are set of features is generated to accurately represents the true content of the image; The second task is to perform *Similarity Measurement* given the features using appropriate similarity functions.

#### Feature Extraction

Several multi-resolution features, i.e., Harr wavelet, dyadic wavelet, Gabor filters, and steerable pyramid, are good choices to extract image features. Within all the multi-resolution and multichannel transform algorithms, Gabor filters and steerable pyramid are verified to outperform others in classification of image textures [27]. In our experiment, steerable pyramid is used, and the subbands for the input image $I$ is represented by $I_i$ which is the $i_{th}$ subband; $z(t)_i$ is the $i_{th}$ subband for frame $t$ of the output video[3].

[2]The length of the output video can be of any number, but for simplicity and consistency of representation, we consider the output video sequence also with $\tau$ frames

[3]the features or the subbands for the texture is represented in the form of feature vectors

**Similarity Measurement**

Given the features generated by the feature extraction algorithm, the similarity of texture is measured as distance of feature vectors. Commonly, the $L_1, L_2, L_\infty$ norm are used:

$$\begin{aligned} L_1 &= \sum_i |I_i - z(t)_i| \\ L_2 &= \sum_i |I_i - z(t)_i|^2 \\ L_\infty &= \max_i |I_i - z(t)_i| \end{aligned}$$

More complex measurement functions could be found in [10], [33], which are more comparable to the human perception of textures. In our experiments, the $L_2$ norm will be used because of its simple representation.

### 6.1.2 Similarity on Dynamic Behavior

Another requirement in motion transfer is that the output dynamic texture should resemble the input video in its dynamic behavior. That is, we want to minimize the distance between the motion flow of $Y_1^\tau$ and $Z_1^\tau$:

$$S_{motion} = \min_{Z_1^\tau} \parallel Y - Z \parallel \tag{6.2}$$

As the state-space model captures the motion flow of the dynamic textures more globally, we would like to take the advantage of LDS to measure the similarity between two dynamic textures. The LDS model representations for $Y$ and $Z$ are:

$$\begin{cases} x_y(t+1) = A_y x_y(t) + v_y(t) & v_y(t) \sim \mathcal{N}(0, Q_y) \\ y(t) = C_y x_y(t) + w_y(t) & w_y(t) \sim \mathcal{N}(0, R_y) \end{cases} \tag{6.3}$$

for the input video and the LDS model parameters are $\Theta_y = \{A_y, B_y, C_y, x_y(0)\}$

$$\begin{cases} x_z(t+1) = A_z x_z(t) + v_z(t) & v_z(t) \sim \mathcal{N}(0, Q_z) \\ z(t) = C_z x_z(t) + w_z(t) & w_z(t) \sim \mathcal{N}(0, R_z) \end{cases} \tag{6.4}$$

for the output video and the LDS model parameters are $\Theta_z = \{A_z, B_z, C_z, x_z(0)\}$.

### Probability Distribution

Dynamic textures exhibit characteristic stochastic motion, and the intrinsic motion behavior is embedded in the state variables ($X_y$ and $X_z$ in this case) in the LDS model. As the problem is modeled as probability distribution of the Gauss-Markov process, the probability distribution for the state variables can be recursively proved to be a Gaussian distribution:

$$x(t) \sim \mathcal{N}(\mu(t), \Sigma(t)) \tag{6.5}$$

where $\mu(t) = A^t x(0)$ and $\Sigma(t) = \sum_{i=0}^{t-1} A^i Q (A^i)^T$ (Please refer to Appendix C for detailed derivation).

### Distance Measures using Kullback–Leibler Divergence

*Kullback-Leibler divergence*(K-L divergence for short) or call the *relative entropy* is a natural measurement for analyzing the difference between two probability distributions, which is widely adopted in information theory and probability theory. Formally, the K-L divergence from a probability distribution $p(x)$ to an arbitrary probability distribution $q(x)$ is defined as:

$$D_{KL}(p(x) \parallel q(x)) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{6.6}$$

with discrete random variables, and

$$D_{KL}(p(x) \parallel q(x)) = \int_X p(x) \log \frac{p(x)}{q(x)} dx \tag{6.7}$$

with continuous random variable over the set $X$. Especially, when the two are gaussian distributed with $\mathcal{N}(\mu_0, \Sigma_0)$ and $\mathcal{N}(\mu_1, \Sigma_1)$, the K-L divergence will become:

$$\begin{aligned} & D_{KL}(p(x) \parallel q(x)) \\ = \ & \tfrac{1}{2}(\log(\tfrac{\det \Sigma_1}{\det \Sigma_0}) + tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1}(\mu_1 - \mu_0)) \end{aligned} \tag{6.8}$$

With the joint probability distribution of state vector sequence, the K-L divergence between two different dynamic system state vectors can be calculated as:

$$\begin{aligned} & D_{KL}(p(X) \parallel q(X)) \\ = \ & \lim_{t \to \infty} \tfrac{1}{\tau} D_{KL}(p(x_1^\tau | p(x_1^\tau)) \\ = \ & \tfrac{1}{2}[\log \tfrac{|Q_q|}{|Q_p|} + tr(Q_q^{-1}Q_p) - n + \tfrac{1}{\tau} \parallel A_p x_p(0) - A_q x_q(0) \parallel_{Q_q}^2 \\ & + \tfrac{1}{\tau} \textstyle\sum_{i=2}^{\tau} tr(\bar{A}^T Q_q^{-1} \bar{A}(\Sigma_p(i-1)) + \mu_p(i-1)\mu_p(i-1)^T)] \end{aligned} \tag{6.9}$$

where the two dynamic systems are parameterized as $\{A_p, Q_p, x_p(0)\}$ and $\{A_q, Q_q, x_q(0)\}$, $\bar{A} = A_p - A_q$. Please refer to [6] for more details of the proof and the KL divergence in observation space. Table 6.2 shows some examples of the KL divergence calculated from 4 dynamic textures. Notice that $D_{KL}(p(x)||q(x)) \neq D_{KL}(q(x)||p(x))$, and $D_{KL}(p(x)||p(x))$ equals to zero, which is the lower bound for KL divergence.

coherence, and recursively change video to the target video to match the dynamic behavior. We call this *Image Based Motion Transfer*. In the following sections, we will talk about these two different ideas in more details and present prototypes to do motion transfer.

## 6.2 Further Work

We have done some preliminary work on the topic of motion transfer: give the formal definition of motion transfer under the concept of LDS model; we form the two objective functions which analyze the appearance and dynamic behavior similarities separately. Further work focus on finding a suitable methodology to solve the objective function.

---

□ **End of chapter.**

Table 6.2: The KL divergence between 4 sample dynamic textures

| | Large wave | Small wave | Smoke | Elevator |
|---|---|---|---|---|
| Large wave | 0.0 | 115.2 | 1692.2 | 892.2 |
| Small wave | 200.4 | 0.0 | 962.7 | 265.6 |
| Smoke | 194.1 | 216.9 | -0.0 | 631.7 |
| Elevator | 214.4 | 134.0 | 886.9 | -0.0 |

## 6.1.3 The Objective Function

The problem can be formulated as the objective function given the two constraints both in appearance space and state space (motion flow) as the following:

Equation 6.1 for computing the similarity in appearance space could be rewritten as:

$$S_{app} = \min_{Z_1^\tau} \sum_t \sum_i |I_i - z(t)_i|^2 \qquad (6.10)$$

where $t$ stands for the $th_{th}$ frame, and $i$ stands for the $i_{th}$ subband of the feature space.

Equation 6.2 for computing the similarity in state space could be formulated as:

$$S_{motion} = \min_{Z_1^\tau} D_{KL}(p(X_Y) \parallel p(X_Z)) \qquad (6.11)$$

where $p(X_Y)$ and $p(X_Z)$ are the joint probability distribution for the input and output dynamic texture sequence separately.

Generally, there are two ways of thinking of the problem: first is that we change the appearance of the input video to match the appearance of the input image while keeping the dynamic behavior of the video. We call this *Video Based Motion Transfer*; And another one is that we start with the input image to form an initial video sequence which is not necessary to have time

# Chapter 7

# Conclusions

Synthesis of dynamic textures such as flowing water, blowing trees, burning fires, is of great importance in game development, virtual and augmented reality, and video processing. The aim of our work is to develop and analyze different parametric models to represent the input sample dynamic textures, which are then used to predict and extrapolating the finite sequence of images to the longer video sharing similar dynamic behaviors.

In this thesis, we have explored the state-of-art methodologies in dynamic texture synthesis and presented a thorough study on the recent advances in this research topic. We first propose the LDS model from the gray level video sequence to colored dynamic textures by compressing the color channels in the YUV space using laplacian pyramid and radial basis functions (RBF). The synthesize results are better than the algorithms that focus on dynamic color textures, and the size of the matrix to be dealt with is greatly reduced than the LDS model. Further, we develop the multi-resolution auto-regressive moving average model (MARMA) in the frequency domain for betterment of the synthesis result, and provide a comparative study on using multi-resolution analysis methods, including laplacian pyramid, Haar wavelets, and steerable pyramid. The experimental results showed improvements in the synthesized dynamic textures.

# Appendix A

# List of Publications

1. **Dynamic Color Texture Synthesis in the YUV Color Sapce**, Leilei Xu, Hanqiu Sun, Jiaya Jia, Chenjun Tao, *In Proceedings of the International Conference on Entertainment Computing 2007 (ICEC07), Lecture Notes in Computer Science, Springer*, 2007.

2. **Hardware-accelerated Parallel-Split Shadow Maps** Fan Zhang, Hanqiu Sun, Leilei Xu and Lee Kit Lun, *To appear in the International Journal of Image and Graphics (IJIG), 2007*, special issue of VRCIA'06, in press, 2006

3. **Parallel-Split Shadow Maps for Large-scale Virtual Environments**, Fan Zhang, Hanqiu Sun, Leilei Xu and KitLun Lee, *in Proceedings of ACM/SIGGRAPH VRCIA 2006*, Published by ACM SIGGRAPH, pp.311-318, 2006

4. **Generalized Linear Perspective Shadow Map Reparameterization**, Fan Zhang, Leilei Xu, Chenjun Tao, Hanqiu Sun, *in Proceedings of ACM/SIGGRAPH VRCIA 2006*, Published by ACM SIGGRAPH, 14-17 June, 2006.

5. **High Quality Shadow Rendering with Parallel-Split Shadow Maps**, Fan Zhang, Hanqiu Sun, Leilei Xu and KitLun Lee, *in Proceedings of the 7th ACM Postgraduate*

*Research Day 2006 (ACM-HK Annually Local Activity)*, Published by ACM Hong Kong Chapter, 2006.

6. **Dynamic Texture Synthesis Using Multi-resolution Analysis**, Leilei Xu, Hanqiu Sun, JianbinShen, Jiaya Jia, *in submission to Journal of Computer Animation and Virtual World (CAVW)*, 2007.

□ **End of chapter.**

# Appendix B

# Degeneracy in LDS Model

## B.1 Equivalence Class

The equivalence class of solutions to LDS model

$$\mathcal{R} \doteq \{[A] = TAT^{-1}, [C] = CT^{-1}, [Q] = TQT^T, [x(0)] = Tx(0)\} \tag{B.1}$$

where $\{A, C, Q, x(0)\}$ is one of the solutions to the model. We can verify the class of solutions as follows:

$$\left.\begin{array}{r} \left.\begin{array}{r} x'(t+1) = TAT^{-1}x'(t) + TBv(t) \\ x'(0) = Tx(0) \end{array}\right\} \Rightarrow x'(t) = Tx(t) \\ \\ y'(t) = CT^{-1}x'(t) + w(t) \end{array}\right\} \Rightarrow y'(t) = y(t) \tag{B.2}$$

where $BB^T = Q$, $v(t) \sim \mathcal{N}(0,1)$ and $w(t) \sim \mathcal{N}(0,R)$. $A, C, Q, x(0)$ is one of the solution to the model.

## B.2 The Choice of the Matrix $Q$

$Q$ could be constrained to an identity matrix without generality particularly since $Q$ is a covariance matrix and is symmetric positive semidefinite and thus can be diagonalized to the form

$E\Lambda E^T$ (the SVD decomposition). Thus for any model in which Q is not the identity matrix, we can generate an equivalent model in which $Q'$ is identity.

$$\begin{aligned}
& x(t+1) = Ax(t) + Bv'(t), \ \mathit{with}\ (v'(t) \sim \mathcal{N}(0,1), Q = BB^T = E\Lambda E^T) \\
\Rightarrow\ & x(t+1) = Ax(t) + E\Lambda^{1/2}v'(t) \\
\Rightarrow\ & (\Lambda^{-1/2}E^T)x(t+1) = (\Lambda^{-1/2}E^T)A(E\Lambda^{1/2})[(\Lambda^{-1/2}E^T)x(t)] + v'(t) \\
\Rightarrow\ & \begin{cases} x'(t) = (\Lambda^{-1/2}E^T)x(t) \\ A' = (\Lambda^{-1/2}E^T)A(E\Lambda^{1/2}) \\ C' = C(E\Lambda^{1/2}) \end{cases}
\end{aligned} \tag{B.3}$$

In this case, the new state vector $x'(t)$ has a covariance of $Q' = I$, the identity matrix.

## B.3 Swapping the Column of $C$ and $A$

Swapping the columns of the matrix $C$ and $A$ corresponds to reorder the components of state vectors.

Because the change of order of the components in $x(t)$ will not affect the covariance $Q$ of the state sequence, then we have

$v(t) = v'(t)$, then we have:

$$\begin{pmatrix} \vdots \\ x^i(t+1) \\ \vdots \\ x^j(t+1) \\ \vdots \end{pmatrix} = \begin{pmatrix} \cdots & A_i & \cdots & A_j & \cdots \end{pmatrix} \begin{pmatrix} \vdots \\ x^i(t) \\ \vdots \\ x^j(t) \\ \vdots \end{pmatrix} + v(t)$$

$$\Rightarrow \begin{cases} \begin{pmatrix} \vdots \\ x^j(t+1) \\ \vdots \\ x^i(t+1) \\ \vdots \end{pmatrix} = \begin{pmatrix} \cdots & A_j & \cdots & A_i & \cdots \end{pmatrix} \begin{pmatrix} \vdots \\ x^j(t) \\ \vdots \\ x^i(t) \\ \vdots \end{pmatrix} + v(t) \\ y(t) = \begin{pmatrix} \vdots \\ y^i(t) \\ \vdots \\ y^j(t) \\ \vdots \end{pmatrix} = \begin{pmatrix} \cdots & C_j & \cdots & C_i & \cdots \end{pmatrix} \begin{pmatrix} \vdots \\ x^j(t) \\ \vdots \\ x^i(t) \\ \vdots \end{pmatrix} + w(t) \end{cases} \tag{B.4}$$

where $x^i$ means the $i_{th}$ component of vector $x$, and $A_i$ means the $i_{th}$ column vector of matrix $C$.

# Appendix C

# Probability Density Functions

## C.1 Probability Distribution

The linear dynamic system is a Gaussian noise driven auto-regressive process. We first obtain the probability density functions of the state vectors $x(t)$. In the following we will assume that $x(0)$ is constant. According to [25], the state vectors are governed by a Gaussian Markov process, hence the conditional probability of the state vector is:

$$\begin{aligned} p(x(t)|x(t-1)) &= \mathcal{G}(x(t), x(t-1), Q) \\ &= \frac{1}{\sqrt{(2\pi)^n |Q|}} e^{-\frac{1}{2}\|x(t)-Ax(t-1)\|_Q^2} \end{aligned} \tag{C.1}$$

where $\| x \|_Q^2 = x^T Q^{-1} x$. Recursively substituting into the state equation we get:

$$x(t) = A^t x(0) + \sum_{t=1}^{t} A^{t-i} v(i) \tag{C.2}$$

As a single state is the linear combination of independently distributed Gaussian random variables, the probability of a single state is also Gaussian:

$$p(x(t)) = \mathcal{N}(\mu(t), \Sigma(t)) \tag{C.3}$$

where

$$\begin{cases} \mu(t) & = & A^t x(0) \\ \Sigma(t) & = & A\Sigma(t-1)A^T + Q = \Sigma_{i=0}^{t-1} A^i Q (A^i)^T \end{cases} \tag{C.4}$$

As the observation vector $y(t)$ is a linear combination of $x(t)$ and the noisy term $w(t)$ which is independent from $x(t)$, we can easily calculate the probability distribution for $y(t)$:

$$p(y(t)) = \mathcal{N}(\gamma(t), \Phi(t)) \tag{C.5}$$

where

$$\begin{cases} \gamma(t) & = & C\mu(t) = CA^t x(0) \\ \Phi(t) & = & C\Sigma(t)C^T + R = C(\Sigma_{i=0}^{t-1} A^i Q (A^i)^T) C^T + R \end{cases} \tag{C.6}$$

## C.2 Joint Probability Distributions

After obtaining the probability distribution for each individual state and observation, we define the join probabilities for state sequence and observation sequence which form the basis for formulating the synthesis and recognition problems. Since the driving process is Gaussian, the join probability of a state sequence $x_1^\tau$ is also Gaussian. Specifically we have:

$$p(x_1^\tau) = \mathcal{N}(\mu, \Sigma) \tag{C.7}$$

where

$$\left\{ \begin{array}{lcl} \mu & = & \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_\tau \end{pmatrix} \\ \Sigma & = & \begin{pmatrix} \Sigma_1 & (A\Sigma_1)^T & (A^2\Sigma_1)^T & \cdots & (A^{\tau-1}\Sigma_1)^T \\ A\Sigma_1 & \Sigma_2 & (A\Sigma_2)^T & \cdots & A^{\tau-2}\Sigma_2 \\ A^2\Sigma_1 & A\Sigma_2 & \Sigma_3 & \cdots & A^{\tau-3}\Sigma_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A^{\tau-1}\Sigma_1 & A^{\tau-2}\Sigma_2 & A^{\tau-3}\Sigma_3 & \cdots & \Sigma_\tau \end{pmatrix} \end{array} \right. \tag{C.8}$$

because

$$\begin{array}{rl} & x(t+k) = A^k x(t) + \Sigma_{i=1}^{k} A^{k-i} B v(t+i-1) \\ \Rightarrow & cov(x(t+k), x(t)) = A^k \Sigma(t) \end{array} \tag{C.9}$$

And the joint probability of a observation sequence $y_1^\tau$ can be represented as:

$$p(y_1^\tau) = \mathcal{N}(\gamma, \Phi) \tag{C.10}$$

where

$$\left\{ \begin{array}{lcl} \gamma & = & \mathbf{C}\mu \\ \Phi & = & \mathbf{C}\Phi\mathbf{C}^T + \mathbf{R} \end{array} \right. \tag{C.11}$$

$$\text{where } \mathbf{C} = \begin{pmatrix} C & 0 & \cdots & 0 \\ 0 & C & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & C \end{pmatrix} \text{ and } \mathbf{R} = \begin{pmatrix} R & 0 & \cdots & 0 \\ 0 & R & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & R \end{pmatrix}$$

# Bibliography

[1] B. Abraham, O. Camps, and M. Sznaier. Dynamic texture with fourier descriptors. *IEEE Texture 2005 in conjunction with ICCV*, Oct. 2005.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, Sept. 2004.

[3] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. *In IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *In IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI)*, volume 23, pages 1222–1239, 2001.

[5] N. Campbell, C. Dalton, D. Gibson, and B. Thomas. Practical generation of video textures using the auto-regressive process. *In Proceedings of British Machine Vision Conference*, pages 434–443, 2002.

[6] A. B. Chan and N. Vasconcelos. Efficient computation of the kl divergence between dynamic textures. Technical report, statistical visual computing laboratory(SVCL), University of California, San Diego, 2004.

[7] D. Chetverikov and R. Péteri. A brief survey of dynamic texture description and recognition. *In Proceedings 4th Int. Conf. on Computer Recognition Systems*, pages 17–26, 2005.

[8] Y. Chuang, D. B. Goldman, K. C. Zheng, B. Curless, D. Salesin, and R. Szeliski. Animating pictures with stochastic motion textures. *ACM Transactions on Graphics*, 24(3):853–860, 2005.

[9] J. M. Coggins. *A Framework for Texture Analysis Based on Spatial Filtering.* PhD thesis, Computer Science Department, Michigan State University, East Lansing, Michigan, 1982.

[10] M. Do. Texture similarity measurement using kullback-leibler distance on wavelet subbands. volume 3, pages 730–733, 2000.

[11] D. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto. Dynamic textures. *International Journal of Computer Vision*, (2):91–109, 2003.

[12] G. Doretto, D. Cremers, P. Favaro, and S. Soatto. Dynamic texture segmentation. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, Washington, DC, USA, 2003. IEEE Computer Society.

[13] G. Doretto and S. Soatto. Editable dynamic textures. *CVPR*, 2003.

[14] J. Filip, M. Haindl, and D. Chetverikov. Fast synthesis of dynamic colour textures. *Proceedings of the 18th IAPR Int. Conf. on Pattern Recognition(to appear)*, 2007.

[15] A. Fitzgibbon. Stochastic rigidity: Image registration for nowhere-static scenes. pages I: 662–669, 2001.

[16] Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural Computation*, (12):831–864, 2000.

[17] G. Golub and C. V. Loan. *Matrix Computations, 2nd edn.* Johns Hopkins University Press, 1989.

[18] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B*, 51, 1989.

[19] J. Hammersley and P. Clifford. *Markov field on finite graphs and lattices.* Unpublished, 1971.

[20] R. M. Haralick. Statistical and structural approaches to texture. In *Proceedings of the IEEE*, pages 786–804. IEEE Computer Society, 1979.

[21] J. K. Hawkins. Textural properties for pattern recognition. In *Picture Processing and Psychopictorics*, New York, 1969. Academic Press.

[22] H. Hotelling. Analysis of a complex of statistical variables into principal components. In *Journal of Educational Psychology*, volume 24, page 417.

[23] K. Jack. *Video Demystified (3rd edition).* lsevier Science and Technology, 2001.

[24] T. Kailath. *Linear Systems.* Prentice Hall, Englewood Cliffs, NJ., 1980.

[25] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory.* Prentice Hall Signal Processing Series, 1993.

[26] S. Li. *Markov random field modeling in computer vision.* Springer-Verlag London, UK, 1995.

[27] S. Li and J. S. Taylor. Comparison and fusion of multiresolution features for texture classification. 26(5):633–638, April 2005.

[28] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Transactions on Graphics (TOG)*, 23:303–308, 2004.

[29] Y. Li, T. Wang, and H. Y. Shum. Motion texture: A two-level statistical model for character motion synthesis. *In Proceedings of ACM SIGGRAPH*, pages 465–472, 2002.

[30] C. Liu, R. Lin, N. Ahuja, and M. Yang. Dynamic textures synthesis as nonlinear manifold learning and traversing. *The 17th British Machine Vision Conference (BMVC 2006)*, pages 859–868, 2006.

[31] L. Ljung. *System Identification – Theory for the User (2nd Edition)*. Prentice Hall, 1999.

[32] C. L. Lu Yuan, Fang Wen and H.-Y. Shum. Synthesizing dynamic texture with closed-loop linear dynamic system. In *ECCV (2)*, pages 603–616, 2004.

[33] J. R. Mathiassen, A. Skavhaug, and K. Bø. Texture similarity measure using kullback-leibler divergence between gamma distributions. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 133–147, London, UK, 2002. Springer-Verlag.

[34] R. Péteri and M. Huiskes. A comprehensive database of dynamic textures. Technical report.

[35] J. H. Piater. Mixture models and expectation-maximization. *Lecture at ENSIMAG*, May 2002.

[36] W. Richards and A. Polit. Texture matching. *Kybernetic*, 16.

[37] R.W.Hunt. *The Reproduction of Colour, 5th edn.* Fisher Books, 1996.

[38] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. *Proceedings of SIGGRAPH 2000*, pages 489–498, July 2000.

[39] A. Schödl and I. A. Essa. Controlled animation of video sprites. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 121–127. ACM Press, 2002.

[40] R. Shumway and D. Stoffer. *Time Series Analysis and Its Applications.* Springer, 2000.

[41] E. P. Simoncelli and W. T. Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *International Conference on Image Processing*, volume 3, pages 444–447, 23-26 Oct. 1995, Washington, DC, USA, 1995.

[42] J. Sklansky. Image segmentation and feature extraction. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 237–247, 1978.

[43] S. Soatto, G. Doretto, and Y. N. Wu. Dynamic textures. pages 439–446, July 2001.

[44] J. Sun, N. Zheng, and H. Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[45] M. Szummer. Temporal texture modeling. Technical Report 346, MIT Media Lab Perceptual Computing, 1995.

[46] M. Szummer and R. W. Picard. Temporal texture modeling. *In Proceedings of IEEE International Conference on Image Processing*, 3(1-2):823–826, 1995.

[47] H. Tamura, S. Mori, and Y. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 460–473, 1978.

[48] B. Thirion, V. R. B. Bascle, and N. Navab. Fusion of color, shading and boundary information for factory pipe segmentation. *In IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[49] M. Tipping and C. Bishop. Probabilistic principal component analysis. Technical report, Neural Computing Research Group, Aston University, 1997.

[50] I. E. G. T. V. Kwatra, A. Schödl and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003*, 22(3):277–286, July 2003.

[51] Y. Wang and S. Zhu. Analysis and synthesis of textured motion: Particles and waves. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10):1348–1363, 2004.

[52] L. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGraph-00*, pages 479–488, 2000.

[53] Weiss and Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEETIT: IEEE Transactions on Information Theory*, 47, 2001.

[54] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, 2001.

[55] F. Woolfe and A. Fitzgibbon. Shift-invariant dynamic texture recognition. In *ECCV06*, pages 549–562, 2006.

[56] C. S. Zhu, N. Y. Wu, and D. Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8), Nov 1997.

[57] S. Zhu, C. Guo, Y. Wang, and Z. Xu. What are textons? *Int. J. Comput. Vision*, 62(1–2):121–143, 2005.

[58] D. L. Ziv Bar-Joseph, Ran El-Yaniv and M. Werman. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):120–135, 2001.

[59] A. Zomet and S. Peleg. Multi-sensor super-resolution. In *WACV '02: Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*, page 27. IEEE Computer Society, 2002.

[60] S. W. Zucker and K. Kant. Multiple-level representations for texture discrimination. In *Proceedings of the IEEE Conference on Pattern Recognition and Image Processing*, pages 609–614, Dallas, TX, 1981. IEEE Computer Society.